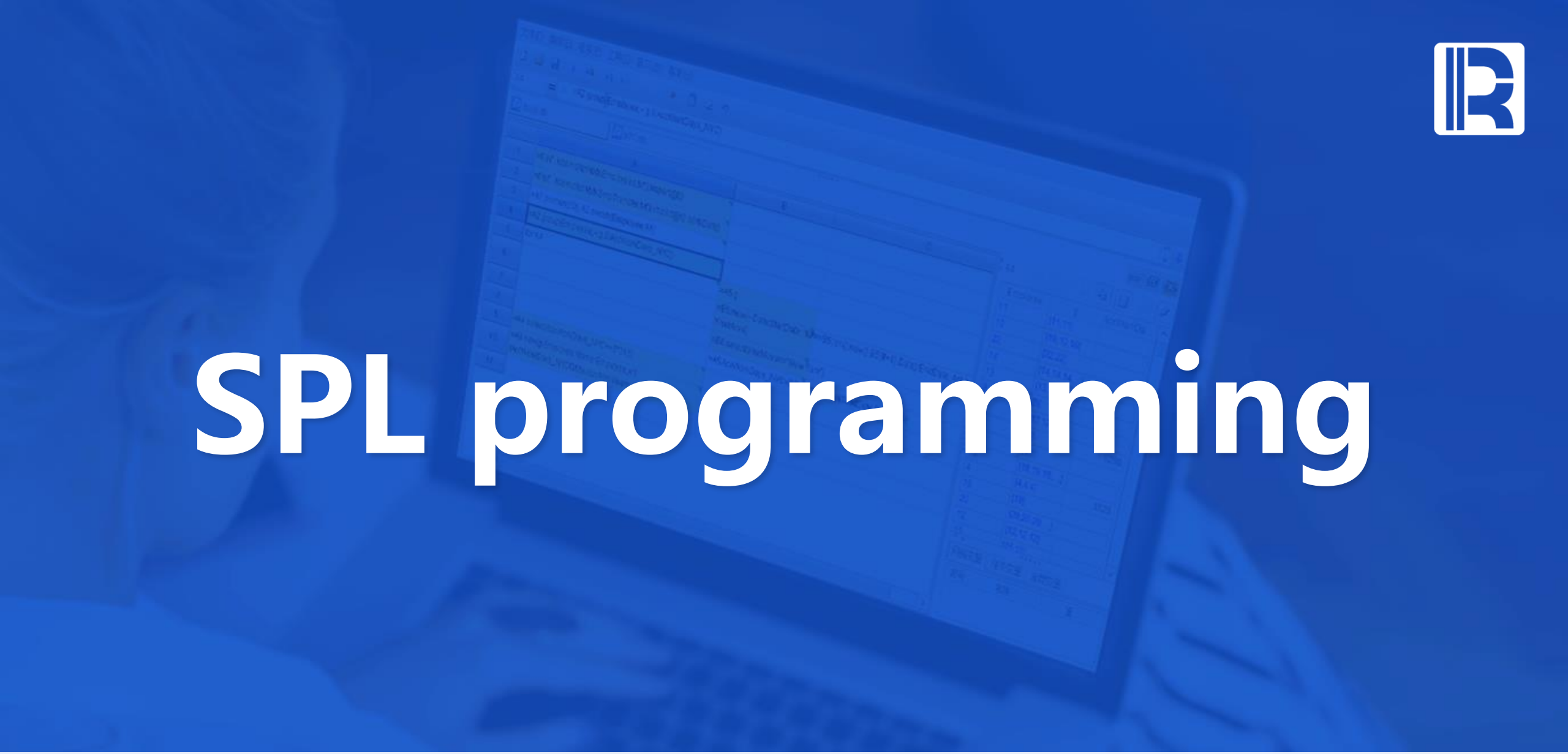


SPL programming

www.raqsoft.com

Contents

Chapter 1 Basic concepts of programming 12

1.1 Computer program and language 13

1.2 esProc installation 19

1.3 The simplest program 24

1.4 More simple program examples 30

1.5 Program structure 42

1.6 The task of programming 46

1.7 Exercises 48

Chapter 2 Algorithm and program 54

2.1 What is algorithm? 55

2.2 Simple algorithm examples 59

2.3 Characteristics of the algorithm 74

2.4 Methods of representing algorithm 76

Represent algorithm by flow chart 78

Represent algorithm by N-S flow chart 91

Represent algorithm by pseudo code 94

Represent algorithm by SPL 96

2.5 Structured programming method 100

2.6 Exercises 102

Chapter 3 Sequential structure programming 111

3.1 Examples of sequential programming 112

3.2 The representation of data and its operation 119

Constant and variable 120

Data type 124

Identification of data types 134

3.3 Operators and Expressions 136

Common operators in SPL 137

Basic arithmetic operators 139

Operator precedence..... 140
Mixed operations of multi type data..... 141
Forced type conversion operators..... 143
3.4 SPL statement..... 144
3.5 Input and output of data..... 158
Input/output examples..... 159
Input and output data in SPL..... 161
3.6 Exercises..... 162
Chapter 4 Selection structure programming..... 170
4.1 Selection structure and condition judgment..... 171
4.2 Implement selection structure with if statement..... 173
Example of selection structure..... 174
The general form of if statement..... 182
Other forms of if statement..... 184
4.3 Relational operators and relational expression..... 186
Relational operators and their precedence..... 187
Relational expression..... 189
4.4 Logical operators and logical expression..... 190
Logical operators and their precedence..... 191
Logical expression..... 193
4.5 Function if..... 196
4.6 Nesting of selection structure..... 202
4.7 Exercises..... 205
Chapter 5 Loop structure programming..... 225
5.1 The need for loop structure..... 226
5.2 Implement loop structure with for x statement..... 228
The general form of for x statement..... 229
Loop of specified cycle number..... 234
Break and next statements..... 240
5.3 For statement that specifies the scope of the loop..... 248

for a, b statement..... 249
for a, b, s statement..... 252
5.4 Nesting of loop structure..... 255
Nesting of loop structure..... 256
Using break and next in multi-layer loops..... 259
5.5 Examples of loop structure..... 262
5.6 Exercises..... 271
Chapter 6 Text and string..... 287
6.1 String..... 288
String constant..... 289
String operator..... 291
Special strings..... 292
Escape character and special characters..... 293
6.2 String conversion..... 296
Character and ASCII..... 297
Coding and conversion..... 301
String and display format..... 303
6.3 Dynamic program..... 310
Macro..... 311
eval function..... 312
6.4 Text file..... 316
6.5 String functions..... 323
Character function..... 324
String functions..... 326
Regular expression..... 330
6.6 Exercises..... 332
Chapter 7 Function and subroutine..... 348
7.1 What is a function? 350
7.2 Calling a function in SPL..... 352
Calling a function..... 353

Parameters of function in SPL..... 357
Function options in SPL..... 359
7.3 Subroutine..... 362
Definition and call of subroutine..... 363
Parameter transfer in subroutine..... 366
Scope of variables..... 367
Calling procedure of subroutine..... 369
Nesting of subroutines..... 370
Recursion of subroutines..... 373
7.4 Cross cellset calling..... 374
Cross cellset calling..... 375
Parameter transfer in cross cellset calling..... 377
Nesting and recursion of cross cellset calling..... 379
7.5 Exercises..... 380
Chapter 8 Sequence..... 394
8.1 The concept of sequence..... 395
8.2 Generation and modification of a sequence..... 401
Sequence constants..... 402
Member access of a sequence..... 409
Add a member to the end of a sequence..... 418
8.3 Loop and calculation of sequence..... 421
Execute the loop with for A..... 422
Aggregation..... 431
Examples of sequence calculation..... 434
8.4 Common sequences..... 439
A sequence of numbers..... 440
A sequence of strings..... 445
8.5 Multidimensional sequence..... 452
What is multidimensional sequence? 453
Access to multidimensional sequence members..... 455

The use of multidimensional sequences..... 457
8.6 Exercises..... 463
Chapter 9 Calculation of sequence..... 483
9.1 Set property of sequence..... 484
Ordered property of sequence..... 485
Sequence and set..... 490
9.2 Sequence functions..... 506
Generation and maintenance of sequence..... 507
String and sequence..... 519
9.3 Aggregate calculation..... 525
Count, Sum & Average..... 527
Ranking..... 535
Max/Min value and median..... 540
Other aggregate functions..... 545
9.4 Loop calculation..... 547
Complete loop calculation with A.(x) 548
Calculation by position..... 556
Loop functions..... 561
9.5 Positioning..... 570
Find the position of the specified member..... 572
Find the position of a member that meets the condition..... 575
Find the position of the Max/Min value..... 578
Overall positioning..... 583
9.6 Select..... 601
Find members that meet the conditions..... 603
Select the Max/Min value..... 606
Sorting and ranking..... 611
9.7 Calculation of multidimensional sequence..... 617
9.8 Examples of sequence calculation..... 623
9.9 Exercises..... 632

Chapter 10 Data table 642
10.1 Structured data 643
Structured data 644
Field and record 650
Data table 651
10.2 Operations of table sequence 654
Create/View a table sequence 655
Record access 657
Basic addition, deletion and modification 661
Examples and extension operations 672
10.3 External data 679
Data file 680
Reading file data 682
Exporting file data 689
Database DBMS 695
10.4 Table sequence operations 696
Aggregation 697
Filter 703
10.5 Record sequence 706
The concept of record sequence 707
Record sequence operations 717
The difference between record sequence and table sequence 729
10.6 Exercises 734
Chapter 11 Calculation of table 744
11.1 Aggregation 745
11.2 Filter and positioning 757
Filter 758
Positioning 761
Calculation of specified location 767
11.3 Aligned calculation 769

Modification..... 770
Generation..... 773
Addition..... 779
11.4 Sorting and ranking..... 782
Sorting..... 783
Sorting and positioning..... 791
Ranking..... 793
11.5 Max/Min value..... 803
Max/Min value..... 804
TOP..... 811
11.6 Exercises..... 820
Chapter 12 Order-related calculations..... 832
12.1 Ordered set..... 833
The concept of ordered set..... 834
Access by sequence number..... 836
12.2 Iterative function..... 851
12.3 Positioning calculations..... 857
Locate members..... 858
Locate max/min value..... 863
Get the sequence numbers of members that meet the conditions..... 867
12.4 Sorting and alignment..... 871
Sorting..... 872
12.5 Exercises..... 881
Chapter 13 Grouping calculations..... 897
13.1 Group aggregation..... 898
Group aggregation..... 899
Calculate unique values..... 913
13.2 The essence of grouping..... 922
Grouped subsets..... 923
Group aggregation vs. grouping..... 933

13.3 Non equivalent grouping..... 939
Equivalent grouping and non equivalent grouping..... 940
Alignment grouping..... 942
Enumeration grouping..... 950
13.4 Ordered grouping..... 960
Order of grouping..... 961
Ordered grouping..... 968
13.5 Exercises..... 978
Chapter 14 Inter table association..... 1000
14.1 Primary key and index..... 1001
Find a record by primary key value..... 1002
Index table..... 1007
14.2 Inter table association..... 1010
Inter table association..... 1011
Common associated relationships..... 1015
14.3 Foreign key table..... 1021
14.4 Same dimension table and main sub table..... 1051
14.5 Cross join and non equivalent join..... 1068
Cross join..... 1069
Non equivalent join..... 1073
14.6 Exercises..... 1077
Chapter 15 Graphics drawing..... 1100
15.1 Canvas..... 1101
Canvas..... 1102
Element..... 1103
15.2 Physical coordinates..... 1107
Absolute coordinates..... 1113
Relative coordinates..... 1116
15.3 Logical coordinates..... 1119
Numeric axis..... 1123

Enumeration axis..... 1132
Date axis..... 1142
15.4 More logical mappings..... 1145
Polar coordinates..... 1146
Combined coordinate system..... 1153
Legend..... 1157
15.5 Finished graph..... 1169
Column graph..... 1171
Dual-axis graph..... 1173
15.6 Exercises..... 1176
Trigonometric function..... 1177
Multiple combinations..... 1180
K-line chart..... 1183
Chapter 16 Big data and cursor..... 1186
16.1 Big data and cursor..... 1187
Big data..... 1188
Concept of cursor..... 1191
16.2 Basic cursor operations..... 1193
Use a cursor to read data..... 1194
Loop the cursor..... 1200
16.3 Cursor functions..... 1209
Aggregate operations of a cursor..... 1210
Use cursor functions..... 1217
Deferred calculation vs. immediate calculation..... 1232
16.4 Ordered cursor..... 1234
16.5 Big cursor..... 1245
Big sorting..... 1246
Big grouping..... 1249
Combine..... 1252
Merge..... 1256

Join..... 1260

Multi cursors..... 1265

16.6 Exercises..... 1271

Chapter 1
Basic concepts of programming

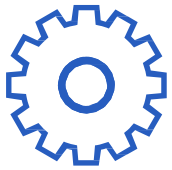
1.1 Computer program and language
1.2 esProc installation
1.3 The simplest program
1.4 More simple program examples
1.5 Program structure
1.6 The task of programming
1.7 Exercises

Chapter 1
Basic concepts of programming

1.1 Computer program and language

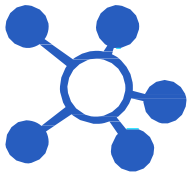
1.1 Computer program and language

Computer program



Instruction

A command that instructs a computer to perform some basic operation



Program

A set of instructions
A specific sequence is executed to perform a specific function



Software

A collection of programs organized in a specific order and possibly required data and files
The final result of programming

1.1 Computer program and language

Computer language

	Advantages	Disadvantages
Machine language	Directly identified by computer The highest execution efficiency	Very poor readability, difficult to learn, check, and modify Relying on computer hardware, difficult to migrate
Assembly language	Better readability than machine language High execution efficiency	Poor readability, long code, not easy to modify It also depends on computer hardware and is difficult to migrate
Advanced language	Easy to learn, easy to understand and easy to modify Powerful and migratable	The execution efficiency is relatively low There are different usages with different concept and grammar

1.1 Computer program and language

Computer languages



Advanced language

1000+100

Advanced language is easy to read and write, and people can easily understand its meaning.

Assembly language

MOV AX, 1000
ADD AX, 100

Assembly language transforms advanced language into machine instructions. Assembly language is different for different instruction sets.

Machine language

10111000
11101000
00000011
00000101
01100100
00000000

Machine language is a binary language used by computers. Machine language is not universal in different platforms.

Compilation and Interpretation

Compilation

- **Compilation** is the process of generating a advanced language at one time into a machine language that the computer can recognize.
- The program that performs the compilation process is called a compiler, and a program needs to be compiled before it can be executed.
- If a program needs to be used on different platforms, or the program changes, it needs to be recompiled and then executed.
- The languages which are execute by compilation are called **static languages**, such as C/C++, Pascal, FORTRAN, etc.

Interpretation

- **Interpretation** is the process of translating a advanced language item by item into target code to execute.
- The program that performs the interpretation process is called an interpreter. The program is executed while it is interpreted.
- The program with interpreted execution does not need to be recompiled when it is migrated or modified.
- The languages with interpreted execution are called **scripting languages**, such as SPL, Python, JavaScript, etc.

1.1 Computer program and language

The development of advanced language



Early stage computer language
Lots of jumps, lack of standard
Low development efficiency, difficult to maintain

The program has a good basic structure, and it is not allowed to jump randomly
The program structure is clear, easy to write and easy to maintain

Object as the basic structure
Provides encapsulation, inheritance and reloading features
It has stronger reusability

It doesn't matter if you don't understand the above contents. It's just a simple understanding of terms, which will not affect the later study.

Chapter 1
Basic concepts of programming

1.2 esProc installation

1.2 esProc installation

Why is esProc needed?

- To solve complex practical problems, we need a simple advanced language
- Frequent changes in operational logic or data require an interpreted execution language to avoid repeated recompilation
- Join calculation and statistical analysis are frequently needed in business, so a tool for structured data calculation is needed

1.2 esProc installation

What is SPL?

- SPL is Structured Process Language used by esProc.
- SPL is a scripting language, which can be interpreted and executed without compiling in advance.
- SPL is an advanced language whose syntax is closer to natural thinking and easier to master

SPL(Structured Process Language)



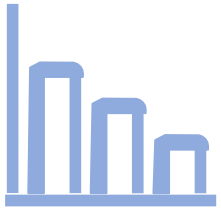
An interpreted execution language

Executed while interpreted, highly efficient
The code is simple, easy to read and write
Easy to use and maintain



Structured query

Clear data structure
Support multi type data
Interactive query with high flexibility



Efficient data analysis

Powerful computing function
Intelligent data processing
Support parallel computing to further improve performance



Code in cells

Programming is simple and fast
Convenient cell variables
Step by step execution, the results are visible

Download and installation of esProc

http://www.raqsoft.com/esproc-download



Chapter 1
Basic concepts of programming

1.3 The simplest program

1.3 The simplest program

【Example 1.1】Output in console:Hello, world!

1.3 The simplest program

【Example 1.1】Output in console:Hello, world!

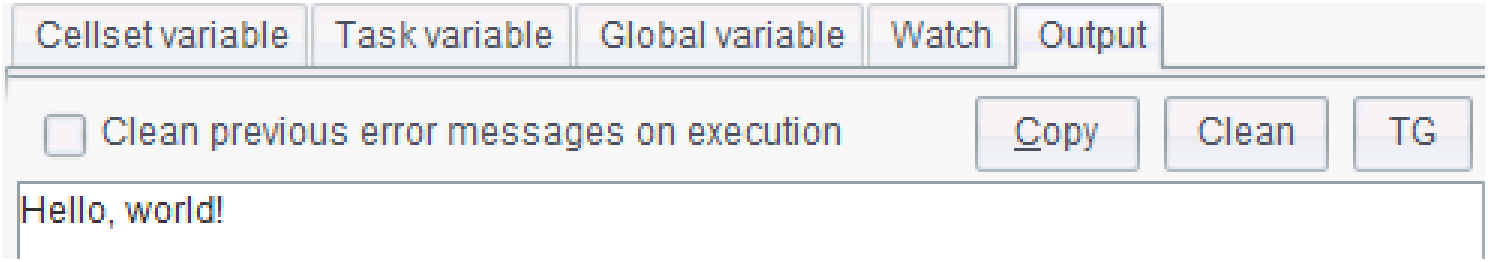
Solution: Output the above string with **output( )** function in the cell.

Press  button to create a new grid file

	A	B
1	>output("Hello, world!")	/Output a line of text

Execution method: no need to compile, press  in the upper toolbar to execute.

Output result: You can see the code execution result in the output window on the right.



1.3 The simplest program

	A	B
1	>output("Hello, world!")	/Output a line of text



Program analysis

- SPL code is written in a grid and stored as a dfx file.
- Each statement is written in a cell.
- Cells beginning with > are execution cells that can be used to perform calculations or operations.
- Cells beginning with / are comment cells that can be used to make necessary description of the program to facilitate maintenance of the code.

1.3 The simplest program

	A	B
1	>output("Hello, world!")	/Output a line of text



Program analysis

- output() is the output function in SPL, which is used to output a line of text in the console. The double quoted string Hello, world! in the output function is output as is.

- The comment cell does not participate in the calculation and has no effect on the running of the program. The text in the comment cell turns green for viewing only.

1.3 The simplest program

	A	B
1	>output("Hello, world!")	/Output a line of text



Program analysis

- SPL program can be executed directly without compiling. The grid code stored in a file with dfx extension can also be invoked in other programs (see Chapter six "Function and subroutine").
- SPL statements are written in cells and are usually executed from top to bottom and left to right.

Chapter 1
Basic concepts of programming

1.4 More simple program examples

1.4 More simple program examples

【Example 1.2】Calculate BMI according to the given height (m) and weight (kg).

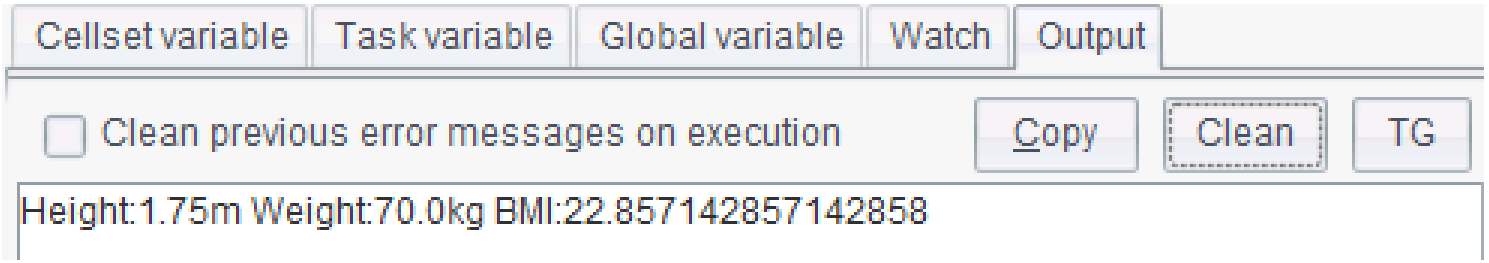
1.4 More simple program examples

【Example 1.2】Calculate BMI according to the given height (m) and weight (kg).

Solution: Calculate according to the height and weight data given in the cell, and output the result.

	A	B
1	1.75	/Height, m
2	70.0	/Weight, kg
3	=A2/A1/A1	/Calculate BMI
4	>output("Height:"/A1/"m Weight:"/A2/"kg BMI:"/A3)	/Output result

Output result:



1.4 More simple program examples

【Example 1.2】Calculate BMI according to the given height (m) and weight (kg).

	A	B
1	1.75	/Height, m
2	70.0	/Weight, kg
3	=A2/A1/A1	/Calculate BMI
4	>output("Height:"/A1/"m Weight:"/A2/"kg BMI:"/A3)	/Output result



Program analysis

- In SPL, data can be directly recorded in cells, such as A1 and A2. Such cells are called **constant cells**, and the data in the grid is displayed in magenta.
- The data in constant cell can be referenced directly by cell name during calculation.
- Cells beginning with = are **calculation cells** that can be used to perform calculations and return results.

1.4 More simple program examples

【Example 1.2】Calculate BMI according to the given height (m) and weight (kg).

	A	B
1	1.75	/Height, m
2	70.0	/Weight, kg
3	=A2/A1/A1	/Calculate BMI
4	>output("Height:"/A1/"m Weight:"/A2/"kg BMI:"/A3)	/Output result

- **Constant** is a fixed value. The value of the constant cell used in the above example is a kind of constant. Its name is the name of the cell position, such as A1 and A2. The initial value of the constant cell is the value of the data in it, such as 1.75 and 70.0. The value of the constant cell is usually not modified.
- **Calculation** refers to calculating the result according to the expression used. The constant (or variable) in the expression needs to be converted into the corresponding value according to the name. For example, the expression in A3 is = A2/A1/A1, and calculating A3 is to get the result of 70 /1.75/1.75.
- **Variable** represents a value that may change. All cells in SPL can be used as variables. Variable values are cell values, and their names are the names of cell positions.

The concepts of constants and variables will be explained in detail in Chapter 3, and only a preliminary understanding is needed here.

1.4 More simple program examples

【Example 1.2】Calculate BMI according to the given height (m) and weight (kg).

	A	B
1	1.75	/Height, m
2	70.0	/Weight, kg
3	=A2/A1/A1	/Calculate BMI
4	>output("Height:"/A1/"m Weight:"/A2/"kg BMI:"/A3)	/Output result

Program analysis

- The string output by the output() function is composed of multiple parts. The string concatenation can use the operator /, and the same operator represents division in numerical calculation.

1.4 More simple program examples

【Example 1.2】Calculate BMI according to the given height (m) and weight (kg).

	A	B
1	1.75	/Height, m
2	70.0	/Weight, kg
3	=A2/A1/A1	/Calculate BMI
4	>output("Height:"/A1/"m Weight:"/A2/"kg BMI:"/A3)	/Output result

Program analysis

- The value of a constant cell is the constant in the cell. You can also select a cell to view it.

Value
1.75

Value
70.0

- Execution, comment, or blank cells have no cell values.

Value
(null)

1.4 More simple program examples

【Example 1.2】Calculate BMI according to the given height (m) and weight (kg).

	A	B
1	1.75	/Height, m
2	70.0	/Weight, kg
3	=A2/A1/A1	/Calculate BMI
4	>output("Height:"/A1/"m Weight:"/A2/"kg BMI:"/A3)	/Output result

	A	B
1	1.75	70.0
2	=B1/A1/A1	>output("Height:"/A1/"m Weight:"/B1/"kg BMI:"/A2)

Program analysis

- SPL program can be written in multi column cells, and the execution order is from top to bottom, from left to right. The effect of the above two programs is the same.

1.4 More simple program examples

【Example 1.3】Enter 3 numbers to find the maximum value.

1.4 More simple program examples

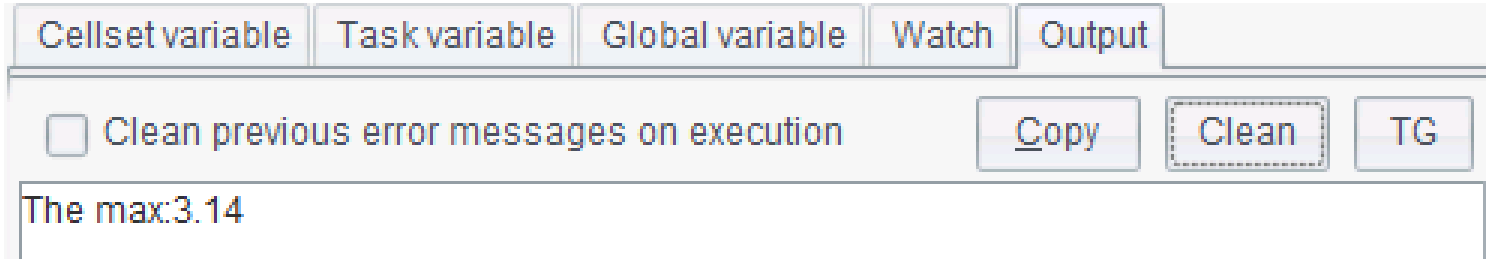
【Example 1.3】Enter 3 numbers to find the maximum value.

Solution: Input three numbers into the constant cells, and get the maximum value by comparison and output it.

Compare three cell constants in the program to get the maximum value and output it:

	A	B	C	D
1	3	3.14	3.139889	/Input three numbers in constant cells
2	=if(A1>B1, A1, B1)			/Get the larger values of A1 and B1
3	=if(C1>A2, C1, A2)			/Get the larger value of C1 and A2
4	>output("The max:"/A3)			/A3 is the maximum value, output the result

Output result:



1.4 More simple program examples

【Example 1.3】Enter 3 numbers to find the maximum value.

	A	B	C	D
1	3	3.14	3.139889	/Input three numbers in constant cells
2	=if(A1>B1, A1, B1)			/Get the larger values of A1 and B1
3	=if(C1>A2, C1, A2)			/Get the larger value of C1 and A2
4	>output("The max:"/A3)			/A3 is the maximum value, output the result



Program analysis

- The if() function has three parameters, which are separated by commas. The first parameter is a condition. If it is true, the second parameter is returned, otherwise the third parameter is returned. In A2, if A1 > B1, A1 is returned, otherwise B1 is returned. The parameters of the function are explained in the next section.

- In SPL, you can access the cell value calculated before and complete the calculation step by step.

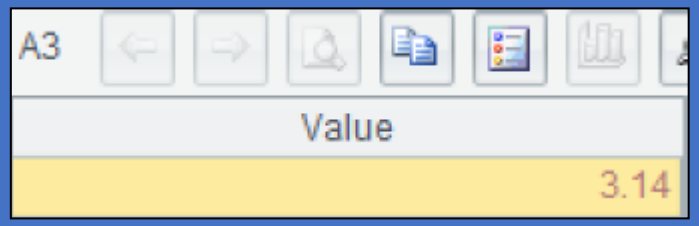
1.4 More simple program examples

【Example 1.3】Enter 3 numbers to find the maximum value.

	A	B	C	D
1	3	3.14	3.139889	/Input three numbers in constant cells
2	=if(A1>B1, A1, B1)			/Get the larger values of A1 and B1
3	=if(C1>A2, C1, A2)			/Get the larger value of C1 and A2

Program analysis

- When using SPL to perform the calculation, you can also view the cell value after calculation instead of output function. As in the above program, after calculation, the cell value of A3 is the maximum value of 3 numbers:



Therefore, SPL program does not necessarily need to output the result with the output function, but also can view the cell value after the calculation is completed.

Chapter 1
Basic concepts of programming

1.5 Program structure

1.5 Program structure

SPL program is written in a grid, composed of statements in each cell.

In the grid, data can be input with constant cell, explanatory text can be recorded with comment cell, calculation result can be obtained by calculation cell, and operation or calculation can be completed by execution cell.

Other advanced languages are similar. They are composed of executive statements and comment statements and are executed in a certain order.

Grid programs are generally executed from top to bottom and from left to right.

When calculating, the cell name can be used to refer to its cell value as a constant or variable.

SPL can input program parameters, and the calculation results can be stored in cell values or output by functions.

The program should include the necessary comments.

Constant cell

Calculation cell

Execution cell

	A	B
1	1.75	/Height, m
2	70.0	/Weight, kg
3	=A2/A1/A1	/Calculate BMI
4	>output("Height:"/A1/"m Weight:"/A2/"kg BMI:"/A3)	/Output result

Comment cell

1.5 Program structure

Use the functions

	A	B	C	D
1	3	3.14	3.139889	/Input three numbers in constant cells
2	=if(A1>B1, A1, B1)			/Get the larger values of A1 and B1
3	=if(C1>A2, C1, A2)			/Get the larger value of C1 and A2



= if (C1>A2, C1, A2)

Function name

Parameters

Parameters can be numerical values, strings, cell names, program parameter names, and so on, or they can be expressions.

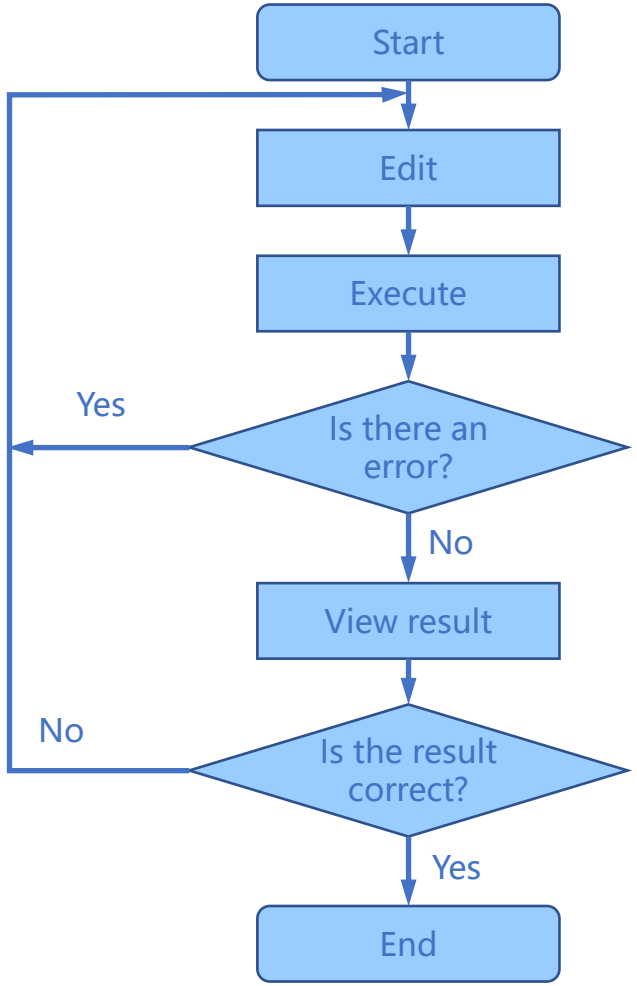
SPL syntax

The if function can also use multiple sets of condition parameters, such as: =if(A1>B1 && A1>C1, A1, B1>A1 && B1>C1, B1, C1)
During execution, when the value in A1 is greater than B1 and greater than C1, the return result is A1; otherwise, when the value in B1 is greater than A1 and greater than C1, the return result is B1; if all the previous conditions are not met, the result C1 is returned.

&& is the logical operator AND, which means that all the conditions on the left and right sides are true. If function and logical operators are described in detail in Chapter 4.

1.5 Program structure

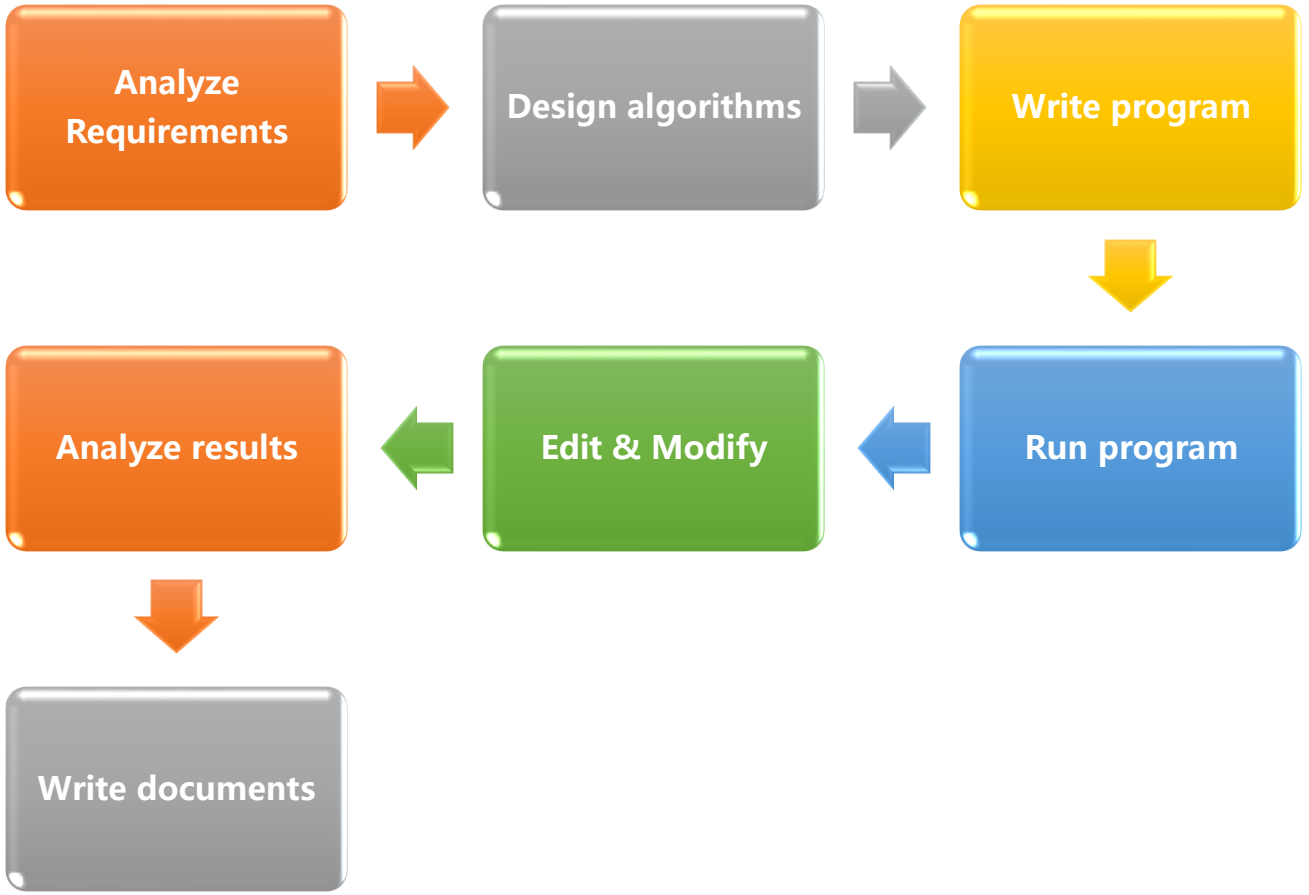
Programming steps



Chapter 1
Basic concepts of programming

1.6 The task of programming

1.6 The task of programming

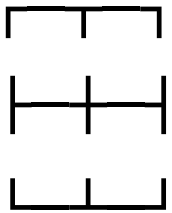


Chapter 1
Basic concepts of programming

1.7 Exercises

1.7 Exercises

【Exercise 1.1】Write a program to output the following grid:



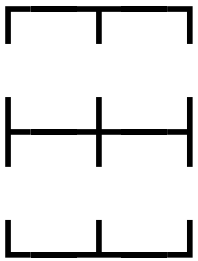
【 Exercise 1.2】Write two English names to output "Hello, XXX. I'm XXX.".

【 Exercise 1.3】Calculate the circumference and area of the circle by writing π = 3.1416 and radius r directly in the cells.

【 Exercise 1.4】Input two numbers and output their relationship, whether a > b, a < b or a == b.

1.7 Exercises

【Exercise 1.1】Write a program to output the following grid:



	A	B
1	>output(" ┌┬┐ ")	/Output table characters line by line, output the first line
2	>output(" ├┼┤ ")	/output the second line
3	>output(" └┴┘ ")	/output the third line

1.7 Exercises

【Exercise 1.2】Write two English names to output "Hello, XXX. I'm XXX.".

	A	B
1	Tom	/Record two English names in two constant cells
2	Jerry	
3	>output("Hello, "/A1/". I'm "/A2/".")	/In the parameters of the output function, use the '/' operator to concatenate strings

1.7 Exercises

【Exercise 1.3】Calculate the circumference and area of the circle by writing π = 3.1416 and radius r directly in the cells.

	A	B	C
1	3.1416	4	/Input π and r in constant cells
2	=2*A1*B1		/Calculate the circumference
3	=A1*B1*B1		/Calculate the area
4	>output("r="/B1/", circumference="/A2/", area="/A3)		/Output results

1.7 Exercises

【Exercise 1.4】Input two numbers and output their relationship, whether a > b, a < b or a == b.

	A	B	C
1	4	4.789	/Input two numbers in constant cells
2	=if(A1>B1, ">", A1<B1, "<", "==")		/Several conditions can be used continuously in if function
3	>output(A1/A2/B1)		/Output result

Chapter 2
Algorithm and program

2.1 What is algorithm?

2.2 Simple algorithm examples

2.3 Characteristics of the algorithm

2.4 Methods of representing algorithm

- Represent algorithm by flow chart
- Represent algorithm by N-S flow chart
- Represent algorithm by pseudo code
- Represent algorithm by SPL

2.5 Structured programming method

2.6 Exercises

Chapter 2
Algorithm and program

2.1 What is algorithm?

2.1 What is algorithm?



Niklaus Wirth

Swiss computer scientist
Turing Award winner
Father of Pascal
Multiple programming languages designer

Algorithm + data structure = Program

Data structure

The way a computer stores and uses data.
Description of various data used in the program, as well as their types and organization format.

Algorithm

A description of the operation of a computer to solve a problem.
The flow and steps of a program to solve a specific problem. It is a set of finite computer instructions.

2.1 What is algorithm?



Algorithm

The process and steps taken to solve specific problems are algorithms.

The algorithm is a complete and accurate description of the operation scheme. In the program, the algorithm is a series of clear and definite instructions.

In order to solve the same problem, different algorithms can be used.

To solve the problem more effectively, we not only need the algorithm to get the correct result, but also need to consider the algorithm's cost, efficiency and other factors to choose a more appropriate algorithm.

2.1 What is algorithm?

Numerical algorithm	Non numerical algorithm
The purpose is to find a numerical solution	Different purposes make it difficult to standardize
The existing numerical analysis methods often can be used	Algorithms often need to be designed by oneself
The algorithms are relatively mature	There are different algorithms for the same problem

Chapter 2
Algorithm and program

2.2 Simple algorithm examples

2.2 Simple algorithm examples

【Example 2.1】Calculate the result of 5×5×5×5×5.

2.2 Simple algorithm examples

【Example 2.1】Calculate the result of 5×5×5×5×5.

Algorithm steps

Step1: multiply 5 by 5 to get the result 25

Step2: Multiply the result of step1 by 5 to get the result 125

Step3: Multiply the result of step2 by 5 to get the result 625

Step4: Multiply 625 by 5 to get the final result 3125

2.2 Simple algorithm examples

【Example 2.1】Calculate the result of 5×5×5×5×5.

Algorithm steps

Step1: multiply 5 by 5 to get the result 25

Step2: Multiply the result of step1 by 5 to get the result 125

Step3: Multiply the result of step2 by 5 to get the result 625

Step4: Multiply 625 by 5 to get the final result 3125

Algorithm steps

Step1: Let p=5, or write p←5 (p is assigned the value of 5)

Step2: Let i=1, or write i←1 (i is assigned the value of 1)

Step3: Multiply p by 5, and the result is still stored in the variable p, or write p←p*5

Step4: Add 1 to the value of i, or write i←i+1

Step5: If i < 5, return to perform step 3 and the following steps; otherwise, it ends.

2.2 Simple algorithm examples

【Example 2.1】Calculate the result of 5×5×5×5×5.

Algorithm steps

Step1: multiply 5 by 5 to get the result 25

Step2: Multiply the result of step1 by 5 to get the result 125

Step3: Multiply the result of step2 by 5 to get the result 625

Step4: Multiply 625 by 5 to get the final result 3125

Modify the example to calculate the result of 9×9×9×9×9.

Algorithm steps

Step1: Let p=9, or write p←9 (p is assigned a value of 9)

Step2: Let i=1, or write i←1 (i is assigned a value of 1)

Step3: Multiply p by 9, and the result is still stored in the variable p, or write p←p*9

Step4: Add 1 to the value of i, or write i←i+1

Step5: If i < 6, return to perform step 3 and the following steps; otherwise, it ends.

- **This algorithm is more flexible, can be used to solve similar problems with generality.**

2.2 Simple algorithm examples

【Example 2.2】There are 44 countries in Europe. List the countries and their capitals with the capital name starting with letter "S".

2.2 Simple algorithm examples

【Example 2.2】There are 44 countries in Europe. List the countries and their capitals with the capital name starting with letter "S".

N: the name of the country

Subscript i:the i^{th} country

N_1: the name of the first country

N_i: the name of the i^{th} country

C: the name of the capital

C_1: the capital name of the first country

C_i: the capital name of the i^{th} country

Algorithm steps

Step1: i←1

Step2: If the initial letter of C_i is S, output N_i and C_i; otherwise, no output.

Step3: i←i+1

Step4: If i≤ 44, return to Step2 to continue execution; otherwise, end.

2.2 Simple algorithm examples

【Example 2.3】Calculate according to Taylor formula :$\pi \approx 4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\frac{4}{11}+\cdots+\frac{4}{197}-\frac{4}{199}$

2.2 Simple algorithm examples

【Example 2.3】Calculate according to Taylor formula :$\pi \approx 4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\frac{4}{11}+\cdots+\frac{4}{197}-\frac{4}{199}$

term:	One of the items in accumulation
sign:	the sign of the current term, 1 or – 1
deno:	The denominator of the current term
res:	current cumulative result

Algorithm steps

Step1: res←4

Step2: sign←-1

Step3: deno←3

Step4: term←sign*(4/deno)

Step5: res←res+term

Step6: sign←sign*(-1)

Step7: deno←deno+2

Step8: If deno≤199, return to Step4 to continue execution; otherwise, end and output res.

2.2 Simple algorithm examples

【Example 2.4】List 30 consecutive leap years after 2019.

2.2 Simple algorithm examples

【Example 2.4】List 30 consecutive leap years after 2019.



2.2 Simple algorithm examples

【Example 2.4】List 30 consecutive leap years after 2019.



Algorithm steps

Step1: year←2019

Step2: num←0

Step3: if year cannot be exactly divided by 4, then this year is not a leap year. Go to Step7 to check the next year.

Step4: If year cannot be divisible by 100, the year is a leap year. Go to Step6 to continue; otherwise, continue to judge.

Step5: If year can be divisible by 400, the year is a leap year, continue to next step; otherwise, it means that it is not a leap year and jump to Step7.

Step6: Find a leap year, output year, num←num+1

Step7: year←year+1

Step8: If num<30, jump to Step3 to continue, judge the next year; Otherwise, the search is completed and the algorithm ends.

2.2 Simple algorithm examples

【Example 2.5】Given a positive integer, output its prime factor decomposition equation, such as 105 = 3 * 5 * 7.

2.2 Simple algorithm examples

【Example 2.5】Given a positive integer, output its prime factor decomposition equation, such as 105 = 3 * 5 * 7.

- To decompose a positive integer into a prime factor, we only need to find its divisor from 2 in turn
- If a positive integer n cannot be divisible by any prime not greater than $\sqrt{n}$, then it must be a prime
- If the positive integer n ≤ 3, it can not be further decomposed

2.2 Simple algorithm examples

【Example 2.5】Given a positive integer, output its prime factor decomposition equation, such as 105 = 3 * 5 * 7.

- To decompose a positive integer into a prime factor, we only need to find its divisor from 2 in turn
- If a positive integer n cannot be divisible by any prime not greater than $\sqrt{n}$, then it must be a prime
- If the positive integer n ≤ 3, it can not be further decomposed

Algorithm steps

Step1: Input the value of n

Step2: i←2, s←n+"="

Step3: If n≤3, the decomposition is completed and execute step8.

Step4: Calculate the remainder r of n divided by i

Step5: If r=0, it means that n can be divisible by i, then n←n/i, s←s+i+"*", return to step3 and continue.

Step6: i←i+1

Step7: If i≤$\sqrt{n}$, return to Step4 and continue.

Step8: s←s+n, output s and end the algorithm.

Chapter 2
Algorithm and program

2.3 Characteristics of the algorithm

2.3 Characteristics of the algorithm

Characteristics of the algorithm



Finiteness

The steps of an algorithm are limited and can be terminated after execution.



Certainty

Every step of the algorithm must have a definite definition and there must be no ambiguity.



Input

The algorithm can obtain 0 or more input information from the outside as the initial condition. When there is 0 input, the algorithm has defined the initial conditions.



Output

An algorithm needs one or more outputs to return the result.



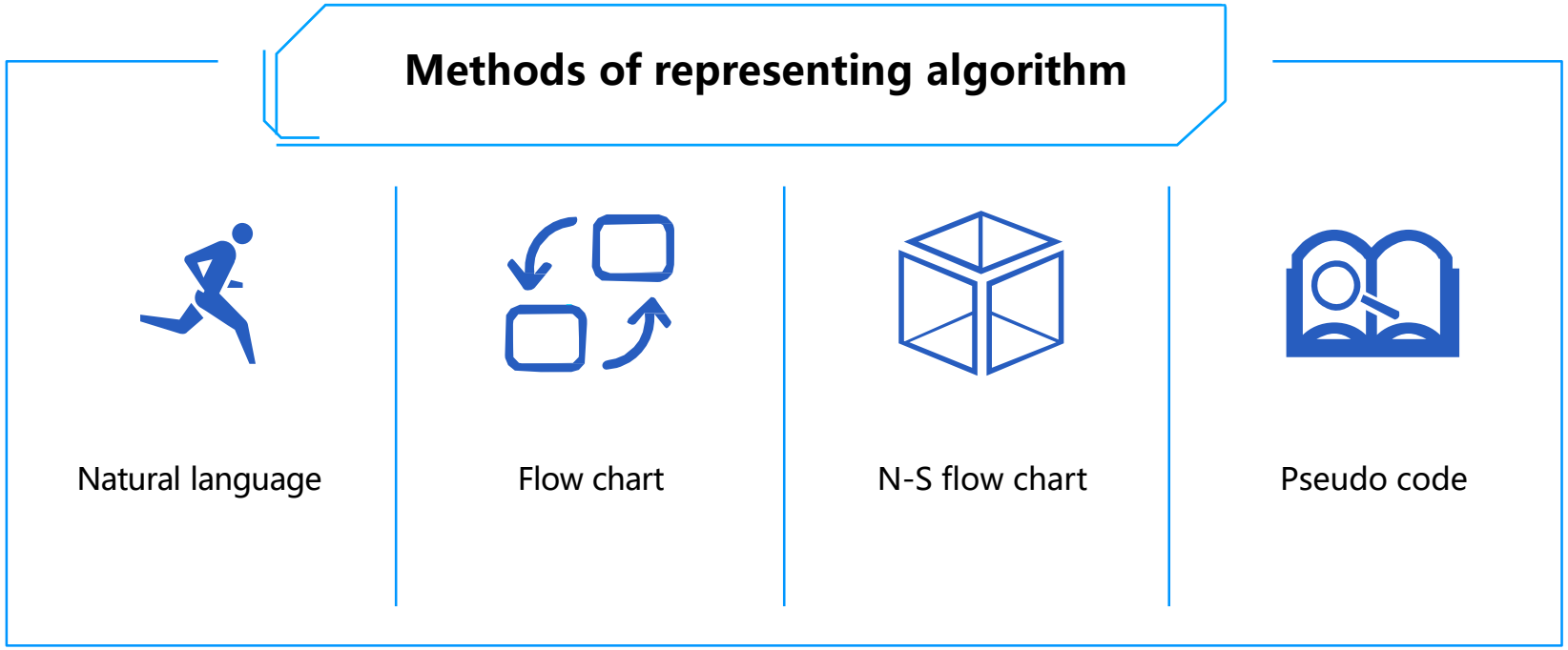
Feasibility

Each step in the algorithm can be effectively executed in a limited time and get the exact results. Also known as effectiveness.

Chapter 2
Algorithm and program

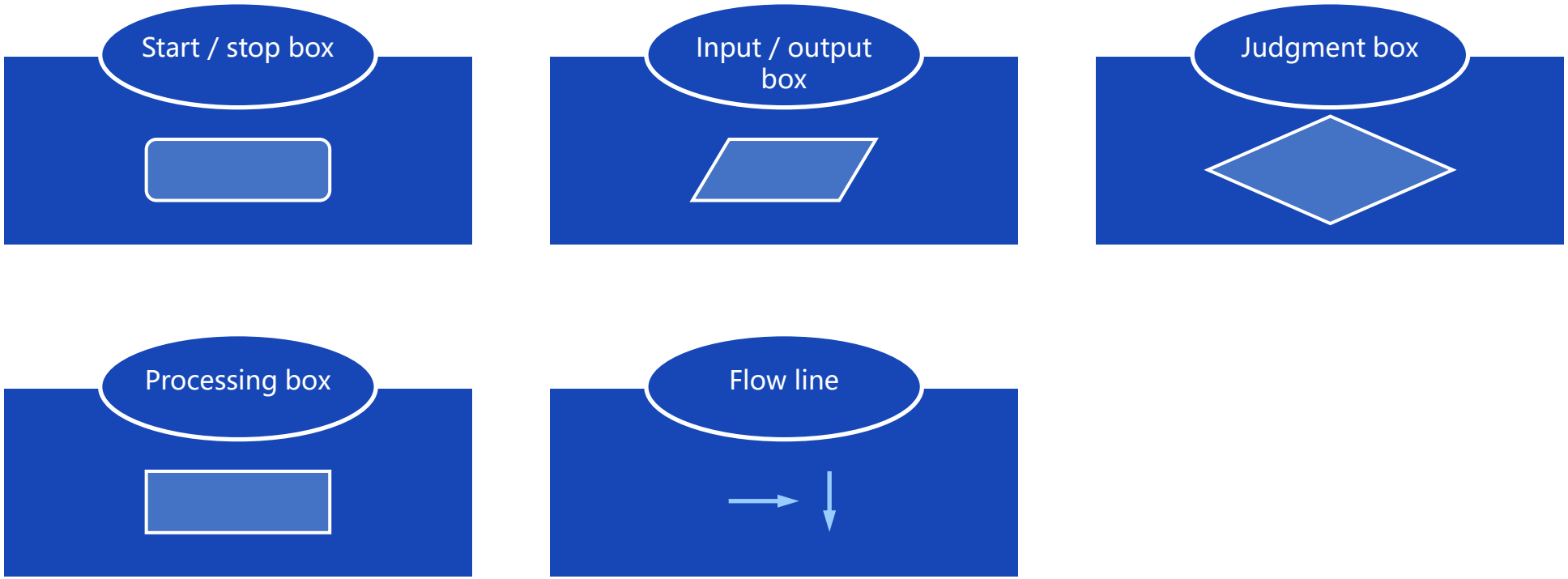
2.4 Methods of representing algorithm

2.4 Methods of representing algorithm



2.4 Methods of representing algorithm

Symbols of flow chart



2.4 Methods of representing algorithm

【Example 2.6】Represent the algorithm of example 2.1 by flow chart.
Calculate the result of 5×5×5×5×5.

Algorithm steps

Step1: p←5

Step2: i←1

Step3: p←p*5

Step4: i←i+1

Step5: If i≥5, output p and end; otherwise return to Step3 and continue.

2.4 Methods of representing algorithm

【Example 2.6】Represent the algorithm of example 2.1 by flow chart.
Calculate the result of 5×5×5×5×5.

Algorithm steps

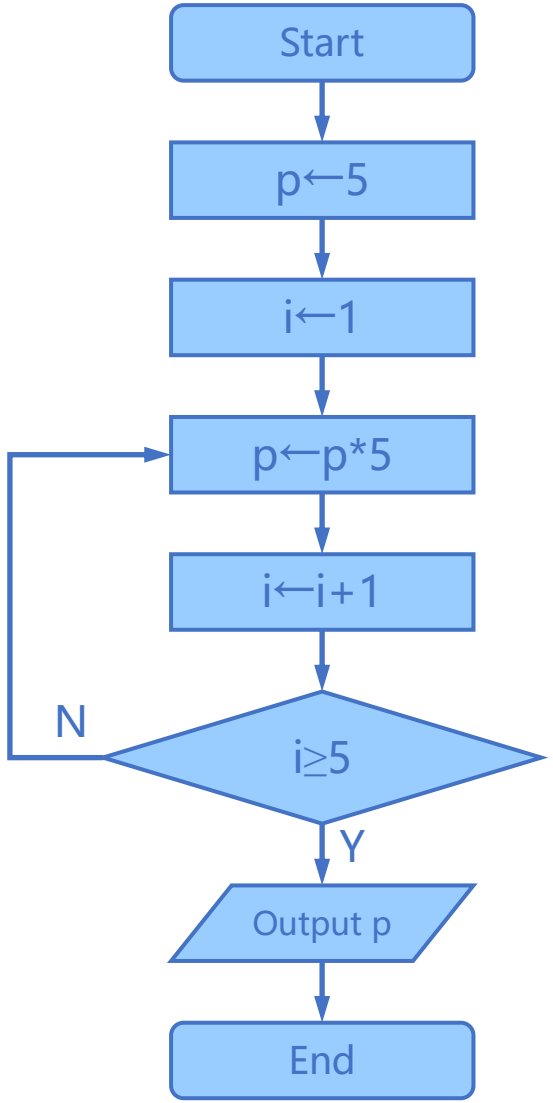
Step1: p←5

Step2: i←1

Step3: p←p*5

Step4: i←i+1

Step5: If i≥5, output p and end; otherwise return to Step3 and continue.



2.4 Methods of representing algorithm

Example 2.7】Represent the algorithm of example 2.2 by flow chart.

There are 44 countries in Europe. List the countries and their capitals with the capital name starting with letter "S".

Algorithm steps

Step1: i←1

Step2: If the initial letter of C_i is S, output N_i and C_i; otherwise, no output.

Step3: i←i+1

Step4: If i≤ 44, return to Step2 to continue execution; otherwise, end.

2.4 Methods of representing algorithm

Example 2.7】Represent the algorithm of example 2.2 by flow chart.

There are 44 countries in Europe. List the countries and their capitals with the capital name starting with letter "S".

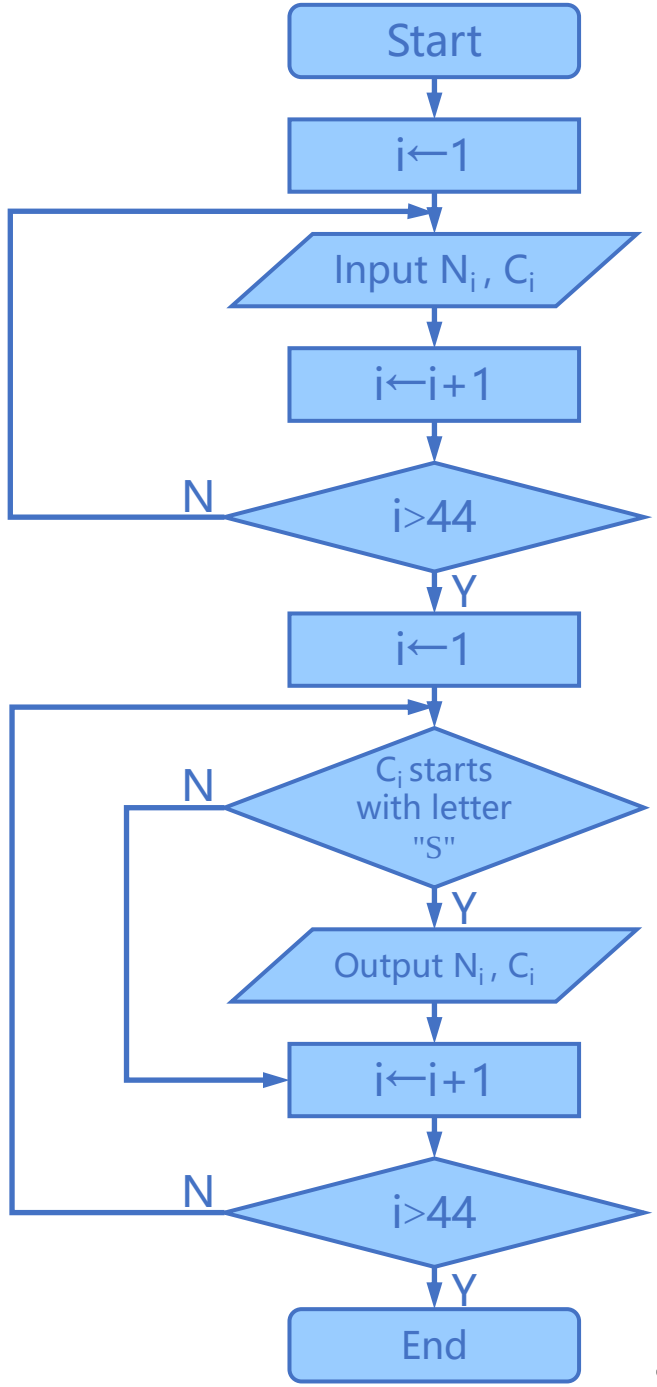
Algorithm steps

Step1: i←1

Step2: If the initial letter of C_i is S, output N_i and C_i; otherwise, no output.

Step3: i←i+1

Step4: If i≤ 44, return to Step2 to continue execution; otherwise, end.



2.4 Methods of representing algorithm

【Example 2.8】Represent the algorithm of example 2.3 by flow chart.

Calculate $\pi \approx 4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\frac{4}{11}+\cdots+\frac{4}{197}-\frac{4}{199}$

Algorithm steps

Step1: res←4

Step2: sign←-1

Step3: deno←3

Step4: term←sign*(4/deno)

Step5: res←res+term

Step6: sign←sign*(-1)

Step7: deno←deno+2

Step8: If deno≤199, return to Step4 to continue execution; otherwise, end and output res.

2.4 Methods of representing algorithm

【Example 2.8】Represent the algorithm of example 2.3 by flow chart.

Calculate $\pi \approx 4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\frac{4}{11}+\cdots+\frac{4}{197}-\frac{4}{199}$

Algorithm steps

Step1: res←4

Step2: sign←-1

Step3: deno←3

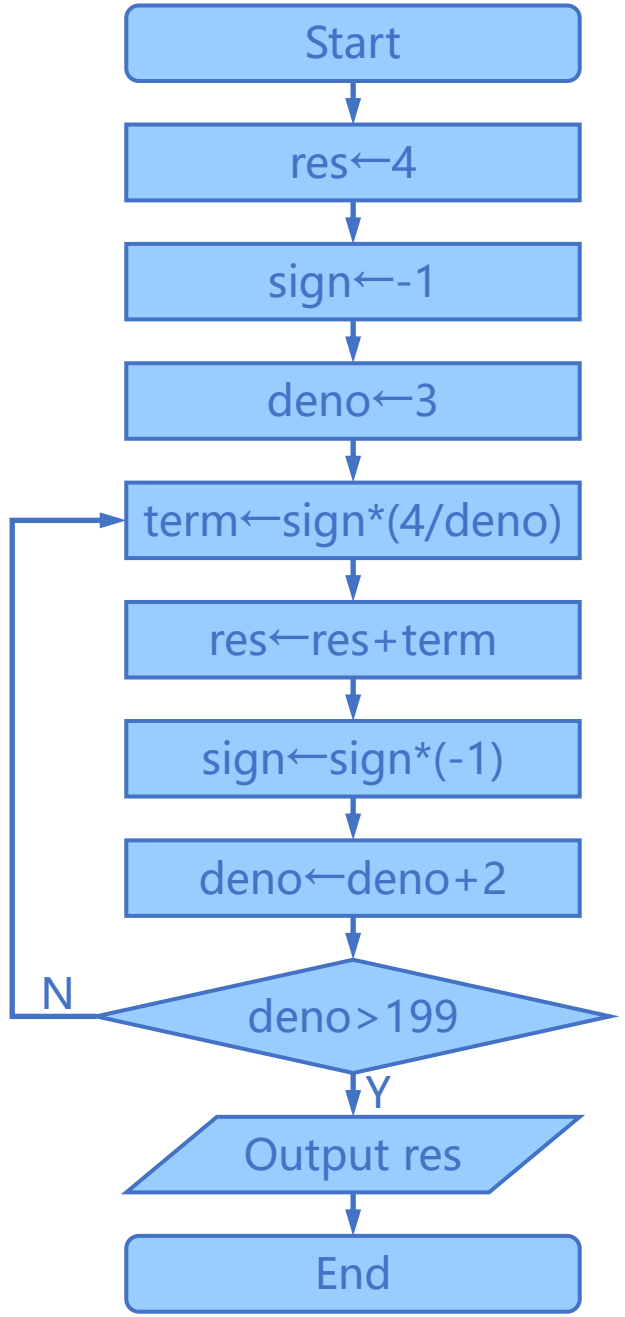
Step4: term←sign*(4/deno)

Step5: res←res+term

Step6: sign←sign*(-1)

Step7: deno←deno+2

Step8: If deno≤199, return to Step4 to continue execution; otherwise, end and output res.



2.4 Methods of representing algorithm

【Example 2.9】Represent the algorithm of example 2.4 by flow chart.

List 30 consecutive leap years after 2019.

Algorithm steps

Step1: year←2019

Step2: num←0

Step3: if year cannot be exactly divided by 4, then this year is not a leap year. Go to step7 to check the next year.

Step4: If year cannot be divisible by 100, the year is a leap year. Go to step6 to continue; otherwise, continue to judge.

Step5: If year can be divisible by 400, the year is a leap year, continue to next step; otherwise, it means that it is not a leap year and jump to step7.

Step6: Find a leap year, output year, num←num+1

Step7: year←year+1

Step8: If num<30, jump to Step3 to continue, judge the next year; Otherwise, the search is completed and the algorithm ends.

2.4 Methods of representing algorithm

【Example 2.9】Represent the algorithm of example 2.4 by flow chart.

List 30 consecutive leap years after 2019.

Algorithm steps

Step1: year←2019

Step2: num←0

Step3: if year cannot be exactly divided by 4, then this year is not a leap year. Go to step7 to check the next year.

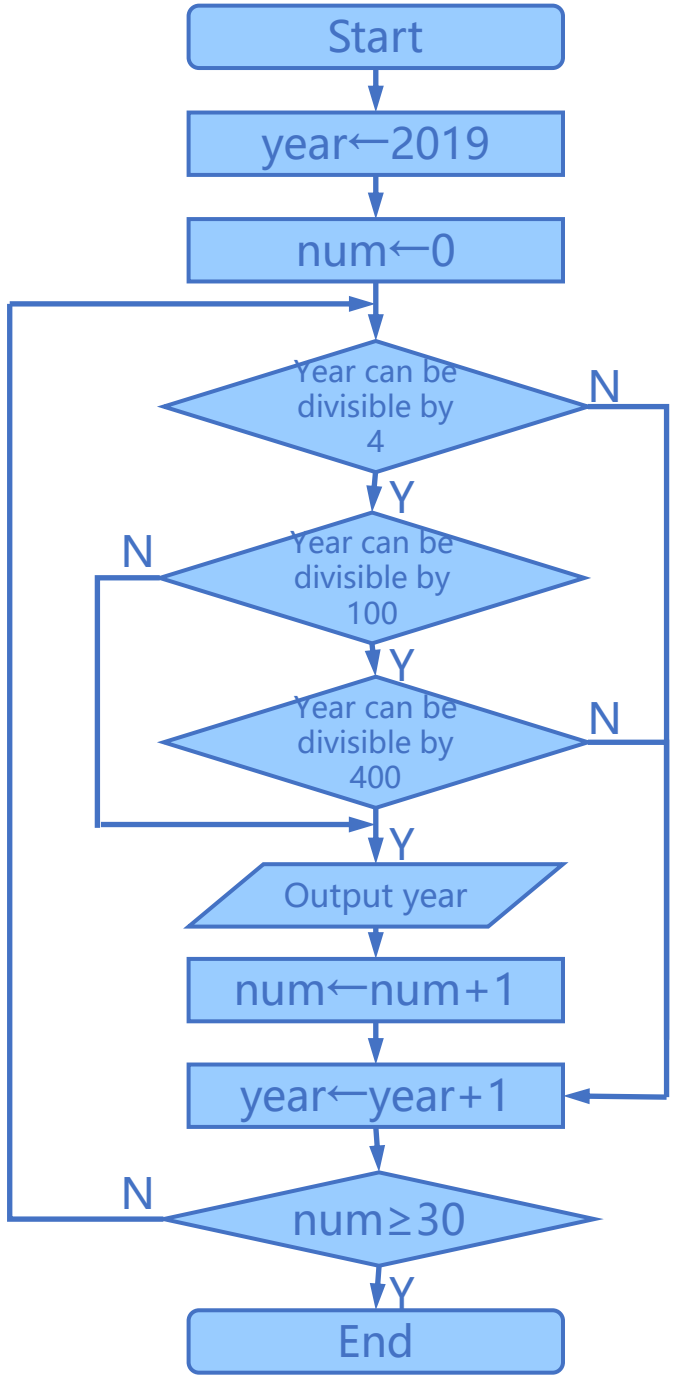
Step4: If year cannot be divisible by 100, the year is a leap year. Go to step6 to continue; otherwise, continue to judge.

Step5: If year can be divisible by 400, the year is a leap year, continue to next step; otherwise, it means that it is not a leap year and jump to step7.

Step6: Find a leap year, output year, num←num+1

Step7: year←year+1

Step8: If num<30, jump to Step3 to continue, judge the next year; Otherwise, the search is completed and the algorithm ends.



2.4 Methods of representing algorithm

【Example 2.10】Represent the algorithm of example 2.5 by flow chart.

Given a positive integer, output its prime factor decomposition equation.

Algorithm steps

Step1: Input the value of n

Step2: i←2 , s←n+"="

Step3: If n≤3, the decomposition is completed and execute step8.

Step4: Calculate the remainder r of n divided by i

Step5: If r=0, it means that n can be divisible by i, then n←n/i, s←s+i+"*", return to step3 and continue.

Step6: i←i+1

Step7: If i≤$\sqrt{n}$, return to Step4 and continue.

Step8: s←s+n, output s and end the algorithm.

2.4 Methods of representing algorithm

【Example 2.10】Represent the algorithm of example 2.5 by flow chart.

Given a positive integer, output its prime factor decomposition equation.

Algorithm steps

Step1: Input the value of n

Step2: i←2 , s←n+"="

Step3: If n≤3, the decomposition is completed and execute step8.

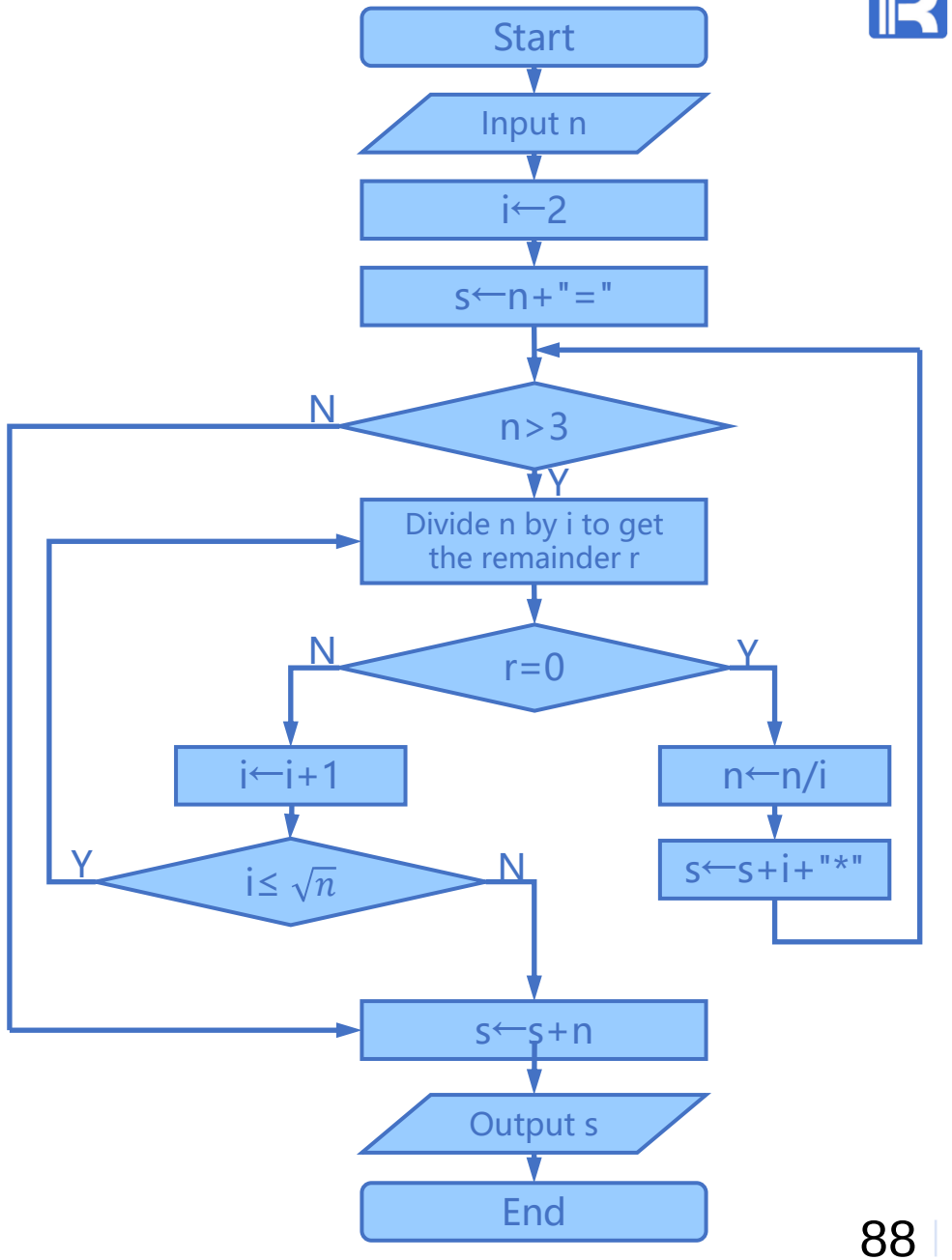
Step4: Calculate the remainder r of n divided by i

Step5: If r=0, it means that n can be divisible by i, then n←n/i, s←s+i+"*", return to step3 and continue.

Step6: i←i+1

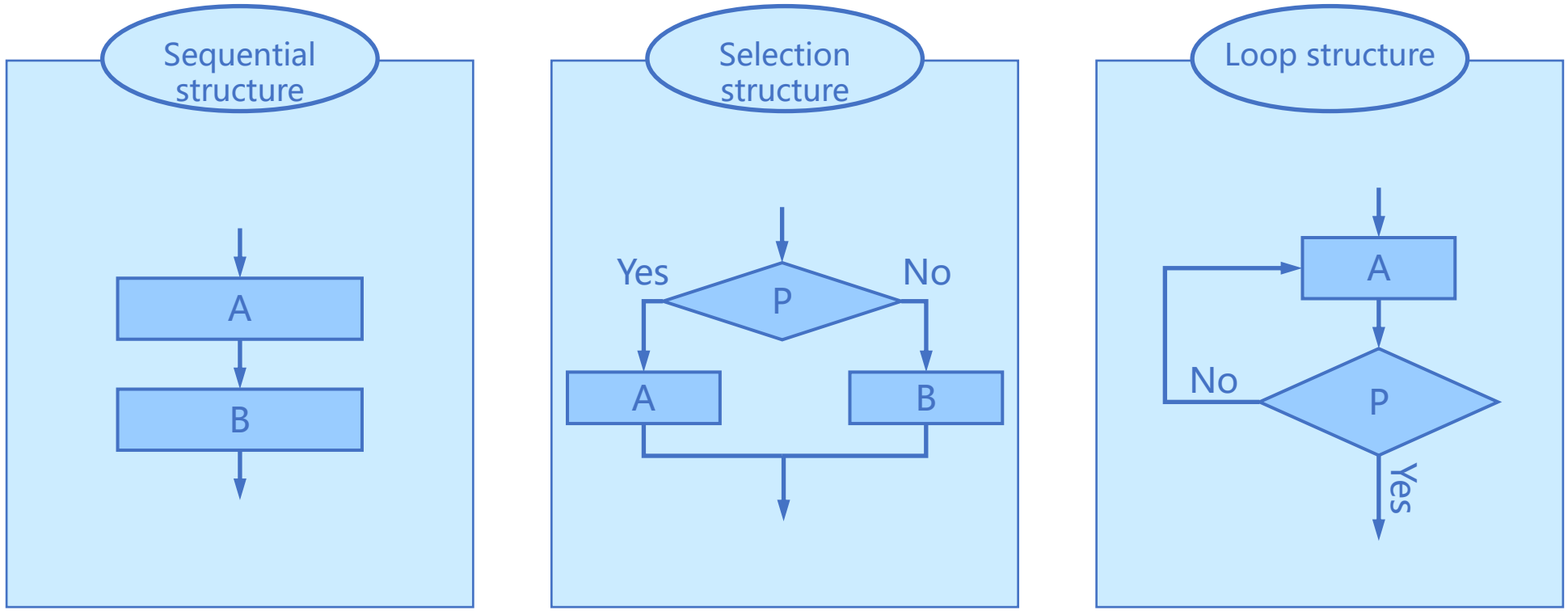
Step7: If i≤$\sqrt{n}$, return to Step4 and continue.

Step8: s←s+n, output s and end the algorithm.



2.4 Methods of representing algorithm

Three basic structures of flow chart

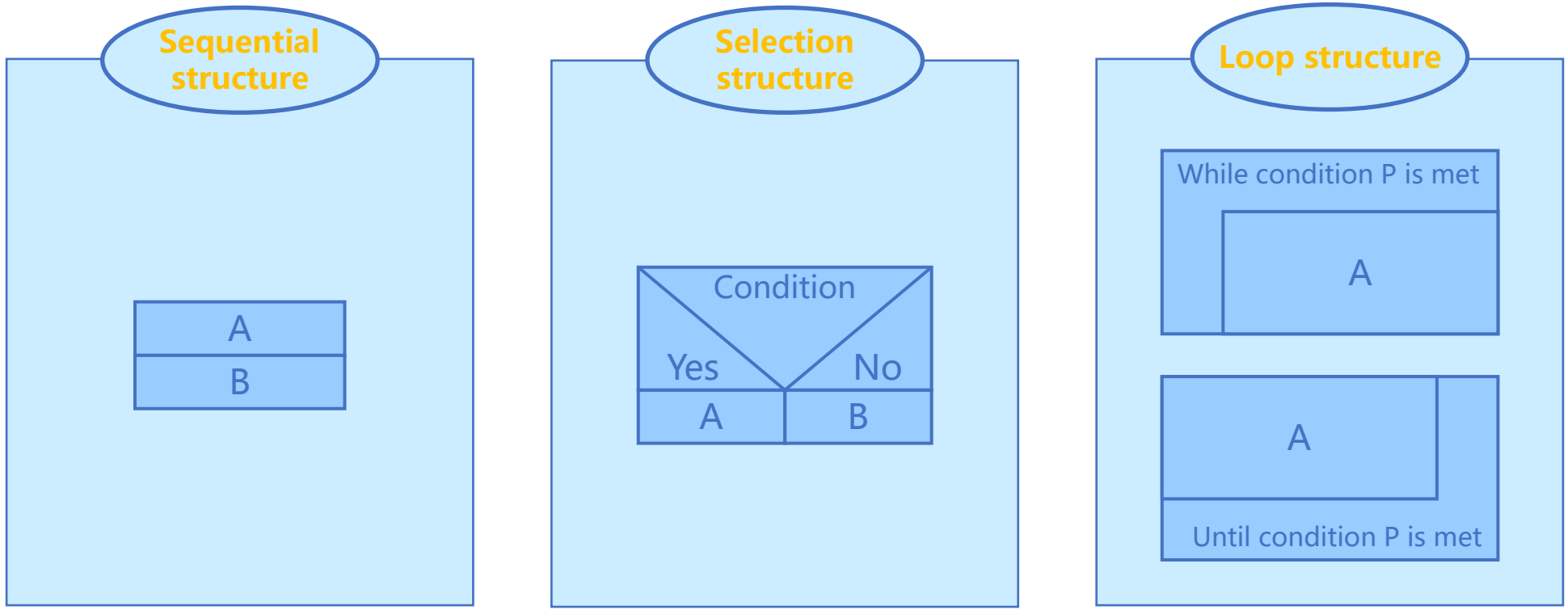


2.4 Methods of representing algorithm

The common characteristics of the three structures

- There's only one entrance.
- There is only one exit.
- Every part of the structure has an opportunity to be executed.
- There is no "dead loop" in the structure (a loop that does not end)

How to represent algorithm with **N-S flow chart**



2.4 Methods of representing algorithm

【Example 2.11】Represent the algorithm of example 2.1 by N-S flow chart.
Calculate the result of 5×5×5×5×5.

Algorithm steps

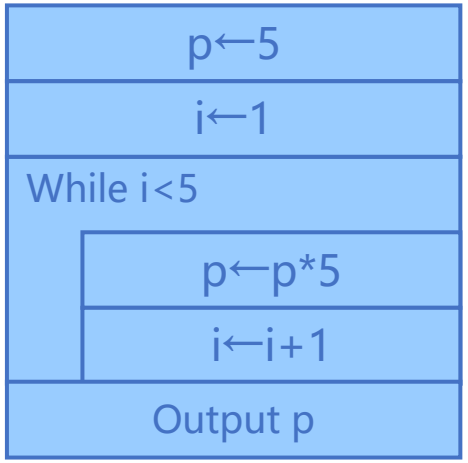
Step1: p←5

Step2: i←1

Step3: p←p*5

Step4: i←i+1

Step5: If i≥5, output p and end; otherwise return to Step3 and continue.



2.4 Methods of representing algorithm

【Example 2.12】Represent the algorithm of example 2.4 by N-S flow chart.
List 30 consecutive leap years after 2019.

Algorithm steps

Step1: year←2019

Step2: num←0

Step3: if year cannot be exactly divided by 4, then this year is not a leap year. Go to step7 to check the next year.

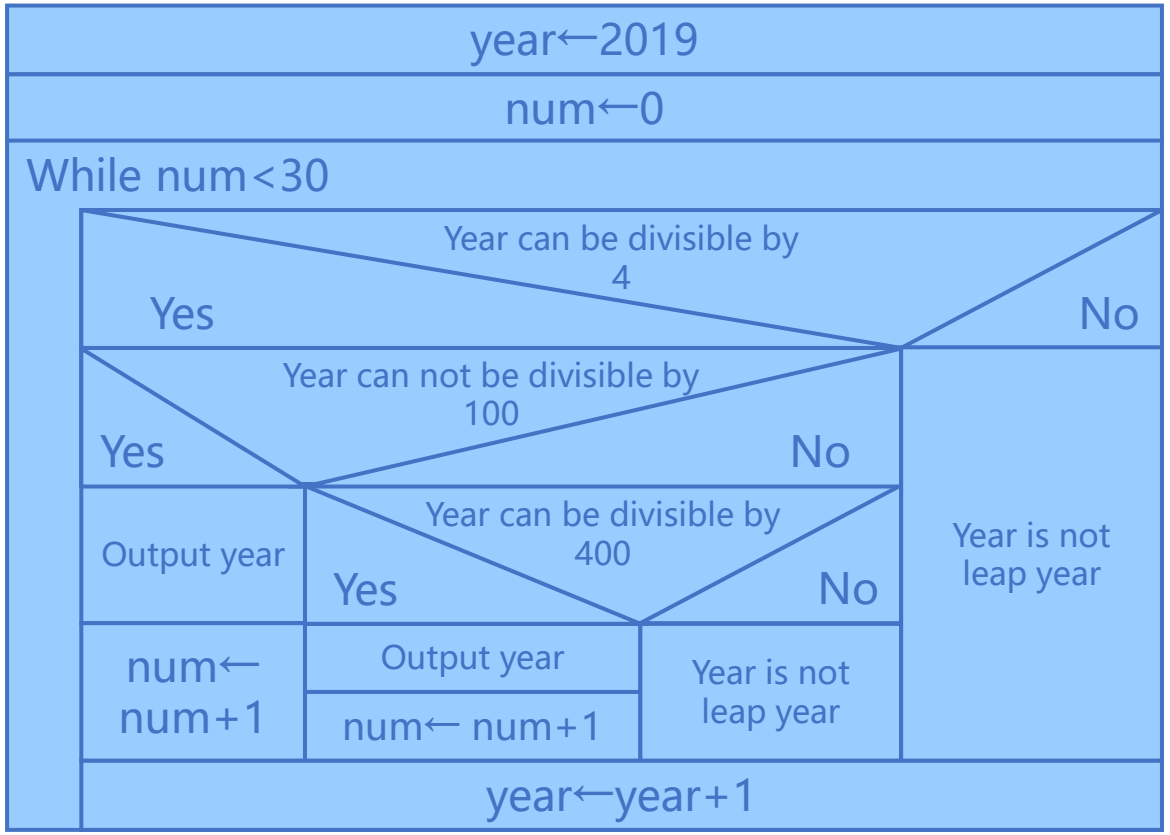
Step4: If year cannot be divisible by 100, the year is a leap year. Go to step6 to continue; otherwise, continue to judge.

Step5: If year can be divisible by 400, the year is a leap year, continue to next step; otherwise, it means that it is not a leap year and jump to step7.

Step6: Find a leap year, output year, num←num+1

Step7: year←year+1

Step8: If num<30, jump to Step3 to continue, judge the next year; Otherwise, the search is completed and the algorithm ends.



What is **pseudo code**?

- An informal language structured between English (or natural language such as Chinese) and computer language, used to describe algorithms.
- Each line or lines of pseudo code represents a basic operation.
- Write from top to bottom and indent the branch structure in the algorithm.
- The structure is clear, the code is simple, and the readability is good; but there may be a lack of rigorousness.

2.4 Methods of representing algorithm

【Example 2.13】Represent the algorithm of example 2.1 by pseudo code.
Calculate the result of 5×5×5×5×5.

Algorithm steps

Step1: p←5

Step2: i←1

Step3: p←p*5

Step4: i←i+1

Step5: If i≥5, output p and end; otherwise return to Step3 and continue.

Pseudo code

```
begin      (Algorithm starts)
   p←5
   i←1
   while i<5
   {
       p←p*5
       i←i+1
   }
   print p
end        (Algorithm ends)
```

2.4 Methods of representing algorithm

SPL

【Example 2.14】Represent the algorithm of example 2.1 by SPL.
Calculate the result of 5×5×5×5×5.

Algorithm steps

Step1: p←5

Step2: i←1

Step3: p←p*5

Step4: i←i+1

Step5: If i≥5, output p and end; otherwise return to Step3 and continue.

2.4 Methods of representing algorithm

【Example 2.14】Represent the algorithm of example 2.1 by SPL.
Calculate the result of 5×5×5×5×5.

Algorithm steps

Step1: p←5

Step2: i←1

Step3: p←p*5

Step4: i←i+1

Step5: If i≥5, output p and end; otherwise return to Step3 and continue.

	A	B
1	5	1
2	for B1<5	>A1=A1*5
3		>B1=B1+1
4	=A1	

SPL syntax

Execution order: from top to bottom, from left to right

For loop :If the condition is met, the code in the code block is executed repeatedly

Code block: A2 is the for loop main cell. In the following rows, if the column of the main cell and all the cells on the left are empty, these rows and the row of the main cell together constitute the code block of the main cell, as shown in the red box in the figure.

The use of SPL will be explained in detail in Chapter 4 and Chapter 5. Here, we only need to understand it briefly.

2.4 Methods of representing algorithm

SPL

【Example 2.15】Represent the algorithm of example 2.3 by SPL.

Calculation:$\pi \approx 4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\frac{4}{11}+\cdots+\frac{4}{197}-\frac{4}{199}$

Algorithm steps

Step1: res←4

Step2: sign←-1

Step3: deno←3

Step4: term←sign*(4/deno)

Step5: res←res+term

Step6: sign←sign*(-1)

Step7: deno←deno+2

Step8: If deno≤199, return to Step4 to continue execution; otherwise, end and output res.

	A	B	C	D
1	4	/A1: res	-1	/C1: sign
2	3	/A2: deno		
3	for A2<=199	=C1*(4.0/A2)	/B3: term	
4		>A1=A1+B3		
5		>C1=C1*(-1)		
6		>A2=A2+2		
7	>output(A1)	/Output π		

2.4 Methods of representing algorithm

【Example 2.16】Represent the algorithm of example 2.4 by SPL.
List 30 consecutive leap years after 2019.

Algorithm steps

Step1: year←2019

Step2: num←0

Step3: if year cannot be exactly divided by 4, then this year is not a leap year. Go to step7 to check the next year.

Step4: If year cannot be divisible by 100, the year is a leap year. Go to step6 to continue; otherwise, continue to judge.

Step5: If year can be divisible by 400, the year is a leap year, continue to next step; otherwise, it means that it is not a leap year and jump to step7.

Step6: Find a leap year, output year, num←num+1

Step7: year←year+1

Step8: If num<30, jump to Step3 to continue, judge the next year; Otherwise, the search is completed and the algorithm ends.

	A	B	C	D
1	2019	/A1: year	0	/C1: num
2	for C1<30	if A1%4 == 0		
3			if A1%100 != 0	>output(A1)
4				>C1=C1+1
5			else if A1%400 == 0	>output(A1)
6				>C1=C1+1
7		>A1=A1+1		

Chapter 2
Algorithm and program

2.5 Structured programming method

2.5 Structured programming method



Structured programming

- When analyzing the requirements, it should be refined from top to bottom.
- Structure oriented, modules are divided according to their functions.
- Structured coding, express the refined algorithm in computer language.

Chapter 2
Algorithm and program

2.6 Exercises

2.6 Exercises

【Exercise 2.1】Find a numerical operation problem and a non numerical operation problem from daily life, and describe their algorithms.

【Exercise 2.2】According to the three branch structure examples of flow chart, design a different selection flow chart and a different loop flow chart.

【Exercise 2.3】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:
Calculation:1+3+5+7+……+197+199

【Exercise 2.4】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:
Input the radius of the circle, calculate its circumference and area, π is 3.14.

【Exercise 2.5】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:
Input a student's test scores in 4 subjects and calculate his total score.

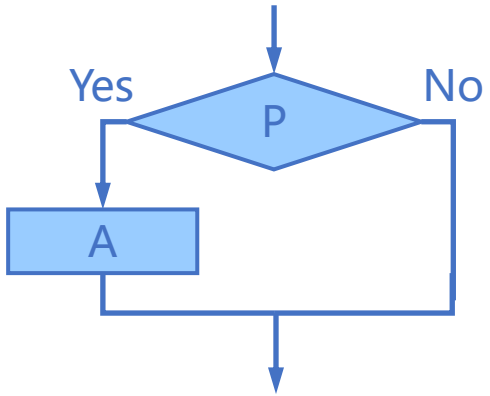
【Exercise 2.6】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:
Output the first 30 Fibonacci Numbers: 1, 1, 2, 3, 5, 8, 13, …

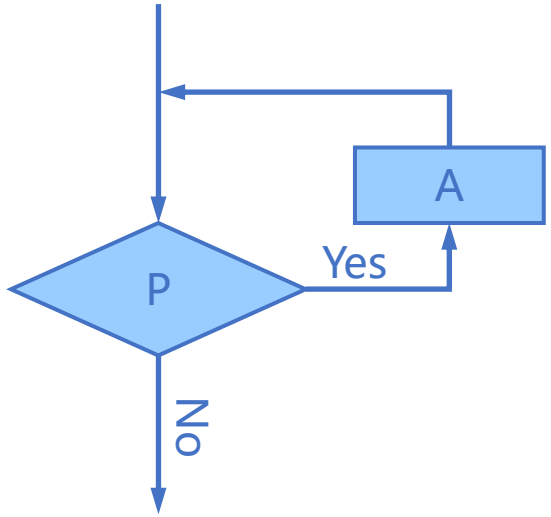
【Exercise 2.7】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:
Find all the 3-digit Narcissistic Numbers: the sum of the cubes of all the digits is equal to itself.

【Exercise 2.8】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:
Now one cock is worth 5 qian, one hen 3 qian and 3 chicks 1 qian. It is required to buy 100 fowls with 100 qian. In each case, find the number of cocks, hens and chicks bought.

2.6 Exercises

【Exercise 2.2】According to the three branch structure examples of flow chart, design a different selection flow chart and a different loop flow chart.

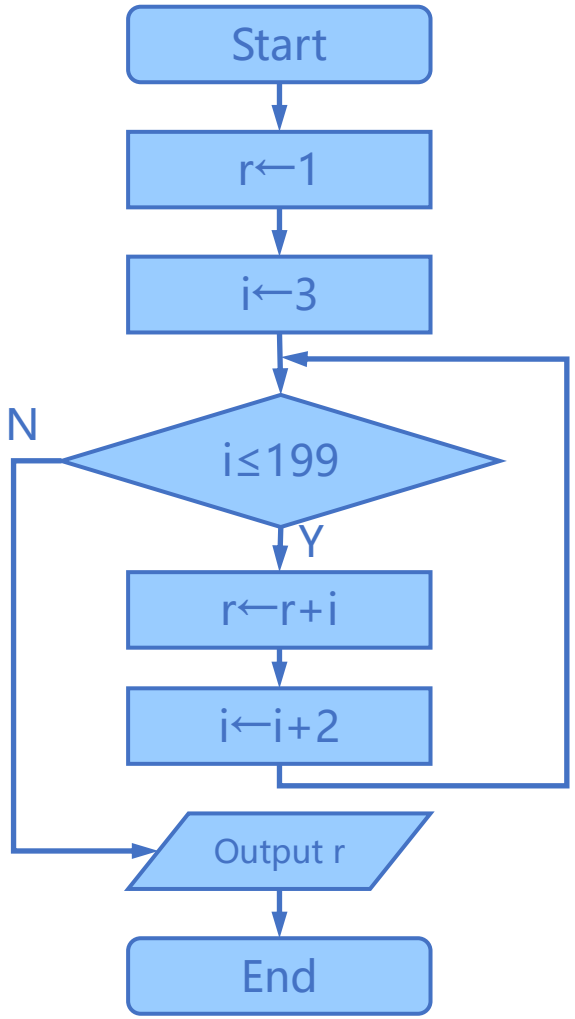


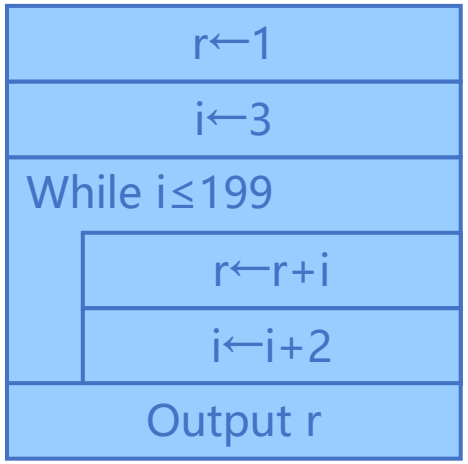


2.6 Exercises

【Exercise 2.3】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:

Calculation:1+3+5+7+.....+197+199





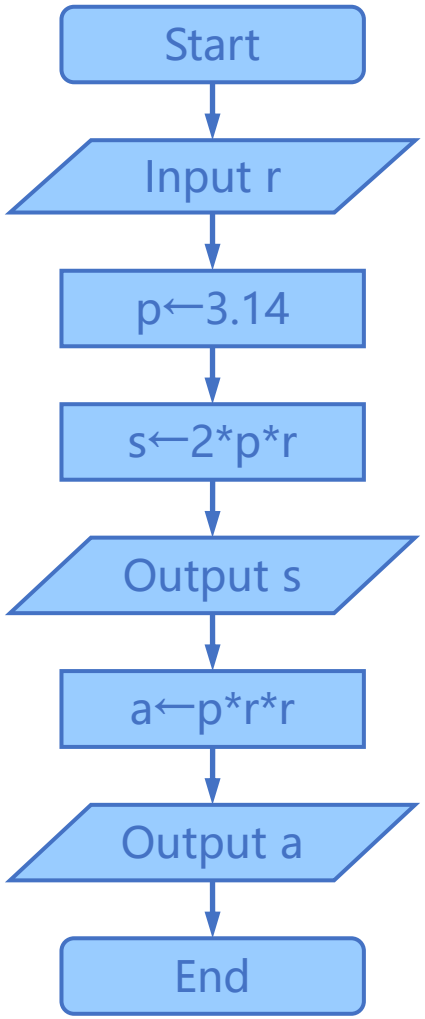
Pseudo code

```
begin        (Algorithm starts)
  r←1
  i←3
  while i≤199
  {
      r←r+i
      i←i+2
  }
  print r
end          (Algorithm ends)
```

2.6 Exercises

【Exercise 2.4】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:

Input the radius of the circle, calculate its circumference and area, π is 3.14.



Input r
p←3.14
s←2*p*r
Output s
a←p*r*r
Output a

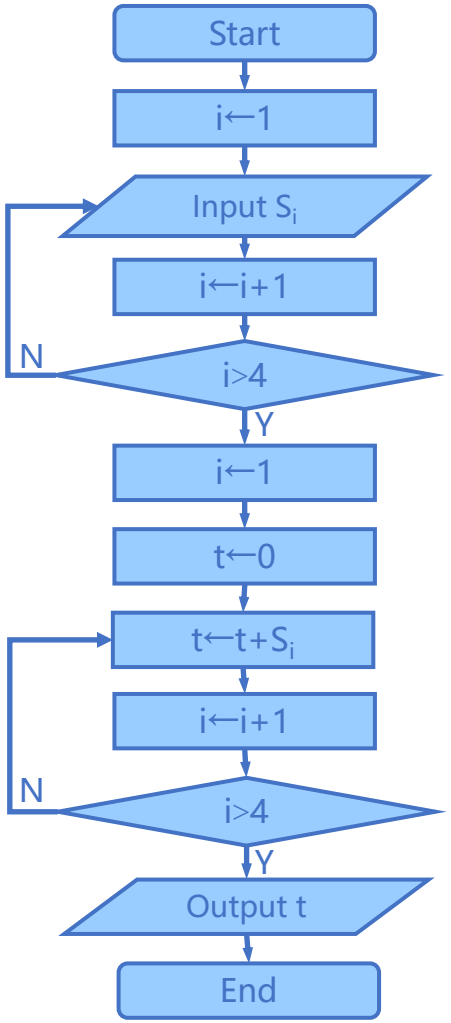
Pseudo code

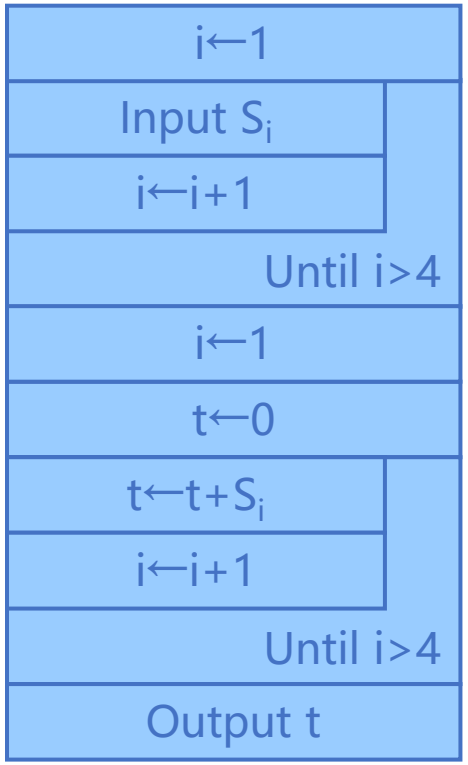
```
begin      (Algorithm starts)
   read r
   p←3.14
   s←2*p*r
   print s
   a←p*r*r
   print a
end        (Algorithm ends)
```

2.6 Exercises

【Exercise 2.5】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:

Input a student's test scores in 4 subjects and calculate his total score.





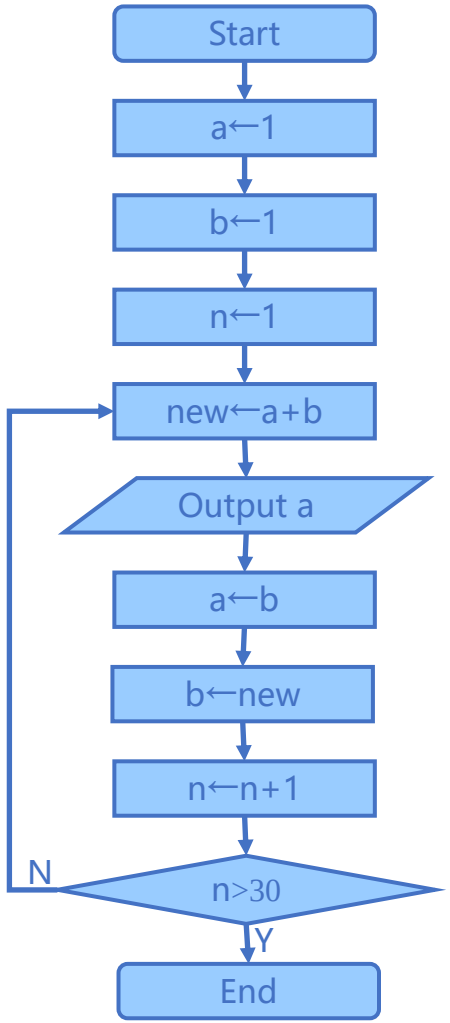
Pseudo code

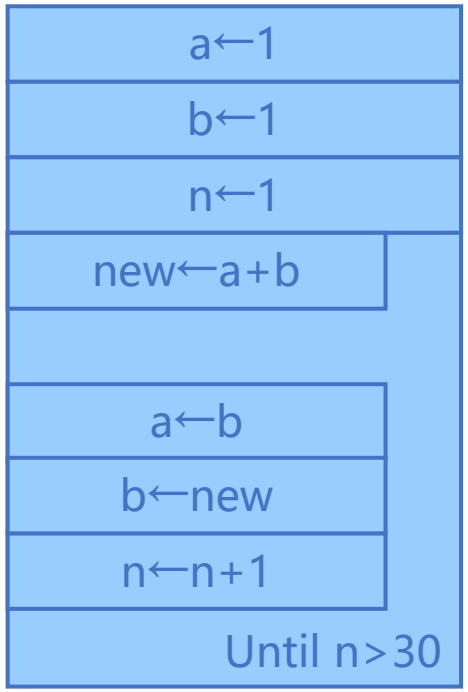
```
begin          (Algorithm starts)
   i←1
   do {
      read Si
      i←i+1
   }
   while i≤4
   i←1
   t←0
   do {
      t←t+Si
      i←i+1
   }
   while i≤4
   print t
end            (Algorithm ends)
```

2.6 Exercises

【Exercise 2.6】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:

Output the first 30 Fibonacci Numbers: 1, 1, 2, 3, 5, 8, 13, …





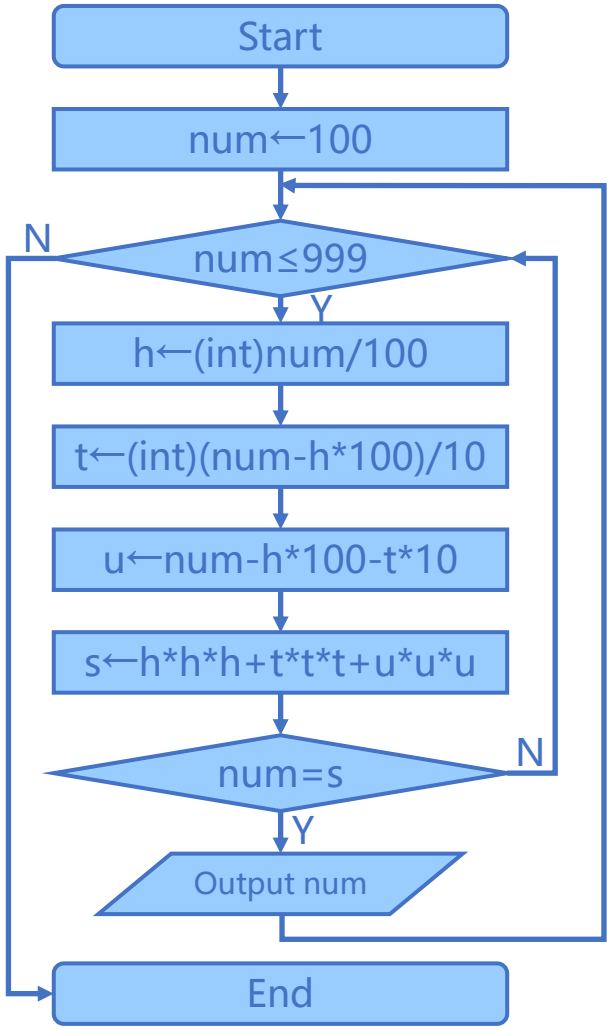
Pseudo code

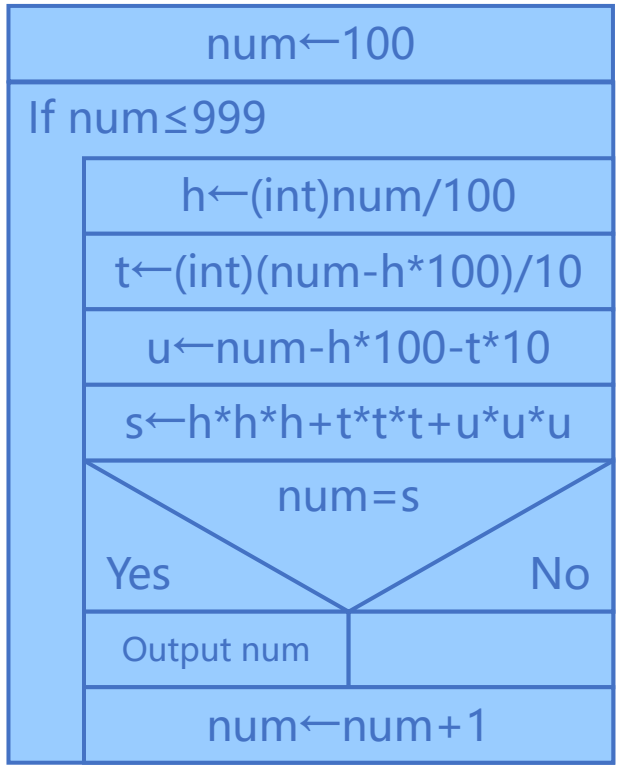
```
begin          (Algorithm starts)
  a←1
  b←1
  n←1
  do {
      new←a+b
      print a
      a←b
      b←new
      n←n+1
  }
  while n≤30
end            (Algorithm ends)
```

2.6 Exercises

【Exercise 2.7】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:

Find all the 3-digit Narcissistic Numbers: the sum of the cubes of all the digits is equal to itself.



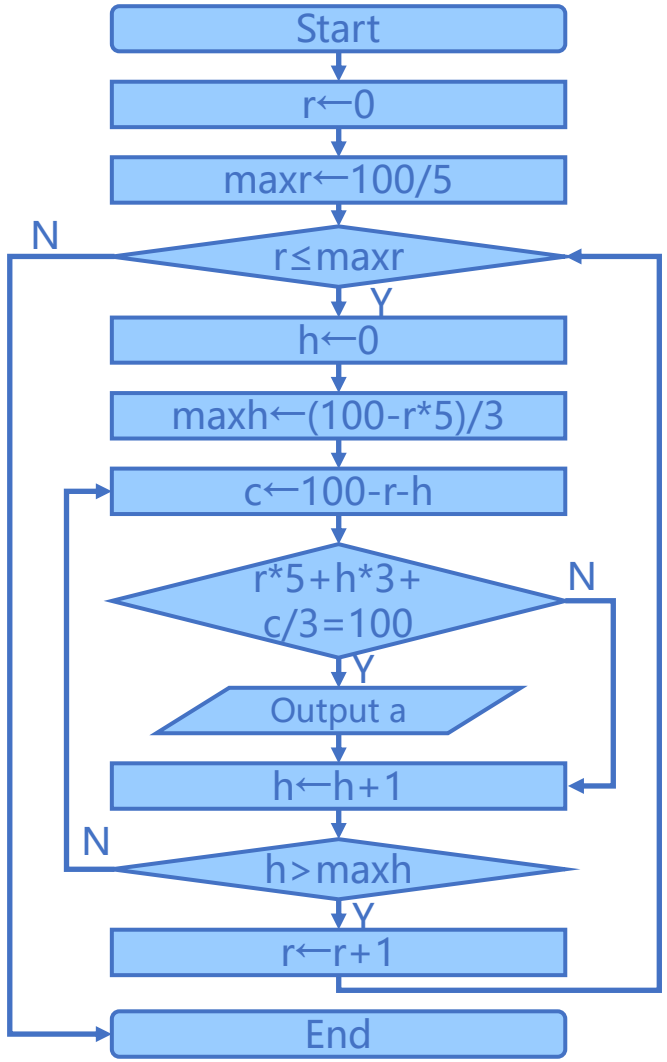


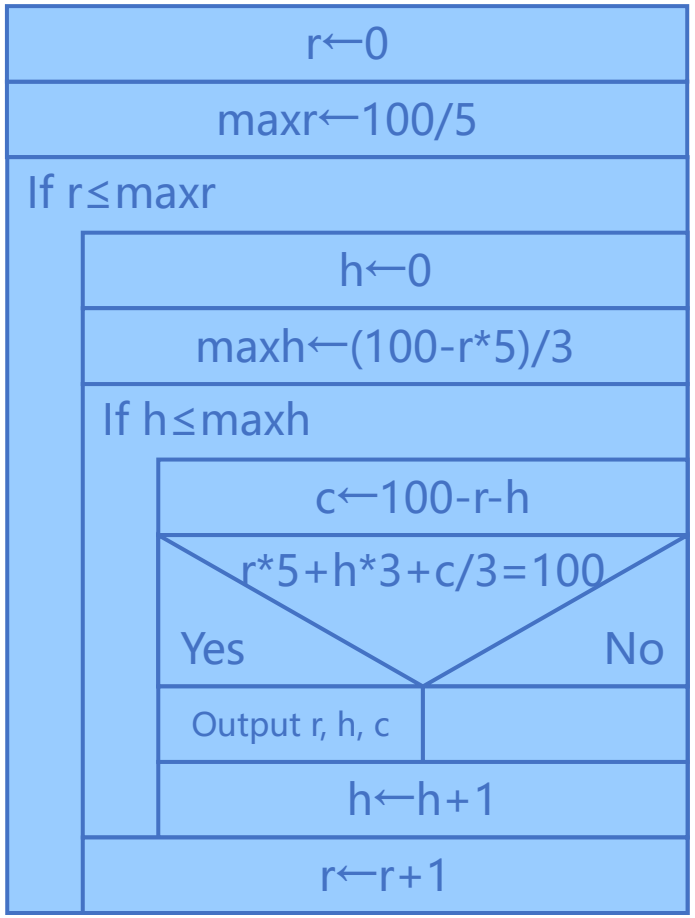
Pseudo code

```
begin        (Algorithm starts)
  num←100
  while num≤999
  {
      h←(int)num/100
      t←(int)(num-h*100)/10
      u←num-h*100-t*10
      if num=h*h*h+t*t*t+u*u*u
      {
          print num
      }
      num←num+1
  }
end          (Algorithm ends)
```

2.6 Exercises

【Exercise 2.8】Represent the algorithm of the following problem by flow chart, NS flow chart and pseudo algorithm respectively:
Now one cock is worth 5 qian, one hen 3 qian and 3 chicks 1 qian. It is required to buy 100 fowls with 100 qian. In each case, find the number of cocks, hens and chicks bought.





Pseudo code

```
begin        (Algorithm starts)
  r←0
  maxr←100/5
  while r≤maxr
  {
     h←0
     maxh←(100-r*5)/3
     while h≤maxh
     {
        c←100-r-h
        if r*5+h*3+c/3=100
        {
           print r, h, c
        }
        h←h+1
     }
     r←r+1
  }
end          (Algorithm ends)
```

Chapter 3
Sequential structure programming

3.1 Examples of sequential programming

3.2 The representation of data and its operation

- Constant and variable
- Data type
- Identification of data types

3.3 Operators and Expressions

- Common operators in SPL
- Basic arithmetic operators
- Operator precedence
- Mixed operations of multi type data
- Forced type conversion operators

3.4 SPL statement

3.5 Input and output of data

- Input/output examples
- Input and output data in SPL

3.6 Exercises

Chapter 3
Sequential structure programming

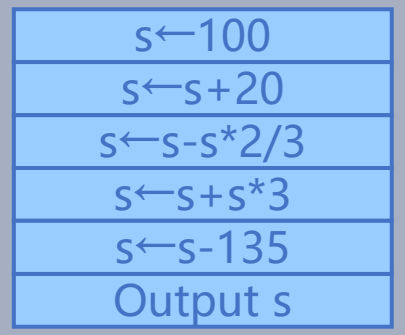
3.1 Examples of sequential programming

【Example 3.1】The stock of a commodity at the beginning of the month is 100 pieces. After purchasing 20 pieces on day 1, two thirds of all goods are sold; on day 2, three times of the remaining stock is purchased, and 135 pieces are sold. What is the inventory balance after day2?

Solution

According to the daily purchase quantity and sales quantity, calculate the daily inventory in turn.

N-S flow chart



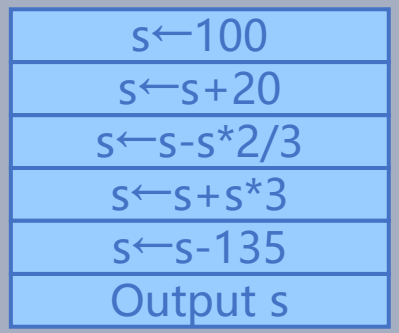
3.1 Examples of sequential programming

【Example 3.1】The stock of a commodity at the beginning of the month is 100 pieces. After purchasing 20 pieces on day 1, two thirds of all goods are sold; on day 2, three times of the remaining stock is purchased, and 135 pieces are sold. What is the inventory balance after day2?

Solution

According to the daily purchase quantity and sales quantity, calculate the daily inventory in turn.

N-S flow chart



	A	B
1	>s=100	/Use variable s to record the stock of goods. The initial value of s is 100
2	>s=s+20	/Calculate the stock after purchasing on day1
3	>s=s-s*2/3	/Calculate the remaining stock after sales on day1
4	>s=s+s*3	/Calculate the stock after purchasing on day2
5	>s=s-135	/Calculate the remaining stock after sales on day2
6	=s	/Get the final result of s

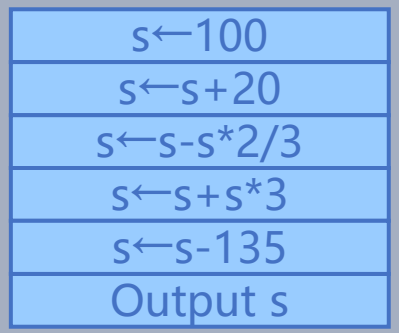
3.1 Examples of sequential programming

【Example 3.1】The stock of a commodity at the beginning of the month is 100 pieces. After purchasing 20 pieces on day 1, two thirds of all goods are sold; on day 2, three times of the remaining stock is purchased, and 135 pieces are sold. What is the inventory balance after day2?

Solution

According to the daily purchase quantity and sales quantity, calculate the daily inventory in turn.

N-S flow chart



	A	B
1	>s=100	/Use variable s to record the stock of goods. The initial value of s is 100
2	>s=s+20	/Calculate the stock after purchasing on day1
3	>s=s-s*2/3	/Calculate the remaining stock after sales on day1
4	>s=s+s*3	/Calculate the stock after purchasing on day2
5	>s=s-135	/Calculate the remaining stock after sales on day2
6	=s	/Get the final result of s

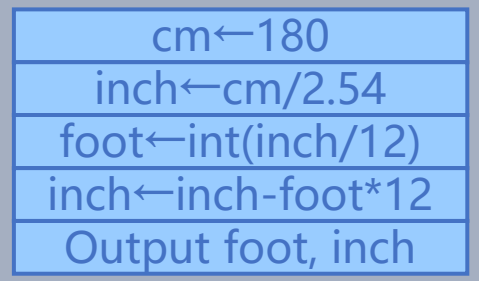
A6	Value
	25.0

【Example 3.2】British height is different from the metric system. For example, 160cm is about 5'3. If someone is 180cm tall, What's the height in feet and inches? (1 inch = 2.54 cm, keep 1 decimal in calculation)

Solution

First convert centimeters to inches, and then calculate the number of feet and inches.

N-S flow chart



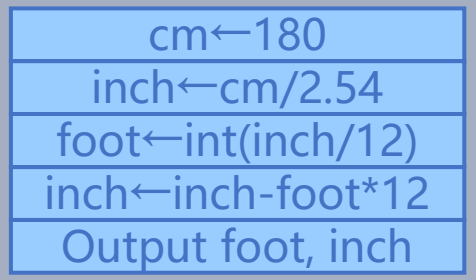
3.1 Examples of sequential programming

【Example 3.2】British height is different from the metric system. For example, 160cm is about 5'3. If someone is 180cm tall, What's the height in feet and inches? (1 inch = 2.54 cm, keep 1 decimal in calculation)

Solution

First convert centimeters to inches, and then calculate the number of feet and inches.

N-S flow chart



	A	B
1	>cm=180	/Use variable cm to record height in cm
2	>inch=cm/2.54	/Calculate the equivalent number of inches
3	>foot=inch\12	/Calculate the value of foot part
4	>inch=round(inch-foot*12, 1)	/Calculate the value of inch part, keeping 1 decimal
5	=foot/"'"/inch	/Concatenate the number of foot and inch

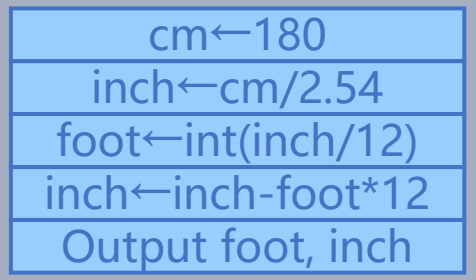
3.1 Examples of sequential programming

【Example 3.2】British height is different from the metric system. For example, 160cm is about 5'3. If someone is 180cm tall, What's the height in feet and inches? (1 inch = 2.54 cm, keep 1 decimal in calculation)

Solution

First convert centimeters to inches, and then calculate the number of feet and inches.

N-S flow chart



	A	B
1	>cm=180	/Use variable cm to record height in cm
2	>inch=cm/2.54	/Calculate the equivalent number of inches
3	>foot=inch\12	/Calculate the value of foot part
4	>inch=round(inch-foot*12, 1)	/Calculate the value of inch part, keeping 1 decimal
5	=foot/"'"/inch	/Concatenate the number of foot and inch

A5	Value
	5'10.9

Chapter 3
Sequential structure programming

3.2 The representation of data and its operation

Constant and variable



Constant

Data that remains unchanged during program operation



Variable

Data that can be changed during program operation

Identifier

- An identifier is the name of an object that can be used to identify constants, variables, and so on.
- Standard identifiers can only have **letters**, **numbers**, or **underscores** _. If necessary, Chinese characters can also be used, but it is necessary to avoid including various symbols that may cause ambiguity, such as plus sign(+) , comma(,), decimal point(.), etc.
- Some program language identifiers are **case sensitive**, i.e. A and a are different. The name of the identifier should be able to indicate its function and meaning.
- Identifiers should **avoid keywords, values**, etc.

Some keywords in SPL

abs	acos	acosh	age	and	arguments	asc	asin	asinh	atan
	atanh	base64	between	bits	break	call	callx	canvas	case
	ceil	channel	char	clear	clipboard	cmp	combin	concat	connect
	cos	cosh	create	cum	cursor	date	datetime	day	days
	deq	directory	elapse	else	end	env	eval	exp	fact
	false	file	filename	fill	floor	for	fork	format	func
	gcd	goto	hash	hosts	hour	httpfile	if	ifa	ifdate
	ifnumber	ifr	ifstring	ift	iftime	ifv	inf	interval	invoke
	isalpha	isdigit	islower	isupper	iterate	join	json	k	lcm
	left	len	lg	like	ln	lock	lower	md5	mid
	millisecond	minute	month	movefile	next	not	now	null	num
	number	or	ord	output	pad	parse	pdate	periods	permut
	pgall	pgcell	pgno	pi	pos	power	product	rand	rands
	range	rank	ranki	reduce	register	replace	return	rgb	right
	round	row	second	seq	shift	sign	sin	sinh	sizeof
	sleep	sqrt	string	substr	sync	system	tan	tanh	time
	to	trim	true	upper	urlencode	vdbase	workday	workdays	xjoin
	xjoinx	xml	xor	year	z	zone			

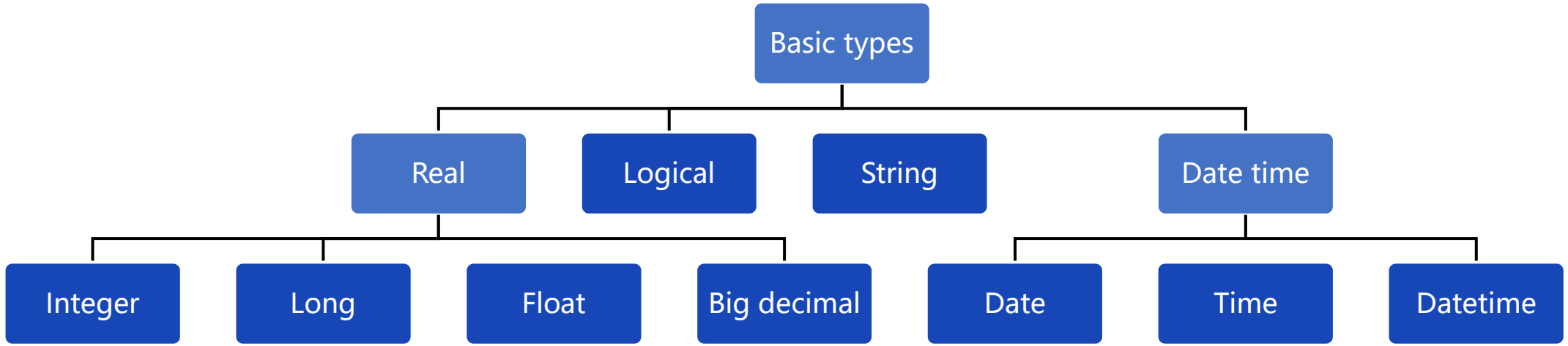
Cell variable

- The name is determined by the location of the cell and consists of **column number(letter) and row number (number)**. For example, B1 represents the first row and the second column.
- Data can be directly entered into the cell as the initial value of the cell variable.
- It can be accessed directly by cell name and can't be named otherwise.
- When a cell variable is assigned, the cell itself is assigned a value.

	A	B	C
1		12.345	
2			
3			

Cell variable is unique in SPL. General programming languages are written as text and do not have cell variables.

Basic data types



The various types of real numbers are described in detail later in this section.

The storage method of integer in computer

The integer in SPL is a 32-bit signed integer stored as 32 bits.



Sign bit, positive number is 0,
negative number is 1

The **Original code** of 155:



The original code sign bit of positive number is 0

$$155 = 2^7 + 2^4 + 2^3 + 2^1 + 1$$

The **Inverse code** of 155: the inverse code of positive integer is the same as the original code



The **Complement code** of 155: the complement code of a positive integer is also equal to the original code



The storage method of integer in computer

The **Original code** of -155:



The original code sign bit of negative number is 1

$$155 = 2^7 + 2^4 + 2^3 + 2^1 + 1$$

The **Inverse code** of -155: the inverse code of negative integers is negated except the sign bit. 0 becomes 1 and 1 becomes 0.



The **Complement** of -155: the complement of a negative integer is equal to the inverse code + 1.



The **Complement code** of integer **is the real storage mode of integer in computer**. The original code is easy to understand but not easy to calculate.

The storage method of integer in computer

Largest positive integer :$2^{31}-1$ =2147483647



0:



-1:



Smallest negative integer : $-2^{31}=-2147483648$



You can have an impression on the above contents, knowing that the binary representation in the computer has some characteristics. It will not affect the follow-up learning if you do not understand the details.

Integer and long integer

Comparison of **integers** and **long integers**

Data type	bits	bytes	Value range
Integer	32 bits	4	-2147483648~2147483647
Long	64 bits	8	-2^{63}~$2^{63}-1$

- When using integers or long integers, pay attention to whether the value itself or the result generated during calculation exceeds the **value range**.

Float

Float

- Float in SPL is **64 bit double precision float**, which is stored as 64 bits (8 bytes).
- The value range of floats is:0 and
  Positive number: 2.2250738585072014E-308 ~ 1.7976931348623157E+308
  Negative number: -1.7976931348623157E+308 ~ -2.2250738585072014E-308
- Floats are stored in binary system, and some values cannot be expressed precisely. Therefore, there may be errors in the calculation of floats. For example, the result of 0.05+0.01 is 0.060000000000000005

Big decimal

Big decimal

- Big decimals can store any real number **without error**.
- Big decimals take up more memory, and the **efficiency is relatively low** in calculation.
- Error free big decimals are usually obtained by **string conversion** or by other **numerical transformations**.
- If a float is converted to a big decimal, because the float itself may have errors, the big decimal obtained **may still be inaccurate**.

String

String

- The string type can be a string of characters of **any length**, a space string or even an empty string.
- Each character in the string can be converted into a corresponding non negative integer.
- There are 128 basic characters, which are corresponded by integers 0 ~ 127 in the **ASCII** table.
- Unlike numeric values, strings are involved in operations such as finding characters, replacing characters, and get substrings.

The use of strings will be explained in detail in Chapter 6, and only a preliminary understanding is needed here.

Logical type

Logical type (**Boolean**)

- There are only two values: **true** and **false**
- Logical type data is used to participate in logical operations, or to represent the result of logical operations.

Date time

Date/Time

- It includes **date**, **time** and **datetime**. They are used to represent date, time, or both.
- Date/time type can be converted to and from a string **in a specific format**. The same date may be converted to a string of different format, such as March 14, 2019, which may be expressed as "2019-3-14", "3/14/19" or "14/3/2019".
- Data of date/time type will participate in some special calculations, and there are many modes of calculation. For example, when calculating the difference, it may be the number of years to obtain the age difference between the two dates, or the number of days or weeks between the two dates.
- When using date/time data, you can use the default format or define the format according to the situation.

Type of a constant

Determination of constant type by SPL

- The numeric value without decimal point is an **integer**, while the value that exceeds 32-bit is a **long integer**, and the long integer can also be represented by the letter **L** after the integer.
- A number with a decimal point or an integer beyond the range of long integers will be judged as a **float**. A float can also be represented by the letter **f** or **d** after the number. If the value exceeds the value range of floats, an error will be reported. Big decimal cannot be defined as constants directly.
- Integers, long integers, and floats can be preceded by a **negative sign(-)** or a **positive sign(+)**.
- Strings that conform to the default **date/time format** are judged as date/time type data.
- Only **lowercase true** and **false** are judged to be Boolean.
- Other strings will be judged as **string** data, but string constants in cells should avoid using keywords. String constants can also be represented by a single quotation mark before a string.

String	Type
123	Integer
-12345678901	Long
+3.1416	Float
false	Boolean
2019-03-25	Date/time
The brown fox.	String

Type of a constant

Display differences of cell constants

- **Integers are shown in blue and aligned to the right.**
- **Long integers are shown in black and aligned to the right.**
- **Floats are shown in deep purple and aligned to the right.**
- **Big decimals are shown in red and aligned to the right.**
- **Booleans are shown in black and aligned to the left.**
- **Date/time values are shown in black and aligned to the left.**
- **Strings are shown in black with blue underline and aligned to the left.**

Type	Cell constant display
Integer	Value 123
Long	Value -12345678901
Float	Value 3.1416
Big decimal	Value 0.06
Boolean	Value false
Date/time	Value 2019-03-25
String	Value The brown fox.

Chapter 3
Sequential structure programming

3.3 Operators and Expressions

3.3 Operators and Expressions

Common operators in SPL

Type	Operators
Arithmetic operators	+ - * / % \
Concatenation operators	+ /
Assignment operators	= ?=
Relational operators	> < == >= <= !=
Logical operators	&& || !
Comma operator	$(x_1, x_2, x_3, \ldots x_k)$
Type conversion operators	int() long() float() decimal() date() time() datetime() string() ""
member operator	. .()
Sequence operators	() | \ ^ & %

Unary operator and binary operator

Unary operator

- Unary operator refers to the operator that only needs **one operand**.
- Unary operators include:

Operator	Meaning
+ -	Positive and negative sign operators
!	Logical non operator
int() long() float() decimal() date() time() datetime() string() ""	Type conversion operators

Binary operator

- Binary operator refers to the operator that requires **two operands**.
- Binary operators include:

Operator	Meaning
+ - * / % \	Addition, subtraction, multiplication, division, remainder, exact division operators
+ /	Concatenation operators
= ?=	Assignment operators
> < == >= <= !=	Relational operators
&& ||	Logic AND, OR
() | \ ^ & %	Sequence operators

Basic arithmetic operators

Operator	Meaning	Example	Result
+	Positive sign operator	+a	a's value
+	Addition operator	a+b	The sum of a and b
-	Negative sign operator	-a	Negative value of a
-	Subtraction operator	a-b	a minus b
*	Multiplication operator	a*b	The product of a times b
/	Division operator	a/b	Quotient of a divided by b
%	Remainder operator	a%b	The remainder of a divided by b
\	Exact division operator	a\b	The integer part of the quotient of a divided by b

Arithmetic expressions and priority of operations

- The formula that connects the operands with arithmetic operators and brackets, conforms to SPL syntax rules, and can realize the calculation requirements is called **SPL arithmetic expression**.
- In arithmetic expressions, different operators have different **priorities**. When calculating, the higher priority operators will be calculated first, such as multiplication, division, exact division and remainder, and then the addition and subtraction will be calculated.
- When calculating ordinary arithmetic expression, it will be calculated from left to right, which is also called "**left associativity**"; however, when processing assignment operator, the right part will be calculated first, which is called "**right associativity**".
- Among the commonly used operators in SPL, bracket priority is the highest level 1, followed by level 2 member operator, positive/negative sign is level 3, level 4 is non operator, level 5 is other arithmetic operators, level 6 is relational operator, level 7 is other logical operators, level 8 is assignment operator, and the priority of comma operator is the lowest level 9.

Arithmetic operations of different types of real numbers

- Among the four types of real numbers, the ranking is as follows:
  Big decimal > float > long > integer. The type of the common arithmetic operation result is higher level in the data involved in calculation. For example, the difference between a long integer minus an integer is a long integer, and the sum of a big decimal and an integer is a big decimal.
- The remainder operator is only allowed between integers or long integers.
- If there is no big decimal in the data involved in the division calculation, the result is a float number. Even if an integer is divided by an integer, the result is a float number.
- When the exact division operation is executed, only two integers are divisible into integers. In other cases, the results are long integers.

Concatenation operation of strings

- In SPL, there are two concatenation operators of strings: + and /.
- Concatenation operator + requires that both operands must be strings, and operands of other types must be converted to strings first by string() or functions format(), char().
- Concatenation operator / requires at least one of the two operands to be a string, and the other will be converted automatically if it is not a string.
- Because of the different requirements for operands, the concatenation operators and the addition and division operators in arithmetic operators will not be confused.

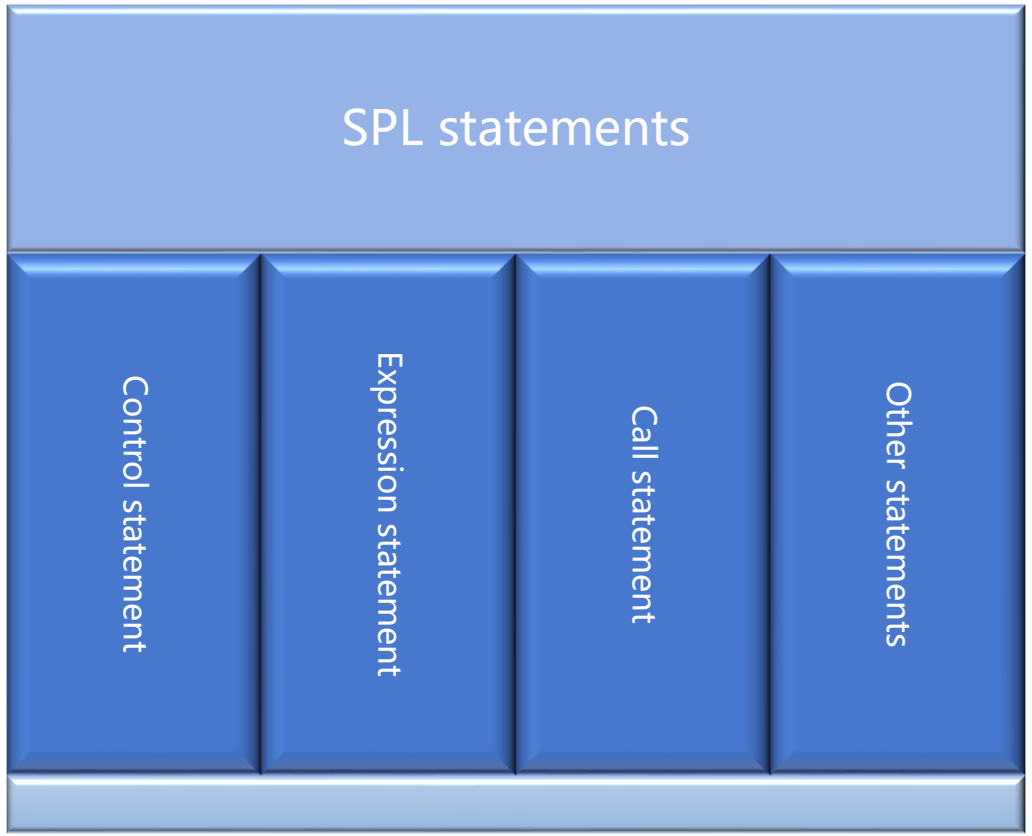
Type(Expression)

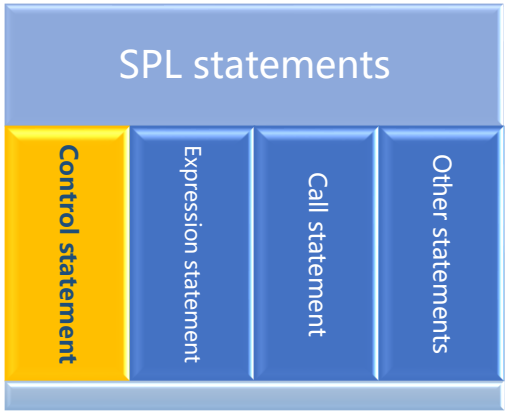
Type	Forced type conversion operators
Integer	int()
Long	long()
Float	float()
Big decimal	decimal()
Boolean	bool()
Date/time	date() time() datetime()
String	string()

Chapter 3
Sequential structure programming

3.4 SPL statement

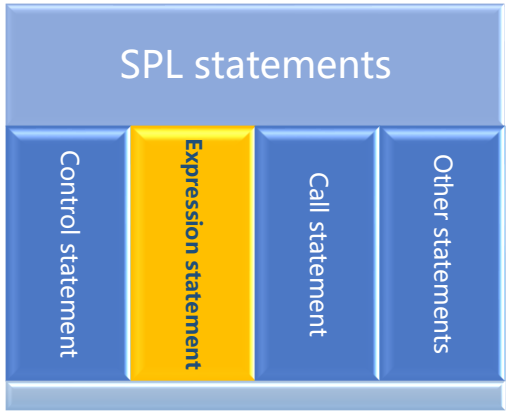
3.4 SPL statements





Control statement

if ... else ...	Conditional statement
goto	Jump statement
for ...	Loop statement
fork ... reduce ...	Multithreaded loop statement
next	Continue loop statement
break	End loop statement
return	End current subroutine and return result
end	Active exception ends and return exception information



Expression statement

- There are two kinds of expression statements: assignment statement and execution statement, which correspond to calculation cell and execution cell.
- The assignment statement starts with =, followed by an expression. After execution, the result of the expression is assigned to the cell. Such as =3
- The execution statement starts with > and is followed by an expression. The executed expression does not necessarily have a result and is not assigned to a cell when there is a result. Such as >A1=A1+4

【Example 3.3】It is known that the three sides of a triangle are 5.6, 7.8 and 9 respectively. What is the area of the triangle?

You can use Heron's formula: $S_{\Delta} = \sqrt{p(p-a)(p-b)(p-c)}$, where a, b, c are the three sides of the triangle, and $p = \frac{a+b+c}{2}$.

【Example 3.3】It is known that the three sides of a triangle are 5.6, 7.8 and 9 respectively. What is the area of the triangle?

You can use Heron's formula: $S_{\Delta} = \sqrt{p(p-a)(p-b)(p-c)}$, where a, b, c are the three sides of the triangle, and $p = \frac{a+b+c}{2}$.

Solution

Using the cell constant to get the triangle's three side length, using Heron's formula, first calculate the value of p, and then calculate the triangle area.

【Example 3.3】It is known that the three sides of a triangle are 5.6, 7.8 and 9 respectively. What is the area of the triangle?

You can use Heron's formula: $S_{\Delta} = \sqrt{p(p-a)(p-b)(p-c)}$, where a, b, c are the three sides of the triangle, and $p = \frac{a+b+c}{2}$.

Solution

Using the cell constant to get the triangle's three side length, using Heron's formula, first calculate the value of p, and then calculate the triangle area.

	A	B	C	D
1	5.6	7.8	9	/Input the length of the triangle's three sides
2	=(A1+B1+C1)/2			/Calculate p in Heron's formula
3	=sqrt(A2*(A2-A1)*(A2-B1)*(A2-C1))			/Calculate triangle area with Heron's formula

A3	Value
	21.659769158511356

Note

Sqrt () is the function that calculates the square root

Assignment operator =

- The function of the **equal sign operator** is to assign the value on the right to the variable on the left. If there is no variable on the left, the value will be assigned to the current cell variable.

For example, constant cells A1, B1 and C1 in example 3.3 can also be assigned with the assignment operator.

	A	B	C	D
1	=5.6	=7.8	=9	/Input the length of the triangle's three sides
2	=(A1+B1+C1)/2			/Calculate p in Heron's formula
3	=sqrt(A2*(A2-A1)*(A2-B1)*(A2-C1))			/Calculate triangle area with Heron's formula

Assignment expression

- If it is an **expression** after the equal sign operator, the expression on the right will be calculated first, and then the result will be assigned to the variable on the left. If there is no variable on the left, the result will be assigned to the current cell.
- Note that the assignment expression is calculated in "right to left" order.

For example:

	A	B	C
1	5.6	7.8	9
2	=A1=B1+C1		

Assignment expression

- If it is an **expression** after the equal sign operator, the expression on the right will be calculated first, and then the result will be assigned to the variable on the left. If there is no variable on the left, the result will be assigned to the current cell.
- Note that the assignment expression is calculated in "right to left" order.

For example:

	A	B	C
1	5.6	7.8	9
2	=A1=B1+C1		

The expression in A2 is equivalent to =(A1=B1+C1) , that is, the result of B1 + C1 is assigned to A1 first and then to A2. After execution, the cell values of A1 and A2 will both be changed. The value written in A1 is only its initial value, and the actual cell value has changed after calculation.

A2	Value
	16.8

A1	Value
	16.8

Compound assignment operators

Adding other arithmetic operators before the assignment operator = can form a **compound assignment operator**, such as A1+=3, which is equivalent to A1=A1+3. Note that = or > should be added before to indicate whether the expression performs calculation or assignment.

Each binary arithmetic operator can form a compound assignment operator with =.

For example:

	A	B	C
1	5.6	7.8	9
2	>A1+=3	>B1/=6	=C1%=4

Note

After A2 and B2 are executed, the values of A1 and B1 will be changed. In fact, A1 and B1 are used as cell variables. When the constant cell is used as a cell variable, the constant written in the cell is only the initial value. The changed value can be viewed on the right after selecting the cell.

Compound assignment operators

Adding other arithmetic operators before the assignment operator = can form a **compound assignment operator**, such as A1+=3, which is equivalent to A1=A1+3. Note that = or > should be added before to indicate whether the expression performs calculation or assignment.

Each binary arithmetic operator can form a compound assignment operator with =.

For example:

	A	B	C
1	5.6	7.8	9
2	>A1+=3	>B1/=6	=C1%=4

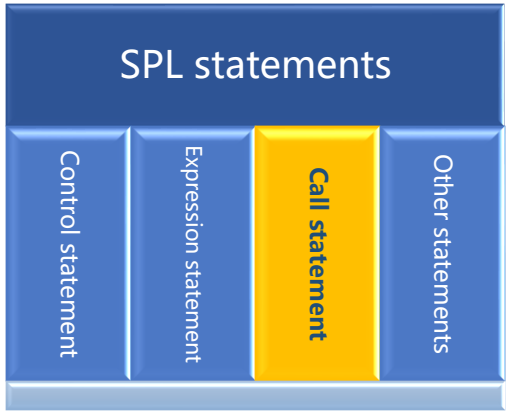
After execution, the results of A1, B1, C1 and C2 are as follows. Since C2 is an assignment cell starting with an equal sign, the cell value of C2 is the same as that of C1.

A1	Value
	8.6

B1	Value
	1.3

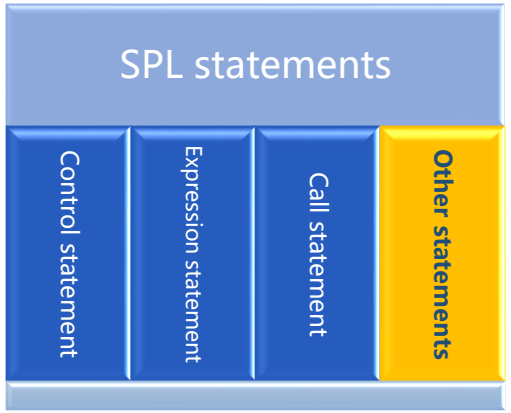
C1	Value
	1.3

C2	Value
	1.3



Call statement

func	Subroutine start statement
return	Return statement
func()	Subroutine call statement
call()	Cross cellset call statement
callx()	Multi thread cross cellset call statement
end	Active ending statement



Other statements

clear	Clear cell value statement
$(db)sql;...	Database query statement
cursor()	Cursor generation statement
for cs	Cursor loop statement
fork cs	Multi thread cursor loop statement

Chapter 3
Sequential structure programming

3.5 Input and output of data

【Example 3.4】Find the solution of $\begin{cases} ax+by+m=0 \\ cx+dy+n=0 \end{cases}$, a, b, m, c, d, n are input as cell parameters. The values in turn are 1, 1, -9, 2, -2, -10 .

3.5 Input and output of data

Input/output examples

【Example 3.4】Find the solution of $\begin{cases} ax+by+m=0 \\ cx+dy+n=0 \end{cases}$, a, b, m, c, d, n are input as cell parameters. The values in turn are 1, 1, -9, 2, -2, -10 .

Solution

The solutions are $x=\frac{bn-dm}{ad-bc}$, $y=\frac{cm-an}{ad-bc}$, output the results after calculation.

	A	B	C	D
1	1	1	-9	/Input parameters of equation 1: a, b, m
2	2	-2	-10	/Input parameters of equation 2: c, d, n
3	=(B1*C2-B2*C1)/(A1*B2-B1*A2)			/Calculate x
4	=(A2*C1-A1*C2)/(A1*B2-B1*A2)			/Calculate y
5	>output("x="/A3/", y="/A4)			/Output result

Note

You can use "..." in the expression to represent a string ...

x=7.0, y=2.0

Input and output data in SPL

- The most commonly used input data method is to input data directly in the cells, using cell constants and cell variables to perform calculations.
- The operation results in SPL can be viewed directly with cell values, and it is not necessary to output them.
- When debugging a program, you can use the **output()** function to output strings and other types of data to the output window.
- In addition to using cell constants directly, you can also input data by defining program parameters. The use of program parameters is described in Chapter 6.

Chapter 3
Sequential structure programming

3.6 Exercises

3.6 Exercises

【Exercise 3.1】If the initial value of x is 5, what is the value of x after executing x+=x-3 ?

【 Exercise 3.2】Input 3 numbers a, b, c, and calculate the result of a+b/c, with 3 decimal places reserved. (reserve 3 decimal places with function round (x, 3))

【 Exercise 3.3】Input a four digit number and output each digit, separated by commas.

【 Exercise 3.4】According to the input Fahrenheit temperature, output the corresponding Centigrade temperature. The formula for converting Fahrenheit temperature F to Centigrade temperature C is as follows:

$$C = \frac{5}{9}(F - 32).$$

【 Exercise 3.5】Input p=4.5, q=321, exchange their values with a program.

【 Exercise 3.6】Read the following program, what is the function of the program? What is the output after execution?

	A	B	C
1	123	23	91
2	=(A1+B1+C1)/3		
3	>output("a="/A1/", b="/B1/", c="/C1/", Avg="/A2)		

3.6 Exercises

【Exercise 3.1】If the initial value of x is 5, what is the value of x after executing x+=x-3 ?

	A	B
1	5	/Input x
2	>A1+=A1-3	/Execute calculation

Or:

	A	B
1	>x=5	/Input x
2	>x+=x-3	/Execute calculation

A1	Value
	7

3.6 Exercises

【Exercise 3.2】Input 3 numbers a, b, c, and calculate the result of a+b/c, with 3 decimal places reserved. (reserve 3 decimal places with function round (x, 3))

	A	B	C	D
1	5	2	3	/Input a, b, c
2	=round(A1+B1/C1, 3)			/Execute calculation

A2	Value
	5.667

3.6 Exercises

【Exercise 3.3】Input a four digit number and output each digit, separated by commas.

	A	B	C	D	E
1	1357				/Input a 4-digit number
2	=A1\1000	=(A1%1000)\100	=(A1%100)\10	=A1%10	/Execute calculation
3	=A2/","/B2/","/C2/","/D2				/Get results

A3	Value
	1,3,5,7

3.6 Exercises

【 Exercise 3.4】According to the input Fahrenheit temperature, output the corresponding Centigrade temperature. The formula for converting Fahrenheit temperature F to Centigrade temperature C is as follows:

$$C = \frac{5}{9}(F - 32).$$

	A	B
1	95	/Input Fahrenheit temperature
2	=round((A1-32)*5/9, 1)	/Convert to Centigrade temperature
3	>output(A1/" C = "/A2/" F")	/Output result

```
95 C = 35.0 F
```

3.6 Exercises

【Exercise 3.5】Input p=4.5, q=321, exchange their values with a program.

SPL script is as follows:

	A	B
1	>p=4.5, q=321	/Input p, q
2	=p	/A2←p
3	>p=q	/p←q
4	>q=A2	/q←A2

No.	Name	Value
1	p	321
2	q	4.5

3.6 Exercises

【Exercise 3.6】Read the following program, what is the function of the program? What is the output after execution?

	A	B	C
1	123	23	91
2	=(A1+B1+C1)/3		
3	>output("a="/A1/", b="/B1/", c="/C1/", Avg="/A2)		

The program is used to calculate the average value of three input numbers, and the result is as follows:

```
a=123,b=23,c=91,Avg=79.0
```

Chapter 4
Selection structure programming

4.1 Selection structure and condition judgment

4.2 Implement selection structure with if statement

- Example of selection structure
- The general form of if statement
- Other forms of if statement

4.3 Relational operators and relational expression

- Relational operators and their precedence
- Relational expression

4.4 Logical operators and logical expression

- Logical operators and their precedence
- Logical expression

4.5 Function if

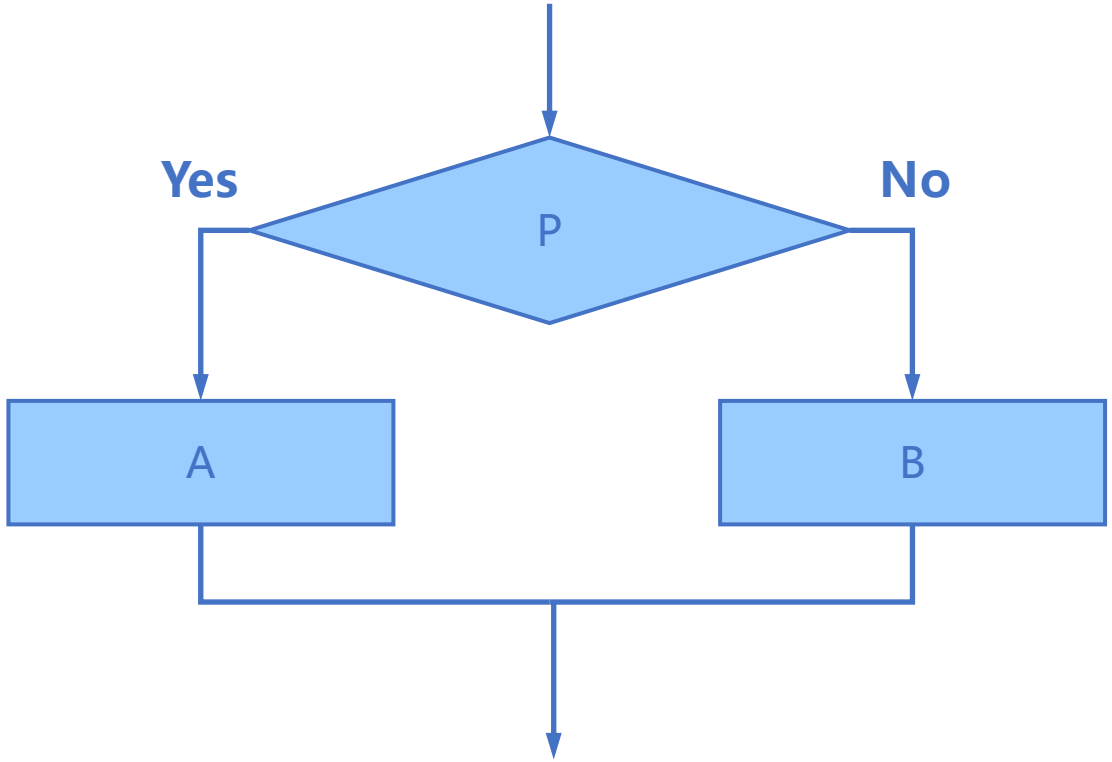
4.6 Nesting of selection structure

4.7 Exercises

Chapter 4
Selection structure programming

4.1 Selection structure and condition judgment

4.1 Selection structure and condition judgment



Chapter 4

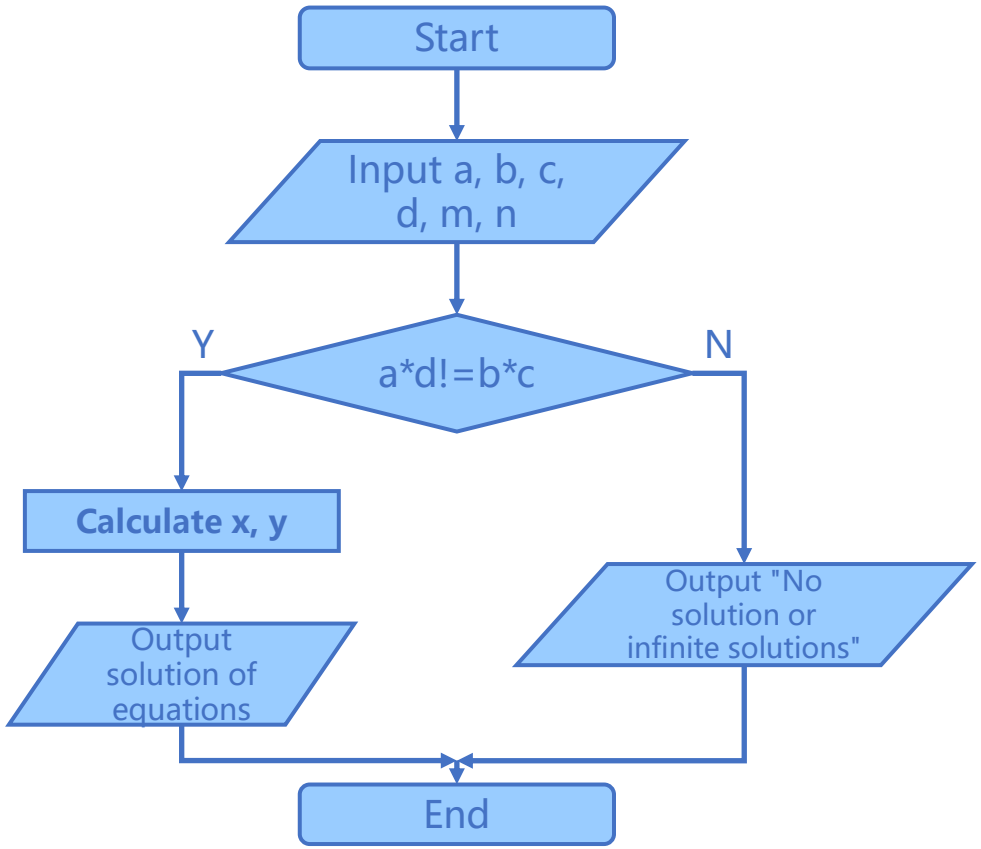
Selection structure programming

4.2 Implement selection structure with if statement

【Example 4.1】In example 3.4, we have studied the problem of finding the solution of Binary Linear Equation Group $\begin{cases} ax + by + m = 0 \\ cx + dy + n = 0 \end{cases}$. In fact, when dealing with the problem, we need to consider the case that $ad\text{-}bc$ is 0 when calculating the results $x = \frac{bn-dm}{ad-bc}$, $y = \frac{cm-an}{ad-bc}$. In this case, there may be no solution or countless solutions.

Example of selection structure

【Example 4.1】In example 3.4, we have studied the problem of finding the solution of Binary Linear Equation Group $\begin{cases} ax+by+m=0 \\ cx+dy+n=0 \end{cases}$. In fact, when dealing with the problem, we need to consider the case that $ad\text{-}bc$ is 0 when calculating the results $x=\frac{bn-dm}{ad-bc}$, $y=\frac{cm-an}{ad-bc}$. In this case, there may be no solution or countless solutions.



	A	B	C	D
1	1	1	-9	/Input parameters a, b, m
2	2	2	-10	/Input parameters c, d, n
3	if A1*B2!=B1*A2	=(B1*C2-B2*C1)/(A1*B2-B1*A2)	=(A2*C1-A1*C2)/(A1*B2-B1*A2)	/Calculate x, y
4		>output("x="/B3/", y="/C3)		/Output result
5	else	>output("No solution or infinite solutions.")		/No solution or infinite solutions

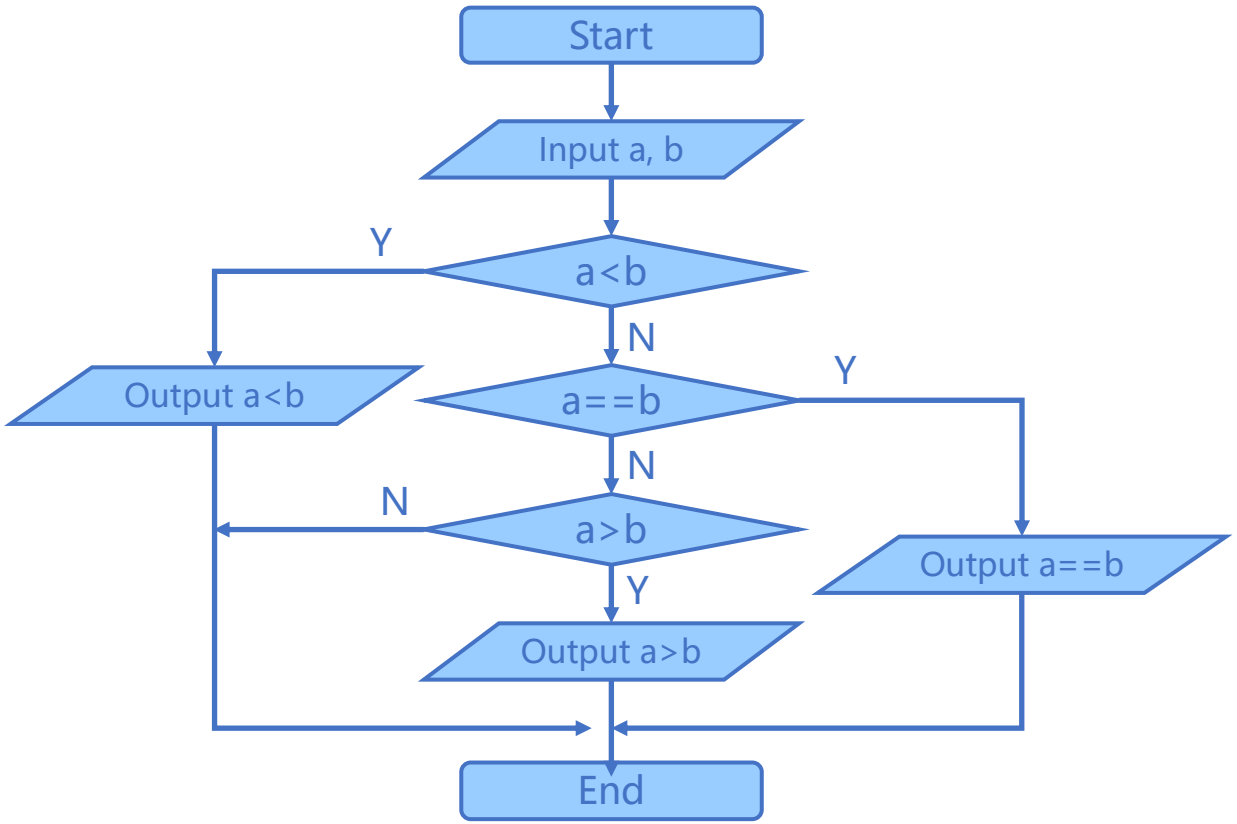
No solution or infinite solutions.

【Example 4.2】Input two real numbers, compare the relationship between them, and output the result.

【Example 4.2】Input two real numbers, compare the relationship between them, and output the result.

Solution

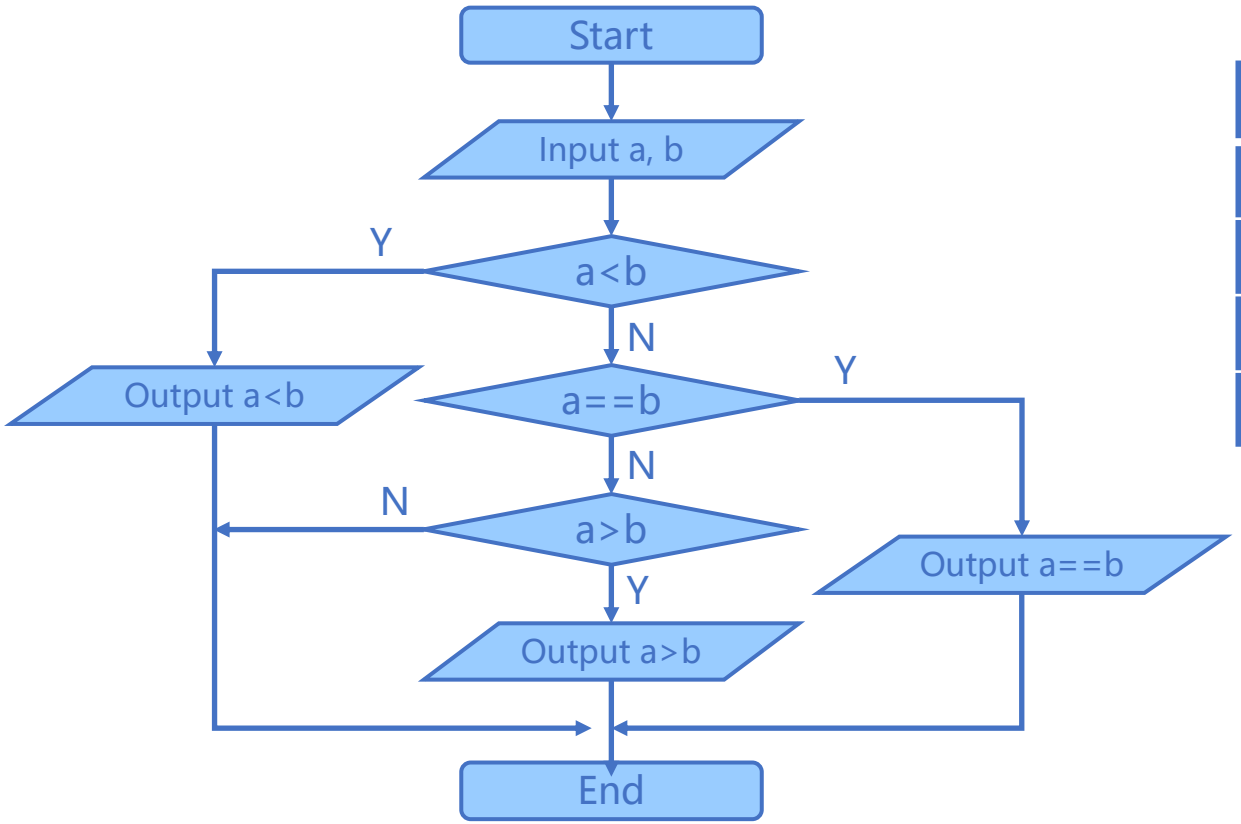
Determine the relationship between them and add different symbols between two real numbers.



【Example 4.2】Input two real numbers, compare the relationship between them, and output the result.

Solution

Determine the relationship between them and add different symbols between two real numbers.



	A	B	C
1	3.1416	3	/Input two real numbers
2	if A1<B1	>output(A1/"<"/B1)	/whether A1 is less than B1
3	if A1==B1	>output(A1/"=="/B1)	/whether A1 is equal to B1
4	if A1>B1	>output(A1/">"/B1)	/whether A1 is greater than B1

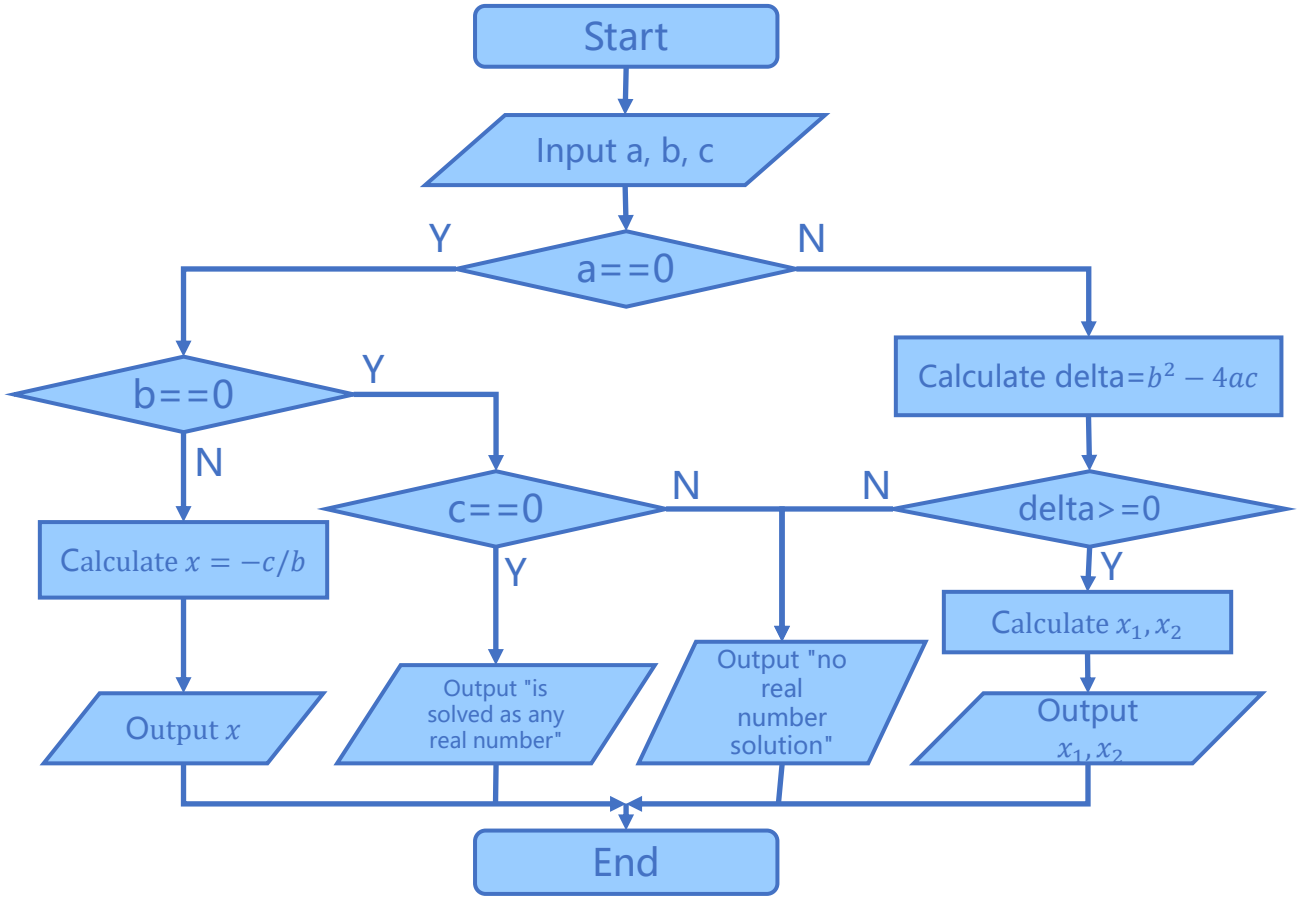
```
3.1416>3
```

【Example 4.3】Input parameters a, b, c to find the real solution of equation $ax^2 + bx + c = 0$.

【Example 4.3】Input parameters a, b, c to find the real solution of equation $ax^2 + bx + c = 0$.

Solution

First we need to judge whether a is 0; for quadratic equation, judge whether the discriminant is greater than 0; For linear equation, further judge the value of b and c.



4.2 Implement selection structure with if statement

Example of selection structure

【Example 4.3】Input parameters a, b, c to find the real solution of equation $ax^2 + bx + c = 0$.

	A	B	C	D	E
1	2	3	-4		/Input a, b, c
2	if A1==0	if B1==0	if C1==0	>output("This equation is solved as any real number.")	/a, b, c are all 0
3			else	>output("This equation has no real number solution.")	/no real number solution
4		else	=round(-C1/B1, 3)	>output("This solution of the equation is "/C4/".")	/Solution of linear equation
5	else	=B1*B1-4*A1*C1			/Calculate the discriminant
6		if B5>=0	=-B1/(2*A1)	=sqrt(B5)/(2*A1)	/Univariate quadratic equation
7			=round(C6+D6, 3)	=round(C6-D6, 3)	/Calculate x_1, x_2
8				>output("The roots of this equation are "/C7/" and "/D7/".")	/Output the solution of quadratic equation of one variable
9		else		>output("This equation has no real number solution.")	/No real number solution

Execution result:

```
The roots of this equation are 0.851 and -2.351.
```

The input equation parameters in A1, B1 and C1 can be adjusted as needed.

The general form of if statement

- if expression code block 1
  [else code block 2]
- The if expression is usually a relational expression, a logical expression, or other expressions. After execution, the calculation result of the expression will be returned as the value of the cell.
- If the result of the expression is null or false, code block 2 is executed; otherwise, the if condition is considered to be true and code block 1 is executed. After the select structure is executed, the code following the if statement will continue to be executed.
- The else clause may not exist, and block 1 is executed only when the condition is true.
- Code block 1 and code block 2 can be a single SPL statement or a series of code composed of multiple SPL statements. If statements can still be used in code blocks.

Code block

- In SPL, a **code block** is used to determine the scope of statements.
- The cell of an **if** statement is the **starting cell** of its code block, also known as the **main cell** of the code block.
- For the several consecutive rows below the main cell, if the column of the main cell and its left are **all empty**, these rows and the row of the main cell together constitute the code block of the main cell.

	A	B	C	D
1				
2		>Code begin		
3				
4				
5				
6	Not null	Not null		

- The scope of the code block is until a certain row where the column of the main cell or any cell to its left is not empty.

if...else... statement in single line

- if expression code block 1 else code block 2

	A	B	C	D
1	-14			
2	if A1>=0	>B1=A1	else	>B1=-A1

- For single line if...else... statement, their respective code blocks can consist of multiple SPL statements, but they should all be written in one line.

else if clause

- if expression 1 code block 1
  else if expression 2 code block 2
  else if expression 3 code block 3
  [else code block 0]

- Multiple else if clauses will be judged one by one until the condition of an expression n is true, and code block n will be executed; or not any condition is met, then code block 0 is executed. After the multiple selection structure is executed, the code following the if statement continues to be executed.

- The else clause can be omitted. In this case, if not any condition is met, the code outside the if statement will continue to be executed.

Chapter 4

Selection structure programming

4.3 Relational operators and relational expression

Relational operators

Operator	Meaning	Example	Result
>	greater than	*a*>*b*	true if a is greater than b, otherwise false
>=	Greater than or equal to (not less than)	*a*>=*b*	true if a is greater than b or a equals b, otherwise false
<	less than	*a*<*b*	true if a is less than b, otherwise false
<=	Less than or equal to (no more than)	*a*<=*b*	true if a is less than b or a equals b, otherwise false
==	equal to	*a*==*b*	true if a equals b, otherwise false
!=	not equal to	*a*!=*b*	true if a is not equal to b, otherwise false

Precedence of relational operators

- In relational operators, the priority of operation is divided into two levels: greater than, greater than or equal to, less than, less than or equal to, the four operators have higher priority and will be calculated first; the two operators, equal to and not equal to, have lower priority.
- Relational operators with the same priority are calculated **from left to right**.
- Relational operators have lower priority than arithmetic operators.
- Relational operators have lower priority than non operator(!), but higher than other logical operators.

Relational expression

- The expression that connects two numerical values or numerical expressions with relational operators is called relational expression.
- The result of a relational expression is a Boolean value, which is **true** or **false** depending on whether the relational expression is true or not.
- In relational expressions of SPL, different types of data cannot be compared which is greater or less, but they can be compared for equality.

Chapter 4

Selection structure programming

4.4 Logical operators and logical expression

Logical operators

Operator	Meaning	Example	Result
!	Logical non operator	!a	If a is true, the result is false; If a is false, the result is true
&&	logical AND operator	a&&b	If both a and b are true, the result is true; otherwise, the result is false
||	logical OR operator	a||b	If at least one of a and b is true, the result is true; otherwise, the result is false

a	*b*	*!a*	*!b*	*a&&b*	*a||b*
true	true	false	false	true	true
true	false	false	true	false	true
false	true	true	false	false	true
false	false	true	true	false	false

Precedence of logical operators

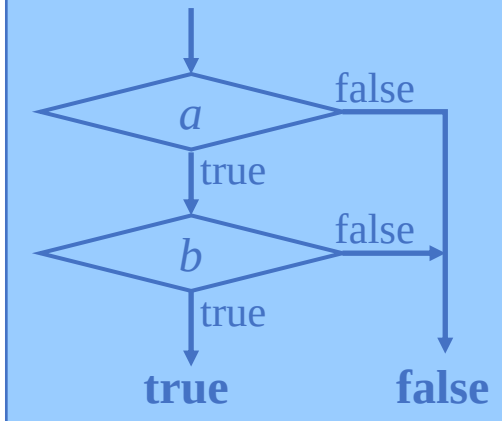
- Logical non operator is unary operations with higher priority than all binary operators. The priority of logical non operator is lower than that of positive and negative sign operators, and higher than other arithmetic operators and relational operators.
- Logical AND and logical OR operators have lower priority than arithmetic and relational operators.
- The priority of logical AND is higher than that of logical OR. When both of them exist, the AND is calculated first and then OR.

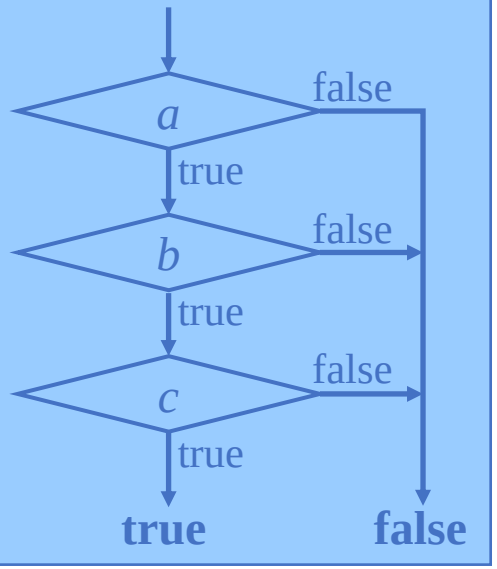
Logical expression

- Logical expressions are expressions that connect relational expressions or other Boolean operands with logical operators.
- The result of a logical expression is also a Boolean value of **true** or **false**.
- In the logic expression of SPL, the operands used by logical operators should be Boolean true or false. Any value of other types will be evaluated as true, and null value will be evaluated as false.

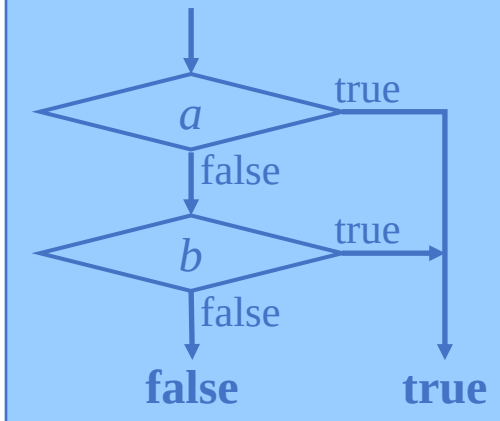
In logical expressions, you can use multiple relational or logical expressions that result in Boolean values. In the calculation of logical expressions, these expressions do not necessarily have to calculate the results.

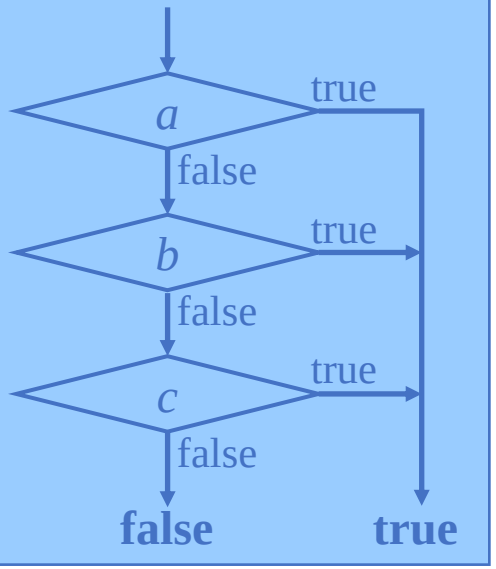
- When calculating $a\&\&b$, only if a is true, the value of b needs to be calculated. When calculating $a\&\&b\&\&c$, only if a is true, the value of b needs to be calculated; Only if both a and b are true, the value of c needs to be calculated.



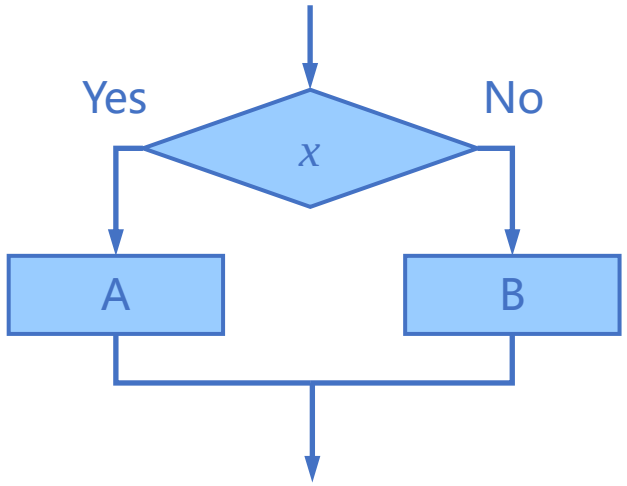


- When calculating $a \| b$, only if a is false, the value of b needs to be calculated. When calculating $a \| b \| c$, only if a is false, the value of b needs to be calculated; Only if both a and b are false, the value of c needs to be calculated.





The results returned by logical expression and relational expression are both Boolean values true or false, so they can be used in the condition of if statement. When the result is true, the condition is met; if the result is false, the condition is not met.



- When x is a Boolean value, if x is true, the condition is met and A is executed; if x is false, the condition is not met and B is executed.
- When x is a relational expression, if the result is true, the condition is met and A is executed; if the result is false, the condition is not met and B is executed.
- When x is a logical expression, if the result is true, the condition is met and A is executed; if the result is false, the condition is not met and B is executed.
- When the calculation result of x is a non Boolean value, such as any integer, character, etc., the condition will be considered to be met and will be executed; if the result is a null value, the condition will be considered to be not met and B will be executed.

Chapter 4
Selection structure programming

4.5 Function If

4.5 Function if

	A	B	C
1	3.1416	3	/Input two real numbers
2	if A1<B1	>max=B1	/If A1<B1, max←B1
3	else	>max=A1	/Otherwise max←A1

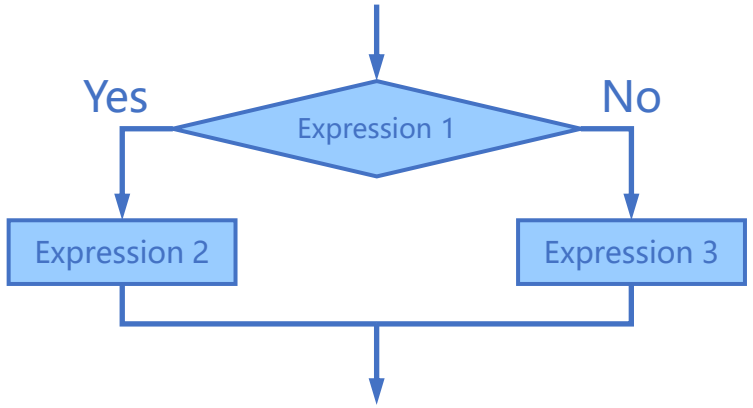
No.	Name	Value
1	max	3.1416

>max=if(A1<B1, B1, A1)

Or >if(A1<B1, max=B1, max=A1)

Function if

- **if(expression1, expression2, expression3)**
- If function of general format has three parameters, and expressions are often used as parameters of function. First calculate expression 1, if false, execute expression 3 and return its result; otherwise, execute expression 2 and return the result.
- If expression 1 is true, only expression 2 will be executed, and expression 3 will not be executed; if expression 1 is not true, only expression 3 will be executed, and expression 2 will not be executed.



4.5 Function if

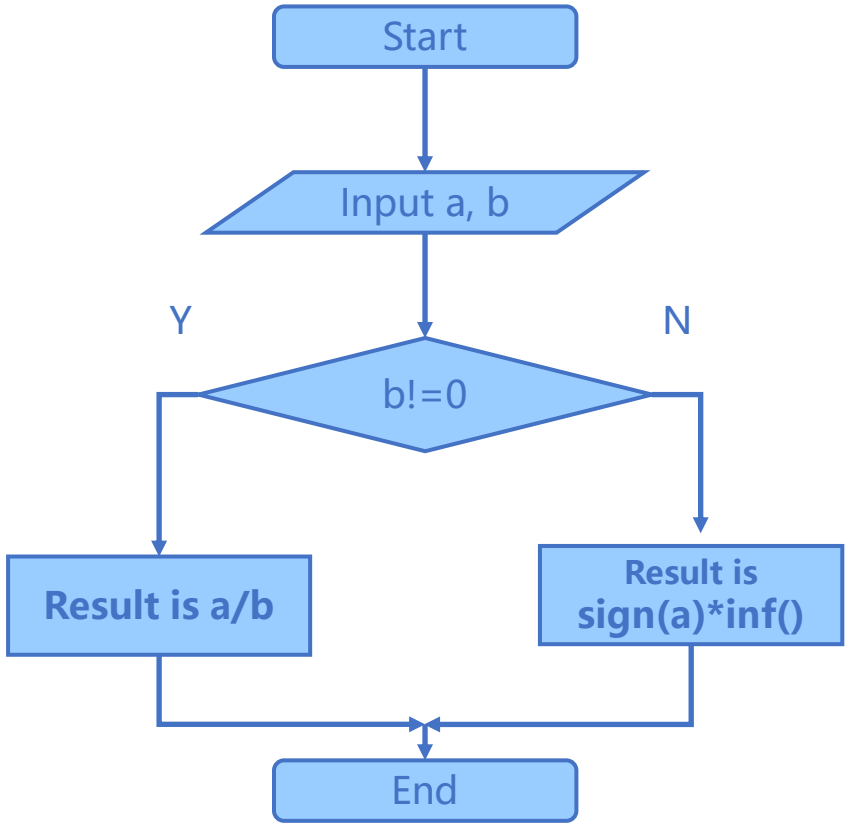
【Example 4.4】Input a, b to calculate the value of a divided by b with 3 decimal places.

4.5 Function if

【Example 4.4】Input a, b to calculate the value of a divided by b with 3 decimal places.

Solution

First, we need to judge whether b is 0. If it is 0, use sign() and inf () functions to calculate the result. The if function can be used directly in the calculation.

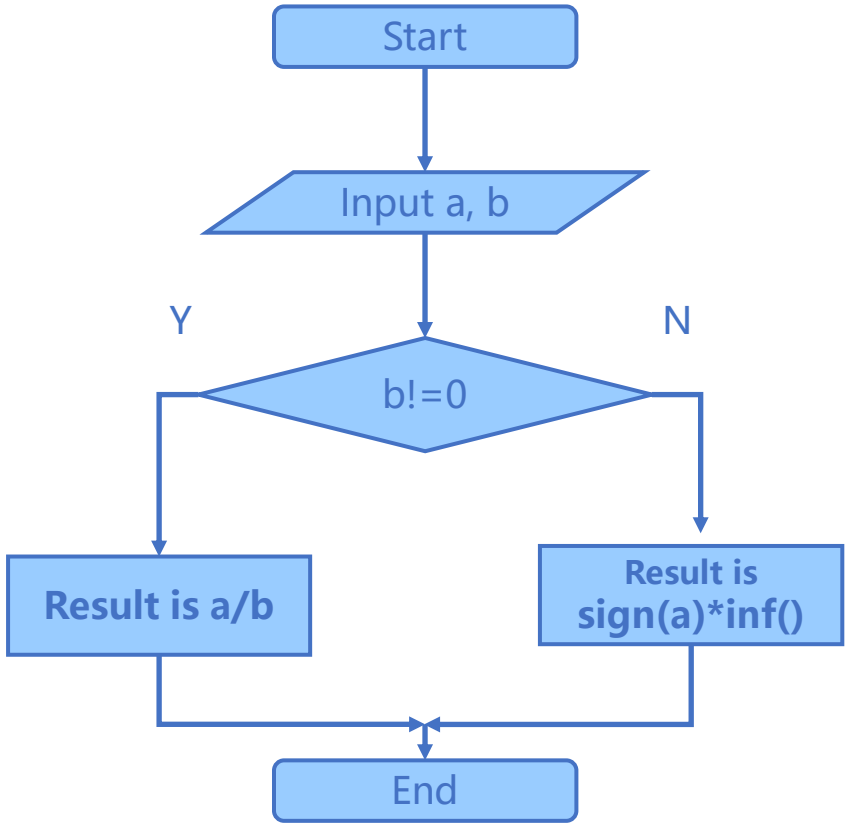


4.5 Function if

【Example 4.4】Input a, b to calculate the value of a divided by b with 3 decimal places.

Solution

First, we need to judge whether b is 0. If it is 0, use sign() and inf () functions to calculate the result. The if function can be used directly in the calculation.



	A	B	C
1	-4	0	/Input parameters a, b
2	=if(B1!=0, round(A1/B1, 3), sign(A1)*inf())		/Calculate a/b, considering whether b is 0

Value
-Infinity

Although A1 / B1 cannot be calculated normally, the calculation above does not report an error because only the corresponding expression can be calculated according to the result of expression 1 in if expression.

4.5 Function if

Similar to the form of if statement, if function also has many formats:

- **if(expression1, expression2)**
  When the result of expression 1 is true, expression 2 is calculated; otherwise, null is returned.
- **if(expression1a, expression1b, expression2a, expression2b, …, expression na, expression nb, expression0)**
  When the result of expression 1a is true, the expression 1b is calculated and the result is returned; otherwise, continue to calculate expression 2a, if its result is true, then calculate expression 2b and return the result; otherwise, continue to calculate expression 3a, … If the expressions has been false until expression na, the expression 0 is calculated and the result is returned. The if function in this format is similar to the case of multiple else if statements.

Chapter 4

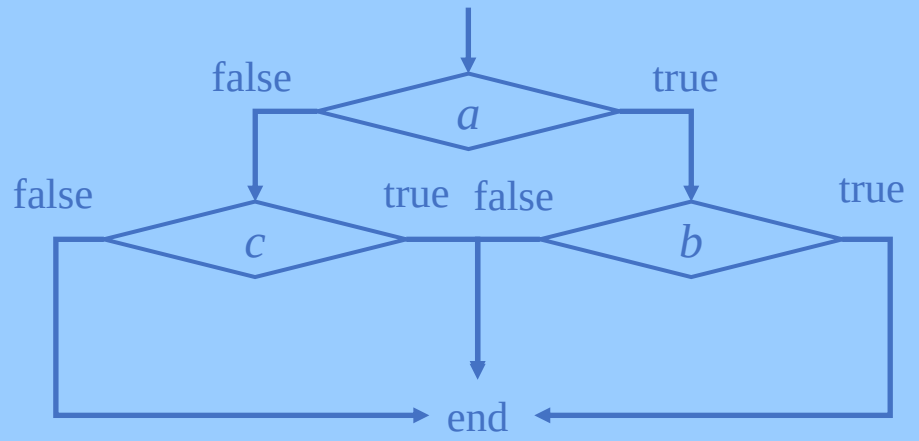
Selection structure programming

4.6 Nesting of selection structure

4.6 Nested selection structure

	A	B	C	D
1	if a			/Outer if judgment
2		if b		/Nested if judgment
3		else		/Nested else statement
4	else			/Outer else statement
5		if c		/Nested if judgment
6		else		/Nested else statement

- Calculate outer selection structure first. If a is true, execute A1's code block and then the nested selection structure if b; if b is true, execute the code after if b, otherwise execute the nested else statement corresponding to if b. If a is false, skip if a code block to directly execute the else code block in the outer selection structure, and then calculate the nested if c code block…



4.6 Nesting of selection structure

	A	B	C	D
1	if a			/Outer if judgment
2		if b		/Nested if judgment
3		else		/Nested else statement
4	else			/Outer else statement
5		if c		/Nested if judgment
6		else		/Nested else statement

- Paired if and else statements need to be aligned.
- If and else can have their own code blocks. When the condition is true, the code block after if will be executed; otherwise, the code block after else will be executed.

Chapter 4
Selection structure programming

4.7 Exercises

4.7 Exercises

【Exercise 4.1】Input two numbers and output them in descending order.

【Exercise 4.2】Input a character and output it after conversion. When converting, lower case letters need to be converted to upper case, and the others remain unchanged.

【Exercise 4.3】In the code, directly write someone's weight (in kg) and height (in m), and calculate his body mass index BMI = weight / (height^2). Simply judge his body level: when BMI is less than 18.5, it is "thin"; when BMI is between 18.5 and 25, it is "fit"; when BMI is between 25 and 27, "overweight"; when BMI is greater than or equal to 27, "obese" .

【Exercise 4.4】The following formula can be used to predict the adult height of a child according to the genetic factors of parents: if fH is the height of the father and mH is the height of the mother, then the adult height of the boy =(fH+mH)*0.54 , and the adult height of the girl =(fH*0.97+mH)/2. Please return the predicted height according to the height of the parents and the gender of the child. The unit is cm, with one decimal place reserved.

【Exercise 4.5】Use if function to solve exercises 4.1 and 4.2.

【Exercise 4.6】Read the following program, what is the output after execution? Why are C1 and C2 outputs different?

	A	B	C
1	>(a=1, b=1, c=1)	=(a=a+1)&&(b=b+1)||(c=c+1)	>output("a="/a/", b="/b/", c="/c)
2	>(a=1, b=1, c=1)	=(a=a+1)||(b=b+1)&&(c=c+1)	>output("a="/a/", b="/b/", c="/c)

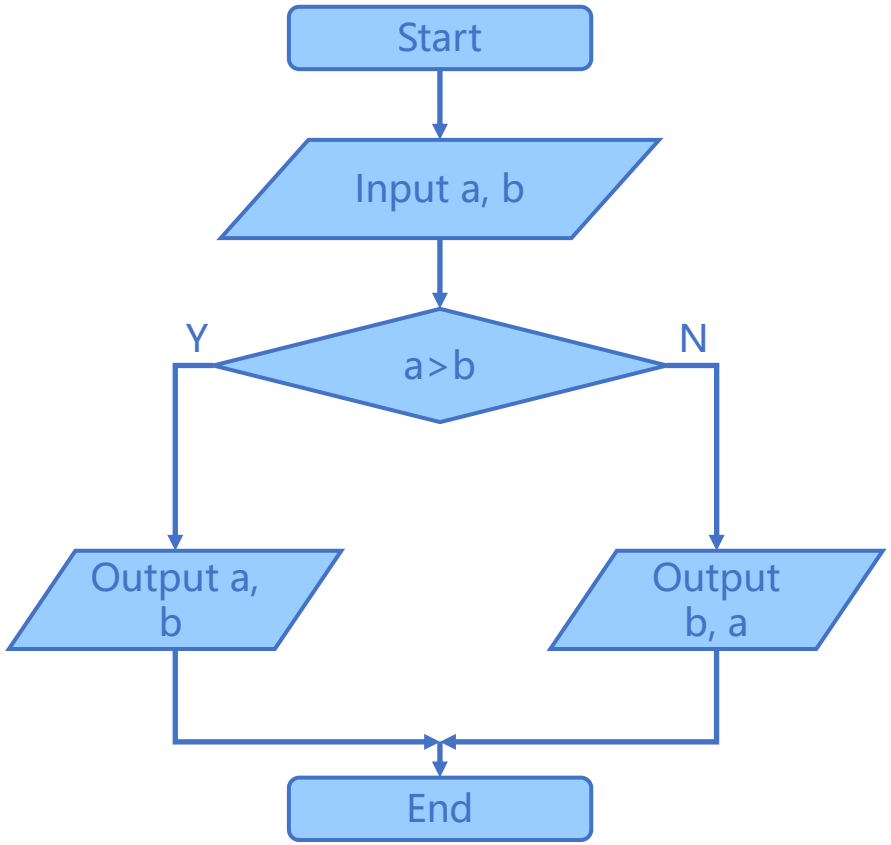
【Exercise 4.7】Input an 8-digit birthday, such as 19800120 for January 20, 1980. It's required that the year of birth is greater than 1899 and less than 2020, judge whether the input date is true or false. You can use the mid() and days() functions. If you want to get the month in the string, you can use int(mid (birthday, 5, 2)) .To judge how many days are in February 1980, you can use days(date(1980, 2, 1)).

4.7 Exercises

【Exercise 4.1】Input two numbers and output them in descending order.

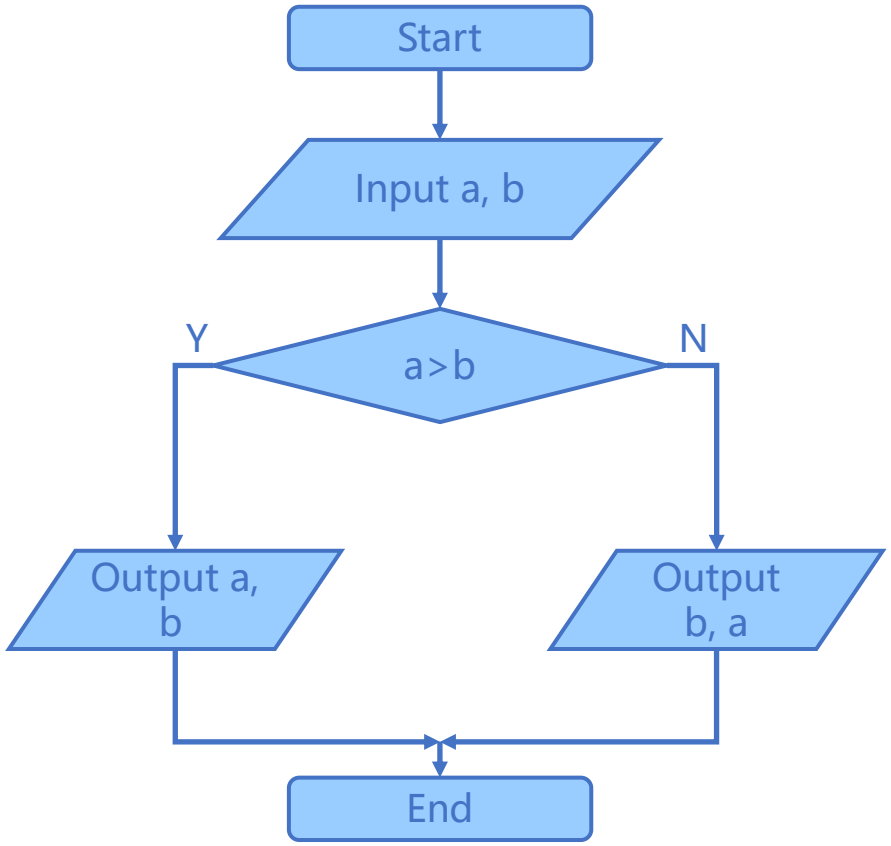
4.7 Exercises

【Exercise 4.1】Input two numbers and output them in descending order.



4.7 Exercises

【Exercise 4.1】Input two numbers and output them in descending order.



	A	B	C
1	10	15	/Input a, b
2	if A1>B1	>output(A1/", "/B1)	/Output a, b
3	else	>output(B1/", "/A1)	/Output b, a

Result(example):

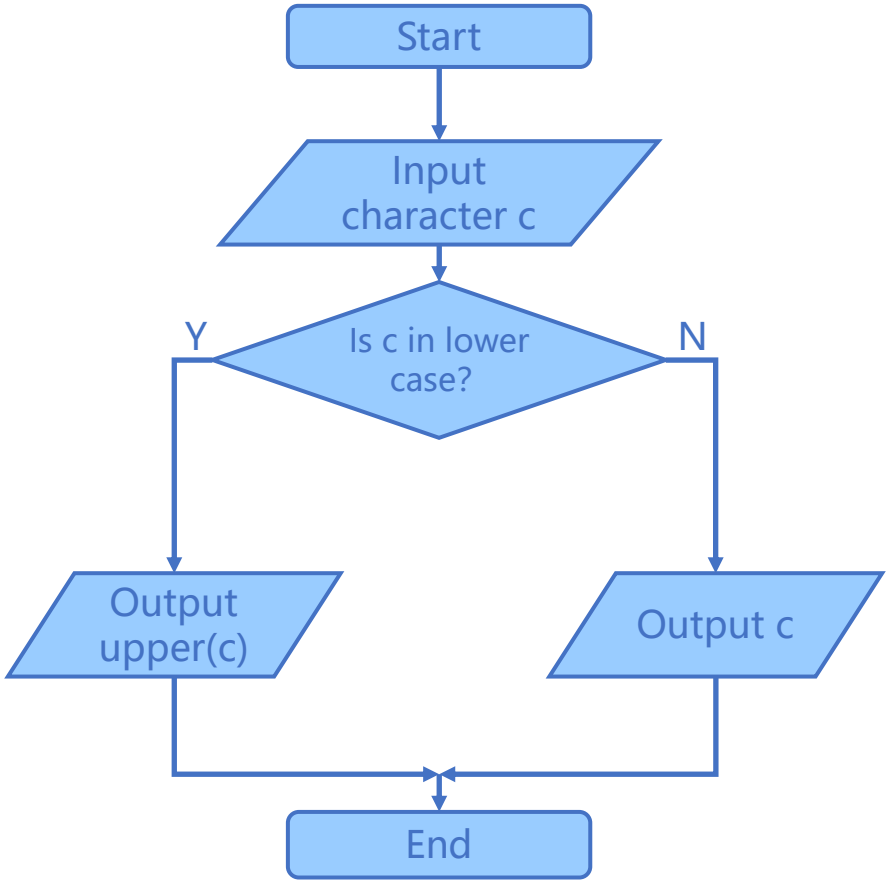
```
15, 10
```

4.7 Exercises

【Exercise 4.2】Input a character and output it after conversion. When converting, lower case letters need to be converted to upper case, and the others remain unchanged.

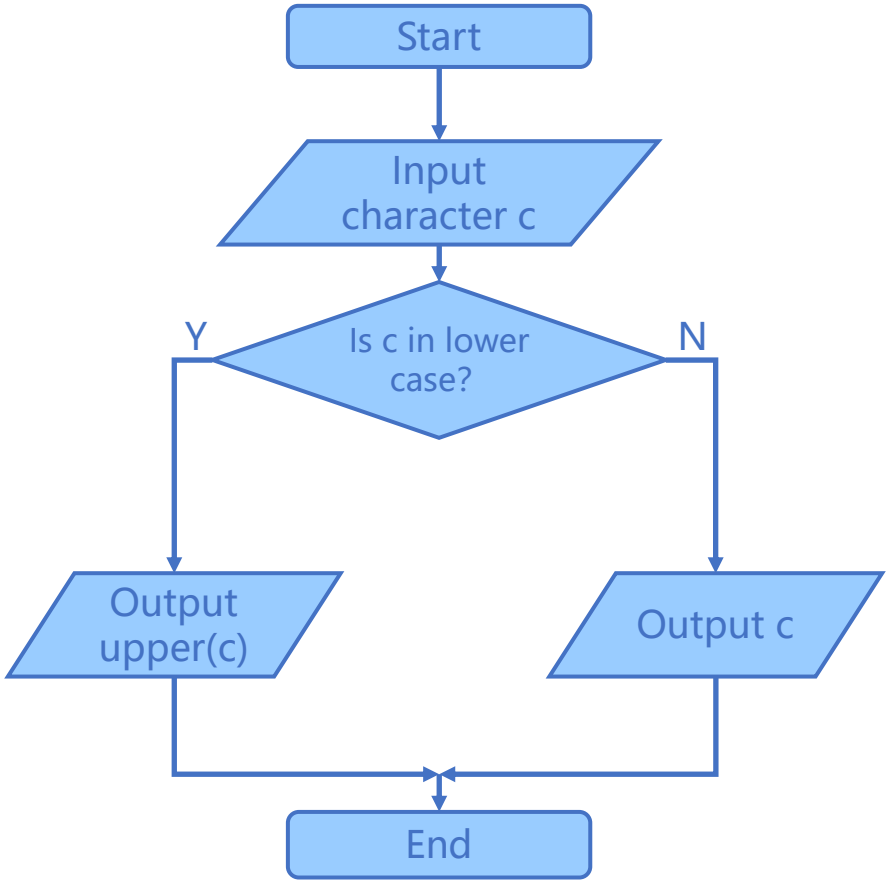
4.7 Exercises

【Exercise 4.2】Input a character and output it after conversion. When converting, lower case letters need to be converted to upper case, and the others remain unchanged.



4.7 Exercises

【Exercise 4.2】Input a character and output it after conversion. When converting, lower case letters need to be converted to upper case, and the others remain unchanged.



	A	B	C
1	b		/Input character c
2	if A1>="a" && A1<="z"	>output(upper(A1))	/Convert lower case letter to upper case to output
3	else	>output(A1)	/Output letter directly

Result(example):

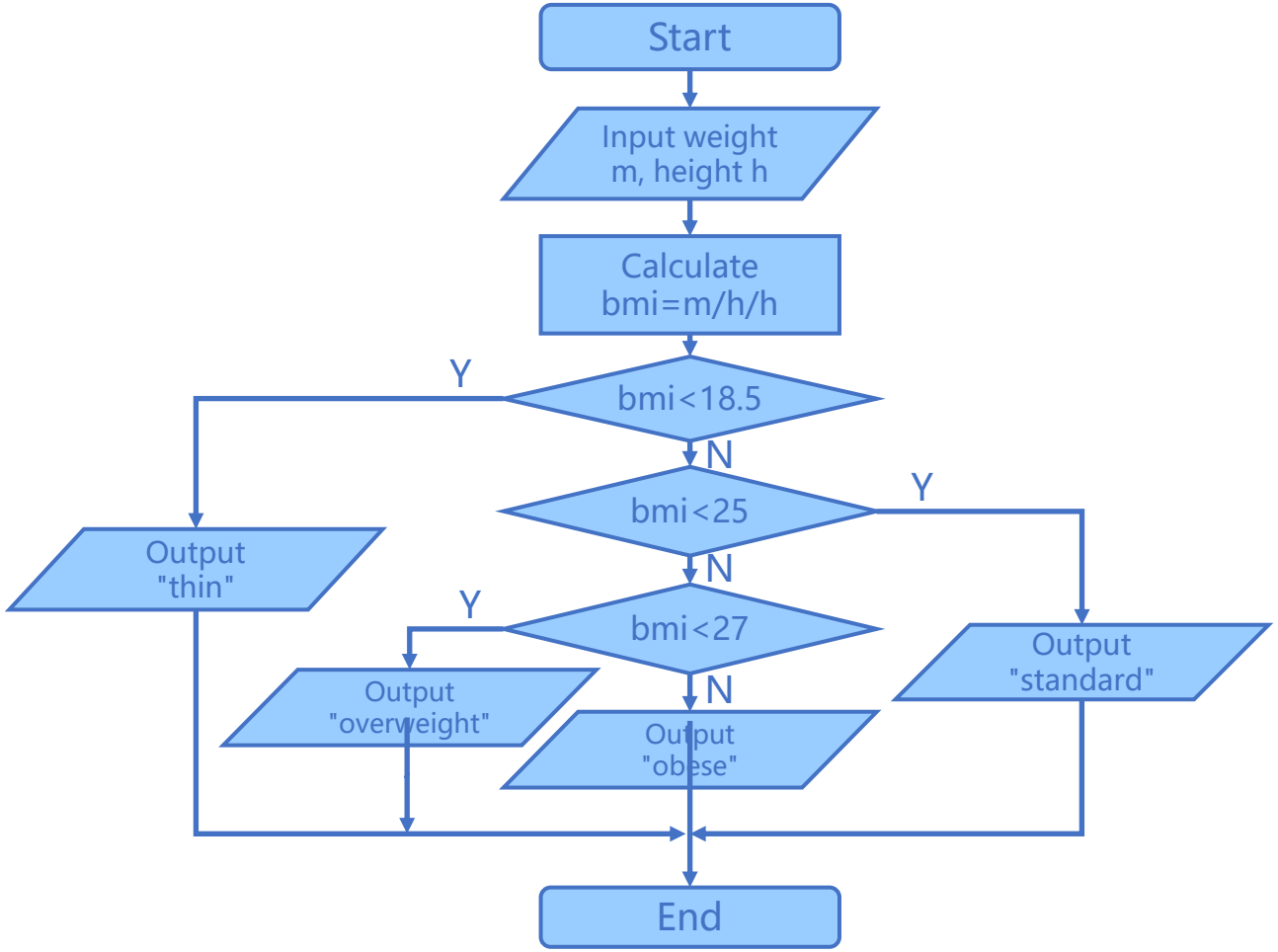


4.7 Exercises

【Exercise 4.3】In the code, directly write someone's weight (in kg) and height (in m), and calculate his body mass index BMI = weight / (height^2). Simply judge his body level: when BMI is less than 18.5, it is "thin"; when BMI is between 18.5 and 25, it is "fit"; when BMI is between 25 and 27, "overweight"; when BMI is greater than or equal to 27, "obese" .

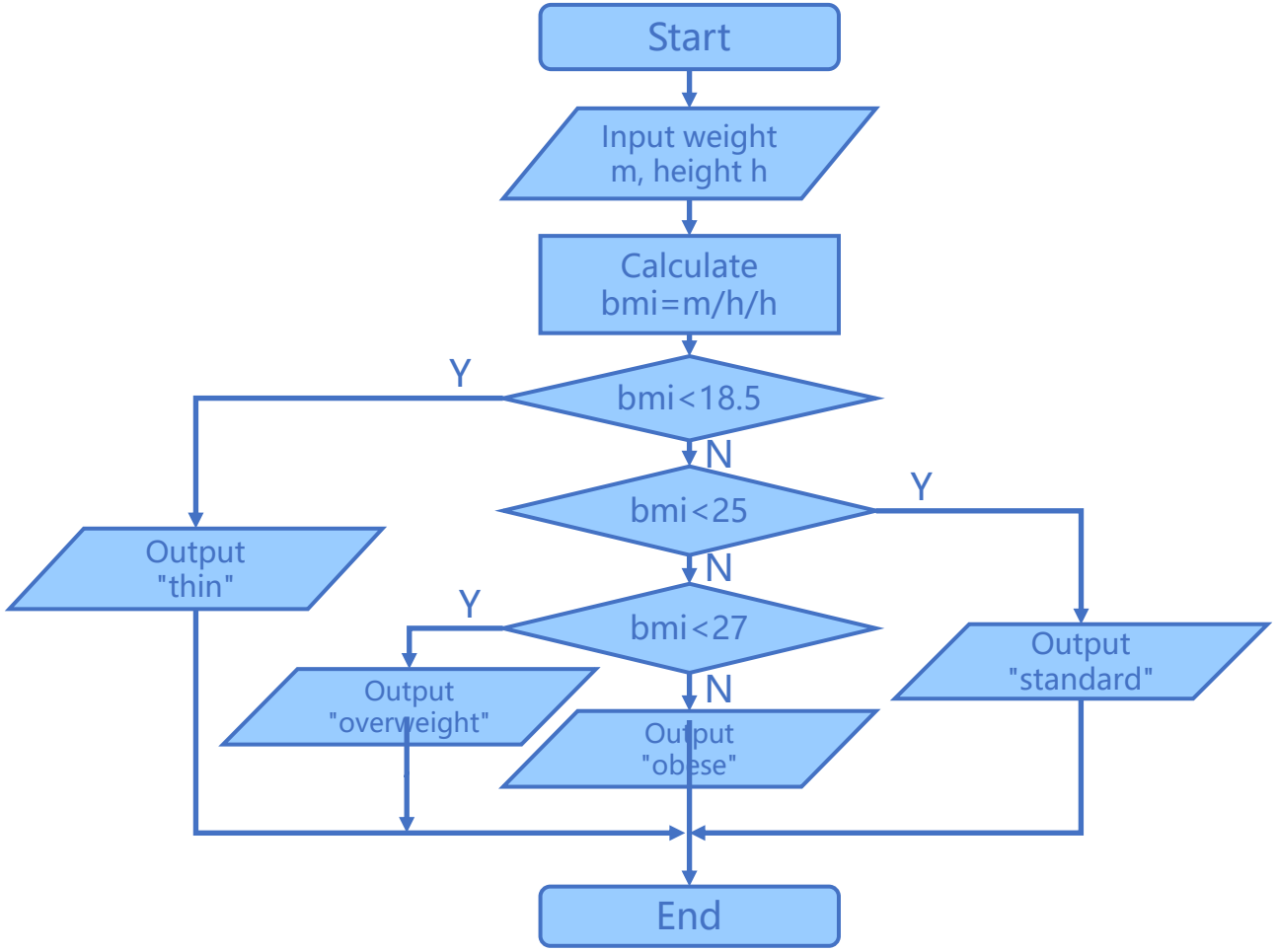
4.7 Exercises

【Exercise 4.3】In the code, directly write someone's weight (in kg) and height (in m), and calculate his body mass index BMI = weight / (height^2). Simply judge his body level: when BMI is less than 18.5, it is "thin"; when BMI is between 18.5 and 25, it is "fit"; when BMI is between 25 and 27, "overweight"; when BMI is greater than or equal to 27, "obese" .



4.7 Exercises

【Exercise 4.3】In the code, directly write someone's weight (in kg) and height (in m), and calculate his body mass index BMI = weight / (height^2). Simply judge his body level: when BMI is less than 18.5, it is "thin"; when BMI is between 18.5 and 25, it is "fit"; when BMI is between 25 and 27, "overweight"; when BMI is greater than or equal to 27, "obese" .



	A	B	C
1	80	1.75	/Input m, h
2	>bmi=A1/B1/B1		/Calculate bmi
3	if bmi<18.5	>output("thin")	/Whether it's thin or not
4	else if bmi<25	>output("fit")	/Whether it's fit or not
5	else if bmi<27	>output("overweight")	/Whether it's overweight or not
6	else	>output("obese")	/It's obese

Result(example):

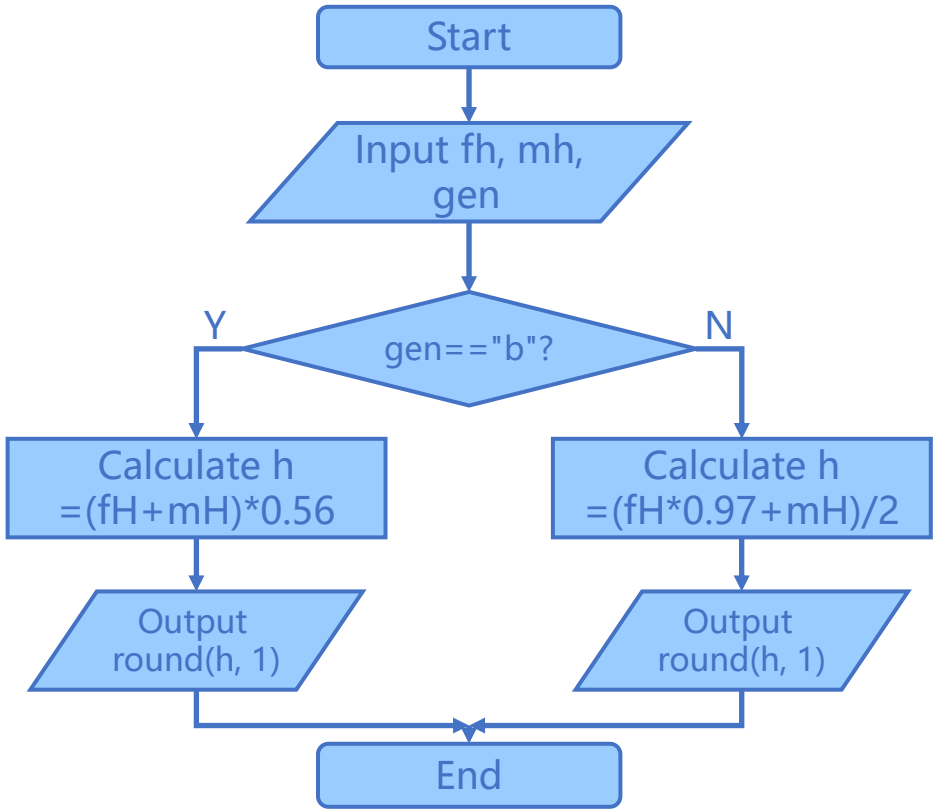
overweight

4.7 Exercises

【Exercise 4.4】The following formula can be used to predict the adult height of a child according to the genetic factors of parents: if fH is the height of the father and mH is the height of the mother, then the adult height of the =(fH+mH)*0.54 , and the adult height of the girl =(fH*0.97+mH)/2. Please return the predicted height according to the height of the parents and the gender of the child. The unit is cm, with one decimal place reserved.

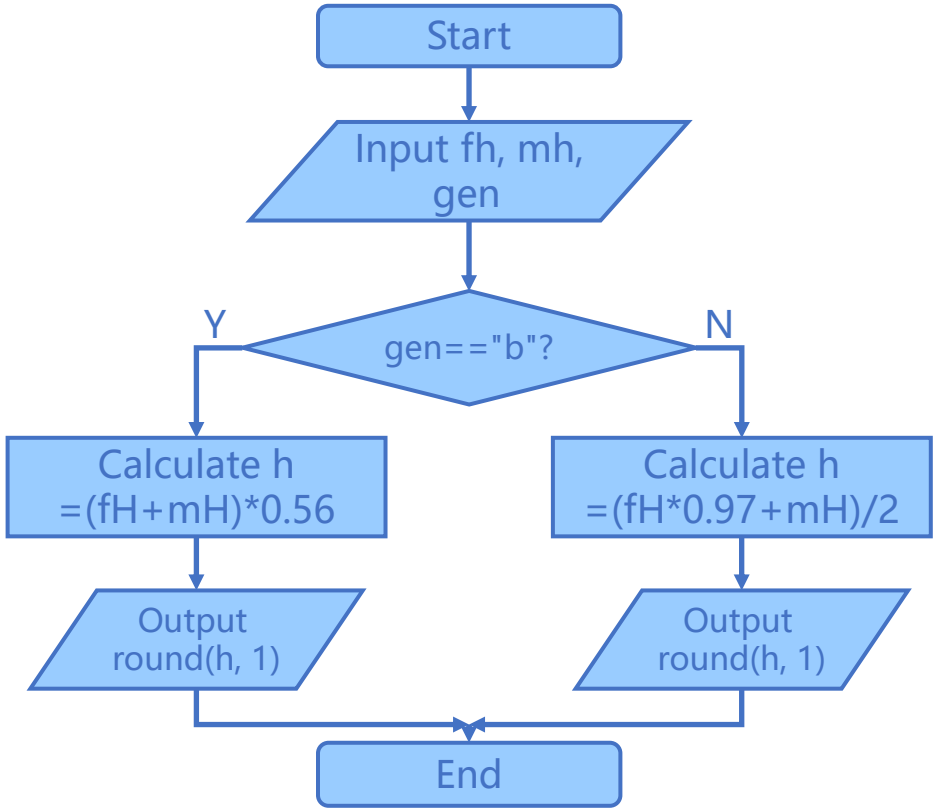
4.7 Exercises

【Exercise 4.4】The following formula can be used to predict the adult height of a child according to the genetic factors of parents: if fH is the height of the father and mH is the height of the mother, then the adult height of the =(fH+mH)*0.54 , and the adult height of the girl =(fH*0.97+mH)/2. Please return the predicted height according to the height of the parents and the gender of the child. The unit is cm, with one decimal place reserved.



4.7 Exercises

【Exercise 4.4】The following formula can be used to predict the adult height of a child according to the genetic factors of parents: if fH is the height of the father and mH is the height of the mother, then the adult height of the =(fH+mH)*0.54 , and the adult height of the girl =(fH*0.97+mH)/2. Please return the predicted height according to the height of the parents and the gender of the child. The unit is cm, with one decimal place reserved.



	A	B	C
1	177	157	/Input fH, mH
2	b		/Input gen
3	if A2=="b"	=(A1+B1)*0.54	/boys
4		>output(round(B3, 1))	/Output the predicted height of boys
5	else	=(A1*0.97+B1)/2	/girls
6		>output(round(B5, 1))	/Output the predicted height of girls

Result(example):

180.4

4.7 Exercises

【Exercise 4.5】Use if function to solve exercises 4.1 and 4.2.

4.1 Input two numbers and output them in descending order.

4.2 Input a character and output it after conversion. When converting, lower case letters need to be converted to upper case, and the others remain unchanged.

4.7 Exercises

【Exercise 4.5】Use if function to solve exercises 4.1 and 4.2.

4.1 Input two numbers and output them in descending order.

	A	B	C
1	10	15	/Input a, b
2	=if(A1>B1, A1/'', ''/B1, B1/'', ''/A1)		/Determine the order
3	>output(A2)		/Output result

Result(example):



4.2 Input a character and output it after conversion. When converting, lower case letters need to be converted to upper case, and the others remain unchanged.

	A	C
1	b	/Input character c
2	=if(A1>=''a''&&A1<=''z'', upper(A1), A1)	/Convert the character
3	>output(A2)	/Output result

Result(example):



4.7 Exercises

【Exercise 4.6】Read the following program, what is the output after execution? Why are C1 and C2 outputs different?

	A	B	C
1	>(a=1, b=1, c=1)	=(a=a+1)&&(b=b+1)||(c=c+1)	>output("a="/a/", b="/b/", c="/c)
2	>(a=1, b=1, c=1)	=(a=a+1)||(b=b+1)&&(c=c+1)	>output("a="/a/", b="/b/", c="/c)

4.7 Exercises

【Exercise 4.6】Read the following program, what is the output after execution? Why are C1 and C2 outputs different?

	A	B	C
1	>(a=1, b=1, c=1)	=(a=a+1)&&(b=b+1)||(c=c+1)	>output("a="/a/", b="/b/", c="/c)
2	>(a=1, b=1, c=1)	=(a=a+1)||(b=b+1)&&(c=c+1)	>output("a="/a/", b="/b/", c="/c)

Result:

```
a=2,b=2,c=1
a=2,b=1,c=1
```

4.7 Exercises

【Exercise 4.6】Read the following program, what is the output after execution? Why are C1 and C2 outputs different?

	A	B	C
1	>(a=1, b=1, c=1)	=(a=a+1)&&(b=b+1)||(c=c+1)	>output("a="/a/", b="/b/", c="/c)
2	>(a=1, b=1, c=1)	=(a=a+1)||(b=b+1)&&(c=c+1)	>output("a="/a/", b="/b/", c="/c)

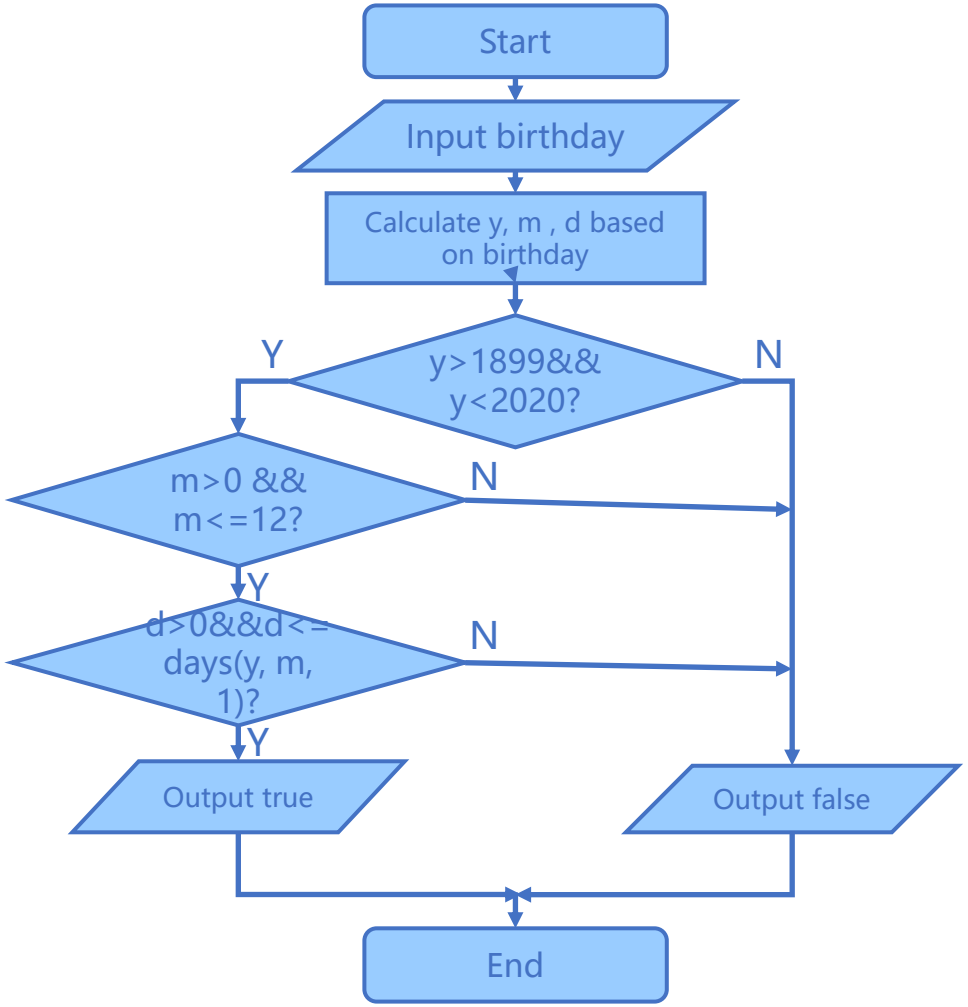
Result:

```
a=2,b=2,c=1
a=2,b=1,c=1
```

- AND (&&) take precedence over OR (||)
- The calculation of && and || does not need to be carried out in full. For example, to calculate a&&b , when a is false, b will not be calculated, and the result will be false. Similarly, to calculate a||b, when a is true, b will also not be calculated, and the result true will be obtained directly.
- When AND and OR are mixed in calculation:
  a&&b||c is equivalent to (a&&b)||c, a will be calculated first, If a is false, there is no need to calculate b, and the result of a&&b is false, only need to calculate c, and the result of c is the final result; If a is true, then there is a need to calculate b, if b is true, the result is true, if b is false, there is a need to calculate c. For example, in B1, the results of expressions a=a+1, b=b+1, c=c+1 are all integers, which is equivalent to true, so the result can be got after calculating the first two expressions. In this case, only expressions 1 and 2 are executed.
  a||b&&c is equivalent to a||(b&&c), a will be calculated first, If a is true, there is no need to calculate other expressions, and the final result is true; if a is false, then continue to calculate b&&c, first calculate b, only if b is true, there is a need to calculate c. In this example, the result of B2 can be got as true after calculating the first expression a=a+1 , and there is no need to calculate the other two expressions. So only the first expression a=a+1 is executed.

4.7 Exercises

【Exercise 4.7】Input an 8-digit birthday, such as 19800120 for January 20, 1980. It's required that the year of birth is greater than 1899 and less than 2020, judge whether the input date is true or false. You can use the mid() and days() functions. If you want to get the month in the string, you can use int(mid (birthday, 5, 2)) .To judge how many days are in February 1980, you can use days (date (1980, 2, 1)).



	A	B	C	D
1	'19900229			/Input birthday and covert it to a string
2	>y=int(mid(A1, 1, 4))	>m=int(mid(A1, 5, 2))	>d=int(mid(A1, 7, 2))	/Calculate y, m, d
3	if y>1899 && y<2020			/Whether year is true
4		if m>0 && m<13	=days(date(y, m, 1))	/Whether month is true
5			>output(d>0 && d<=C4)	/Whether date is true or not
6		else	>output(false)	/month is false, output false
7	else	>output(false)		/year is false, output false

Result(example):



Chapter 5
Loop structure programming

5.1 The need for loop structure

5.2 Implement loop structure with for x statement

- The general form of for x statement
- Loop of specified cycle number
- Break and next statements

5.3 For statement that specifies the scope of the loop

- for a, b statement
- for a, b, s statement

5.4 Nesting of loop structure

- Nesting of loop structure
- Using break and next in multi-layer loops

5.5 Examples of loop structure

5.6 Exercises

Chapter 5
Loop structure programming

5.1 The need for loop structure

5.1 The need for loop structure

Why do we need a loop structure?

- When solving a problem, it is often necessary to perform a specific series of calculations or operations repeatedly.
- When the program is repeatedly executed, the process is the same each time, but the data used or the specific operation performed may be different.
- The loop needs to be able to stop effectively.

Chapter 5

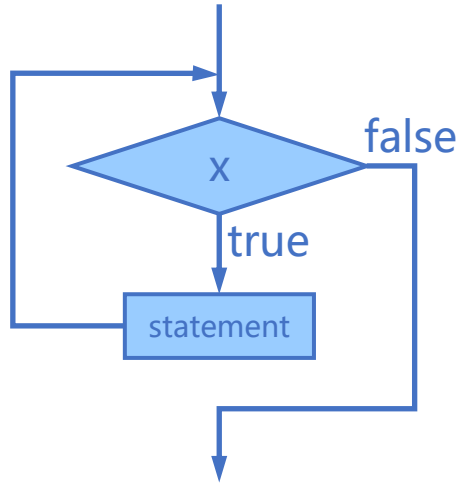
Loop structure programming

5.2 Implement loop structure with for x statement

The general form of for x statement

The form of the for x statement

- for x statement
  Other statements
- When the calculation result of expression x is true, execute the loop statement; otherwise, execute the following other statements.
- x is used to control whether the loop is executed, called loop condition. The statement executed during the loop can be a statement or a code block composed of multiple statements, which is called the loop body.

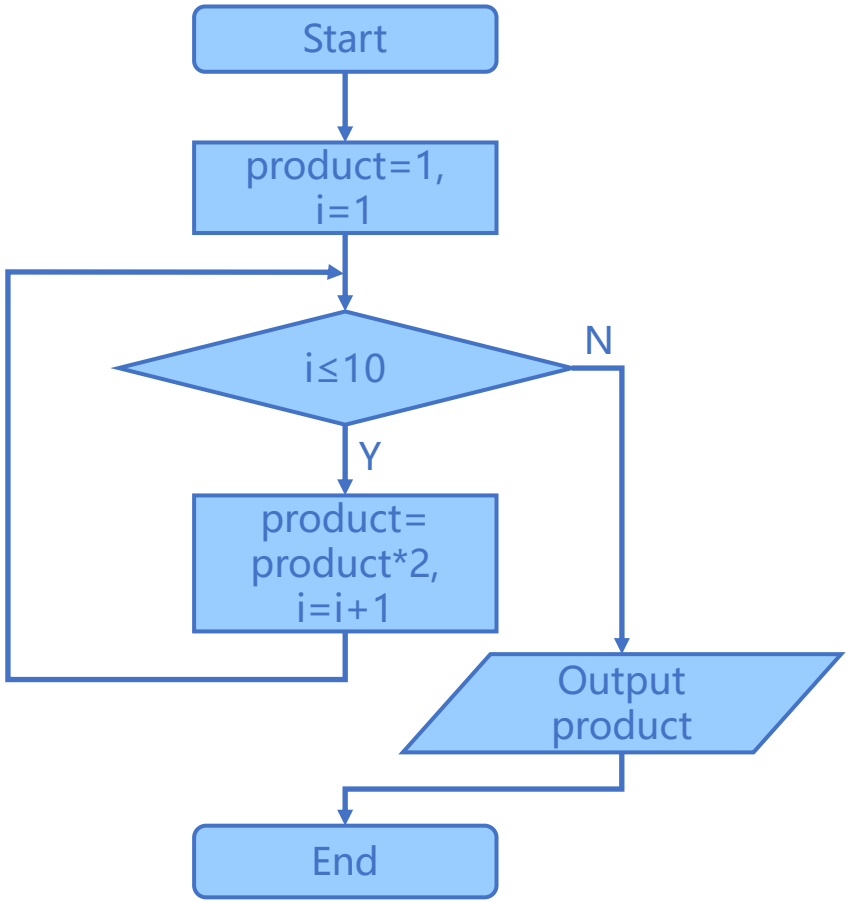


【Example 5.1】Find 2^{10}, that is, the result of 2×2×2×2×2×2×2×2×2×2.

5.2 Implement loop structure with for x statement

The general form of for x statement

【Example 5.1】Find 2^{10}, that is, the result of 2×2×2×2×2×2×2×2×2×2.



	A	B	C
1	1	1	/Initializing product and i
2	for B1<=10	>A1=A1*2	/If i≤10, calculate product in loop
3		>B1=B1+1	/Number of cycles i+1
4	>output("product="/A1)		/Output result

Result:

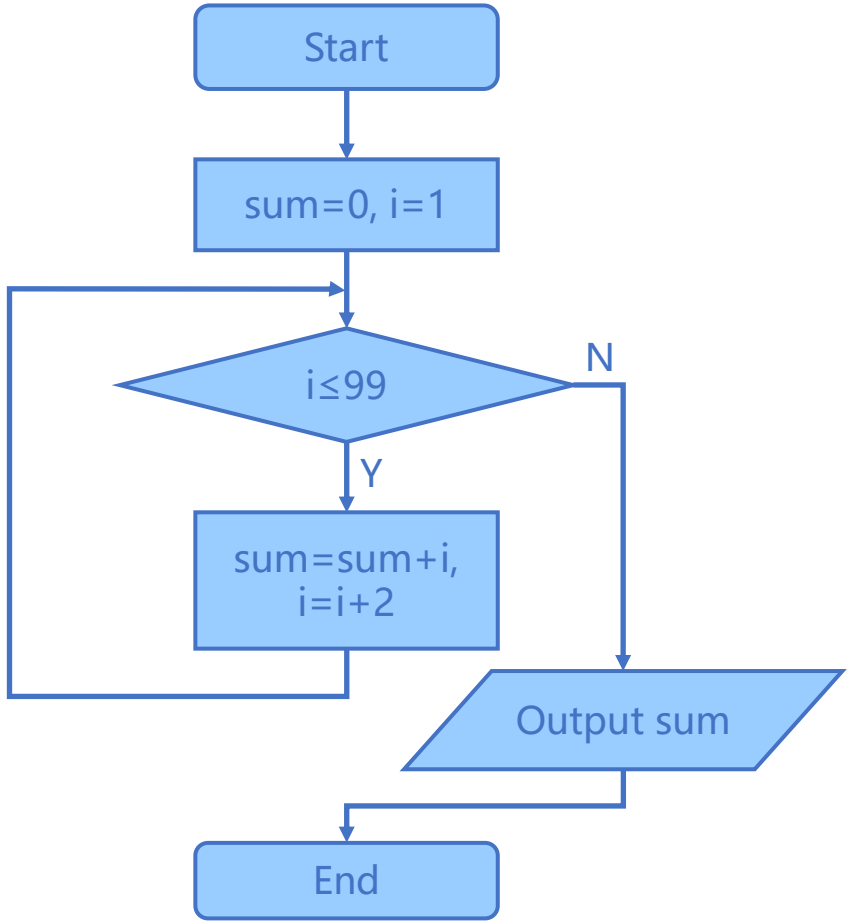
```
product=1024
```

Note

- If the loop body contains more than one statement, it should be placed in the code block of the for statement.
- Pay attention to assign initial values for product and i.
- After each execution of the loop, the value of i should be changed to change the condition of the loop so that the loop can end normally.

【Example 5.2】Calculate the result of 1+3+5+7+…+97+99.

【Example 5.2】Calculate the result of 1+3+5+7+…+97+99.



	A	B	C
1	0	1	/Initializing sum and i
2	for B1<=99	>A1=A1+B1	/If i≤99, calculate sum in loop
3		>B1=B1+2	/i+2
4	>output("sum="/A1)		/Output result

Result:

```
sum=2500
```

The execution of the for loop

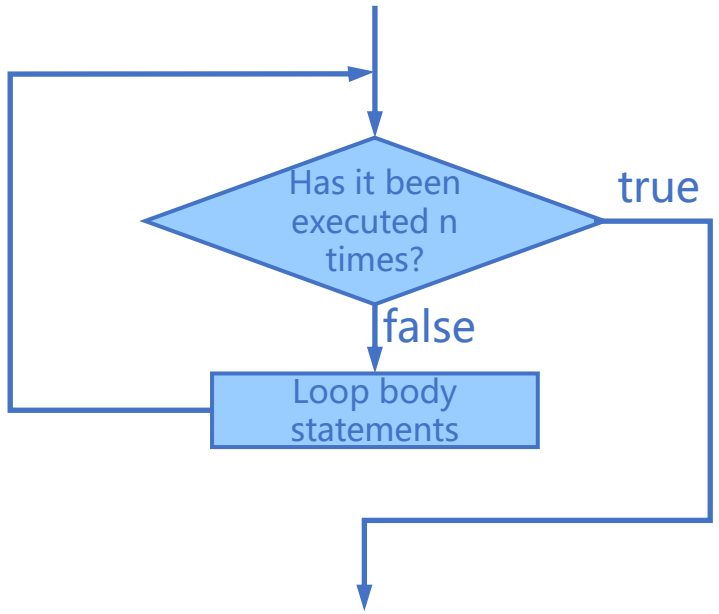
- First calculate the expression in the loop condition. If the result is true, execute the loop body. Otherwise, skip the whole loop body and continue to execute the statement following the loop structure.
- The loop body can be composed of one statement or a code block composed of multiple statements. After the statement in the loop body is executed, it will return to the for cell, recalculate the expression in the loop condition, and judge whether to execute the loop or not.
- In the loop body, it is necessary to execute the statement that makes the loop close to the end, such as changing the loop variable, otherwise the loop will execute repeatedly and never end, which is called dead loop.

5.2 Implement loop structure with for x statement

Loop of specified cycle number

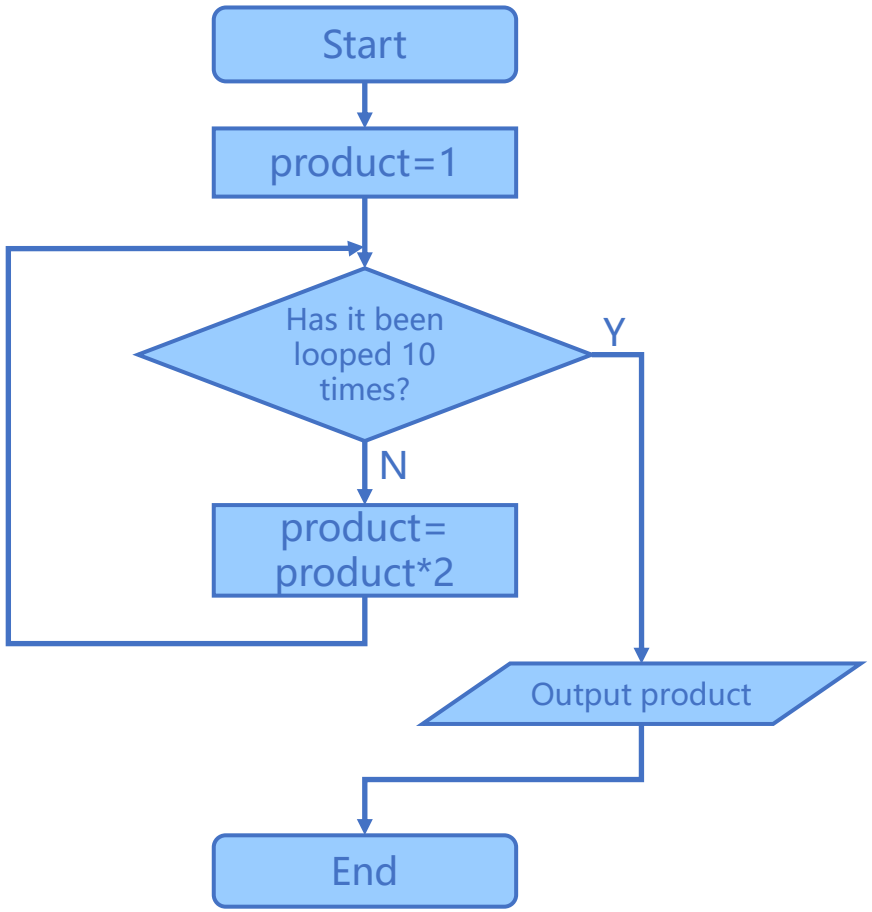
The form of the for n statement

- for n
  loop body statement
  Other statements
- The statements in the loop body will be executed n times, and other statements following the loop structure will be executed after the execution.
- n is a positive integer that specifies the number of cycles. When the loop is executed, the values of the main cell of the loop are from 1 to n.



【Example 5.3】Use loop structure of specified cycle number to calculate example 5.1:Find 2^{10}, that is, the result of 2×2×2×2×2×2×2×2×2×2.

【Example 5.3】Use loop structure of specified cycle number to calculate example 5.1:Find 2^{10}, that is, the result of 2×2×2×2×2×2×2×2×2×2.



	A	B	C
1	1		/Initializing product
2	for 10	>A1=A1*2	/If i≤10, calculate product in loop
3	>output("product="/A1)		/Output result

Result:

product=1024

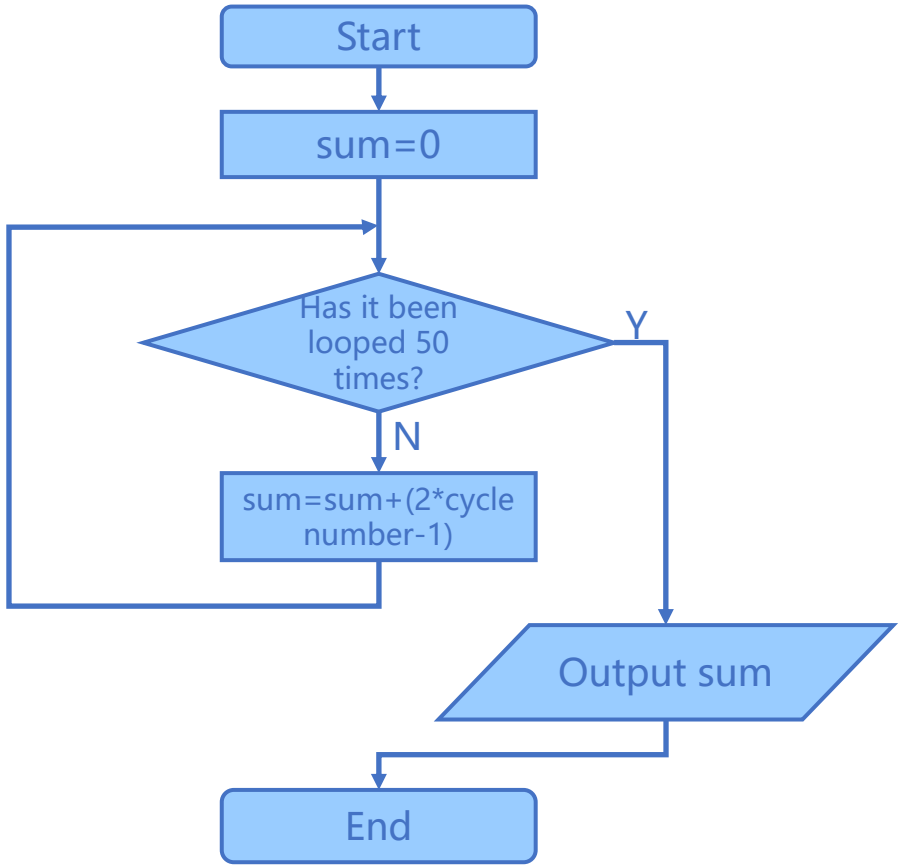
【Example 5.4】Use loop structure of specified cycle number to calculate example 5.2:Calculate the result of 1+3+5+7+…+97+99.

5.2 Implement loop structure with for x statement

【Example 5.4】Use loop structure of specified cycle number to calculate example 5.2:Calculate the result of 1+3+5+7+…+97+99.

Solution

The problem needs to calculate the sum of the first 50 odd numbers. The odd numbers used to add each time can be calculated from the number of cycles.



	A	B	C
1	0		/Initializing sum
2	for 50	>A1=A1+(2*A2-1)	/Calculate sum in loop
3	>output("sum="/A1)		/Output result

Result:

```
sum=2500
```

Note

- When a loop with specified cycle number is executed, the value of the loop main cell is the loop number each time.
- At the end of a loop with specified cycle number, the value of the loop main cell is the total number of cycles + 1.

Change the state of the loop



break

Early termination of loop

Jump out of the loop body and directly execute the statement after the loop structure.
The break statement can only be used in the code block of the for loop.



next

Early termination of current cycle

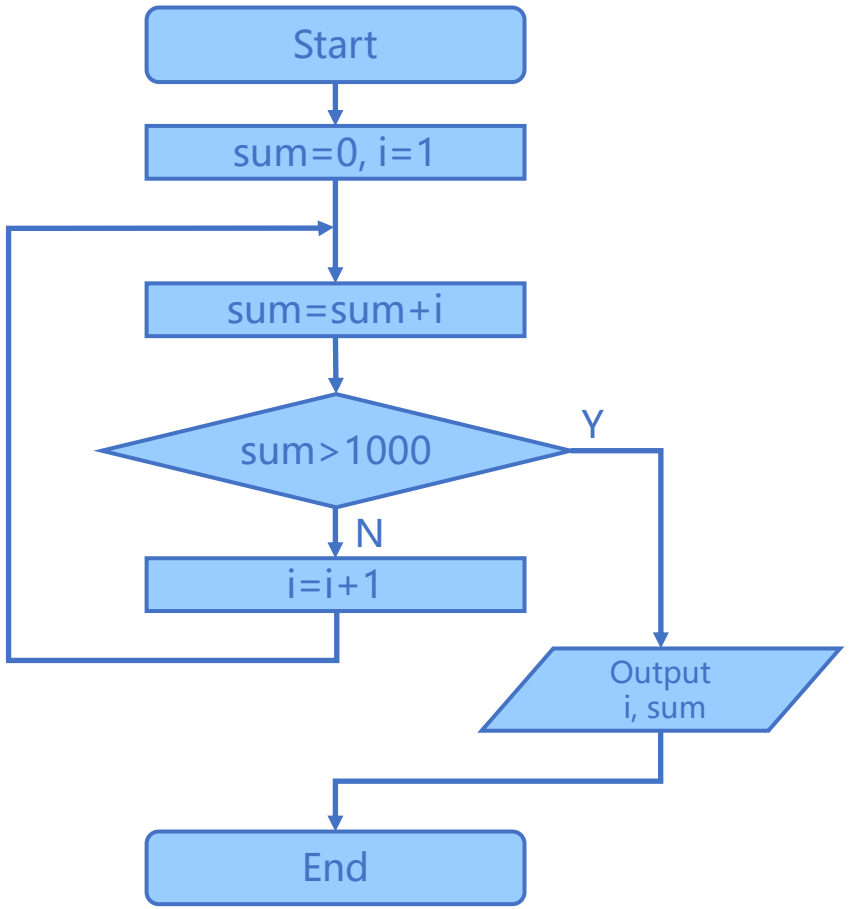
Skip the remaining statements in the loop body and return to the loop judgment statement to determine whether to execute the next loop.
The next statement can only be used in the code block of the for loop.



【Example 5.5】Calculate the sum of continuous positive integers 1+2+3+4+……, when the sum is just over 1000, what's the number just added?

5.2 Implement loop structure with for x statement

【Example 5.5】Calculate the sum of continuous positive integers 1+2+3+4+....., when the sum is just over 1000, what's the number just added?



	A	B	C	D
1	0	1		/Initializing sum and i
2	for	>A1=A1+B1		/Loop, and calculate sum
3		if A1>1000	break	/When sum is just over 1000, end loop
4		>B1=B1+1		/i+1
5	>output("Add to "/B1/". The sum is "/A1/".")			/Output i and sum

Result:

```
Add to 45. The sum is 1035.
```

Note

- When you do not add a condition to a for loop, the loop always executes, which is called an endless loop.
- When executing the endless loop, the break statement must be used in the loop body to ensure the effective termination of the loop.



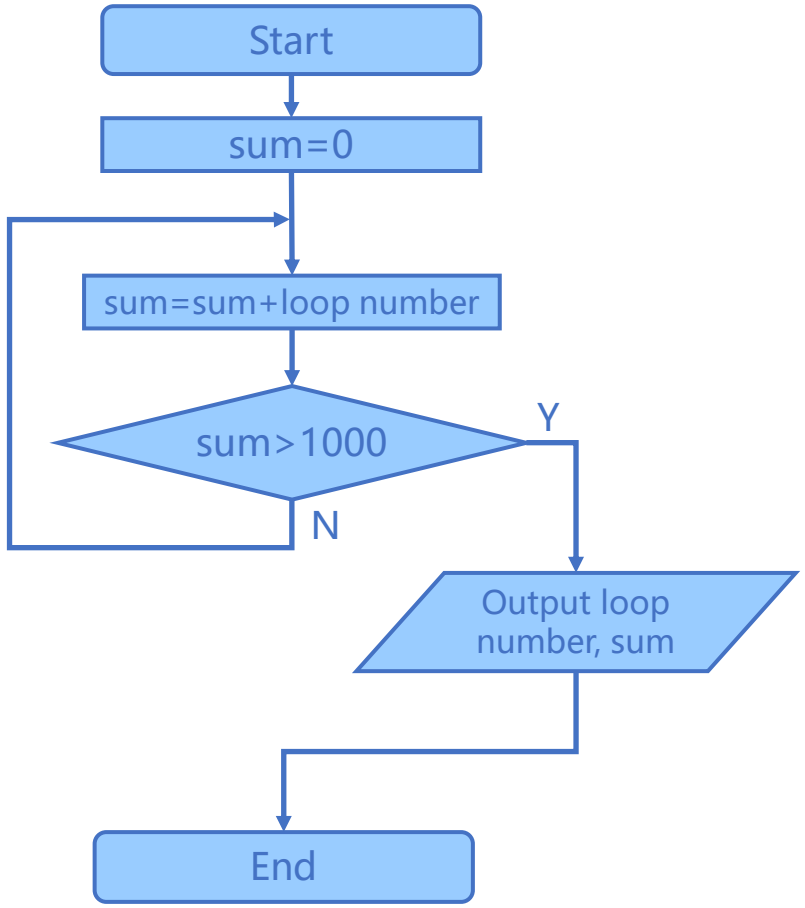
【Example 5.5】Calculate the sum of continuous positive integers 1+2+3+4+....., when the sum is just over 1000, what's the number just added?



Even if you use for to execute an endless loop, you can still get the number of loops in the main cell of the loop, and the code can be simplified accordingly.

【Example 5.5】Calculate the sum of continuous positive integers 1+2+3+4+....., when the sum is just over 1000, what's the number just added?

Even if you use for to execute an endless loop, you can still get the number of loops in the main cell of the loop, and the code can be simplified accordingly.



	A	B	C	D
1	0			/Initializing sum
2	for	>A1=A1+A2		/Loop, and calculate sum
3		if A1>1000	break	/When sum is just over 1000, end loop
4	>output("Add to "/A2/". The sum is "/A1/".")			/Output I and sum

Result:

```
Add to 45. The sum is 1035.
```

【Example 5.6】Use the next statement to calculate 1+3+5+7+…+97+99.

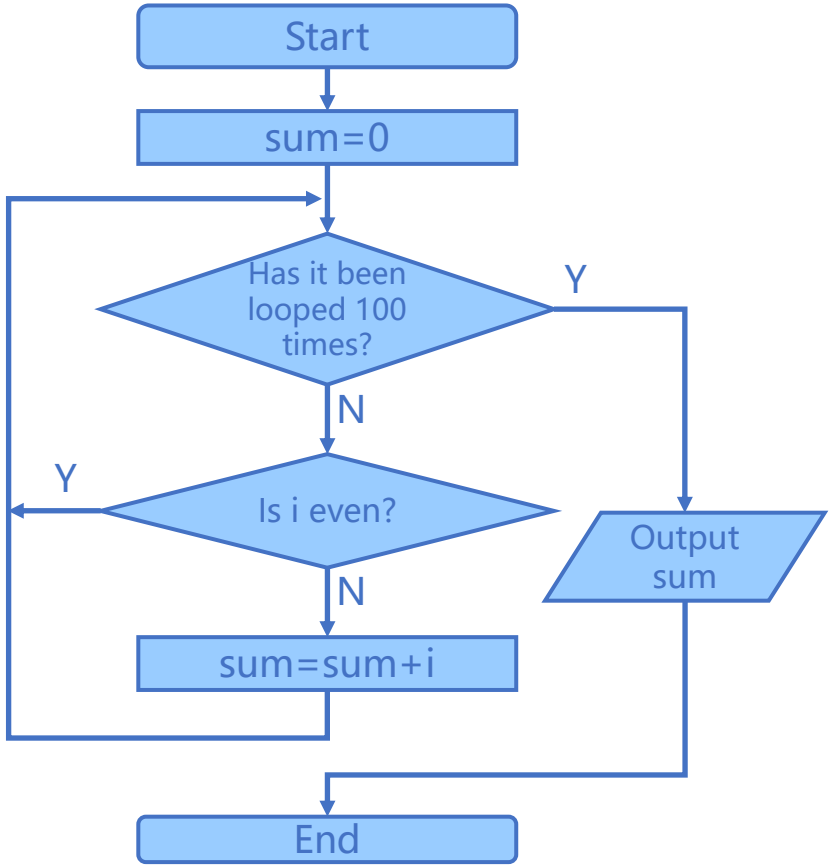
Solution

Loop 100 times, use loop number directly as the addend i. If i is even, skip the current loop and take the next number directly for judgement. If i is odd, it is accumulated to sum.

【Example 5.6】Use the next statement to calculate 1+3+5+7+...+97+99.

Solution

Loop 100 times, use loop number directly as the addend i. If i is even, skip the current loop and take the next number directly for judgement. If i is odd, it is accumulated to sum.



	A	B	C	D
1	0			/Initializing sum
2	for 100	if A2%2==0	next	/If i is even, execute the next loop
3		>A1=A1+A2		/If i is odd, add it to sum
4	>output("The sum is "/A1/".")			/Output sum

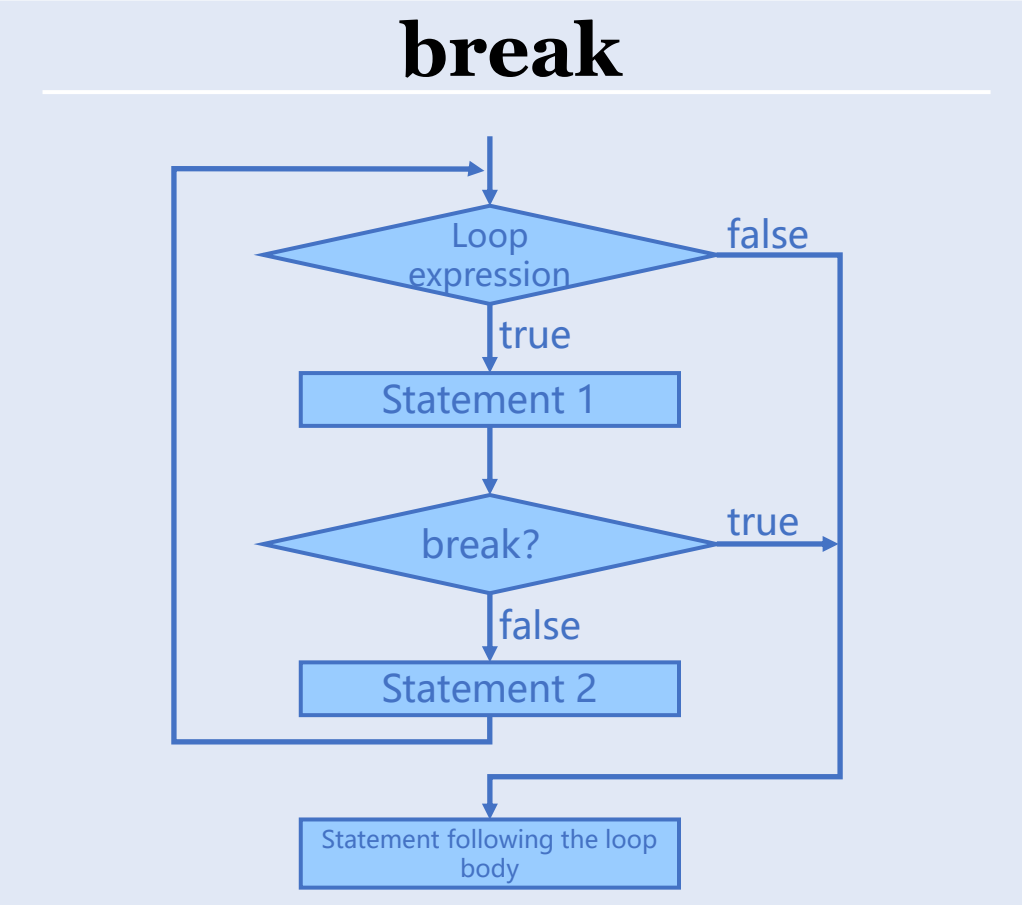
Result:

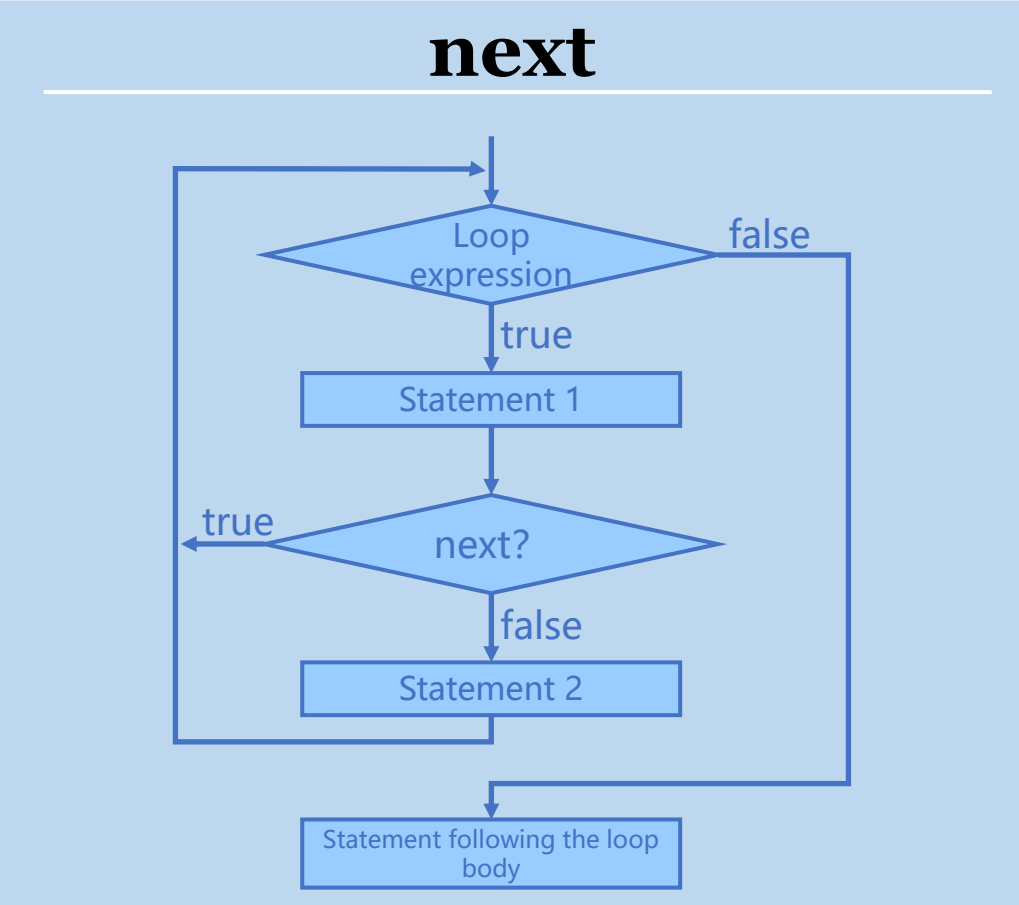
```
The sum is 2500.
```

Note

- The next statement is used to terminate the current loop and return to the main cell of the loop to execute the next loop. The break statement is used to terminate all loops and exit the loop directly to execute the following statements.

Comparison of break and next statements





Chapter 5

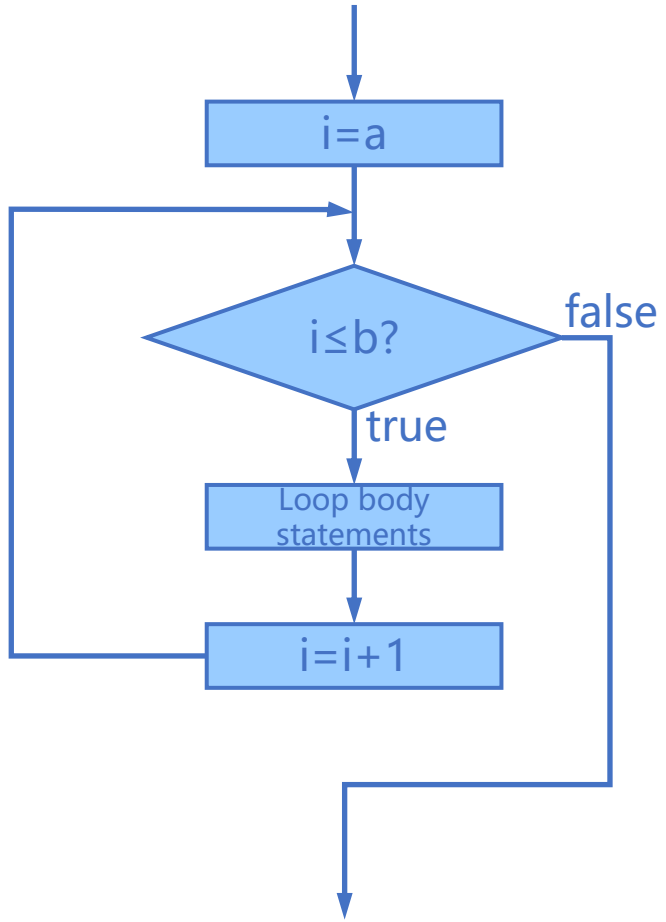
Loop structure programming

5.3 For statement that specifies the scope of the loop

for a, b statement

The form of for a, b statement

- for a, b loop body statements
  Other statements
- When executing the loop, the value of the main cell of the loop is from a to b, plus 1 each time.
- When the value of the main cell of the loop is increased to greater than b, the execution of the loop is completed, and other statements following the loop structure will be executed.



【Example 5.7】Find all 3-digit numbers that the sum of each digit is 10.

Solution

Loop through all 3-digit numbers from 100 to 999, judge whether the sum of each digit is 10, record the qualified 3-digit numbers and output after the loop is completed.

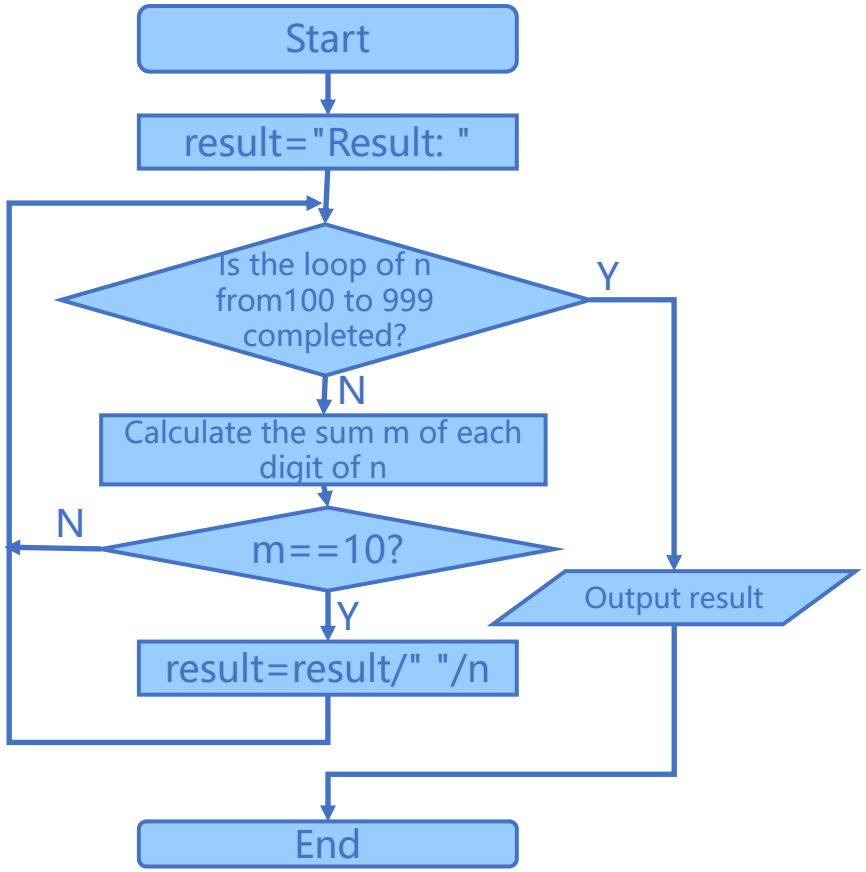
5.3 For statement that specifies the scope of the loop

for a, b statement

【Example 5.7】Find all 3-digit numbers that the sum of each digit is 10.

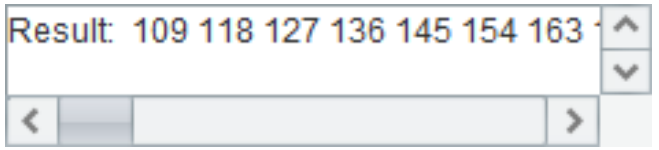
Solution

Loop through all 3-digit numbers from 100 to 999, judge whether the sum of each digit is 10, record the qualified 3-digit numbers and output after the loop is completed.



	A	B	C	D
1	Result:			/Record the output with A1
2	for 100, 999	=A2%10+(A2\10)%10+A2\100		/Calculate the sum m of each digit of n
3		if B2==10	>A1=A1/" "/A2	/if m is 10, record the current number
4	>output(A1)			/Output A1 as result

Result:



Note

- \ is an exact division operator when the operands are all integers.

The form of for a, b, s statement

- for a, b, s loop body statements
  Other statements
- When executing the loop, the value of the main cell of the loop is from a to b, plus s each time.
- When the value of the main cell of the loop is increased to greater than b, the execution of the loop is completed, and other statements following the loop structure will be executed.

【Example 5.8】Use the first 500, 000 items of the formula $\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right)$ to calculate the approximate value of π.

Solution

The positive and negative signs of each item in the formula are rotated, and the denominators are 1, 3, 5, ... The denominator of item 500000 is 999999.

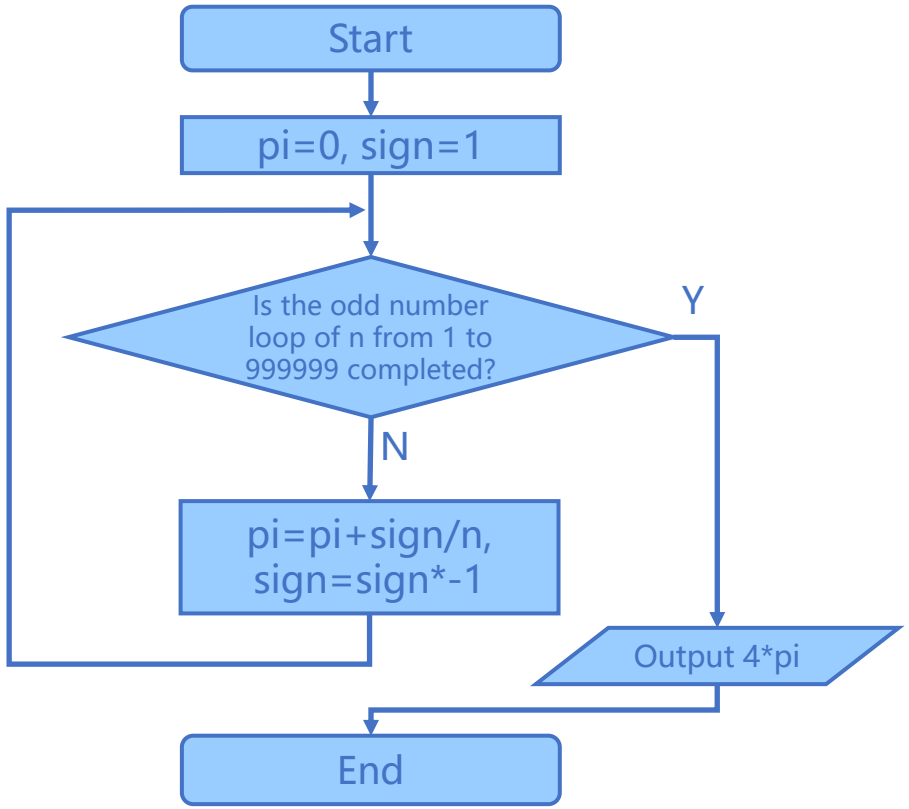
5.3 For statement that specifies the scope of the loop

for a, b, s statement

【Example 5.8】Use the first 500, 000 items of the formula $\pi = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots\right)$ to calculate the approximate value of π.

Solution

The positive and negative signs of each item in the formula are rotated, and the denominators are 1, 3, 5, ... The denominator of item 500000 is 999999.



	A	B	C
1	0	0	/Initializing pi, sign
2	for 1, 999999, 2	>A1=A1+B1/A2	/Use formula to accumulate items
3		>B1=B1*(-1)	/Transform signs
4	>output("π≈"/(A1*4))		/Output the approximate value of π

Result:

π≈3.141590653589692

Note

- for a, b, s will end when the loop value is greater than b, if it is for 1, 1000000, 2 in A2, the execution result is the same.

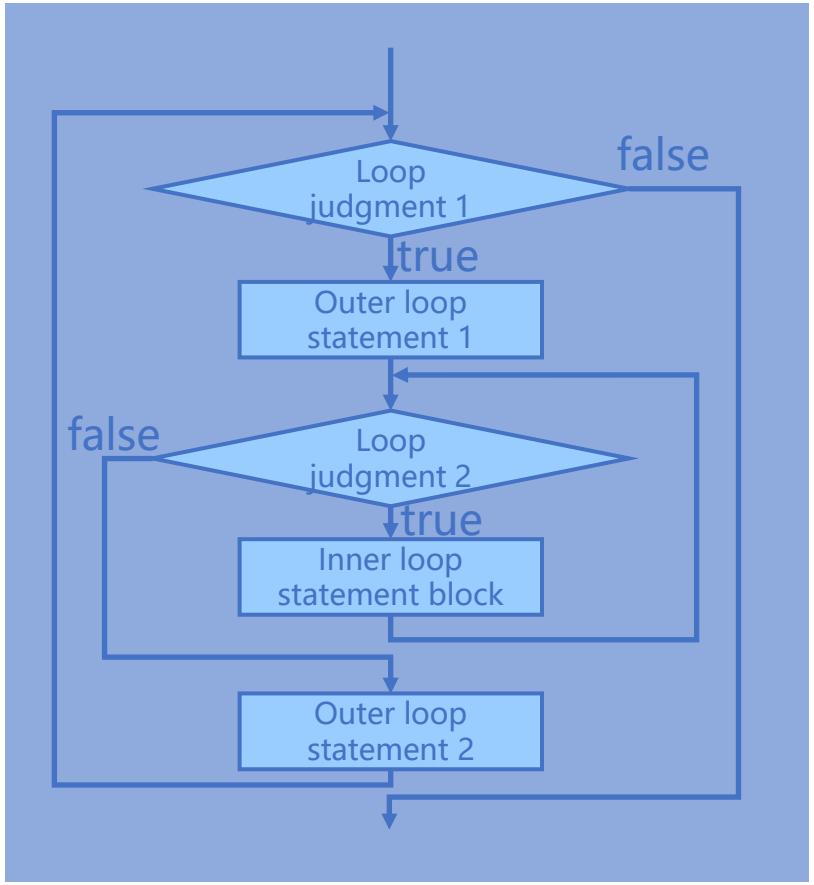
Chapter 5
Loop structure programming

5.4 Nesting of loop structure

5.4 Nesting of loop structure

	A	B	C	D
1	for …	Outer loop statement 1		/Outer loop statement
2		for …	Inner loop statement 1	/Inner loop statement
3			Inner loop statement 2	/Inner loop statement
4		Outer loop statement 2		/Outer loop statement
5	Other statement			/Statement out of loop body

- Nesting loop is also known as multi-layer loop. In each execution of the outer loop, the entire inner loop needs to be executed.
- After the outer loop has been executed, the statement after the loop structure is executed.



【Example 5.9】Output Fibonacci sequence, one line per 6 items, output 5 lines. The first two items of Fibonacci sequence are all 1. From the third item, it is the sum of the previous two items, that is, 1, 1, 2, 3, 5, 8, ...

Solution

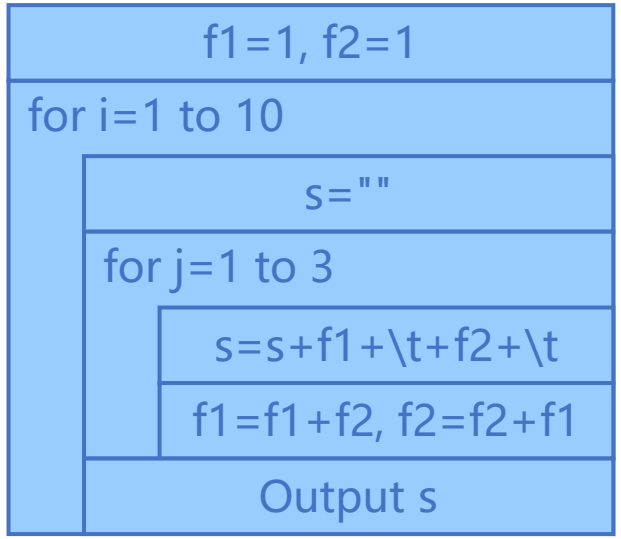
The outer loop outputs one line at a time. The inner loop circulates three times per line, recording two items each time, and calculating the values of the next two items.

5.4 Nesting of loop structure

【Example 5.9】Output Fibonacci sequence, one line per 6 items, output 5 lines. The first two items of Fibonacci sequence are all 1. From the third item, it is the sum of the previous two items, that is, 1, 1, 2, 3, 5, 8, ...

Solution

The outer loop outputs one line at a time. The inner loop circulates three times per line, recording two items each time, and calculating the values of the next two items.



	A	B	C	D	E
1	1	1			/Initializing the first two items
2	for 5	=""			/Initializing output string
3		for 3	B2=B2/A1/"\t"/B1/"\t"		/Add items to the output string
4			>A1=A1+B1	>B1=B1+A1	/Calculate the following two items
5		>output(B2)			/Output one line

Result:

1	1	2	3	5	8
13	21	34	55	89	144
233	377	610	987	1597	2584
4181	6765	10946	17711	28657	46368
75025	121393	196418	317811	514229	832040

Using break and next in multi-layer loops

- When you use break and next, you can specify the loop main cell C. By using break C and next C, indicate which layer terminates or enters the next loop.
- When C is omitted, break and next statements control the innermost loop of the statement.

【Example 5.10】Output the first 50 prime numbers greater than 200.

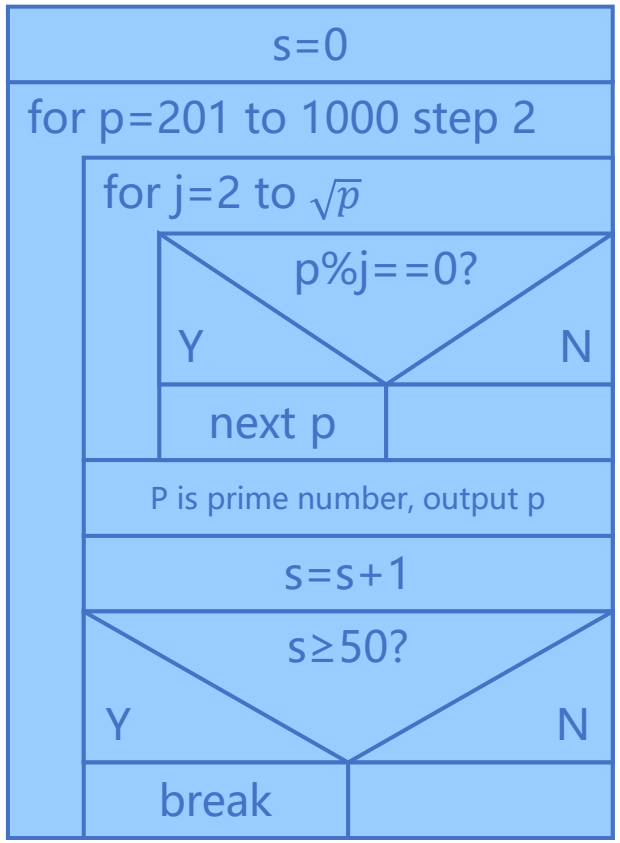
Solution

The prime number p needs to satisfy: it cannot be divisible by all integers from 2 to $\sqrt{p}$. Judge whether an odd number is a prime number starting from 201.

【Example 5.10】Output the first 50 prime numbers greater than 200.

Solution

The prime number p needs to satisfy: it cannot be divisible by all integers from 2 to $\sqrt{p}$. Judge whether an odd number is a prime number starting from 201.



	A	B	C	D	E
1	0				/A1 records the number of prime numbers
2	for 201,1000, 2	=sqrt(A2)			/Loop, calculate sqrt
3		for 2, B2	if A2%B3==0	next A2	/composite number, judge next
4		>output(A2)	>A1=A1+1		/prime number, output and count
5		if A1>=50	break		/Output completed, end

Result:

```
479
487
491
499
503
```

Chapter 5
Loop structure programming

5.5 Examples of loop structure

【Example 5.11】Given two positive integers, find their greatest common divisor and least common multiple.

Solution

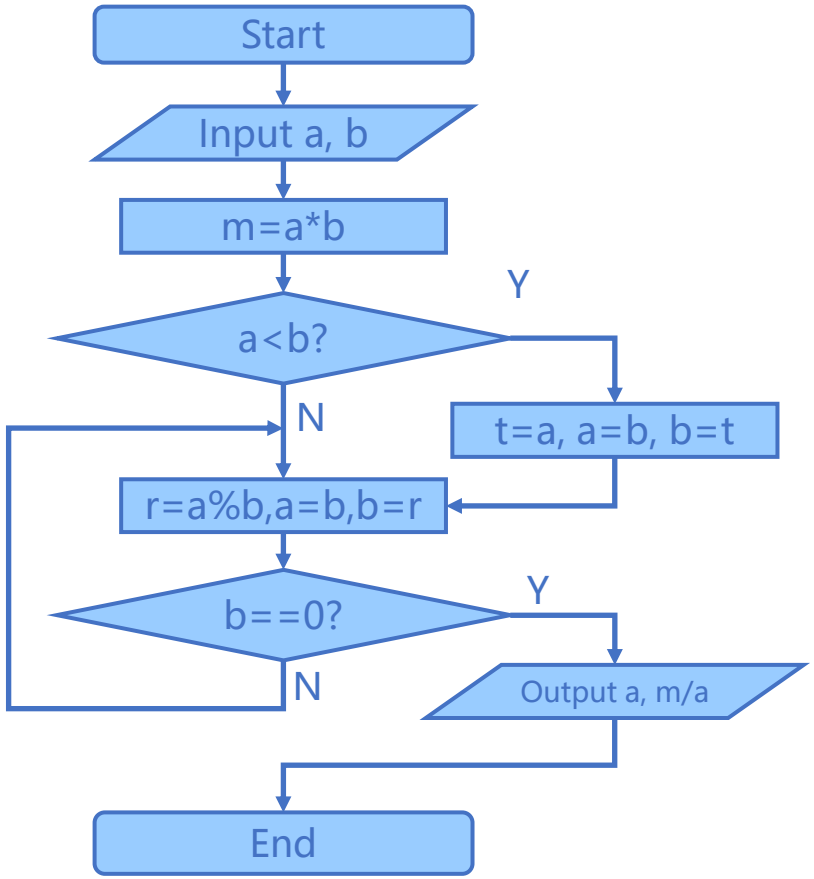
Using the division algorithm, the GCD of two positive integers is equal to the GCD of the remainder of their division and the smaller integer. Divide in loop until the smaller number is the divisor of the larger number. The multiplication of the LCM and the GCD is equal to the product of two numbers.

5.5 Examples of loop structure

【Example 5.11】Given two positive integers, find their greatest common divisor and least common multiple.

Solution

Using the division algorithm, the GCD of two positive integers is equal to the GCD of the remainder of their division and the smaller integer. Divide in loop until the smaller number is the divisor of the larger number. The multiplication of the LCM and the GCD is equal to the product of two numbers.



	A	B	C	D
1	144	96	=A1*B1	/Input two numbers, and calculate their product.
2	if A1<B1	>(t=A1, A1=B1, B1=t)		/If a<b, exchange two numbers
3	for B1!=0	>(r=A1%B1, A1=B1, B1=r)		/division algorithm
4	>output("GCD:"/A1/", LCM:"/(C1\A1))			/Output GCD and LCM

Result: GCD:48, LCM:288

Note

- Bracket operator: $(x_1, x_1, \cdots)$ can calculate $x_1, x_1, \cdots$ in turn, which can effectively simplify the code. Use \ operator to calculate integer division.

【Example 5.12】A frog in the well climbs up 5 feet in the daytime and 2 feet down at night. The well is 20 feet high. How many days can the frog climb out of the well?

Solution

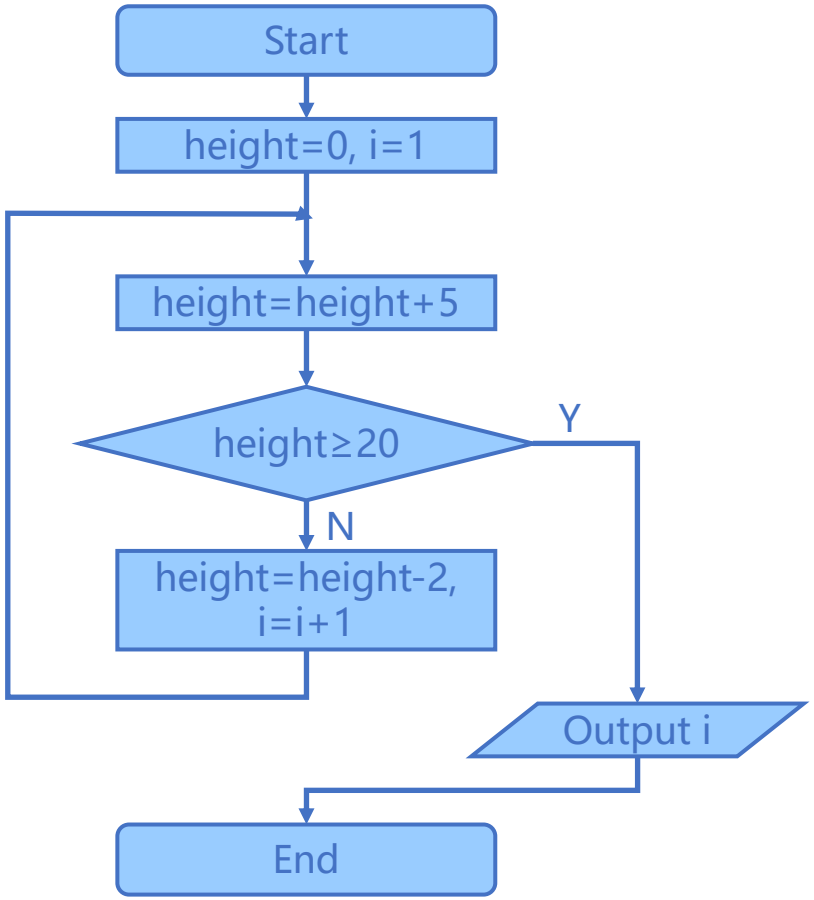
Calculate in loop, judge whether the frog has climbed out of the well in the daytime each time.

5.5 Examples of loop structure

【Example 5.12】A frog in the well climbs up 5 feet in the daytime and 2 feet down at night. The well is 20 feet high. How many days can the frog climb out of the well?

Solution

Calculate in loop, judge whether the frog has climbed out of the well in the daytime each time.



	A	B	C	D
1	0	1		/Initializing height and i
2	for	>A1=A1+5		/Always circulate when not climbing out of the well
3		if A1>=20	break	/Terminate the loop if the frog climb out of the well in the daytime
4		>A1=A1-2	>B1=B1+1	/Slide down at night, days i+1
5	>output("Cost "/B1/" days.")			/Output days

Result:

```
Cost 6 days.
```

【Example 5.13】Find all 4-digit square numbers of AABB type.

Solution

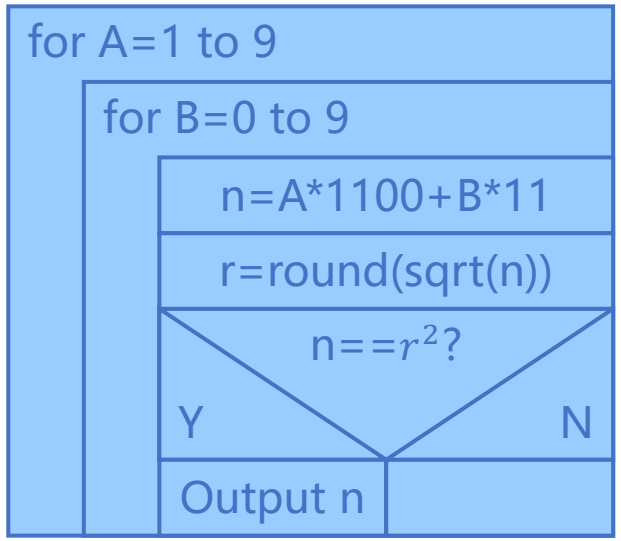
Loop A through 1 ~ 9, loop B through 0 ~ 9, combine into 4-digits AABB, and then judge whether it is square or not. When judging, there may be errors in the calculation of double precision numbers, so we need to use round function to round the square root, and then check whether the square is equal to the original number.

5.5 Examples of loop structure

【Example 5.13】Find all 4-digit square numbers of AABB type.

Solution

Loop A through 1 ~ 9, loop B through 0 ~ 9, combine into 4-digits AABB, and then judge whether it is square or not. When judging, there may be errors in the calculation of double precision numbers, so we need to use round function to round the square root, and then check whether the square is equal to the original number.



	A	B	C	D	E
1	for 9	for 0, 9	=A1*1100+B1*11		/Loop through all possible AABB
2			=round(sqrt(C1))		/Calculate the rounded square foot
3			if C2*C2==C1	>output(C1)	/If AABB is square, output it

Result:

```
7744
```

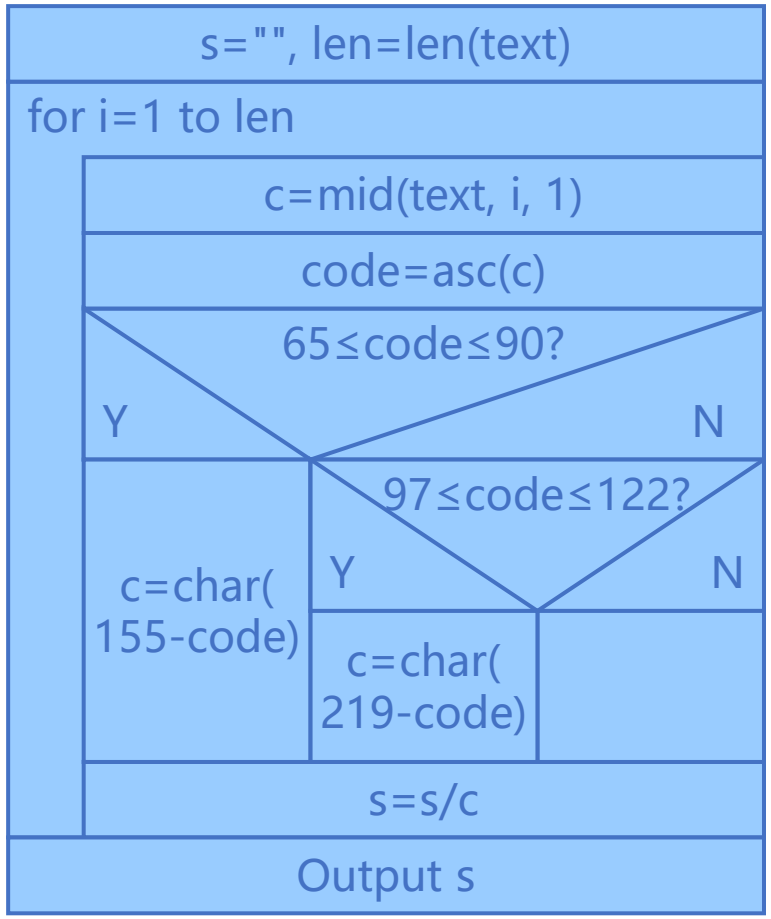
【Example 5.14】Take a paragraph of English as a parameter to input, and perform simple encryption conversion on it: convert a to z, b to y , …, z to a, A to Z, B to Y , Z to A, and other characters remain unchanged. Output encrypted text. (if you need to translate, you can input the encrypted text as a parameter and perform the conversion again.)

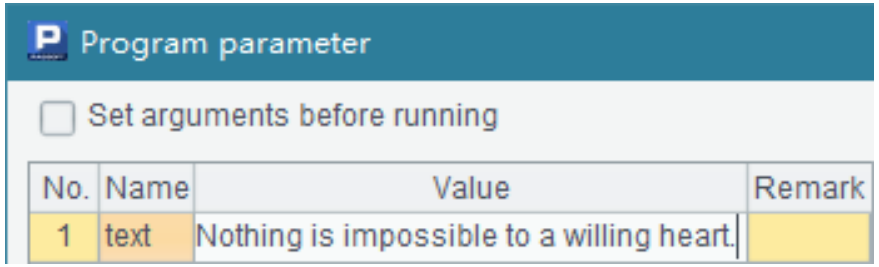
Solution

The ASCII codes of upper case letters from A to Z are 65 ~ 90, and those of lower case letters from a to z are 97 ~ 122. In this way, after transcoding, the ASCII code of a lower case letter is 90 + 65-n, and the ASCII code of the upper case letter is 97 + 122-n.

5.5 Examples of loop structure

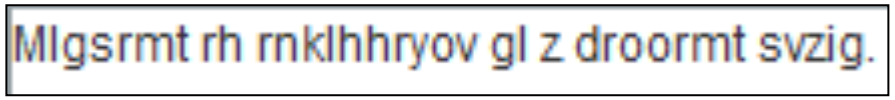
【Example 5.14】Take a paragraph of English as a parameter to input, and perform simple encryption conversion on it: convert a to z, b to y , …, z to a, A to Z, B to Y , Z to A, and other characters remain unchanged. Output encrypted text. (if you need to translate, you can input the encrypted text as a parameter and perform the conversion again.)





	A	B	C	D
1		=len(text)		/Initializing text and number of characters
2	for B1	=mid(text, A2, 1)	=asc(B2)	/Get a single character and calculate its ASCII
3		if C2>=65&&C2<=90	>B2=char(155-C2)	/encrypt upper case letters
4		else if C2>=97&&C2<=122	>B2=char(219-C2)	/encrypt lower case letters
5		>A1=A1/B2		/Add characters to result
6	>output(A1)			/Out encrypted string

Result:



Chapter 5
Loop structure programming

5.6 Exercises

5.6 Exercises

【Exercise 5.1】Enter a positive integer n (such as 10) and calculate its factorial n!.

【 Exercise 5.2】Find all the 3-digit Narcissus numbers: the sum of the cubes of all the digits is equal to itself.

【 Exercise 5.3】Calculate the square root of 1234 by the iterative method. The iterative formula of getting $x = \sqrt{a}$ is :$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$, the error between the two items should be less than 1^{-10}.

【 Exercise 5.4】There is a problem of "one hundred monks and one hundred buns" in the book suanfatongzong written by Cheng Dawei in the Ming Dynasty: there are one hundred buns and one hundred monks, each elder monk will have 3 buns, and every three younger monks will have one bun, what's the possible number of elder and younger monks?

【 Exercise 5.5】There are four suspects in a theft trial. A said, "it's not me. It's B who stole it." B said, "the criminal is either C or D." C said, "what A said is right." D said, "The criminal is among the three of them, " If two out of four people told the truth and two told lies, and there was only one criminal, who was the criminal?

【 Exercise 5.6】Input a piece of text and count the number of English letters and non English letters. For example, input:

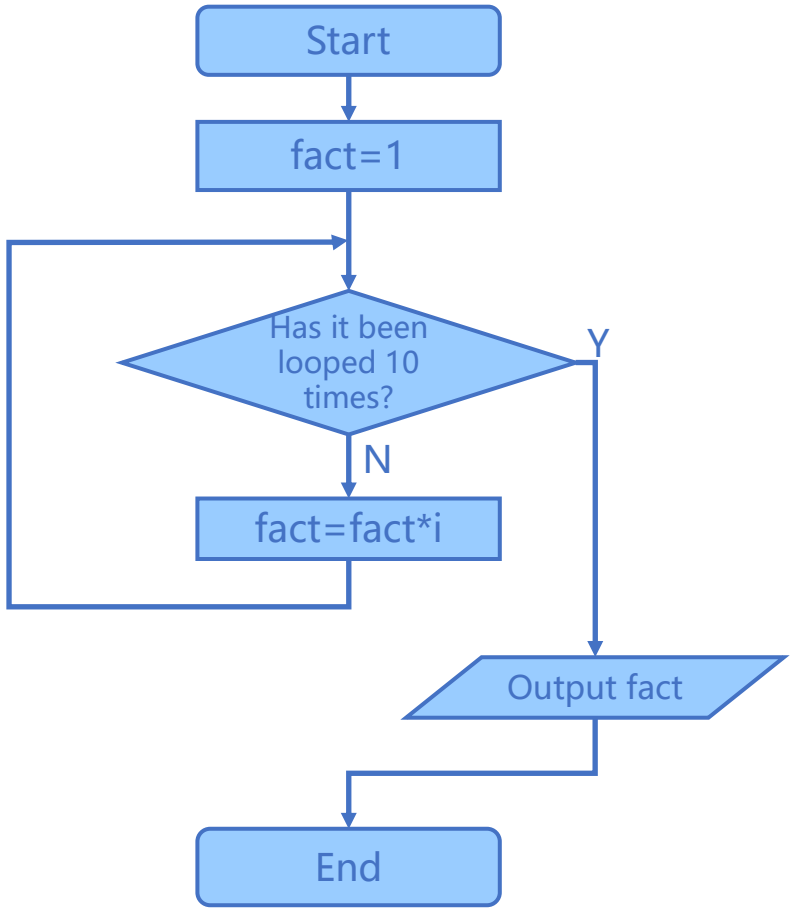
One man's fault is another man's lesson.

【 Exercise 5.7】January 1, 2020 is Wednesday, please output the calendar of the first three months. In a monthly calendar, each week is a line and is displayed from Sunday. You can use the days function to calculate the number of days in a month. For example, to calculate the number of days in March 2020, you can use days(date(2020, 3, 1)).

【Exercise 5.1】Enter a positive integer n (such as 10) and calculate its factorial n!.

5.6 Exercises

【Exercise 5.1】Enter a positive integer n (such as 10) and calculate its factorial n!.



	A	B	C
1	10	1	/Define n, and initialize fact
2	for A1	>B1=B1*A2	/Calculate factorial
3	>output(A1/"!="/B1)		/Output result

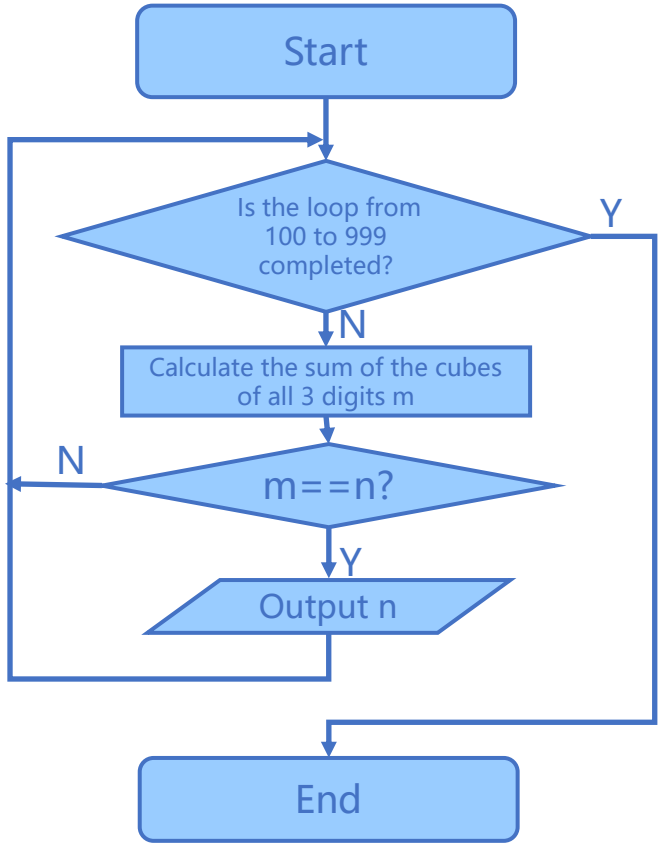
Result:

```
10!=3628800
```

【 Exercise 5.2】Find all the 3-digit Narcissus: the sum of the cubes of all the digits is equal to itself.

5.6 Exercises

【Exercise 5.2】Find all the 3-digit Narcissus: the sum of the cubes of all the digits is equal to itself.



	A	B	C	D
1	for 100, 999	=A1%10	=(A1\10)%10	/Calculate all 3 digits of n
2		=A1\100	=B1*B1*B1+C1*C1*C1+B2*B2*B2	/Calculate the sum of the cubes of all 3 digits
3		if C2==A1	>output(A1)	/If it is a Narcissus, output it

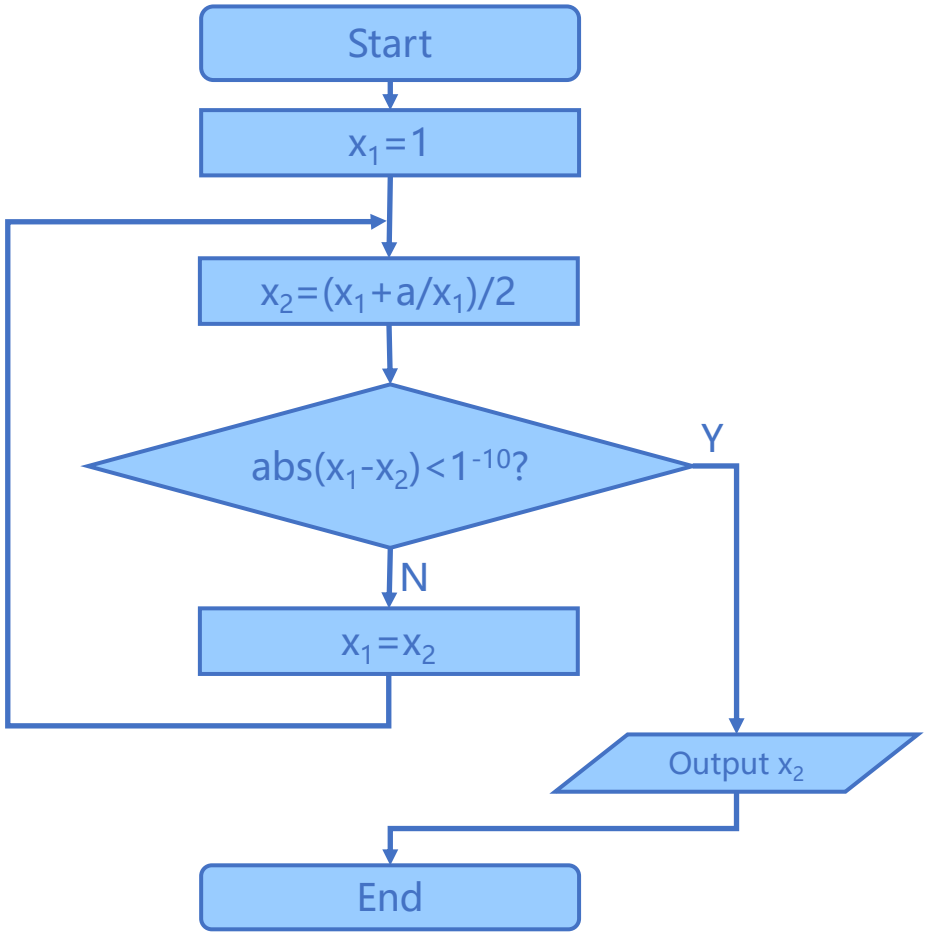
Result:

```
153
370
371
407
```

【 Exercise 5.3】Calculate the square root of 1234 by the iterative method. The iterative formula of getting $x = \sqrt{a}$ is :$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$, the error between the two items should be less than 1^{-10}.

5.6 Exercises

【Exercise 5.3】Calculate the square root of 1234 by the iterative method. The iterative formula of getting $x = \sqrt{a}$ is :$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$, the error between the two items should be less than 1^{-10}.



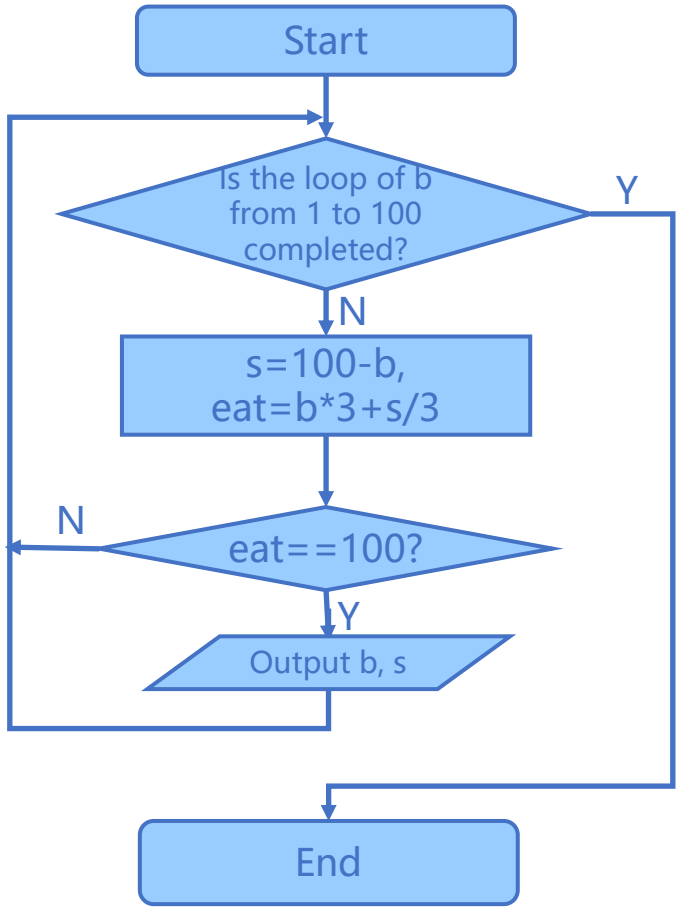
	A	B	C	D
1	1234	1e-10	1	/Define n, precision and initial value
2	for	=(C1+A1/C1)/2		/Loop the iterative formula
3		if abs(C1-B2)<B1	break	/If the error satisfies the condition, end
4		>C1=B2		/Update the square root
5	>output("root="/B 2)			/Output result

Result: root=35.12833614050059

【 Exercise 5.4】There is a problem of "one hundred monks and one hundred buns" in the book suanfatongzong written by Cheng Dawei in the Ming Dynasty: there are one hundred buns and one hundred monks, each elder monk will have 3 buns, and every three younger monks will have one bun, what's the possible number of elder and younger monks?

5.6 Exercises

【Exercise 5.4】There is a problem of "one hundred monks and one hundred buns" in the book suanfatongzong written by Cheng Dawei in the Ming Dynasty: there are one hundred buns and one hundred monks, each elder monk will have 3 buns, and every three younger monks will have one bun, what's the possible number of elder and younger monks?



	A	B	C	D
1	for 100	=100-A1	=A1*3+B1/3	/Loop, calculate the number of monks and buns
2		if C1==100	>output(A1/'', ''/B1)	/If the condition is met, output result.

Result:

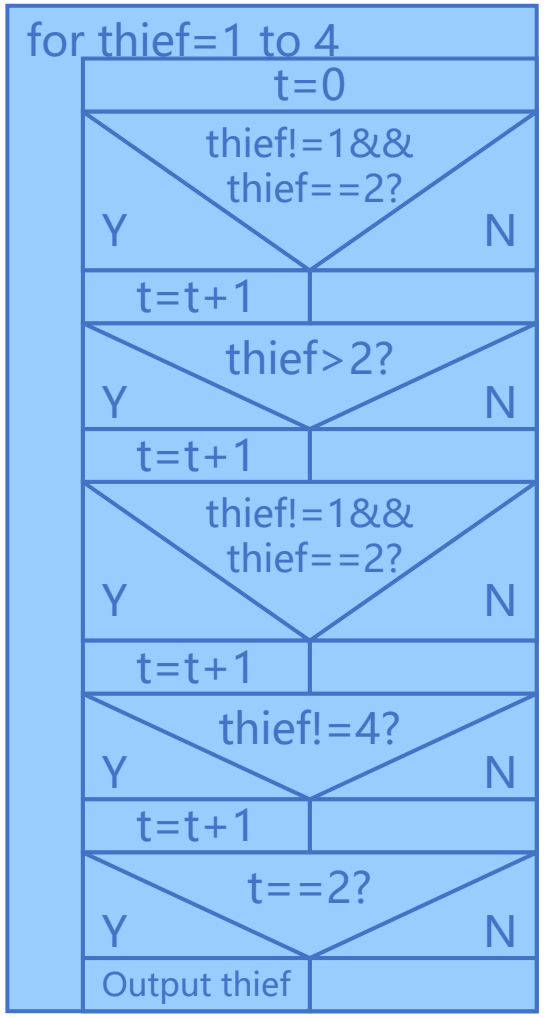
```
25, 75
```

5.6 Exercises

【Exercise 5.5】There are four suspects in a theft trial. A said, "it's not me. It's B who stole it." B said, "the criminal is either C or D." C said, "what A said is right." D said, "The criminal is among the three of them, " If two out of four people told the truth and two told lies, and there was only one criminal, who was the criminal?

5.6 Exercises

【Exercise 5.5】There are four suspects in a theft trial. A said, "it's not me. It's B who stole it." B said, "the criminal is either C or D." C said, "what A said is right." D said, "The criminal is among the three of them, " If two out of four people told the truth and two told lies, and there was only one criminal, who was the criminal?



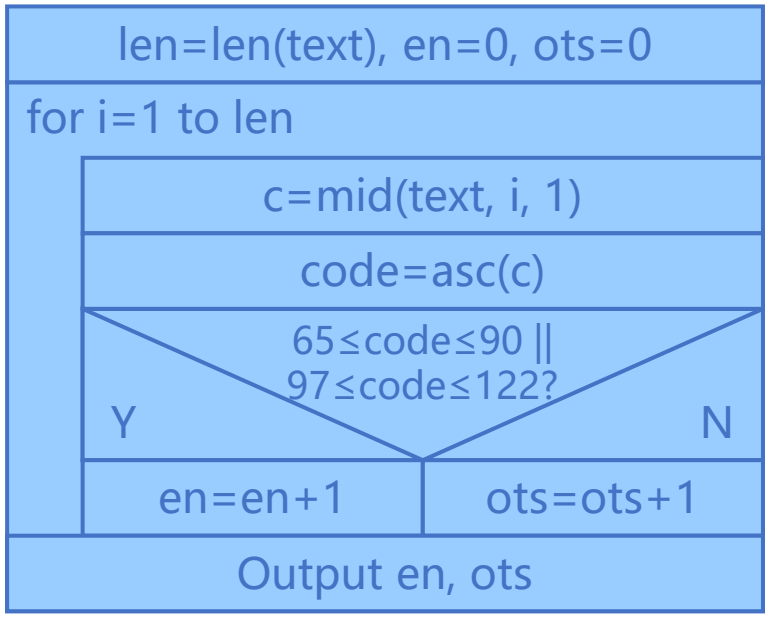
	A	B	C	D
1	for 4	=0		/for thief= 1 to 4
2		if A1!=1 && A1==2	>B1=B1+1	/Whether A told the truth or not
3		if A1>2	>B1=B1+1	/Whether B told the truth or not
4		if A1!=1 && A1==2	>B1=B1+1	/Whether C told the truth or not
5		if A1!=4	>B1=B1+1	/Whether D told the truth or not
6		if B1==2	>output("The thief is "/A1)	/If two people told the truth, output the thief

Result: The thief is 3

【 Exercise 5.6】Input a piece of text and count the number of English letters and non English letters. For example, input: One man's fault is another man's lesson.

5.6 Exercises

【Exercise 5.6】Input a piece of text and count the number of English letters and non English letters. For example, input: One man's fault is another man's lesson.



Program parameter

Set arguments before running

No.	Name	Value	Remark
1	text	another man's lesson.	

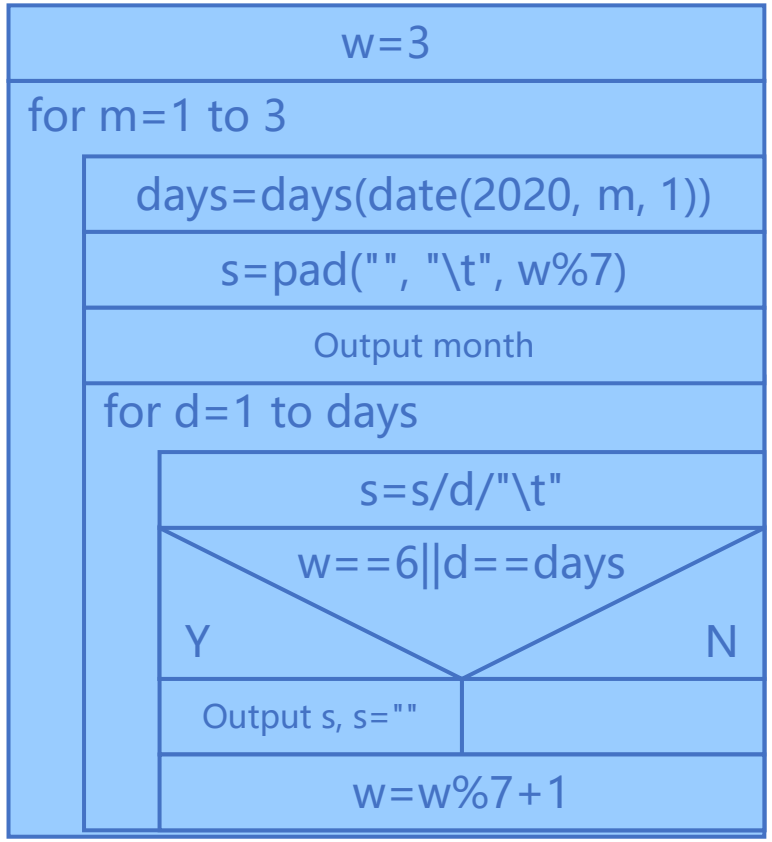
	A	B	C	D
1	=len(text)	0	0	/Initializing the number of characters
2	for A1	=mid(text, A2, 1)	=asc(B2)	/Take a single character and calculate its ASCII
3		if C2>=65&&C2<=90 || C2>=97&&C2<=122	>B1=B1+1	/English letter
4		else	>C1=C1+1	/Non-English-letter
5	>output("En:"/B1/", Ots:"/C1)			/Output result

Result: En:31, Ots:9

【 Exercise 5.7】January 1, 2020 is Wednesday, please output the calendar of the first three months. In a monthly calendar, each week is a line and is displayed from Sunday. You can use the days function to calculate the number of days in a month. For example, to calculate the number of days in March 2020, you can use days (date (2020, 3, 1)).

5.6 Exercises

【Exercise 5.7】January 1, 2020 is Wednesday, please output the calendar of the first three months. In a monthly calendar, each week is a line and is displayed from Sunday. You can use the days function to calculate the number of days in a month. For example, to calculate the number of days in March 2020, you can use days (date (2020, 3, 1)).



	A	B	C		D
1	3				/Initializing
2	for 3	=date(2020, A2, 1)	=days(B2)	=pad("", "\t", A1%7	/The number of days in current month, and fill in the blank days in the first week
3		=string(B2, "MMMM"	>output(B3)		/Output month
4		for C2	>D2=D2/B4/"\t"		/Add current date
5			if A1==6 || B4==C2	>output(D2)	/Weekend or month end, output
6				>D2=""	/Clear string after output
7			>A1=A1%7+1		/week plus 1

Result:

16	17	18	19	20	21	22
23	24	25	26	27	28	29
March						
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Chapter 6
Text and string

6.1 String

- String constant
- String operator
- Special strings
- Escape character and special characters

6.2 String conversion

- Character and ASCII
- Charset and conversion
- String and display format

6.3 Dynamic program

- Macro
- eval function

6.4 Text file

6.5 String functions

- Character functions
- String functions
- Regular expression

6.6 Exercises

Chapter 6
Text and string

6.1 String

String and string constant

- A **string** is a string of characters composed of numbers, letters or other symbols.
- In SPL, **string constants** are constants of string type.
- In SPL, constants can be written directly to cells, and the data type is **automatically parsed**. If the data is not identified as a numeric, Boolean, date/time, etc., it is parsed to a string.

	A	B
1	Name	3rd
2	user@website.com	24*3

Note

- In the above example, constants in each cell are parsed into strings. B1 contains both numbers and letters; while string in B2 has the form of an expression, but does not start with an equal sign, so they are all considered as string constants.

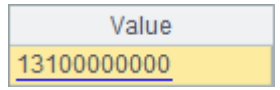
String constant

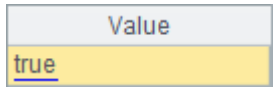
String constant cell

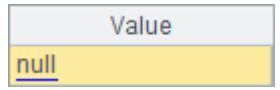
- In some cases, the string may be automatically resolved to other types. In this case, the **string constant cell** can be used to define the string constant.
- In SPL, the **string constant cell** begins with a **half angle single quotation mark '**. This single quotation mark only indicates that the type of constant in this cell is **string** and will not appear in the cell value.
- The rule of string constant cell is **only used when defining constant cell**.

	A	B
1	'13100000000	'true
2	'null	

The values of string constants in each cell are as follows:







Note

- In the above example, since the cell strings start with ', each cell is a string constant cell. A1 is not identified as a long integer, B1 is not identified as a Boolean value, and A2 is not identified as a null value.
- String type values are shown with an underline and left aligned.

String Operator

- In addition to using string constants, you can also use **string operators** to define a string in an expression.
- In SPL expression, the common string operator is a pair of half-width double quotation marks "...", the characters in quotation marks make up the string.
- Another string operator is **$[...]**, which can be used similar to the double quotation mark operator.

	A	B
1	="3.1416"	=$[3.1416]
2	="doctor"	="I am a "/A2

The calculated results of A1, B1, A2 and B2 are as follows:

Value
3.1416

Value
3.1416

Value
doctor

Value
I am a doctor

Note

- The string inside the double quotation marks is evaluated as string type data in the expression. Operator $[...] in B1 can be used similar to double quotation marks.
- B1 uses the division sign / as the string concatenation operator, at least one of the operands is required to be a string; the string concatenation operator can also use the plus sign +, in this case, both operands are required to be strings.

Special strings

- Sometimes, a string may start with an equal sign or a greater than sign, or it may contain special characters or keywords such as double quotation marks, which cannot be parsed automatically, or string operators cannot be used normally, which requires some special processing.
- Using string constant cell can avoid the abnormal parsing of a special string, but it can only be used to define constants.
- If the string contains double quotation marks, it will cause parsing error if it is represented by string operator directly in the expression, which can be avoided by using string constant cell.

	A	B	C
1	'"I live here."	'=== dividing line ===	'>100
2	="He said, "+C1	="He said, "+""I live here.""	

The values of A1, B1 and C1 are as follows:

Value
"I live here."

Value
=== dividing line ===

Value
>100

A2 concatenates two strings and the result is as follows:

Value
He said, "I live here."

Note

- In the example, B2 will report an error when trying to calculate. This is because the string to be used: "I live here." contains double quotation marks, which causes an abnormal parsing of the expression.

Escape character and special characters

- When defining a string directly with string operator in an expression, if the string contains " or \, it can be marked with an **escape character ** before it, and represented by \" or \\.
- If the string contains more than one " or \ character, an escape character needs to be added before each of these characters.
- When the escape character is combined with some letters, it will form some special characters, including line break character **\n** and tab character **\t**. The line break character can make the string break line display, and the tab character can make the text after it align in some fixed positions.

	A
1	="He said, "+"\"I live here.\""
2	="C:\\Program Files\\"+"Common Files"

The calculated results of A1 and A2 are as follows:

Value
He said, "I live here."

Value
C:\Program Files\Common Files

Note

- By adding the escape character in the string operator of the expression, the special string can be parsed normally and errors can be avoided.

The use of line break character \n

	A
1	="Tom\nJerry"
2	>output(A1)

\n in A1 is line break character, and the output result of A2 is as follows:



Note

- The line break character cannot be represented in the value display area on the right. It can only take effect when the characters are output.

The use of tab character \t

	A
1	="1\t22\t333\t4444\n55555\t666666\t\t88888888"
2	>output(A1)

The string of A1 contains a line break character in addition to the tab character. It will output two lines of text. The output of A2 is as follows:

1	22	333	4444
55555	666666		88888888

You can see the alignment of the characters after the tab, and the effect of two consecutive tabs in line 2.

Note

- Tabs are also not represented in the value display area on the right. They can only take effect when the characters are output.
- The characters following the tab will be displayed in some fixed positions, which are different in different software. You can use multiple tab characters to move the following characters to the right by multiple fixed positions.
- Tabs are usually used in multiline text for alignment.

Chapter 6
Text and string

6.2 String conversion

Character and ASCII

- **ASCII** (American Standard Code for information exchange) is a computer encoding system based on English letters (see next page).
- **The basic ASCII** converts the letters, numbers, symbols and some special characters needed to display English characters into **0 ~ 127** codes.
- In addition to the basic ASCII code, there are many extended encoding standards to support the display of various languages.

	A	B
1	=asc("a")	=asc("Z")
2	=char(43)	=char(B1)

The calculated results of A1, B1, A2 and B2 are as follows:

Value
97

Value
90

Value
+

Value
Z

Note

- The **asc(c)** function is used to obtain the ASCII of character **c**, and the **char(n)** function is used to convert the ASCII **n** into the corresponding character.
- **Functions** in programs refer to methods with specific names that can perform processing or return results according to the input parameters. The concept of functions will be explained in Chapter 7. You only need to know how to use some functions related to strings in this chapter.

ASCII table

Code	Character	Code	Character	Code	Character	Code	Character	Code	Character	Code	Character	Code	Character	Code	Character
0	NUL	16	DLE	32	(space)	48	0	64	@	80	P	96	`	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29	GS	45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

【Example 6.1】Randomly generate a string of 4 English characters, the first letter is upper case, the others are lower case.

6.2 String conversion

【Example 6.1】Randomly generate a string of 4 English characters, the first letter is upper case, the others are lower case.

Solution

Using ASCII to convert to the required characters, random function **rand(n)** can generate 0 ~ n-1 integers randomly.

	A	B	C
1	4	=char(65+rand(26))	/First, generate initial letter randomly in B1
2	for A1-1	>B1+=char(97+rand(26))	/Loop, add the following random letters

Each execution will generate a string randomly in B1, such as:

Value
Rfau

Note

- As can be seen from the basic ASCII table on the previous page, the codes of upper case English letters correspond to 65 ~ 90, and the codes of lower case English letters correspond to 97 ~ 122.

Charset and conversion

- The basic ASCII code can only be used to display English. If you need to display other languages or other characters, you need to use the extended coding standard, also known as **charset**.
- There are many character set systems, such as Chinese national standard **GB**, international standard **ISO**, etc. Each system has different character sets, for example, the commonly used character sets of Chinese characters are **Unicode**, **GBK** and **GB2312**.
- When converting non English characters such as Chinese characters to their corresponding codes, the charset must be specified.
- **Unicode** is a charset that can meet the requirements of cross language and cross platform text conversion, and is most commonly used.

	A	B
1	=asc("字")	=char(A1)
2	=urlencode("字符串", "Unicode")	=urlencode@r(A2, "Unicode")
3	=urlencode("字符串", "GBK")	=urlencode@r(A2, "GBK")

The calculation results of each cell are as follows:

Value
23383

Value
字

Value
%FE%FF%5B%57%7B%26%4E%32

Value
字符串

Value
%D7%D6%B7%FB%B4%AE

Value
字符串

Note

- The **asc(c)** and **char(n)** functions described previously actually convert characters and Unicode to each other. However, Unicode is compatible with basic ASCII, so c can be either English characters or other characters such as Chinese characters.
- In order to view the different encoding conversion of character s, we can borrow the transcoding function **urlencode(s, cs)** of URL string, where cs is the character set used, and the transcoded c can be converted to a character with **urlencode@r(c, cs)**.

Common charsets and languages

Charset	Languages
ASCII	English, Western European
ISO-8859-1	English, Western European
GBK	Simplified Chinese, traditional Chinese
GB2312	Simplified Chinese
BIG5	Traditional Chinese
GB18030	Chinese, Japanese, Korean
UTF-8	languages of various countries
Unicode	languages of various countries

Display format

- Any type of data, such as integer, Boolean, time and date, is displayed as a string. If you need to make the data display according to the desired effect, you can set the display format of the data.
- The display format is a string, and certain type of value can be converted into a string according to its rules. For example, a display format of floating-point numbers "$#, ##0.00". In this display format string, some are placeholders, such as "**#**" and "**0**", which indicate the display mode of numbers; some are ordinary characters, such as "**$**", which indicate that a character is output; some are function symbols, such as decimal point "**.**" and segment symbol "**,**", indicating that the data has a decimal point or needs to be segmented.

	A	B
1	'$#, ##0.00	=string(1234567.8, A1)
2	'MM/dd/yyyy	=string(date(2020,1,20), A2)

The data in B1 and B2 are converted according to the display format as follows:

Value
$1,234,567.80

Value
01/20/2020

Note

- A1 and A2 store two display formats with string constants. The display format in A1 is usually used to display the amount, the display format in B1 is usually used to display the date, and the **string()** function is used to convert the display format.
- The **date (y,m,d)** function in B2 generates a date data according to the date parameter.
- After conversion, the string obtained in B1 starts with $and the integer part of the amount is divided into sections with a comma every 3 digits, and the decimal part is displayed with two digits. If the number of digits is not enough, use 0 to supplement. The data in B2 is displayed in month/day/year.

Common characters in numerical data display format

Character	Function
0	Display one digit number, fill in 0 if the number of digit is not enough.
#	Display one digit number, the digit is not displayed if the number of digit is not enough.
.	Decimal point, which separates the integer and decimal part of a real number. When there is no decimal part in the display result, the decimal point is not displayed.
,	Segment symbol, used when displaying integer parts, indicates that each segment is separated by a comma, usually a segment for every 3 digits.
E	In scientific notation, the format of mantissa is indicated in front of E, and the format of index is shown after E
%	Percentage sign, used to display percentage data, multiply the value by 100 and display by percentage
‰	The thousandth ratio is used to display the thousandth ratio data. Multiply the data by 1000 and display it according to the thousandth ratio.
;	Sub mode separator, which separates two display formats. After the separator, it indicates the symbol used for negative numbers
-	Negative sign, used to display negative sign. Generally, it does not need to be used.
'…"	It is used to enclose some characters that may cause ambiguity to indicate that they need to be displayed.

Common characters in date data display format

Character	Function
y/yy	Year, two digits
yyyy	Year, four digits
M	Month
MM	Month, two digits. If the number of digits is not enough, use 0 to supplement.
MMM	Month, English abbreviation in English environment
MMMM	Month, the full name in English
d	Day
dd	Day, two digits. If the number of digits is not enough, use 0 to supplement.
E	Week, English abbreviation in English environment
EEEE	Week, , the full name in English
G	Era identifier, BC / AD, abbreviation
w	What week of the year
W	What week of the month
D	What day of the year

Common characters in time data display format

Character	Function
H	Hour, 24-hour, 0-23
HH	Hour, 24-hour, 00~23, two digits. If the number of digits is not enough, use 0 to supplement.
k	Hour, 24-hour, 1~24
kk	Hour, 24-hour, 01~24, two digits. If the number of digits is not enough, use 0 to supplement.
h	Hour, 12-hour, 1~12
hh	Hour, 12-hour, 01~12, two digits. If the number of digits is not enough, use 0 to supplement.
K	Hour, 12-hour, 0~11
KK	Hour, 12-hour, 00~11, two digits. If the number of digits is not enough, use 0 to supplement.
m	Minute
mm	Minute, two digits. If the number of digits is not enough, use 0 to supplement.
s	Second
ss	Second, two digits. If the number of digits is not enough, use 0 to supplement.
S	millisecond
a	AM/PM
z	time zone

【Example 6.2】January 1, 2020 is Wednesday, please output the calendar of the first three months. In a monthly calendar, each week is a line and is displayed from Sunday.

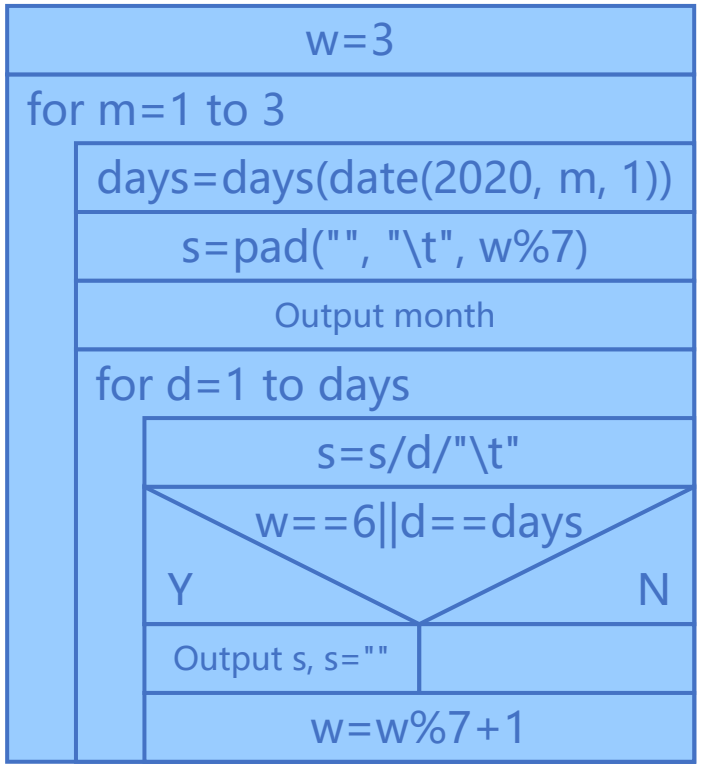
Solution

You can use the **days** function to calculate the number of days in a month. For example, to calculate the number of days in March 2020, you can use days(date(2020, 3, 1)).

When displaying the monthly calendar of a month, you can first output the month in the display format.

The days of the week are separated by a tab \t.

In the first week, you need to fill in multiple tabs according to the week of the first day. In this case, **pad** function can be used. pad(s, c, l) function can add multiple characters c before the string s, so that the total length of s reaches l.



【Example 6.2】January 1, 2020 is Wednesday, please output the calendar of the first three months. In a monthly calendar, each week is a line and is displayed from Sunday.

	A	B	C	D	E
1	3				/Initializing
2	for 3	=date(2020, A2, 1)	=days(B2)	=pad("", "\t", A1%7	/The number of days in current month, and fill in the blank days in the first week
3		=string(B2, "MMMM")	>output(B3)		/Output month
4		for C2	>D2=D2/B4/"\t"		/Add current date
5			if A1==6 || B4==C2	>output(D2)	/Weekend or month end, output
6				>D2=""	/Clear string after output
7			>A1=A1%7+1		/week plus 1

Result:

```
16  17  18  19  20  21  22
23  24  25  26  27  28  29
March
1   2   3   4   5   6   7
8   9   10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  28
29  30  31
```

Parsing data in display format

- If the initial data is stored in the **display format**, you can also specify the display format to parse.
- To parse data according to the format, you can use the functions **number(s, fmt)**, **date(s, fmt)**, **time(s, fmt)** and **datetime(s, fmt)**, where **s** is the data string and **fmt** is the corresponding display format string.
- When parsing data, it is necessary to select different analytic functions according to the data type of numerical, date, time or datetime.

	A	B
1	'$#, ##0.00	=number("$1,234,567.80", A1)
2	'MM/dd/yyyy	=date("01/20/2020", A2)

The data obtained in B1 and B2 is as follows:

Value
1234567.8

Value
2020-01-20

Note

- "$1,234,567.80" is not a numerical value in the standard format, so it can not be directly identified. You need to use the **number()** function to indicate the display format to resolve it to a floating-point number.
- There are many formats of date and time. If the format does not conform to the default format, it is also necessary to specify the display format to be able to use the corresponding parsing function for identification.

Chapter 6
Text and string

6.3 Dynamic program

Macro

- The macro in SPL refers to the string used to compose the expression, which is usually dynamically generated.
- Macro is represented with **${...}** , in brackets is an expression or variable, and the result is a string.
- The macro is not valid inside the quotation marks of the string operator " ... ".

	A	B
1	+	-
2	=3${A1}4	=3${B1}4

The results of A2 and B2 are as follows:

Value
7

Value
-1

Note

- In the above example, the macro is used in A2 and B2, and the expression is calculated by using the symbols in cell variables. In A2, it is equivalent to calculating 3+4, and B2 is equivalent to calculating 3-4.

eval function

- The **eval(x)** function in SPL can reparse the string calculated by parameter x as an expression and calculate the result.
- The macro is only part of an expression, and the eval function parses a new expression and evaluates it.
- Parameters can be used in the expression string of **eval(x, ...)** function. The parameters are input after x. The parameter in the expression string is represented by ? and multiple parameters are represented by **?n**.

	A	B
1	10/2	(?1+?2)/2
2	=eval(A1)	=eval(B1, 3, 7.8)

The results of A2 and B2 are as follows:

Value
5.0

Value
5.4

Note

- The eval function is used in A2 and B2 to evaluate the above two expressions. Where A2 calculates a division formula and B2 calculates the average value of two numbers. These two numbers are input in the eval function in turn and are represented by ?1 and ?2 in the original expression.

The eval function uses $[...] to represent a string

When we introduced string operators in 6.1, we mentioned that there are two string operators in SPL, "..." and $[...]. When used in eval function or macro, there are some differences between the two string representations:

	A	B
1	10	23
2	25	75
3	=eval("A1+B1+A2+B2")	=eval($[A1+B1+A2+B2]



	A	B	C
1	10		23
2			
3	25		75
4	=eval("A1+B1+A2+B2")		=eval($[A1+C1+A3+C3])

In the above grid code, A3 and B3 calculate the sum of the four cell variables A1, B1, A2 and B2 according to an expression string, and their execution effect is the same.

However, if you need to insert a column between two columns of code and a row between the first two rows of code, the grid becomes the grid on the right. In this case, the expression in C4 is still correct, but the expression in A4 is not corresponding.

Note

- As you can see from the above example, although both string operators can represent an expression string, the cell name that appears in the content of $[...] will be automatically adjusted as the grid is edited, and the string in double quotation marks will not change.

【Example 6.3】Randomly generate five oral arithmetic questions. The questions are required to be the addition, subtraction or multiplication between two 2-digit numbers. The answers are given after all the questions are output.

【Example 6.3】Randomly generate five oral arithmetic questions. The questions are required to be the addition, subtraction or multiplication between two 2-digit numbers. The answers are given after all the questions are output.

Solution

With **rand(90)+10**, a 2-digit number can be generated randomly. The symbol can be obtained from the cell constant by macro. The answer needs to be calculated by eval function.

	A	B	C	D
1	+	-	*	/Operation symbol
2	=""			/Initializing the answer
3	for 5	=rand(90)+10	=rand(90)+10	/Loop, generate random 2-digit number
4		=char(65+rand(3))	=B3/${B4}1/C3	/Generate a question randomly
5		>A2=A2/eval(C4)/", "	>output(C4)	/Record the answer and output the question
6	>output(A2)			/Output answer A2

The output result is as follows:

```
12+52
86+44
93+59
34+63
34+85
64,130,152,97,119,
```

▶ In B4, a letter of A ~ C is randomly generated, and then in C4, a symbol is obtained by using a macro to generate a question. For example, If "A" is obtained in B4, then C4 is equivalent to calculate =B3/A1/C3, concatenating two 2-digit numbers and the middle symbol.

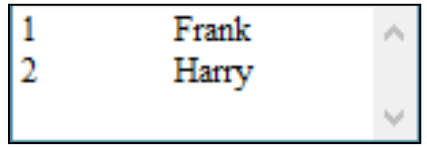
Chapter 6
Text and string

6.4 Text file

6.4 Text file

Text file

- **Text file** is a computer file stored in text mode. The data stored in the file is composed of several lines of strings.
- Text files usually use "**txt**" as the extension. There are also text files with other extensions, such as the csv file mentioned in Chapter 10.
- When a text file is stored, it needs to adopt a consistent encoding method. The commonly used character set is Unicode, and other character sets may be used. If the wrong encoding mode is selected when opening the text file, there may be garbled code and the correct characters cannot be read out.



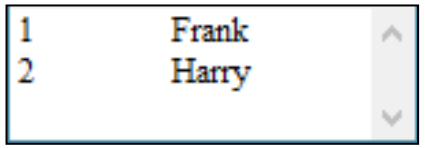
6.4 Text file

Input or output of text file

- In SPL, the **f.read()** function is used to read data from a text file.
- In SPL, you can use the **f.write(s)** function to write a line of string to a text file, or you can add an option, using **f.write@a(s)** function to write the string s after the existing content of the text file.

	A	B
1	D:\files\info.txt	
2	>file(A1).write("1\tFrank")	>file(A1).write@a("2\tHenry")
3	=file(A1).read()	

A2 and B2 write two lines of data to the text file:



Note

- Txt file is a kind of widely used text file. In windows and other systems, you can double-click it and use Notepad and other software to view the contents of the file.
- f.read() function will read all the characters in the text file, and the special characters such as newline character and tab character cannot be viewed in the value display area on the right.

6.4 Text file

【Example 6.4】From text file paragraph.txt, read the English text and count the frequencies of letter e.

At heart he was an arrant coward, which is the way with bullies among apes as well as among men; so he did not remain to fight and die, but tore himself away from them as quickly as he could and fled into the sheltering boughs of the forest.

6.4 Text file

【Example 6.4】From text file paragraph.txt, read the English text and count the frequencies of letter e.

At heart he was an arrant coward, which is the way with bullies among apes as well as among men; so he did not remain to fight and die, but tore himself away from them as quickly as he could and fled into the sheltering boughs of the forest.

Solution

According to the file name, use **file(fn).read()** to read the string in the file, then select characters from the string in loop, and judge whether the character is "e". When looping, the **len(s)** function is used to calculate the string length, and the **mid(s, n, 1)** function is used to select characters.

	A	B	C	D
1	=file("paragraph.txt").read()	0		/Read string from file
2	for len(A1)	=mid(A1, A2, 1)		/Select character from string in loop
3		if B2=="e"	>B1=B1+1	/If character is e, count
4	>output(B1)			/Output result

The result is as follows:

```
20
```

6.4 Text file

【Example 6.5】Please calculate the following four operations and output the results to the file D:\files\answers.txt.

25*8*4*1.25

(107-1)/2+9

123456789*8+9

6.4 Text file

【Example 6.5】Please calculate the following four operations and output the results to the file D:\files\answers.txt.

25*8*4*1.25
(107-1)/2+9
123456789*8+9

Solution

Use **eval()** function to evaluate the result of an expression, and use **file.write()** function to write the text to a file.

	A	B	C	D
1	25*8*4*1.25	(107-1)/2+9	123456789*8+9	/Input 3 expressions
2	D:\files\answers.txt			/The path of output file
3	>A1+="="/eval(A1)	>B1+="="/eval(B1)	>C1+="="/eval(C1)	/Calculate each expression and add result
4	>file(A2).write(A1)	>file(A2).write@a(B1)	>file(A2).write@a(C1)	/Output results

The results in the output file are as follows:

```
25*8*4*1.25=1000.0
(107-1)/2+9=62.0
123456789*8+9=987654321
```

Chapter 6
Text and string

6.5 String functions

6.5 String functions

Character function

- The character function in SPL is used to perform string conversion, string type judgment and other functions.
- The asc, char, urlencode and string functions used earlier in this chapter are all character functions used to perform string conversion.

	A	B
1	'$#, ##0.00	=string(1234567.8, A1)
2	=asc("Z")	=isalpha("Alphabet")

Note

- In the above example, B1 converts the numeric value into a string according to the display format, A2 calculates the code of the character, and B2 determines whether the string is all composed of letters.

6.5 String functions

Common character functions

Function	Effect
asc(s)	Unicode encoding of the first character of string s
asc(s, n))	Unicode encoding of the n^{th} character of string s
char(n))	Character encoded as n
isdigit(s)	Whether the string s consists entirely of numbers or not
isalpha(s)	Whether the string s consists entirely of letters or not
islower(s)	Whether the string s consists entirely of lower case letters or not
isupper(s)	Whether the string s consists entirely of upper case letters or not
lower(s)	Make the letters in the string s lowercase
upper(s)	Make the letters in the string s uppercase
string(x, fmt)	Convert the data according to the display format FMT
urlencode(s, cs)	Convert URL string according to character set CS
urlencode@r(s, cs)	Reverse the URL string according to the character set CS

6.5 String functions

String functions

- The string function in SPL refers to the function used to process strings or perform string related calculation.
- String functions can calculate string length, modify string, intercept string, generate string and so on.

	A	B
1	Helen Keller	=len(A1)
2	=left(A1, 5)	=replace(A1, "e", "E")

Note

In the above example, B1 calculates the number of characters in the string, A2 gets 5 characters from the left of the string, and B2 changes all the lowercase letter e in the string to uppercase.

6.5 String functions

Common string functions

Function	Effect
len(s)	The length of the string s, i.e. the total number of characters of s
fill(s, n))	N strings of s are concatenated into a long string
left(s, n))	A string of n characters taken from the left of string s
right(s, n)	A string of n characters taken from the right of string s
mid(s, n, len)	A string of len characters taken from string s from the n^{th} character
pos(s_1, s_2)	Find the position of the substring s2 in the string s1. If it is not found, null is returned
replace(s, a, b)	Replace substring a in string s with b
like(s, fmt)	Whether the string s meets the format of FMT or not
trim(s)	Remove the spaces on both sides of the string s
pad(s, c, len)	Add the character c before the string s until the character length reaches len
rands(s, len)	Using the characters in the string s, randomly generate a string with the length of len

6.5 String functions

【Example 6.6】Input a full English name and output the last name on the right, such as Helen Keller.

6.5 String functions

【Example 6.6】Input a full English name and output the last name on the right, such as Helen Keller.

Solution

To get the rightmost word, you need to find the position of the rightmost space first, and then take out the string composed of all the characters to the right of this position. Use **pos@z()** function to find the character position from the right.

	A	B
1	Helen Keller	/Input full name
2	=pos@z(A1, " ")	/The position of the rightmost space
3	>output(right(A1, A2))	/Output surname

The output result is as follows:

Keller

Regular expression

- **Regular Expression** is a string that describes the rule of constituting a certain text.
- In SPL, with the **s.regex(re)** function, the regular expression **re** can be used to match the string **s**. if **s** can match the regular expression, the part marked with brackets in **re** will be taken out.
- When the regex function is used in SPL, the regular expression can have multiple brackets to mark the extracted parts. Each extracted string will form a sequence. The concept of sequence will be introduced in detail in Chapter 8.

	A	B
1	a1253b	
2	=A1.regex("(a[0-9])")	=A1.regex("a([0-9]*)")

The results of A2 and B2 are as follows:

Index	Member
1	a1

Index	Member
1	1253

Note

- In the above example, the regular expression **"(a[0-9])"** in A2 means that it starts with letter a, followed by one digit, and takes out this part of the string; the regular expression **"a([0-9]*)"** in B2 means that the string starts with letter a and takes out the consecutive numbers after it.

6.5 String functions

Common symbols in regular expressions

Symbol	Meaning
.	The decimal point represents any character
(…)	Gets the string that matches the part in brackets
x|y	Match x or y
[a-z]	Any letter from a to z, a and z can be changed. For example, [0-9] represents any number between 0 and 9
[^a-z]	Any character beyond a to z
[xyz]	Match any character in x, y, z
[^xyz]	Match any character other than x, y, z
*	Any number of continuous preceding subexpressions
?	1 or 0 preceding subexpression
{n}	Continuous n preceding subexpressions
\d	For any numeric character, the escape character is \\d
\D	For any non numeric character, the escape character is \\D
\t	To match a tab character, the escape character is \\t
\s	o match an invisible character, such as space, tab, newline, etc., you need to add the escape character as \\s
\S	To match a visible character, you need to add the escape character as \\S
\uXXXX	To match a Unicode character, where XXXX is four hexadecimal digits, you need to add the escape character as \\uXXXX

The rules of regular expression are difficult. Here we only give a preliminary introduction, and you need to consult the relevant data in depth when applying.

Chapter 6
Text and string

6.6 Exercises

6.6 Exercises

【Exercise 6.1】Convert the input 3 words into capital letters and output to a file. For example, input Quick, brown and foX.

【 Exercise 6.2】Output the monthly calendar of June 2020 to june.txt. In the calendar, there is one line per week, starting from Sunday. The first line shows the English month and the second line shows the abbreviation of the week. June 1, 2020 is Monday.

【 Exercise 6.3】Input a paragraph of English, convert them to ASCII and output to code.txt. Each character corresponds to a 3-digit code. For example, input: We know only that on a bright May morning in 1888, John, Lord Greystoke, and Lady Alice sailed from Dover on their way to Africa.

【 Exercise 6.4】Read the codes from code.txt in the last question and restore them into English.

【 Exercise 6.5】Input an English name and use regular expression to extract the first name from it. For example, input: Helen Keller.

【 Exercise 6.6】Input an English sentence and output each English word in the sentence in turn. For example, input: The quick brown fox jumps over a lazy dog.

【 Exercise 6.7】Generate a calculation question randomly, and the question is required to be the addition and subtraction operation among three 3-digit numbers. The answer is given in the disjunctive format. Brackets may appear in the next step. The format is as follows:

XXX+(XXX-XXX)

=XXX+XXX

=XXX

6.6 Exercises

【Exercise 6.1】Convert the input 3 words into capital letters and output to a file. For example, input Quick, brown and foX.

6.6 Exercises

【Exercise 6.1】Convert the input 3 words into capital letters and output to a file. For example, input Quick, brown and foX.

	A	B	C	D
1	Quick	brown	foX	/Input 3 words
2	D:\files\words.txt			/Path of output file
3	>file(A2).write(upper(A1))	>file(A2).write@a(upper(B1))	>file(A2).write@a(upper(C1))	/Output result to file

Result:

```
QUICK
BROWN
FOX
```

6.6 Exercises

【Exercise 6.2】Output the monthly calendar of June 2020 to june.txt. In the calendar, there is one line per week, starting from Sunday. The first line shows the English month and the second line shows the abbreviation of the week. June 1, 2020 is Monday.

6.6 Exercises

【Exercise 6.2】Output the monthly calendar of June 2020 to june.txt. In the calendar, there is one line per week, starting from Sunday. The first line shows the English month and the second line shows the abbreviation of the week. June 1, 2020 is Monday.

	A	B	C	D
1	1	=date(2020, 6, 1)	=days(B1)	/The number of days in current month
2	=""	=file("D:/files/june.txt")	>B2.write(string(B1, "MMMM"))	/Output month
3	for 7	=pdate(B1)+A3-1	>A2=A2/string(B3, "E")/"\t"	/Generate data of week line
4	>B2.write@a(A2)	>A2=pad("", "\t", A1%7)		/Output the week line, and fill in the blank days in the first week
5	for C1	>A2=A2/A5/"\t"	>A1=A1%7+1	/Add date of current day, and set the week of the next day
6		if A1==7 || A5==C1	>B2.write@a(A2)	/Weekend or month end, output
7			>A2=""	/Clear string after output

Result:

June

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

Note

- The pdate(date) function can be used to obtain the first day of the week which the date is belonged to. The international convention is to take Sunday as the first day of the week.

6.6 Exercises

【Exercise 6.3】Input a paragraph of English, convert them to ASCII and output to code.txt. Each character corresponds to a 3-digit code. For example, input: We know only that on a bright May morning in 1888, John, Lord Greystoke, and Lady Alice sailed from Dover on their way to Africa.

6.6 Exercises

【Exercise 6.3】Input a paragraph of English, convert them to ASCII and output to code.txt. Each character corresponds to a 3-digit code. For example, input: We know only that on a bright May morning in 1888, John, Lord Greystoke, and Lady Alice sailed from Dover on their way to Africa.

	A	B	C
1	We know only that on a bright May morning in 1888, John, Lord Greystoke, and Lady Alice sailed from Dover on their way to Africa.	=""	/Input English paragraph, initialize output string
2	for len(A1)	=mid(A1, A2, 1)	/Loop through the total number of characters, get one character at a time
3		=asc(B2)	/Convert character to ASCII
4		>B1=B1+string(B3, "000")	/Add the code of one character. The display format 000 means 3-digit integer, if the number of digits is not enough, use 0 to supplement.
5	>file("D:/files/code.txt").write(B1)		/Output ASCII codes to file

Result:

```
087101032107110111190321111101081210321161040971160321 1
111003209703209811410510310411603207709712103210911111411
010511010303210511003204905605605604403207411110411004 4
032076111114100032071114101121115116111107101044032097 11
010003207609710012103206510810509910103211509710510810 1
100032102114111109032068111118101114032111100321161041 01
105114032119097121032116111032065102114105099097046
```

6.6 Exercises

【 Exercise 6.4】Read the codes from code.txt in the last question and restore them into English.

6.6 Exercises

【Exercise 6.4】Read the codes from code.txt in the last question and restore them into English.

	A	B	C
1	=file("D:/files/code.txt").read()	=""	/Read text from file, initialize the decoded text
2	for len(A1)/3	=mid(A1, A2*3-2, 3)	/The number of cycles is calculated according to one third of the total number of characters, and three characters are taken out each time, which is a complete code
3		=number(B2)	/Convert the read-out characters to numerical values
4		>B1=B1+char(B3)	/Decoding a character, recording
5	>output(B1)		/Output decoding result

Result:

We know only that on a bright May morning in 1888, John, Lord Greystoke, and Lady Alice sailed from Dover on their way to Africa.

6.6 Exercises

【 Exercise 6.5】Input an English name and use regular expression to extract the first name from it. For example, input: Helen Keller.

6.6 Exercises

【Exercise 6.5】Input an English name and use regular expression to extract the first name from it. For example, input: Helen Keller.

	A	B
1	Helen Keller	/Input English name
2	=A1.regex("(\\S*)\\s")	/Use regular expression to get first name, which is the part before the first invisible character \s

Result:

Index	Member
1	Helen

6.6 Exercises

【 Exercise 6.6】Input an English sentence and output each English word in the sentence in turn. For example, input: The quick brown fox jumps over a lazy dog.

6.6 Exercises

【Exercise 6.6】Input an English sentence and output each English word in the sentence in turn. For example, input: The quick brown fox jumps over a lazy dog.

	A	B	C	D
1	The quick brown fox jumps over a lazy dog.	=""		/Input English, initialize word record
2	for len(A1)	=mid(A1, A2, 1)		/Get each character in loop
3		if isalpha(B2)	>B1=B1+B2	/If the character is an English letter, record it to the word
4		else	>output(B1)	/If the character is not an English letter, output word
5			>B1=""	/Clear word record

Result:

```
The
quick
brown
fox
jumps
over
a
lazy
dog
```

6.6 Exercises

【Exercise 6.7】Generate a calculation question randomly, and the question is required to be the addition and subtraction operation among three 3-digit numbers. The answer is given in the disjunctive format. Brackets may appear in the next step.

6.6 Exercises

【Exercise 6.7】Generate a calculation question randomly, and the question is required to be the addition and subtraction operation among three 3-digit numbers. The answer is given in the disjunctive format. Brackets may appear in the next step.

	A	B	C	D
1	+	-	=char(65+rand(2))	/Record symbol, C1 selects the column of symbol randomly(A or B)
2	=rand(900)+100	=rand(900)+100	=rand(900)+100	/Generate three 3-digit numbers randomly
3	=A2/${C1}1/B2	=eval(A3)	>C1=char(65+rand(2))	/Generate the formula of the first step by macro, and C1 randomly selects the column where the symbol of the second step is located
4	if rand(2)==0	>A3=A3/${C1}1/C2	>B3=B3/${C1}1/C2	/There is no need for brackets. Calculate step2, generate the complete form and result of step2
5	else	>A3=C2/${C1}1/"("/A3/")"		/The first step is actually at the back, and brackets should be added when generating the complete form
6		if B3<0	>B3=C2/${C1}1/"("/B3/")"	/When the result of the first step is at the back, it is necessary to consider whether the calculation result of the first step is a negative number, and brackets should be added if it's a negative number
7		else	>B3=C2/${C1}1/B3	/The calculation result of step1 is not a negative number, generate result of step2 normally
8	>output(" "/A3)	>output("="/B3)	>output("="/eval(B3))	/Output the result of each row

Result:

```
960+(471-479)
=960+(-8)
=952
```

Chapter 7
Function and subroutine

7.1 What is a function?

7.2 Calling a function in SPL

- Calling a function
- Parameters of function in SPL
- Function options in SPL

7.3 Subroutine

- Definition and call of subroutine
- Parameter transfer in subroutine
- Scope of variables
- Calling procedure of subroutine
- Nesting of subroutines
- Recursion of subroutines

7.4 Cross cellset calling

- Cross cellset calling
- Parameter transfer in cross cellset calling
- Nesting and recursion of cross cellset calling

7.5 Exercises

Chapter 7
Function and subroutine

7.1 What is a function?

What is a function?

- Each **function** in computer programs is used to implement a specific function.
- A function can be executed by its name, which often represents its function.
- Executing a function in a program is called calling a function. When calling a function, the data that needs to be input is called parameters.
- Functions can be divided into nonparametric functions and parametric functions according to whether or not to input data.

	A	B
1	3.1416	3
2	=if(A1<B1, B1, A1)	

Note

- The **if (a, b, c)** function used in A2 has been introduced in Chapter 4. This is a function, and the function name is **if**. Its function is to calculate b or c and return it as a result according to whether a is true or not. After execution, the value obtained in A2 is the larger value in A1 and B1.



Why do we use functions?

- Functions can achieve specific functions, by calling functions, you can avoid repeated programming, effectively reducing the workload.
- In advanced languages, some commonly used function modules will be written as functions, which can be used directly by user. It can not only improve the efficiency, but also increase the reliability of the program.

Chapter 7
Function and subroutine

7.2 Calling a function in SPL

Calling a function

- The call of a function is very simple, just call the function name in the expression.
- Functions usually need input parameters when calling, and some functions may not have input parameters, such as **pi()** and **now()**.
- Some functions have return values, and some functions do not return values, but only change some parameters.
- In SPL, functions with the same name can provide multiple function modes by adding function options or changing input parameters.

	A	B
1	=pi()	=pi(5)
2	=now()	

Note

- The **pi()** function in A1 does not use parameters, and the result is π; while the parameter 5 is input in B1, and the result is 5π.
- The function **now()** in A2 will return the datetime of the current time.
- In a program, the same function can be called any number of times.

【Example 7.1】Input the radius of the circle (e.g. 5.5) to calculate the circumference and area of the circle.

7.2 Calling a function in SPL

【Example 7.1】Input the radius of the circle (e.g. 5.5) to calculate the circumference and area of the circle.

Solution

Use the function pi(n) that has just been introduced to calculate the result.

	A	B	C
1	5.5		/Input radius
2	=2*pi()*A1	=pi(A1*A1)	/Use pi function to calculate circumference and area.

A2 calculates the circumference of the circle and B2 calculates the area of the circle. Select a cell to view the result on the right side:

Value
34.557519189487 72

Value
95.033177771091 25

Note

- The pi() function without a parameter is used in A2, which is equivalent to directly returning the pi. The expression in A2 can also be changed to =pi(2*A1), and the result is the same.
- The parameters of the function can be input directly or calculated by an expression.

In the previous chapters, we have introduced string related functions, if function, etc. Here we will learn about the commonly used date time functions:

Common date time functions

Function	Effect
date(y, m, d)	Generate date type data by specified year, month and day
time(h, min, s)	Generate time type data by specified hour, minute and second
datetime(y, m, d, h, min, s)	Generate datetime type data by specified year, month, day, hour, minute and second
now()	Current time, datetime type, accurate to millisecond
pdate(dt)	The first day of the same period as the specified date. By default the first day of the week
days(dt)	The number of days in the month which the date is belonged to
age(dt)	Calculate current age according to the birthday. It is accurate to the day by default
elapse(dt, n)	By default, calculate the datetime after n days of the specified datetime
interval(dt1, dt2)	By default, calculate the number of days between two datetimes

Parameters of function in SPL

- Different functions have different number of parameters and parameter types, which are input when calling the function.
- In SPL, if a function needs to input multiple parameters, commas, semicolons or colons should be used between the parameters. Pay attention to the different requirements of the function for the separators.
- Some functions use point syntax, such as various functions related to sequence and table sequence.

	A	B
1	=round(3.1416)	=round(3.1416, 2)
2	[1, 3, 5.5, 2]	=A2.len()

The calculated results of A1, B1 and B2 are as follows:

Value
3.0

Value
3.14

Value
4

Note

- **round()** function is used in both A1 and B1.The difference is that only one parameter is input in A1 to calculate the rounded result. In B1, two parameters are input, separated by comma, and the second parameter "2" indicates that the rounded result reserves two decimal places.
- The functions that use **point syntax** are for some specific data types. For example, the function A2.len() in B2 for calculating the number of sequence members is the calculation of sequence A2. These functions are explained in detail in Chapters 8 to 11.

Separation of function parameters in SPL

In SPL, multiple separators can be used between the parameters of some functions, such as if function. Let's look at the following program for judging the fitness level:

	A	B	C
1	77.2	1.78	/Input weight(kg) and height(m)
2	=A1/B1/B1		/Calculate bmi
3	=if(A2<18.5:"Underweight", A2<24:"Fit", A2<28:"Overweight"; "Obesity")	>output(A3)	/Judge the fitness level, output result

The output result is: Overweight

Note

- In the above example, the standard writing method of if function is used, **if(c_1:v_1, c_2:v_2, ...; v_0)**, where c_1:v_1 represents returning value v_1 when the condition expression c_1 is true, such a group of parameters are separated by colons; each condition is separated by a comma; and v_0 is the result returned by the if function when all the previous conditions are not met, so it is separated from other parameters by a semicolon. Compared with using commas as separators, this method is clearer, more accurate, and less prone to errors when writing functions.

Function options in SPL

- The way to call a function with option f *@o*(...), add the @ symbol and the option character o after the function name f.
- In SPL, many functions can provide different functions by adding options. By using options, we can extend the function modes and avoid using too many function names.
- Each function option represents a function mode. Multiple function modes may need to be used together. In this case, multiple options can be used at the same time.

	A
1	=now@t()
1	=now@d()

Note

- The function **now()** used in A1 and A2 is with options. The option **@t** in A1 returns only the current time; the option **@d** in A2 returns only the date.

【Example 7.2】Kevin was born on June 23, 1992. How old is he by March 1, 2020? How many days, months and weeks from his birth?

Example 7.2】Kevin was born on June 23, 1992. How old is he by March 1, 2020? How many days, months and weeks from his birth?

Solution

The **interval** function should be used to calculate the date difference. When different date units need to be calculated, different options should be added. @y is in years, @m is in months, and @w is in weeks.

	A	B	C
1	=date(1992, 6, 23)	=date(2020, 3, 1)	/Generate two dates
2	>output(interval@y(A1, B1)/" years")	>output(interval(A1, B1)/" days")	/Output year and day difference
3	>output(interval@m(A1, B1)/" months")	>output(interval@w(A1, B1)/" weeks")	/Output month and week difference

The output results are as follows:

```
28 years
10113 days
333 months
1444 weeks
```

Note

- When calculating the date difference, if the time enters the next time period, the count will be increased by 1. For example, the calculation of age is based on January 1, it will be considered that the age will be increased by 1 by January 1 of the next year. Therefore, even if the birthday has not arrived in 2020, it will be counted as 28 years old.

Chapter 7
Function and subroutine

7.3 Subroutine

What is a subroutine?

- A **subroutine** is a section of program that can realize a certain function. The subroutine can be called by the main program, and will return to the main program to continue execution after the call is completed.
- In SPL, a subroutine is a code block whose main cell is the statement **func**, and the result needs to be returned by **return** statement. If there is no return statement, the subroutine will return after the code block is executed, and there is no return result.
- In SPL, calling subroutine needs to use **func(C, ...)** function, where **C** is the main cell of the subroutine, and the parameters used by the subroutine can be input as needed.

	A	B	C	D
1	=func(A2)	=func(A2)	=A1*10+B1	/Randomly generate a two digit number of 1 ~ 6
2	func	=rand(6)+1		/subroutine
3		return B2		/Returns a number from 1 to 6

Note

- The random function **rand(n)** used in B2 will generate a 0 ~ n-1 random number, and add 1 to generate a 1 ~ 6 random number.
- In A2, the subroutine is defined by func statement. The subroutine does not need to input parameters, and returns a number from 1 to 6 randomly, which is returned by B3.
- A1 and B1 call the subroutine in A2 to generate a number of 1 ~ 6, and C1 gets a 2-digit randomly composed of 1 ~ 6 .

【Example 7.3】Use subroutine to generate ten oral arithmetic questions randomly. The questions are required to be the addition, subtraction or multiplication between two 2-digit. The answer is given with an equal sign after each question. Output questions to question.txt, each question occupies one line.

【Example 7.3】Use subroutine to generate ten oral arithmetic questions randomly. The questions are required to be the addition, subtraction or multiplication between two 2-digit. The answer is given with an equal sign after each question. Output questions to question.txt, each question occupies one line.

	A	B	C	D
1	+	-	*	/Operation symbol
2	=file("question.txt")	>A2.write("")		/Initialize the output file
3	for 10	>func(A3)		/Loop 10 times to call the subroutine
4	func	=rand(90)+10	=rand(90)+10	/Subroutine, generate random 2-digit
5		=char(65+rand(3))	=B4/${B5}1/C4	/Use random symbol to generate a question
6		>C5=C5/"="/eval(C5)	>A2.write@a(C5)	/Record answer and output one line

The output results are as follows:

```
19+68=87
18+23=41
96-96=0
51-76=-25
84-79=5
63-12=51
73+34=107
45*80=3600
28*92=2576
62-10=52
```

Note

- The subroutine part does not execute on its own, only when it is called. In the example above, the first three lines are the main program part, and in the third line, the subroutine is called several times with a loop.
- A subroutine can be placed anywhere in the cellset, and the code block of the main cell of the subroutine is the scope of the subroutine.
- In B2, it is necessary to output a null character to the text file first to clear the data that may exist in the file. The subroutine in this example only outputs data to the file and has no return value.

Parameters in subroutine

- The subroutine in SPL is similar to the custom function, and it can also input parameters to the subroutine when it is called.
- When calling a subroutine, parameters x_1, x_2, … are input to a subroutine in the format of **func(C, x_1, x_2, …)**.
- When a subroutine is input with only one parameter, the parameter value will be assigned to the main cell of the subroutine during the execution of the subroutine. If there are more than one parameter, the values from the second parameter on will be assigned to the cells on the right side of the main cell of the subroutine in turn.

	A	B	C
1	=func(A2, 4, 6)	=func(A2, 23, 32)	/Calculate the average of 2 numbers
2	func		/Subroutine
3		return (A2+B2)/2	/Return the average of 2 numbers

A1 and B1 call subroutines to calculate the results as follows:

Value
5.0

Value
27.5

Note

- Each time the subroutine in A2 is called, the two input parameters are assigned to A2 and B2 in turn. The value returned by subroutine in B3 can be calculated from the two values.
- If the subroutine has input parameters, no code should be written in the relevant cells. The code of the subroutine can start from the code block area of the next line.

Scope of variables in subroutine

- Cell variables or variables that have been assigned in the main program are valid throughout the program. If the subroutine is modified, the values of these variables will change.
- Each input parameter in the subroutine is valid only when the subroutine is executed. At the end of the subroutine execution, the relevant cell variables will be cleared.
- In a subroutine, the values of other cell variables in the code block of the main cell are also valid when the subroutine is executed. If there are constant cells in the code block, the cell values are reset every time the subroutine executes.

	A	B	C	D
1	func	0	>B1=B1+1	/Subroutine
2		>A3=A3+A1	return A3/B1	/Prepare to return the average
3	0			/A3 stores the cumulated sum
4	=func(A1, 1)	=func(A1, 2)	=func(A1, 3)	/Call subroutine 3 times

The calculation results of A4, B4 and C4 are as follows:

Value
1.0

Value
3.0

Value
6.0

Note

- In this example, the subroutine is written in the first two rows. The subroutine will be executed only when it is called, and the code location has nothing to do with the execution effect.
- The subroutine needs one input parameter. Each time the subroutine is executed, A3 will calculate the sum of the parameters, and each time it will add 1 to the value in B1, and return the result of A3/ B1. However, it can be seen from the results that the subroutine only returns the cumulative sum of the input parameters. This is because B1 is assigned the input constant value 0 every time the subroutine is executed.

Scope of variables in subroutine

- Cell variables or variables that have been assigned in the main program are valid throughout the program. If the subroutine is modified, the values of these variables will change.
- Each input parameter in the subroutine is valid only when the subroutine is executed. At the end of the subroutine execution, the relevant cell variables will be cleared.
- In a subroutine, the values of other cell variables in the code block of the main cell are also valid when the subroutine is executed. If there are constant cells in the code block, the cell values are reset every time the subroutine executes.

	A	B	C	D
1	func	0	>B3=B3+1	/Subroutine
2		>A3=A3+A1	return A3/B3	/Prepare to return the average
3	0	0		/A3 stores the cumulated sum
4	=func(A1, 1)	=func(A1, 2)	=func(A1, 3)	/Call subroutine 3 times

The calculation results of A4, B4 and C4 are as follows:

Value
1.0

Value
1.5

Value
2.0

Note

- If the cell variable in the main program is also used to calculate the number of data in C1, the average value of data returned in C2 is correct, and the three cumulative mean values are 1.0, 1.5 and 2.0 respectively.

	A	B	C	D
1	func	0	>B3=B3+1	/Subroutine
2		>A3=A3+A1	return A3/B3	/Prepare to return the average
3	0	0		/A3 stores the cumulated sum
4	=func(A1, 1)	=func(A1, 2)	=func(A1, 3)	/Call subroutine 3 times

Let's take the subroutine we just used as an example to understand the calling process of the subroutine:

In this program, the first two lines are a subroutine, which is executed only when called, otherwise it will not be executed. So, the main program runs from line 3, with A3 and B3 assigned a value of 0.

When executing A4, the subroutine in A1 needs to be called. The subroutine has a parameter value of 1. This will jump to A1 and assign parameter value 1 to A1. Next, execute the code block of subroutine main cell A1, first execute line 1, B1 is assigned as the initial value 0 (here, the value of B1 is not used in the subroutine); execute C1, the cell parameter B3 in the main program is added with 1, recording the execution times of subroutine. Then execute B2, the cell parameter A3 in the main program adds the input parameter in A1, recording the cumulative value of the parameter. In C2, the cumulative average value of 1 / 1 = 1.0 is returned, the subroutine call ends, and the result of subroutine is returned to A4.

Next, B4 is executed, the subroutine in A1 is called again, and the subroutine is executed in A1. The parameter value 2 is assigned to A1, and B1 is assigned to 0. In C1, calculate the cumulative execution times B3; in B2, calculate the cumulative sum A3. C2 returns the cumulative average value of 3 / 2 = 1.5. The subroutine ends, the result is returned to B4.

Finally, C4 is executed and subroutine in A1 is called again. Parameter value 3 is assigned to A1 and B1 is assigned to constant value 0. In C1, calculate the cumulative execution times B3; in B2, calculate the cumulative sum A3. C2 returns the cumulative average value of 6 / 3 = 2.0. The subroutine ends, the result is returned to C4.

At the end of program execution, the cell values of the first two lines of subroutine are null, the cell value of A3 is the cumulative sum of 6, and the cell value of B3 is the cumulative number of times 3.

	A	B	C	D
1	=func(A2, …)			/The main program calls subroutine 1
2	func	…	…	/Subroutine 1
3		=func(A5, ...)	...	/Call subroutine2
4		…	return …	/Subroutine1 returns
5	func	...	...	/Subroutine2
6		...	return ...	/Subroutine2 returns

Nesting of subroutines

- In SPL, a subroutine can also call other subroutines. Calling other subroutines in a subroutine is called nesting of subroutines.
- When a nested subroutine is called, after it ends, it will return to the place where it is called and continue to execute the original subroutine, which is similar to the execution sequence of calling a subroutine in the main program.

【Example 7.4】Using nested subroutines, output the nine times nine multiplication table to multable.txt. The n^{th} line outputs the formula of 1 * n ~ n * n.

7.3 Subroutine

Nesting of subroutines

【Example 7.4】Using nested subroutines, output the nine times nine multiplication table to multable.txt. The n^{th} line outputs the formula of 1 * n ~ n * n.

	A	B	C	D
1	9	=file("multable.txt")	>B1.write("")	/The size of multiplication table, initialize output table
2	for A1	>func(A5, A2)		/Loop 9 times to output formula of each line
3	func		=A3*B3	/Subroutine2, input two multipliers to calculate the product
4		=A3/"*"/B3/"="/C3/"\t"	return B4	/Generate one formula and return
5	func			/Subroutine1, output formula of one line
6		for A5	>B5=B5+func(A3, B6, A5)	/Loop subroutine2, generate one line of formula in B5
7		>B1.write@a(B5)		/Output one line to file

The output result is as follows:

```
1*1=1
1*2=2   2*2=4
1*3=3   2*3=6   3*3=9
1*4=4   2*4=8   3*4=12  4*4=16
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

Note

- The position of subroutine 1 and subroutine 2 in the code does not affect the use.
- In subroutine 1, subroutine 2 is called circularly, and one piece formula is generated each time to get the whole line of formula. Through the use of subroutines, the main program structure is simple and clear, and easier to understand.

Recursion of subroutines

- In SPL, a subroutine can call other subroutines, and if it calls itself, it becomes a **recursion of subroutines**.
- When using the recursion of subroutine, it is necessary to ensure that the recursive call can end effectively and avoid falling into an endless loop.
- When a subroutine is called recursively, each execution process does not affect each other, and the parameters of the subroutine do not interfere with each other, and are only used in their own execution scope.

	A	B	C	D
1	func	if A1<=1	return 1	/Subroutine, when n is not greater than 1, set the factorial to 1
2		=A1*func(A1,A1-1)	return B2	/Recursion, n!=n*((n-1)!)
3	=func(A1,4)	=func(A1,7)	=func(A1,10)	/Call subroutine 3 times

The results of A3, B3, C3 are as follows:

Value
24

Value
5040

Value
3628800

Note

- When a subroutine is called recursively, the subroutine in A1 is called many times, and different parameter values are input in the main cell A1 respectively. However, the operation of each subroutine is independent, and the value of A1 is only valid in the calling process of the subroutine and will not interfere with each other.

Chapter 7
Function and subroutine

7.4 Cross Cellset calling

7.4 Cross cellset calling

Cross cellset calling

- In SPL, in addition to calling subroutines, you can also call a DFX program file directly across the cellset, execute all the codes and return.
- Similar to subroutines, when calling across cellsets, if the called cellset needs to return results, it can return them with **return** statement.
- Cross cellset calling needs to use **call(fn, ...)** function in the main program, where fn is file name of the DFX file.

Grid file exer.dfx generates a 2-digit arithmetic question randomly.

	A	B	C
1	+	-	*
2	=rand(90)+10	=rand(90)+10	=char(65+rand(3))
3	=A2/${C2}1/B2	>A3=A3/"="/eval(A3)	return A3

Call the above cellset file in the main grid:

	A	B	C
1	for 10	>output(call("exer.dfx"))	/Execute the cellset file in loop

Note

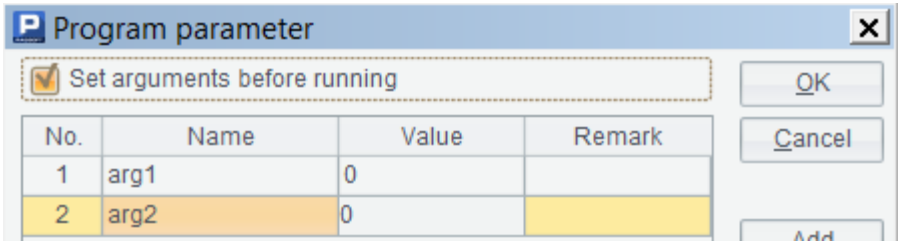
- The cross cellset calling is basically the same as that of the subroutine. Calling a subroutine needs to use **func(C, ...)** to specify the main cell C of the subroutine; and cross cellset calling needs to use **call(fn, ...)** to specify the DFX file name.
- Cross cellset calling is more universal than subroutines, and can be called in different cellset codes. At the same time, the security is higher. It can encrypt the file's code without affecting the calling.

7.4 Cross cellset calling

Parameter transfer in cross cellset calling

- Similar to calling a subroutine, parameters can also be used in cross cellset calling. Parameters can be input after the DFX file name in call(fn, ...).
- If the called grid needs to use parameters, you need to define program parameters in the cellset.
- To define program parameters in SPL, you can click Program>Parameter to set in the menu bar of esProc.

Cellset file bigger.dfx returns the larger of the two numbers, you need to enter two program parameters first:



	A	B
1	if arg1>arg2	return arg1
2	else	return arg2

Call the above cellset file in the main cellset:

	A	B
1	=call("bigger.dfx", 72, 75)	/Call cellset file to calculate the larger value.

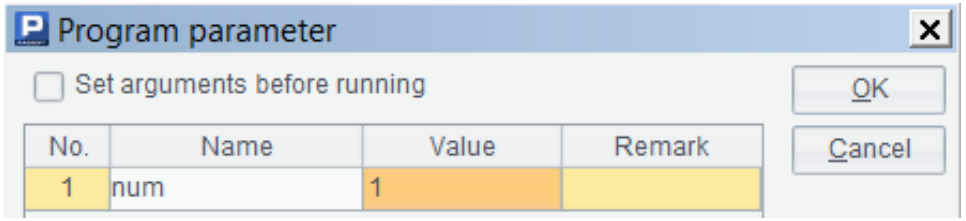
Note

- When calling the cellset file with parameters, it is not necessary to write the parameter name, and the parameters will be set in the order of the list.

【Example 7.5】Using cross cellset calling, output the nine times nine multiplication table to multable.txt. The n^{th} line outputs the formula of 1 * n ~ n * n.

【Example 7.5】Using cross cellset calling, output the nine times nine multiplication table to multable.txt. The n^{th} line outputs the formula of 1 * n ~ n * n.

Cellset file mulline.dfx outputs a line of formula from 1*num to num*num with input parameter num:



	A	B	C	D
1	func		=A1*B1	/Subroutine, input two multipliers to calculate the product
2		=A1/"*"/B1/"="/C1/"\t"	return B2	/Generate one formula and return
3	=""	for num	>A3=A3+func(A1, B3, num)	/Loop the subroutine, generate the formula in A3
4	return A3			/Return the whole line

In the main program, the cellset file is called circularly, and all multiplication tables are output into the file.

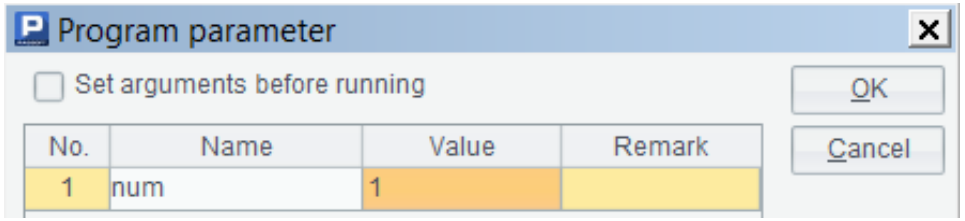
	A	B	C	D
1	9	=file("multable.txt")	>B1.write("")	/The size of multiplication table, initialize output table
2	for A1	=call("mulline.dfx", A2)	>B1.write@a(B2)	/Loop 9 times to output the formula of each line

The output is the same as before.

Nesting and recursion of cross cellset calling

- In the previous example, a subroutine was used in the called grid. In fact, there are no restrictions in the called grid except for the return statement to return the required value. You can use a subroutine or call other cellsets.
- In DFX that is called across cellset, another cellset is called, that is the nesting of cross cellset calling. If the grid itself is called in the cellset, it is the recursion of cross cellset calling.
- The grid codes that are called are independent of each other and will not affect each other.

Cellset file factorial.dfx calculates the factorial of num:



	A	B
1	if num<=1	return 1
2	else	return num*call("factorial.dfx", num-1)

Call the above cellset file in the main cellset:

	A	B
1	=call("factorial.dfx", 5)	/Call cellset file to calculate factorial

Chapter 7
Function and subroutine

7.5 Exercises

7.5 Exercises

【Exercise 7.1】Enter your birthday and calculate the dates of 50 years, 500 months, and 5000 days after that (using the elapse function, the options of which are similar to that of interval function).

【 Exercise 7.2】By calling a subroutine, output Fibonacci sequence of the first 20 items.

【 Exercise 7.3】Call a subroutine recursively to calculate the n^{th} power of a, where n is a positive integer.

【 Exercise 7.4】By calling a nested subroutine, output all pentagram number. A pentagram number refers to such a 5-digit number: the sum of the quintic power of its digits is equal to itself.

【 Exercise 7.5】By calling a subroutine to output the annual calendar of 2020 to calendar.txt. In the calendar, there is one line per week, starting from Sunday. The first line shows the English month and the second line shows the abbreviation of the week. January 1, 2020 is Wednesday.

【 Exercise 7.6】Using cross cellset call to solve the previous problem, output the calendar of 2020.

7.5 Exercises

【Exercise 7.1】Enter your birthday and calculate the dates of 50 years, 500 months, and 5000 days after that (using the elapse function, the options of which are similar to that of interval function).

7.5 Exercises

【Exercise 7.1】Enter your birthday and calculate the dates of 50 years, 500 months, and 5000 days after that (using the elapse function, the options of which are similar to that of interval function).

	A	B
1	=date(1992, 6, 23)	/Input birthday
2	>output(elapse@y(A1, 50))	/Output the date 50 years after it
3	>output(elapse@m(A1, 500))	/Output the date 500 months after it
4	>output(elapse(A1, 5000))	/Output the date 5000 days after it

Result:

2042-06-23
2034-02-23
2006-03-02

7.5 Exercises

【 Exercise 7.2】By calling a subroutine, output Fibonacci sequence of the first 20 items.

7.5 Exercises

【Exercise 7.2】By calling a subroutine, output Fibonacci sequence of the first 20 items.

	A	B	C	D
1	20			/Input the number of items
2	for A1	>output(func(A3, A2))		/Call subroutine in loop to output result
3	func	if A3<3	return 1	/Subroutine, the first two items are 1
4		else	return func(A3,A3-1)+func(A3,A3-2)	/From the 3rd item on, it's the sum of the previous two items

Result:

```
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
```

7.5 Exercises

【 Exercise 7.3】Call a subroutine recursively to calculate the n^{th} power of a, where n is a positive integer.

7.5 Exercises

【 Exercise 7.3】Call a subroutine recursively to calculate the n^{th} power of a, where n is a positive integer.

	A	B	C	D
1	1.5	3	>output(func(A2, A1, B1))	/Input a and n, call subroutine to calculate
2	func			/Subroutine. The first line is empty to receive parameters
3		if B2<=1	return A2	/If n=1, return a
4		else	return a*func(A2, A2, B2-1)	/If n>1, return a*f(n-1)

Result:

3.375

7.5 Exercises

【 Exercise 7.4】By calling a nested subroutine, output all pentagram number. A pentagram number refers to such a 5-digit number: the sum of the quintic power of its digits is equal to itself.

7.5 Exercises

【Exercise 7.4】By calling a nested subroutine, output all pentagram number. A pentagram number refers to such a 5-digit number: the sum of the quintic power of its digits is equal to itself.

	A	B	C	D
1	for 10000, 99999	if func(A2, A1)	>output(A1)	/Loop through all 5-digit numbers, call subroutine1 to judge whether it's a five pointed star number or not.
2	func	=string(A2)		/Subroutine1, convert the 5-digit number to a string
3		=func(A6, left(B2, 1))	=func(A6, mid(B2, 2, 1))	/Get the first 2 digits from the string, call subroutine2
4		=func(A6, mid(B2, 3, 1))	=func(A6, mid(B2, 4, 1))	/Get the third and fourth digits from the string
5		=func(A6, right(B2, 1))	return B3+C3+B4+C4+B5==A2	/Get the fifth digit, judge whether it's a pentagram number or not.
6	func	=int(A6)	return B6*B6*B6*B6*B6	/Subroutine2, according to a one-digit character, calculate its quintic power

Result:

```
54748
92727
93084
```

7.5 Exercises

【 Exercise 7.5】By calling a subroutine to output the annual calendar of 2020 to calendar.txt. In the calendar, there is one line per week, starting from Sunday. The first line shows the English month and the second line shows the abbreviation of the week. January 1, 2020 is Wednesday.

7.5 Exercises

【Exercise 7.5】By calling a subroutine to output the annual calendar of 2020 to calendar.txt. In the calendar, there is one line per week, starting from Sunday. The first line shows the English month and the second line shows the abbreviation of the week. January 1, 2020 is Wednesday.

	A	B	C	D	E
1	=date(2020, 1, 1)	=A1-pdate(A1)	=file("calendar.txt")	>C1.write("2020")	/Calculate the initial values of week, initialize the output file
2	for 7	=pdate(A1)+A2-1	>D1=D1/string(B2, "E")/"\t"		/Prepare the title line of abbreviated week and record it in D1
3	for 12	>func(A4, A3)			/Call subroutine to output the monthly calendar of each month
4	func	=date(2020, A4, 1)	=days(B4)	=""	/Subroutine, Output the monthly calendar, calculate the number of days in the month
5		>C1.write@a(string(B4, "MMMM"))	>C1.write@a(D1)	>D4=pad("", "\t", B1%7)	/Output the title lines of month and week
6		for C4	>D4=D4/B6/"\t"	>B1=B1%7+1	/Add the current date and set the next day's week
7			if B1==7 || B6==C4	>C1.write@a(D4), D4=""	/If it's weekend or end of month, output the record of the week

Result:

```
22      23      24      25      26      27      28
29      30
December
Sun     Mon     Tue     Wed     Thu     Fri     Sat
                1       2       3       4       5
6       7       8       9       10      11      12
13      14      15      16      17      18      19
20      21      22      23      24      25      26
27      28      29      30      31
```

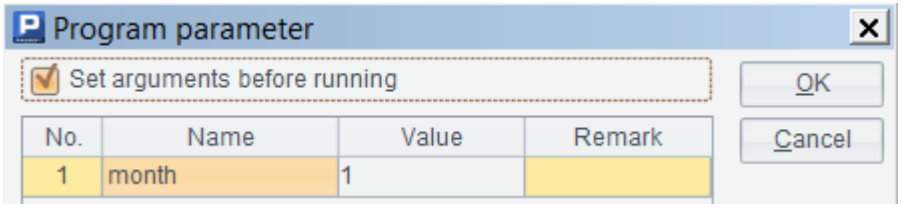
7.5 Exercises

【Exercise 7.6】Using cross cellset call to solve the previous problem, output the calendar of 2020.

7.5 Exercises

【Exercise 7.6】Using cross cellset call to solve the previous problem, output the calendar of 2020.

Cellset file mcalendar.dfx outputs a month's calendar with input parameter month:



	A	B	C	D	E
1	=date(2020, month, 1)	=A1-pdate(A1)	=days(A1)	=string(A1, "MMMM")+"\r\n"	/Calculate the weekday of the initial day of the month
2	for 7	=pdate(A1)+A2-1	>D1=D1/string(B2, "E")/"\t"		/The title line of weekdays
3	=pad("", "\t", B1%7)	for C1	>A3=A3/B3/"\t"	>B1=B1%7+1	/Initialize the vacancy of the first week, loop through the days of the month
4			if B1==7 || B3==C1	>D1=D1/"\r\n"/A3, A3=""	/If it's weekend or end of month, output the record of the week
5	return D1				/Return the monthly calendar of current month

In the main program, the cellset file is called circularly, and the annual calendar is output to the file:

	A	B	C
1	=file("calendar.txt")	>A1.write("2020")	/Initialize the output file
2	for 12	>A1.write@a(call("mcalendar.dfx", A2))	/The cellset file is called circularly to output the monthly calendar of each month

The output result is the same as before. Note that when multiple lines of data need to be output in a text file, the line break character is \r\n.

Chapter 8 Sequence

8.1 The concept of sequence

8.2 Generation and modification of a sequence

- Sequence constants
- Member access of a sequence
- Add a member to the end of a sequence

8.3 Loop and calculation of sequence

- Execute the loop with for A
- Aggregation
- Examples of sequence calculation

8.4 Common sequences

- Integer sequence
- String sequence

8.5 Multidimensional sequence

- What is multidimensional sequence?
- Access to multidimensional sequence members
- The use of multidimensional sequences

8.6 Exercises

Chapter 8
Sequence

8.1 The concept of sequence

Why do we need sequence?

- In solving problems, we often need to deal with many data with the same purpose and meaning. To process these data concisely and efficiently, **sequences** are needed.
- Using sequence, it is convenient to store and read data, and also to realize cyclic calculation.
- The data in sequences can also be used to perform inter-set operations. Or calling some special functions to realize many common data processing methods, such as sum, sorting, etc.

8.1 The concept of sequence

Suppose we go to the supermarket and buy 10 pieces of goods. Referring to the contents of the previous chapters, if we want to calculate the total price, we may need to deal with it as follows:

	A	B	C	D	E	F
1	23.5	29.1	15.8	4.6	5.2	/Original data
2	5.2	2.5	7.6	12.9	43.8	/Original data
3	=A1+B1+C1+D1+E1+A2+B2+C2+D2+E2					/Getting sum directly

A3	Value
	150.2

Obviously, such a writing method is both cumbersome and a waste of space. If the amount of data is further expanded, the space occupied and the complexity of calculation expression will be unacceptable.

8.1 The concept of sequence

If the sequence and related special functions are used, the calculation in the above example can be simplified.

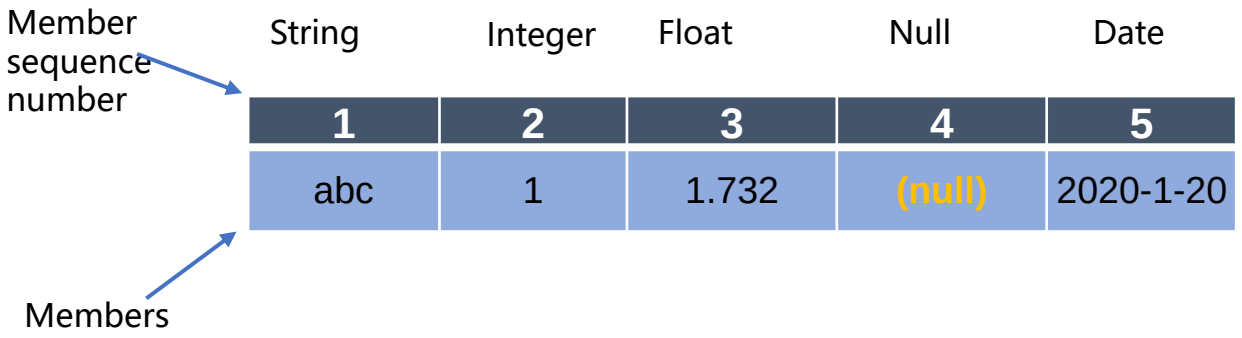
	A	B
1	[23.5, 29.1, 15.8, 4.6, 5.2, 5.2, 2.5, 7.6, 12.9, 43.8]	/Save data in a sequence
2	=A1.sum()	/Use aggregation function to get SUM

A2	Value
	150.2

Now that we talked about using a sequence in the example, what is a sequence?

The concept of sequence

- **Sequence** is a set of ordered data. The data that makes up a sequence is called a **member** of the sequence. Members of a sequence can be of any data type, such as string, number, float, date, or null. The values of members in a sequence can be repeated.
- Sequence is similar to array in some programming languages.



Note

- **(null)** represents null.

8.1 The concept of sequence

Because the sequence is ordered, we can use its order to deal with many practical problems. For example, you can use a sequence to store the math scores of a group of five students. The order of data in the sequence is the order of students' student IDs.

Math scores of a 5-student group

1	2	3	4	5
91	87	65	23	59

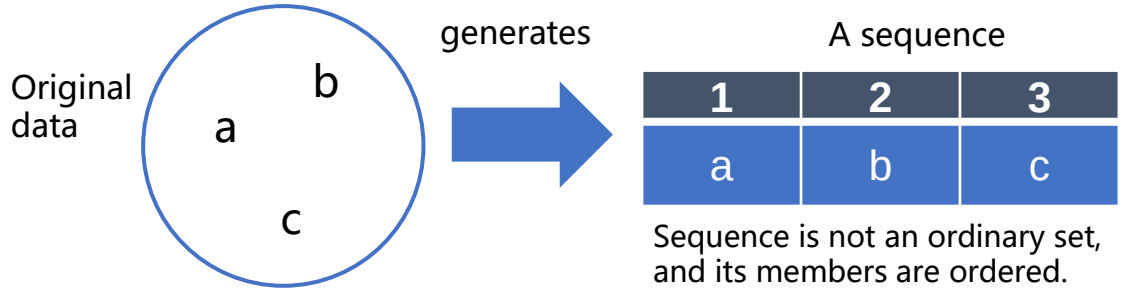
If you need to find the student's math score of student ID 3, you only need to find the value of sequence member with sequence number 3: 65.

Next, we will introduce the generation, read-write, loop, common types of sequences, and the concept and use of multidimensional sequences.

Chapter 8
Sequence

8.2 Generation and modification of a sequence

8.2.1 Sequence constants



Sequence is not an ordinary set, and its members are ordered.

Generation of a sequence

- There are two common methods of constant construction and function construction, and a sequence can also be generated by calculation. Here we mainly introduce the constant construction method.

Constant construction method

	A	B
1	[a, b, c]	/Generating a sequence with constant construction method in SPL

Click A1 to view the sequence in the value display area on the right:

A1	Index	Member
	1	a
	2	b
	3	c

When displayed, the index column is an auxiliary column to display the sequence number. The total number of members in a sequence is called the length of the sequence.

8.2.1 Sequence constants

Special types of sequences

- An **integer sequence** is a sequence whose members are all integers.
- An **empty sequence** is a sequence without members. Note that an empty sequence is different from a sequence of null values.

A1	Index	Member
	1	1
	2	2
	3	3

An integer sequence: the members are all integers

	A	B
1	[1, 2, 3]	/Define a sequence of numbers with constant construction method

Empty sequence: without member

A1	Index	Member

An empty sequence is a sequence without members; while a null value will be displayed as (null)

	A	B
1	[]	/Define an empty sequence with constant construction method

8.2.1 Sequence constants

In addition to setting sequence members directly, you can also create a new sequence with existing cell values as sequence members.

	A	B	C
1	s	a	/Original data
2	d	t	/Original data
3	=[A1:A2]	=[A1:B2]	/Define a sequence with cell values

A3	Index	Member
	1	s
	2	d

B3	Index	Member
	1	s
	2	a
	3	d
	4	t

Note

- When the original data occupies multiple rows and columns, such as writing as [A1: Bn], SPL will put the data into the sequence in the order of first by row and then by column: [A1, B1, A2, B2, ..., An, Bn]。

8.2.1 Sequence constants

The existing sequence can also be used for multiplication to generate a new sequence.

	A	B
1	=["a"]*4	/When the data type of sequence members is normal data, all members copy existing sequence members in the new sequence formed by multiplication.
2	[0]	
3	=[A2]*4	/When A2 itself is a sequence, [A2] indicates a sequence whose member is also a sequence. In the new sequence formed by [A2]*4, all sequence members are the sequence A2.

A1	Index	Member
	1	a
	2	a
	3	a
	4	a

A3	Index	Member
	1	[0]
	2	[0]
	3	[0]
	4	[0]

Note

- In A1, sequence constants such as ["a"] are directly used in expressions. When sequence constants are directly used in expressions, member parsing rules in sequences are different from constant cell. Members of string type need to be enclosed in quotation marks, and strings of date time type need to be converted first.
- For [A]*n, when a is A sequence, all members in the generated sequence B actually reference the same object. If A is modified, all members in B will be modified.

8.2.1 Sequence constants

【Example 8.1】Suppose a hospital has a simple queuing system, try to simulate the queuing system with sequence. Suppose that the order of sequence members is the order of patients queuing. There are 5 patients in the queue, in order: James, Kate, Bill, Mary, Lucy.

8.2.1 Sequence constants

【Example 8.1】Suppose a hospital has a simple queuing system, try to simulate the queuing system with sequence. Suppose that the order of sequence members is the order of patients queuing. There are 5 patients in the queue, in order: James, Kate, Bill, Mary, Lucy.

Solution

Let the member order in the sequence be consistent with the actual patient queuing order.

	A	B
1	[James, Kate, Bill, Mary, Lucy]	/Using cell constant to construct sequence, constant sequence can be displayed directly
2	=["James", "Kate", "Bill", "Mary", "Lucy"]	/To construct a sequence with an expression, the string members of the sequence in the expression should be expressed in quotation marks. The expression needs to be evaluated before it has a result. The result is the same as A1

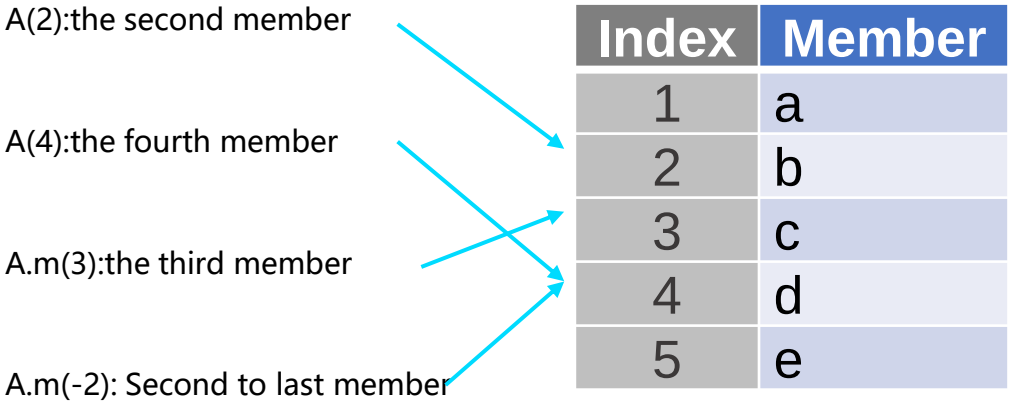
A1	Index	Member
	1	James
	2	Kate
	3	Bill
	4	Mary
	5	Lucy

A2	Index	Member
	1	James
	2	Kate
	3	Bill
	4	Mary
	5	Lucy

8.2.2 Member access of a sequence

Member access of a sequence

- The most basic way to access sequence members is to access them with **A(n)** in the expression. According to the sequence number **n**, the member at the specified position in **A** is accessed. In addition to obtaining the members in the specified position, we can also use this access method to assign values to sequence members: **A(n)=x**. However, we can only change the value of existing members in the sequence in this way, and cannot add additional sequence members, that is, n cannot exceed the length of the sequence.
- We can also access the members of sequence **A** in the way of function **A.m(n)**. **n** in the parameter can be the sequence number of the member, or a negative number to represent the member counting backwards.



8.2.2 Member access of a sequence

【Example 8.2】Suppose that in example 8.1, Bill, the third patient, is about to finish his consultation, and the hospital is ready to remind the next patient behind him to be ready by radio. Then how to get the name of the next patient? If the hospital is going to give a small gift to the last patient in the queue in order to appease the lower ranking patients, how can we quickly find the last patient in the queue?

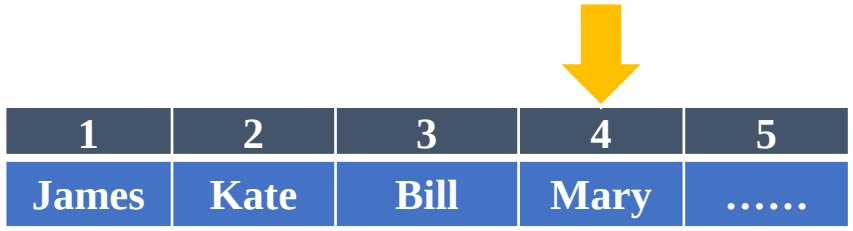
8.2.2 Member access of a sequence

【Example 8.2】Suppose that in example 8.1, Bill, the third patient, is about to finish his consultation, and the hospital is ready to remind the next patient behind him to be ready by radio. Then how to get the name of the next patient? If the hospital is going to give a small gift to the last patient in the queue in order to appease the lower ranking patients, how can we quickly find the last patient in the queue?

Solution

Members of a sequence can be accessed by sequence numbers. For the members near the end of the sequence, we can also use the relative position access method of sequence members to quickly find them.

In the expression of SPL, **A(i)** can be used to access the members of sequence **A** according to the sequence number **i**. **i** can only be a positive number. If **i** exceeds the sequence number range of **A**, an error will be reported. You can also use the function **A.m(i)** to access the members in sequence **A** reversely. when **i** is a negative number, it means to access the **-i** member from the end of the sequence. If **i** exceeds the sequence number range of **A**, it will return a null value.



The value of the fourth member in the sequence: Mary



The value of the last member in the sequence: Lucy

8.2.2 Member access of a sequence

【Example 8.2】Suppose that in example 8.1, Bill, the third patient, is about to finish his consultation, and the hospital is ready to remind the next patient behind him to be ready by radio. Then how to get the name of the next patient? If the hospital is going to give a small gift to the last patient in the queue in order to appease the lower ranking patients, how can we quickly find the last patient in the queue?

	A	B
1	[James, Kate, Bill, Mary, Lucy]	/Define sequence A1 with constant construction method
2	=A1(4)	/Get the value of the fourth member in the sequence: Mary
3	=A1.m(4)	/Same as A2
4	=A1.m(-1)	/Reversely access the value of the last member in the sequence: Lucy

A2	Value
	Mary

A3	Value
	Mary

A4	Value
	Lucy

8.2.2 Member access of a sequence

【Example 8.3】Suppose in example 8.1, the hospital mistakenly registered the name of the fourth patient, Marie, as Mary. How to quickly correct the patient's name?

8.2.2 Member access of a sequence

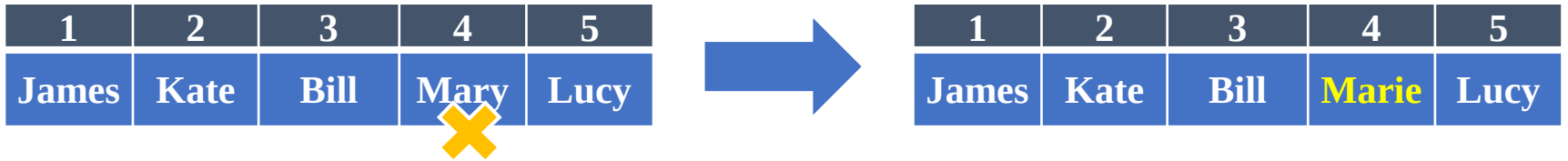
【Example 8.3】Suppose in example 8.1, the hospital mistakenly registered the name of the fourth patient, Marie, as Mary. How to quickly correct the patient's name?

Solution

You can assign values directly to the sequence members in the specified position.

In SPL, the expression can be written as **A(i)=x**. Where **A** represents the sequence, **i** represents the sequence number of the member to be accessed, and **x** represents the new value to be assigned to the member.

Reassign a value to the fourth member



8.2.2 Member access of a sequence

【Example 8.3】Suppose in example 8.1, the hospital mistakenly registered the name of the fourth patient, Marie, as Mary. How to quickly correct the patient's name?

	A	B
1	[James, Kate, Bill, Mary, Lucy]	/Define sequence A1 with constant construction method
2	>A1(4)="Marie"	/Modify the value of the fourth member to: Marie

After execution, the sequence in A1 is modified:

A1	Index	Member
	1	James
	2	Kate
	3	Bill
	4	Marie
	5	Lucy

8.2.2 Member access of a sequence

【Example 8.4】Suppose that in example 8.1, the hospital has written the fourth patient and the fifth patient in wrong order, how to quickly correct it?

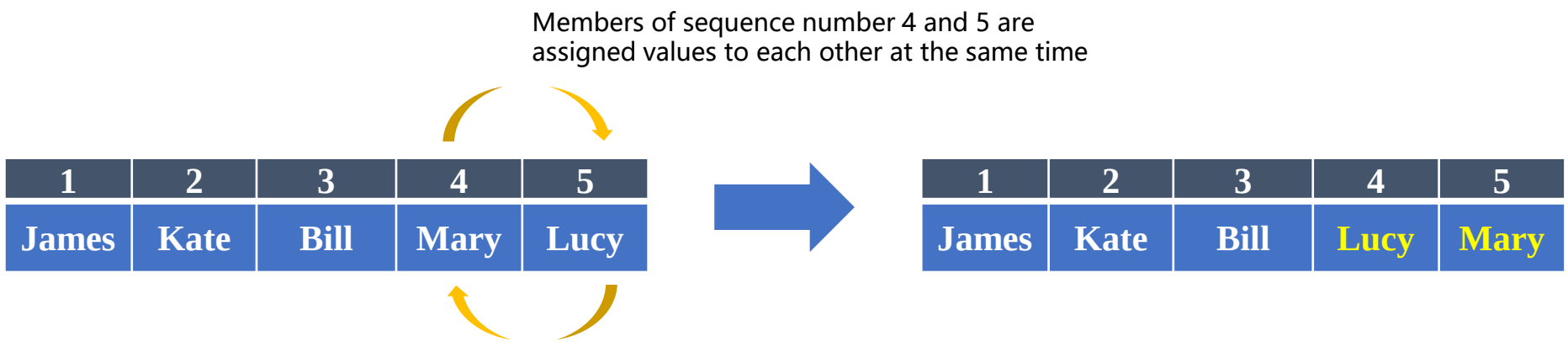
8.2.2 Member access of a sequence

【Example 8.4】Suppose that in example 8.1, the hospital has written the fourth patient and the fifth patient in wrong order, how to quickly correct it?

Solution

You can assign values to multiple referenced sequence members at the same time.

In SPL, multiple members of sequence **A** can be assigned values in the form of function **A(p)=X**, where **p** is a sequence of sequence numbers and **X** is a sequence. This means that the members of sequence **A** corresponding to sequence **p** are assigned to the data in sequence **X** in turn, where the sequence length of **p** and **X** must be equal, and the sequence numbers in **p** cannot exceed the length of **A**.



8.2.2 Member access of a sequence

【Example 8.4】Suppose that in example 8.1, the hospital has written the fourth patient and the fifth patient in wrong order, how to quickly correct it?

	A	B
1	[James, Kate, Bill, Mary, Lucy]	/Define sequence A1 with constant construction method
2	>A1([4, 5])=A1([5, 4])	/Swap the values of the fourth and fifth members of the sequence

After execution, the sequence in A1 is modified:

A1	Index	Member
	1	James
	2	Kate
	3	Bill
	4	Lucy
	5	Mary

8.2.3 Add a member to the end of a sequence

Add a member to the end of a sequence

- To add a member to the end of sequence A, you can use the expression **>A=A|x**. In fact, this method is to assign new values to A instead of directly changing the original sequence itself.
- The modification and maintenance of the sequence will be described in detail in the next chapter.

	A	B
1	[a, b, c]	/Define a sequence of numbers with constant construction method
2	>A1=A1|"new"	/Update A1, adding a member to the end of the sequence

A1	Index	Member
	1	a
	2	b
	3	c

Add a member

A1	Index	Member
	1	a
	2	b
	3	c
	4	new

8.2.3 Add a member to the end of a sequence

【Example 8.5】Suppose that in example 8.1, two new patients, Ella and David, have come in succession. Please list the waiting queue after they came in.

8.2.3 Add a member to the end of a sequence

【Example 8.5】Suppose that in example 8.1, two new patients, Ella and David, have come in succession. Please list the waiting queue after they came in.

Solution

Add members to the end of the sequence in turn. In order to note the change of the number of members in the sequence, the change of sequence length in the process of sequence change is calculated in column B. The total number of members in sequence A can be obtained by using function **A.len()**, which is the sequence length of sequence A.

	A	B	C
1	[James, Kate, Bill, Mary, Lucy]	=A1.len()	/Using cell constant to construct sequence, constant sequence can be displayed directly
2	>A1=A1|"Ella"	=A1.len()	/Add a member Ella to the end of the sequence
3	>A1=A1|"David"	=A1.len()	/Add a member David to the end of the sequence

A1	Index	Member
	1	James
	2	Kate
	3	Bill
	4	Mary
	5	Lucy

B1	Value
	5

A1	Index	Member
	1	James
	2	Kate
	3	Bill
	4	Mary
	5	Lucy
	6	Ella

B2	Value
	6

A1	Index	Member
	1	James
	2	Kate
	3	Bill
	4	Mary
	5	Lucy
	6	Ella
	7	David

B3	Value
	7

Chapter 8
Sequence

8.3 Loop and calculation of sequence

8.3.1 Execute the loop with for A

for A

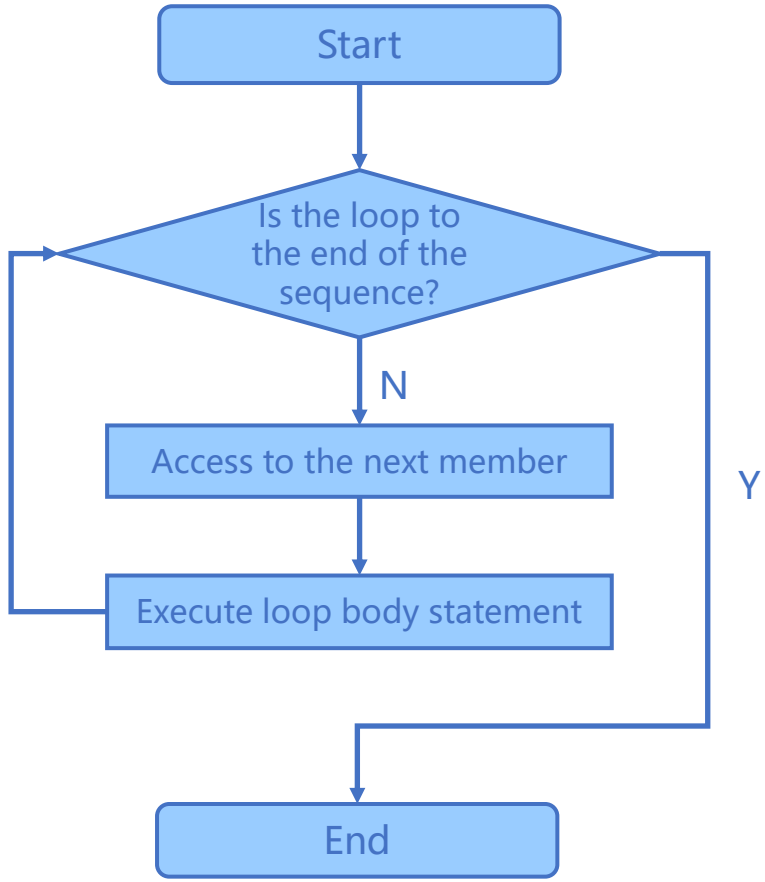
- The for A loop of a sequence, that is, the repetitive calculation is performed for each member of the sequence.

for A loop

	A	B
1	[a, b, c]	
2	for A1	>output(A2)

Output result:

```
a
b
c
```



8.3.1 Execute the loop with for A

【Example 8.6】It is known that the date of birth of all members of an R & D team are: 1978-04-25, 1992-05-28, 1983-06-16, 1988-08-08, 1992-05-17, 1986-12-11, find the age of all members. (based on the current date of June 6, 2020)

8.3.1 Execute the loop with for A

【Example 8.6】It is known that the date of birth of all members of an R & D team are: 1978-04-25, 1992-05-28, 1983-06-16, 1988-08-08, 1992-05-17, 1986-12-11, find the age of all members. (based on the current date of June 6, 2020)

Solution

For this problem, we can save the birth date of all members of the R & D team as a sequence, and then use the loop calculation for the sequence to calculate the age of all members in turn.

for A: Loop through the members of the sequence, the value of the main cell is the current member of the sequence A in turn.

	A	B	C	D
1	[1978-04-25, 1992-05-28, 1983-06-16, 1988-08-08, 1992-05-17, 1986-12-11]	[]		/A1 stores the birth date sequence, and B1 defines an empty sequence to store the result
2	for A1	=age(A2)	>B1=B1|B2	/Calculate the current age according to the date of birth of the group member and add a member to B1

B1	Index	Member
	1	42
	2	27
	3	36
	4	31
	5	28
	6	33

8.3.1 Execute the loop with for A

【Example 8.7】In example 8.6, if the team leader is 33 years old and there is no members of the same age in the group, and the member's sequence number corresponding to the birth date in the sequence is the employee ID of the member. Find the employee ID of the team leader.

8.3.1 Execute the loop with for A

【Example 8.7】In example 8.6, if the team leader is 33 years old and there is no members of the same age in the group, and the member's sequence number corresponding to the birth date in the sequence is the employee ID of the member. Find the employee ID of the team leader.

Solution

To solve this problem, you must be able to obtain the sequence number of a sequence member while looping through it. Because for A is a sequential loop, the sequence number of the member is equal to the current loop number.

#A: Get the current loop number, where A is the main cell of the loop body.

	A	B	C	D
1	[1978-04-25, 1992-05-28, 1983-06-16, 1988-08-08, 1992-05-17, 1986-12-11]	0		/A1 stores the birth date sequence, and B1 stores the result(initialized as 0)
2	for A1	=age(A2)		/Calculate the current age according to the date of birth of the group member
3		if B2==33>B1=#A2		/If the age is 33, store the result in B1

B1	Value
	6

8.3.1 Execute the loop with for A

【Example 8.8】In example 8.6, find the employee ID of the oldest member in the group.

8.3.1 Execute the loop with for A

【Example 8.8】In example 8.6, find the employee ID of the oldest member in the group.

Solution

When solving the problem, you need to save the calculation results in each cycle and continue to use them in the next cycle. The variables used to save the results of the loop should be defined outside the loop body and the initial values should be set.

	A	B	C	D
1	[1978-04-25, 1992-05-28, 1983-06-16, 1988-08-08, 1992-05-17, 1986-12-11]	0	0	/A1 stores the birth date sequence, and B1 stores the maximum age, C1 stores the corresponding employee ID
2	for A1	=age(A2)		/Calculate the current age according to the date of birth of the group member
3		if B2>B1	>B1=B2	/B1 records the maximum age of the group
4			>C1=#A2	/C1 records the employee ID of the oldest group member

C1	Value
	1

8.3.1 Execute the loop with for A

【Example 8.9】How many 3-digit numbers(each digit is different) can be made up of 2, 3, 6, 9 and what are they?

8.3.1 Execute the loop with for A

【Example 8.9】How many 3-digit numbers(each digit is different) can be made up of 2, 3, 6, 9 and what are they?

Solution

The numbers used in the example are not continuous, so we need to use a sequence to store them and then loop through the sequence to calculate. The results can be stored in a sequence, and each time a qualified 3-digit number is found, it is added to the result sequence. The count can be calculated by the length of the result sequence.

	A	B	C	D	E	F
1	[2, 3, 6, 9]	[]				/Prepare the number sequence, and define the empty sequence in B1 to store the result
2	for A1	for A1	for A1	if A2!=B2 && A2!=C2 && B2!=C2	>B1=B1|(A2*100+B2*10+C2)	/Loop through the digits to find a 3-digit number with different digits and store it in B1
3	=B1.len()					/Output the total number of the sequence

B1	Index	Member
	1	236
	2	239
	3	263
	4	269
	5	293
	6	296
	7	326
	…	…

A3	Value
	24

8.3.2 Aggregation

Aggregation

- In the loop, it is sometimes necessary to calculate the aggregate results for all members, such as counting, sum, average, Max/Min, etc.
- With the **for A** loop, you can calculate the aggregate result by looping through each member of the sequence.

Count: calculate the total number of sequence members

	A	B
1	[a, b, c]	0
2	for A1	>B1=B1+1

B1	Value
	3

8.3.2 Aggregation

【Example 8.10】It is known that the scores of several students in a certain test are as follows: 66, null, 82, 96. Someone did not take the test for some reason, so he/she did not get any score. Please calculate their total score and average score. Student who is absent from the test will not participate in the statistics.

8.3.2 Aggregation

【Example 8.10】It is known that the scores of several students in a certain test are as follows: 66, null, 82, 96. Someone did not take the test for some reason, so he/she did not get any score. Please calculate their total score and average score. Student who is absent from the test will not participate in the statistics.

Solution

To calculate the average score, we need to count the total score and the total number of students participating in the test.

	A	B	C	D
1	[66, null, 82, 96]	0	0	/Generate score sequence, B1 and C1 store the total number of students and total score respectively
2	for A1	if A2 != null	>B1=B1+1	/If there is a score, accumulate the total number of students
3			>C1=C1+A2	/Accumulate the total score
4	=round(C1/B1, 2)			/Calculate the average score and keep 2 decimal places

B1	Value
	3

C1	Value
	244

A4	Value
	81.33

Note

- A more convenient way to calculate aggregations is to use **aggregate functions**, which will be explained in the next chapter.

8.3.3 Comprehensive calculation of a sequence

【Example 8.11】It is known that the date of birth of all members of an R & D team are: 1978-04-25, 1992-05-28, 1983-06-16, 1988-08-08, 1992-05-17, 1986-12-11. If the last member of the sequence is the birthday of the team leader, please find the employee IDs of all members who are older than the team leader.

8.3.3 Comprehensive calculation of a sequence

【Example 8.11】It is known that the date of birth of all members of an R & D team are: 1978-04-25, 1992-05-28, 1983-06-16, 1988-08-08, 1992-05-17, 1986-12-11. If the last member of the sequence is the birthday of the team leader, please find the employee IDs of all members who are older than the team leader.

Solution

It is recommended to use the **for n** form of loop. The loop number can be used as the sequence number to obtain the members of the sequence to calculate the age. In this way, it is convenient to exclude a meaningless cycle in the for A loop in which the team leader makes a comparison with himself.

	A	B	C	D
1	[1978-04-25, 1992-05-28, 1983-06-16, 1988-08-08, 1992-05-17, 1986-12-11]	=age(A1.m(-1))	[]	/A1 defines the birth date sequence, B1 calculates the age of the team leader, and C1 stores the result sequence
2	for A1.len()-1	=age(A1(A2))		/Calculate the current age according to the date of birth of the group member
3		if B2>B1	>C1=C1|#A2	/If the age is greater than the team leader, the corresponding employee ID will be output

C1	Index	Member
	1	1
	2	3

Note

- When performing loop calculation with sequence, you can choose **for A** or **for n** according to your needs, and the latter is more flexible.
- When using **for n** to perform loop calculation with sequence, it is often necessary to obtain the members from the sequence according to the sequence number before calculating. Note that the value of n should not exceed the length of the sequence, otherwise an error of index overrun will be reported.

8.3.3 Comprehensive calculation of a sequence

【Example 8.12】Suppose the scores of a subject in a class are as follows:82, 77, 65, 85, 73, 91, 68, 76, 65. Rank the scores in descending order.

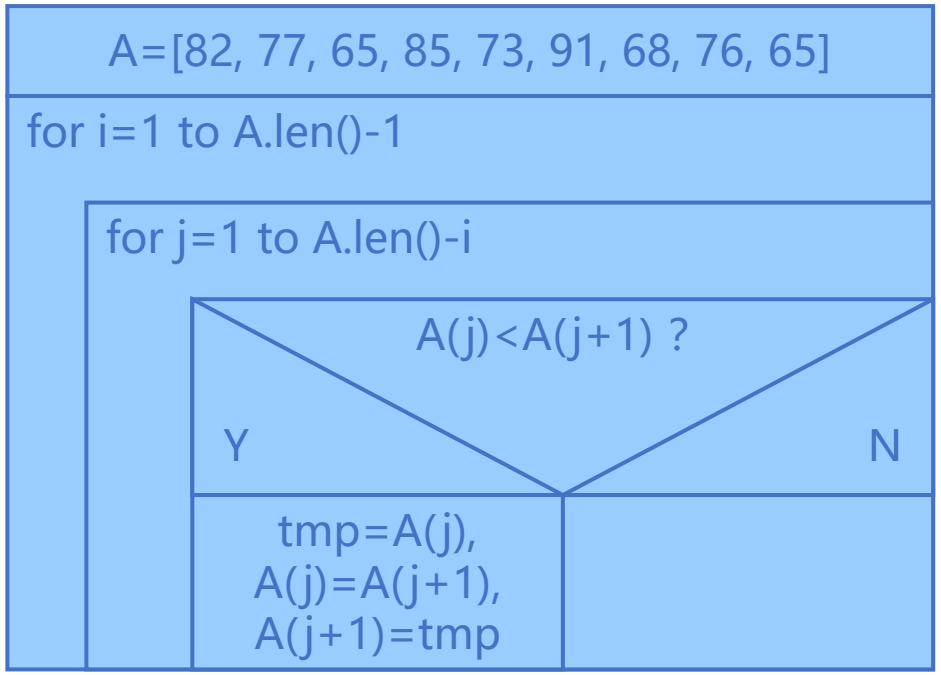
8.3.3 Comprehensive calculation of a sequence

【Example 8.12】Suppose the scores of a subject in a class are as follows:82, 77, 65, 85, 73, 91, 68, 76, 65. Rank the scores in descending order.

Solution

Put the scores in the sequence first. Each pair of adjacent scores is compared in a loop. If the front one is less than the latter, the positions of the two are exchanged. After each round of comparison, the lowest score in this round will be placed at the bottom. The end point of the next round of comparison is set at the top of the lowest score obtained in the previous round.

In ascending (descending) order, the smaller (larger) elements will slowly "float" to the edge of the sequence through exchange, just like the carbon dioxide bubble in carbonated drinks will eventually float to the surface, so the name of the algorithm is "bubble sorting".



8.3.3 Comprehensive calculation of a sequence

【Example 8.12】Suppose the scores of a subject in a class are as follows:82, 77, 65, 85, 73, 91, 68, 76, 65. Rank the scores in descending order.

	A	B	C	D	E
1	[82,77,65,85,73,91,68,76,65]				/The original data is stored in the sequence
2	for A1.len()-1	for A1.len()-A2	if A1(B2)<A1(B2+1)	>A1([B2, B2+1])=A1([B2+1, B2])	/Loops over each pair of adjacent values, swapping positions if the front is less than the back

A1	Index	Member
	1	91
	2	85
	3	82
	4	77
	5	76
	6	73
	7	68
	8	65
	9	65

Note

- This example also uses the method of **for n** (n is a positive integer) to deal with the cyclic calculation of the sequence.
- The value of n in B2 changes with the value of A2.

Chapter 8
Sequence

8.4 Common sequences

8.4.1 Integer sequence

Integer sequence

- An **integer sequence** is a sequence whose members are all integers. The sequence of numbers in SPL is more often used as the sequence of sequence numbers of members.
- It supports some special calculations that require that the sequence members must be of numerical type, such as sum or average.
- The member of the sequence will not be **(null)** , so the length of the sequence **A.len()** is equal to the total count calculation **A.count()**. The calculation of **A.count()** will be explained in the next chapter.

Integer sequence

Index	Member
1	91
2	85
3	82
4	77
5	76
6	73
7	68
8	65
9	65

The members are all integers

【Example 8.13】Suppose the test scores of a 5-people group are: 80, 62, 53, 60, 45. Calculate the average score of the group.

8.4.1 Integer sequence

【Example 8.13】Suppose the test scores of a 5-people group are: 80, 62, 53, 60, 45. Calculate the average score of the group.

Solution

Traverse all members of the sequence to calculate the total score by accumulation, and then divide it by the total number of people (i.e., the length of the sequence) to get the average score.

	A	B	C
1	[80, 62, 53, 60, 45]	0	/Initializing sequence, B1 is used to record the total score
2	for A1	>B1=B1+A2	/Accumulate total score in loop
3	=B1/A1.len()		/Calculate the average score

A3	Value
	60.0

【Example 8.14】Suppose that in example 8.13, if the test score reaches 60, it will be regarded as qualified; otherwise, it will be regarded as unqualified. Find the qualified rate of the group.

8.4.1 Integer sequence

【Example 8.14】Suppose that in example 8.13, if the test score reaches 60, it will be regarded as qualified; otherwise, it will be regarded as unqualified. Find the qualified rate of the group.

Solution

Traverse all members of the sequence, and add 1 to the result if the member is 60 or above, otherwise omit the member. Finally divide the result by the total number of people (i.e. the length of the sequence) to get the qualified rate.

	A	B	C	D
1	[80,62,53,60,45]	0		/Initializing sequence, B1 is used to count the number of qualified people
2	for A1	if A2>=60	>B1=B1+1	/If the score reaches 60, the qualified number will be increased by 1, otherwise it will not be counted
3	=B1/A1.len()	=string(A3,"#.0 %")		/Calculate the qualified rate and display it as a percentage in B3

A3	Value
	0.6

B3	Value
	60.0 %

8.4.2 String sequence

String sequence

- A sequence consisting entirely of string members is a **string sequence**.
- Because the members of string sequence are strings, special functions of strings can be called to perform some special calculations, such as string splitting and merging.

String sequence

1	2	3	4	5
apple	orange	banana	grape	peach

All members are strings

8.4.2 String sequence

【Example 8.15】There are some names: Dennis, Rose, John, James, Robert, Jimmy. Output the names with initial letter(Capital by default) J.

8.4.2 String sequence

【Example 8.15】There are some names: Dennis, Rose, John, James, Robert, Jimmy. Output the names with initial letter(Capital by default) J.

Solution

Traverse all members of the sequence, intercept the first letter, judge whether it is the capital letter J.

	A	B	C	D
1	[Dennis, Rose, John, James, Robert, Jimmy]	[]		
2	for A1	if left(A2, 1) == "J"	>B1=B1|A2	/Names with initial letter of J are stored in B1

B1	Index	Member
	1	John
	2	James
	3	Jimmy

8.4.2 String sequence

【Example 8.16】In example 8.15, output the names with the last letter (lowercase by default) s.

8.4.2 String sequence

【Example 8.16】In example 8.15, output the names with the last letter (lowercase by default) s.

Solution

Traverse all members of the sequence, intercept the last letter, and judge whether it is the lowercase letter s.

	A	B	C	D
1	[Dennis, Rose, John, James, Robert, Jimmy]	[]		
2	for A1	if right(A2, 1) == "s"	>B1=B1|A2	/Names with last letter of s are stored in B1

B1	Index	Member
	1	Dennis
	2	James

8.4.2 String sequence

【Example 8.17】In example 8.15, the full names of these people are: Dennis White, Rose Smith, John Davis, James Smith, Robert Jones, Jimmy Thomas. Only one of them, whose surname begins with D, is the leader of the others. Find who the leader is.

8.4.2 String sequence

【Example 8.17】In example 8.15, the full names of these people are: Dennis White, Rose Smith, John Davis, James Smith, Robert Jones, Jimmy Thomas. Only one of them, whose surname begins with D, is the leader of the others. Find who the leader is.

Solution

Traversing all members of the sequence, first with a space as a separator to intercept the last part of the string, and then intercept the first letter, if it is the capital letter D, then store the full name of the person and end the loop.

	A	B	C	D
1	[Dennis White, Rose Smith, John Davis, James Smith, Robert Jones, Jimmy Thomas]			
2	for A1	if mid(A2, pos@z(A2, " ")+1, 1)=="D"	>B1=A2	/Stores the full name of the first person whose last name begins with D
3			break	/Break the loop

B1	Value
	John Davis

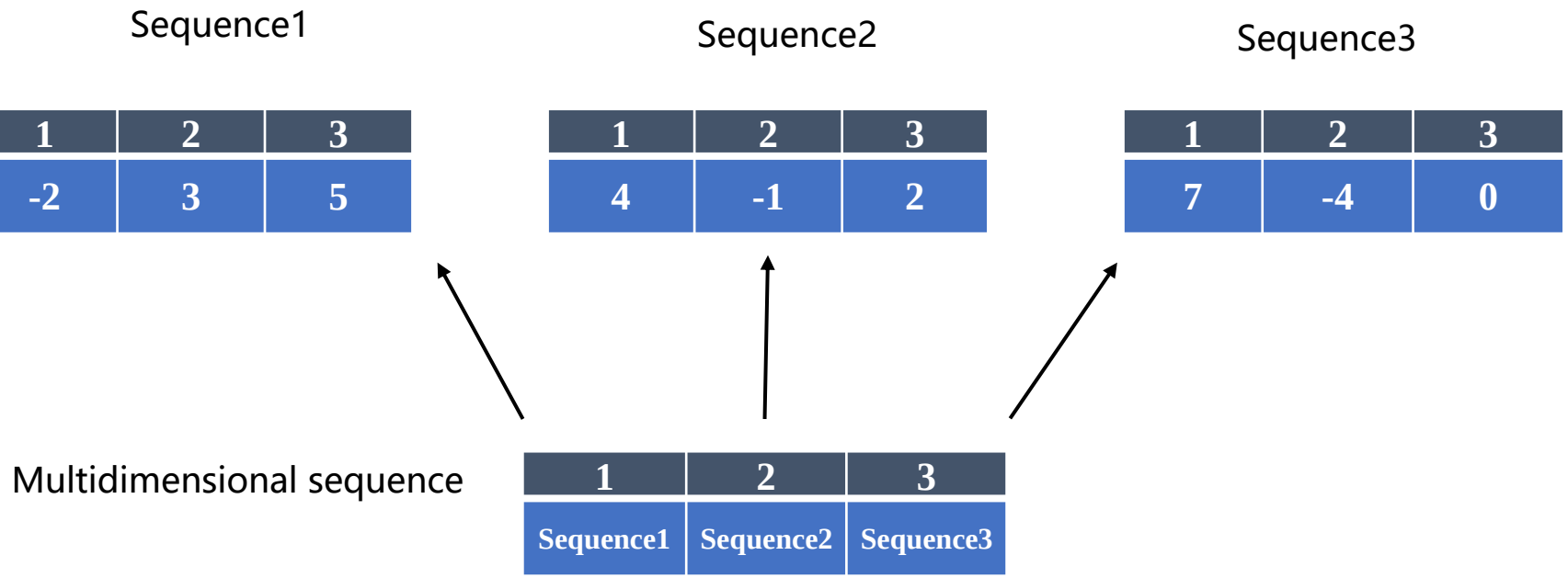
Chapter 8
Sequence

8.5 Multidimensional sequence

8.5.1 What is multidimensional sequence?

Multidimensional sequence

- Members in a sequence can be any data type. When the members of a sequence are also sequences, the sequence is called **multidimensional sequence**.



8.5.1 What is multidimensional sequence?

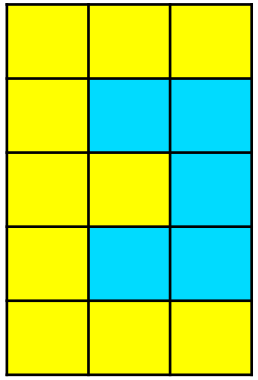
For example, a matrix can be considered as a multidimensional sequence.

Use SPL to generate matrix : $\begin{bmatrix} 11 & 23 \\ 59 & 17 \end{bmatrix}$

	A	B
1	[[11, 23] , [59, 17]]	/Using constant construction method to create matrix

In real life, the dot matrix used for text display can also be represented by multi-dimensional sequence.

Use dot matrix to display text:E



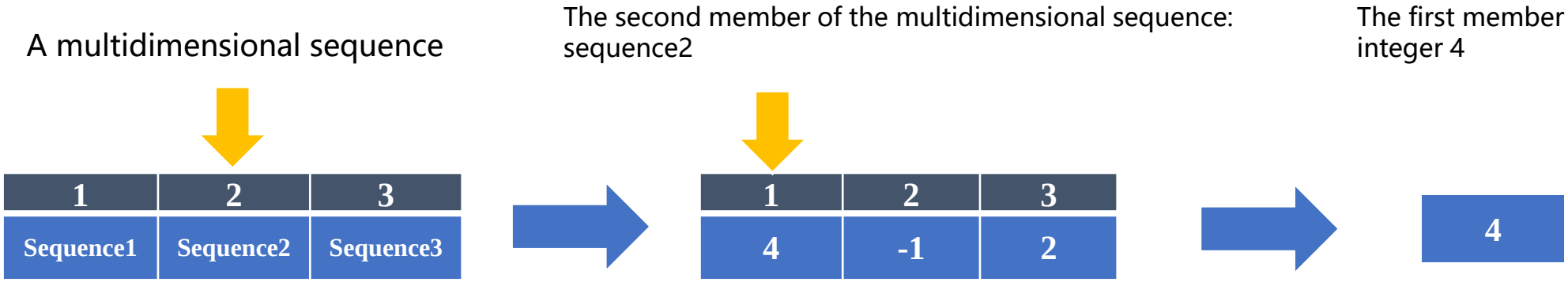
This dot matrix can be saved in SPL as:

	A
1	[[1, 1, 1], [1, 0, 0], [1, 1, 0], [1, 0, 0], [1, 1, 1]]

8.5.2 Access to multidimensional sequence members

Access a multidimensional sequence

- The member access of a multidimensional sequence is **hierarchical**. First access the sequence as a member, and then access the members in the member sequence.



8.5.2 Access to multidimensional sequence members

Accessing the elements of matrix step by step in SPL

	A	B
1	[[11, 23] , [59, 17]]	/Generating multidimensional sequence by constant construction method:$\begin{bmatrix}11 & 23\\59 & 17\end{bmatrix}$
2	=A1(1)(1)	/Access the first member of the first sequence member:11
3	=A1(2)(2)	/Access the second member of the second sequence member:17

Output the dot matrix text of the previous example in SPL : E

	A	B	C	D
1	[[1, 1, 1], [1, 0, 0], [1, 1, 0], [1, 0, 0], [1, 1, 1]]			
2	for A1	=""		
3		for A2	if B3==1	>B2=B2+"@"
4			else	>B2=B2+" "
5		>output(B2)		

Output result:

```
@@@
@
@@
@
@@@
```

8.5.3 The use of multidimensional sequences

【Example 8.18】Use multidimensional sequence to implement the addition calculation of matrix $\begin{bmatrix} 8 & 3 \\ -4 & -5 \end{bmatrix}$ and matrix $\begin{bmatrix} 9 & 7 \\ -1 & 2 \end{bmatrix}$.

8.5.3 The use of multidimensional sequences

【Example 8.18】Use multidimensional sequence to implement the addition calculation of matrix $\begin{bmatrix} 8 & 3 \\ -4 & -5 \end{bmatrix}$ and matrix $\begin{bmatrix} 9 & 7 \\ -1 & 2 \end{bmatrix}$.

Solution

Traverse each numerical member of each sequence member in A1, and then use the member's sequence number to find the corresponding numerical member in B1. After the addition calculation, save the result in the corresponding position in C1.

	A	B	C	D
1	[[8, 3], [-4, -5]]	[[9, 7], [-1, 2]]	[[0, 0], [0, 0]]	/A1:$\begin{bmatrix} 8 & 3 \\ -4 & -5 \end{bmatrix}$, B1:$\begin{bmatrix} 9 & 7 \\ -1 & 2 \end{bmatrix}$, C1 saves the result
2	for A1	for A2	>C1(#A2)(#B2)=B2+B1(#A2)(#B2)	/Using nested loop, implement the addition calculation of matrix by position.

C1	Index	Member
	1	[17, 10]
	2	[-5, -3]

i.e. the calculation result: $\begin{bmatrix} 17 & 10 \\ -5 & -3 \end{bmatrix}$

8.5.3 The use of multidimensional sequences

【Example 8.19】Matthew wants to show his love to Fannie who works in a building on Valentine's day. The building manager promised to make the windows of the building heart-shaped by lighting and turning off the lights on Valentine's day. The building has 20 floors with 25 windows on each floor. How to record the rooms with lights on and off?

8.5.3 The use of multidimensional sequences

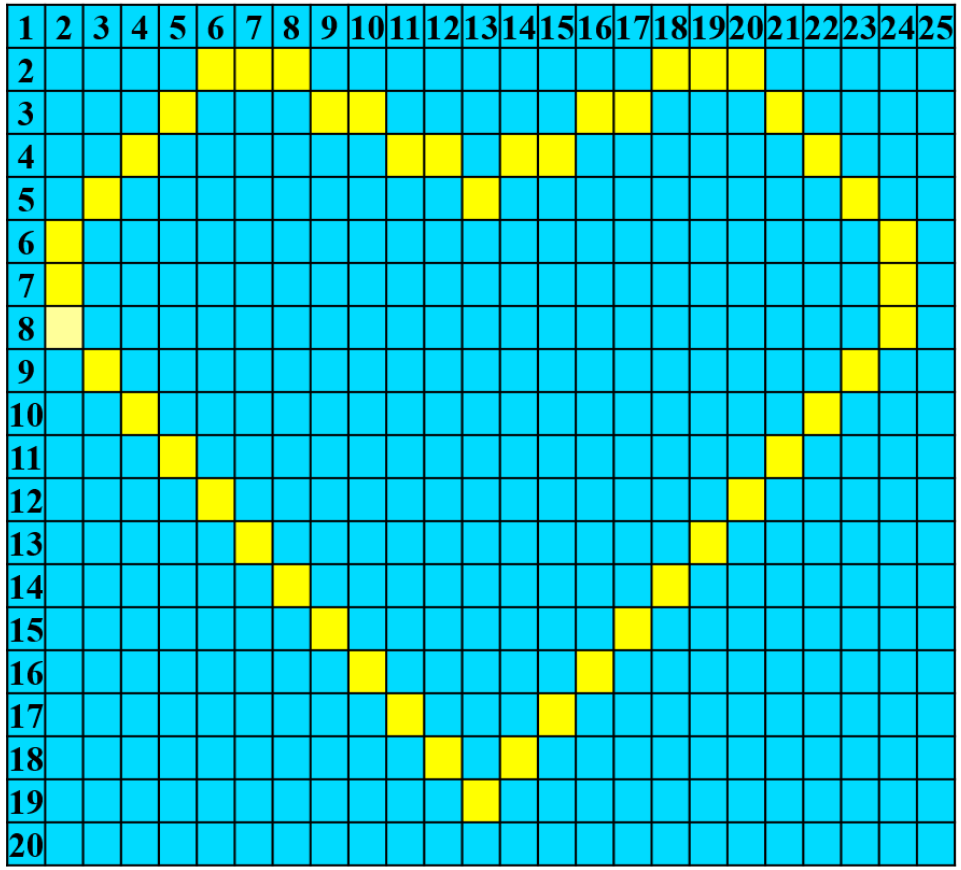
【Example 8.19】Matthew wants to show his love to Fannie who works in a building on Valentine's day. The building manager promised to make the windows of the building heart-shaped by lighting and turning off the lights on Valentine's day. The building has 20 floors with 25 windows on each floor. How to record the rooms with lights on and off?

Solution

To deal with such problems, it is undoubtedly the most convenient way to draw an intuitive schematic diagram.

It can be seen from the schematic diagram that this problem is actually the same kind of problem as dot matrix displaying text. So you can also use multidimensional sequences for processing.

Because there are few rooms with lights on in each row, we can improve the previous method: let the sequence members in the multi-dimensional sequence record only the corresponding values of the lighted rooms in each row.



8.5.3 The use of multidimensional sequences

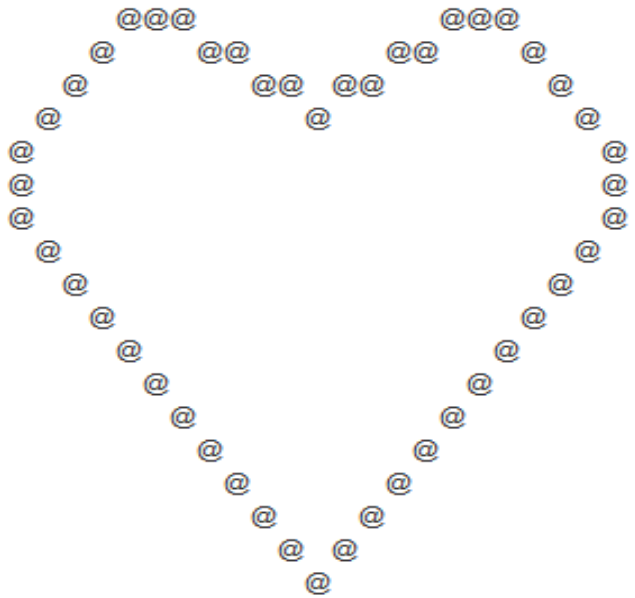
【Example 8.19】Matthew wants to show his love to Fannie who works in a building on Valentine's day. The building manager promised to make the windows of the building heart-shaped by lighting and turning off the lights on Valentine's day. The building has 20 floors with 25 windows on each floor. How to record the rooms with lights on and off?

	A	B	C	D	E
1	[[], [6, 7, 8, 18, 19, 20], [5, 9, 10, 16, 17, 21], [4, 11, 12, 14, 15, 22], [3, 13, 23], [2, 24], [2, 24], [2, 24], [3, 23], [4, 22], [5, 21], [6, 20], [7, 19], [8, 18], [9, 17], [10, 16], [11 15], [12, 14], [13], []]	=[[0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25, [0]*25]			/A1 is a multi-dimensional matrix for recording the room with lights on in each row, and B1 is used to restore it to a dot matrix including the lighting status of all rooms
2	for A1	for A2	>B1(#A2)(B2)=1		/the dot matrix to restore the lights on and off status of all rooms
3	for B1	=""			/Use the method in the example to verify the results displayed
4		for A3	if B4==1	>B3=B3+"@"	
5			else	>B3=B3+" "	
6		>output(B3)			

8.5.3 The use of multidimensional sequences

【Example 8.19】Matthew wants to show his love to Fannie who works in a building on Valentine's day. The building manager promised to make the windows of the building heart-shaped by lighting and turning off the lights on Valentine's day. The building has 20 floors with 25 windows on each floor. How to record the rooms with lights on and off?

The output result in SPL:



Note

- In practical use, some appropriate adjustments should be made according to the different display sizes of the elements used to display lights on or off.

Chapter 8
Sequence

8.6 Exercises

8.6 Exercises

【Exercise 8.1】There are salary records of employees (4500, 6000, 5500, 8000, 12000, 8500, 15000, 7000, 18000, 25000, 20000) and individual income tax rate table (as shown in the figure). Assuming that the maximum salary of the employees before tax is 25000 / month, find the salary of each employee after tax.

【 Exercise 8.2】There is a string filename@xxxx.com.rar. Replace "@xxxx.com" with "@yyyy.net".

【Exercise 8.3】If you want to take the train from Handan to Shenyang, the direct ticket has been sold out and you need to transfer once. Considering the time requirement of arriving in Shenyang, some trains in three cities are available for transit: Qinhuangdao, Tangshan and Baoding. Detailed information is recorded in the sequences (data format in A1 is train number - transfer station - arrival time, and data format in B1 is train number - transfer station - departure time). List all feasible schemes (transfer train numbers and transfer station).
[[G6272, Baoding, 11:28:00], [G6272, Qinhuangdao, 14:42:00], [G6272, Tangshan, 13:46:00]] and [[G1290, Baoding, 14:16:00], [G1221, Qinhuangdao, 15:49:00], [G1221, Tangshan, 15:01:00]]

【Exercise 8.4】Both Martin and Helen have outstanding physics scores. They have four recent physical simulation test scores (Mike: 97, 99, 90, 86; Helen: 93, 94, 91, 96). The weights of the four tests are 0.23, 0.30, 0.21 and 0.26 according to the difficulty coefficient. Based on this, who should be sent to participate in the competition?

【Exercise 8.5】There are three members in a website development team, each with different division of work: Dennis (Photoshop), Rose (HTML, JavaScript), and Jimmy (PHP, MySQL). Now Rose wants to leave and Mike, a new recruit, will take over her work. What's the latest work arrangement?

【Exercise 8.6】There are two sequences. The first is the names (Mike, Tom, Andy) and the second is the addresses (London, Manchester, Liverpool). The two sets of data are one-to-one corresponding. Merge the contents of the two sequences and write them to a file.

【Exercise 8.7】There is an anonymous and ordered physics test result of a learning group (98, 97, 93, 92, 89, 89, 87, 83, 79, 76, 74, 74, 68, 65, 63, 62, 59, 58, 58, 52), George wants to know the ranking of his score (79) (assuming that no one else has the same score as George), how to find it quickly?

【Exercise 8.8】Dennis White, Rose Smith, John Davis, James Smith, Robert Jones, Jimmy Thomas. Among these people, there are only two brother/sister with the same surname. Find these two people.

【Exercise 8.9】In a movie, the agent has to meet the informant. There are three pre-arranged meeting places: Tokyo, London or Yaren. One day, the agent received a telegram from the informant, which contained five strings of numbers: 10001, 01010, 00100, 00100, 00100. Where does the informant suggest to meet?

8.6 Exercises

【Exercise 8.1】There are salary records of employees (4500, 6000, 5500, 8000, 12000, 8500, 15000, 7000, 18000, 25000, 20000) and individual income tax rate table (as shown in the figure). Assuming that the maximum salary of the employees before tax is 25000 / month, find the salary of each employee after tax.

Individual income tax rate table:

1. 0-5000, tax rate is 0%
2. 5000-8000, tax rate is 3%
3. 8000-17000, tax rate is 10%
4. 17000-30000, tax rate is 20%

8.6 Exercises

【Exercise 8.1】There are salary records of employees (4500, 6000, 5500, 8000, 12000, 8500, 15000, 7000, 18000, 25000, 20000) and individual income tax rate table (as shown in the figure). Assuming that the maximum salary of the employees before tax is 25000 / month, find the salary of each employee after tax.

Individual income tax rate table:

1. 0-5000, tax rate is 0%
2. 5000-8000, tax rate is 3%
3. 8000-17000, tax rate is 10%
4. 17000-30000, tax rate is 20%

	A	B	C	D
1	[4500, 6000, 5500, 8000, 12000, 8500, 15000, 7000, 18000, 25000 20000]	=(3000-0)*0.03	=(12000-3000)*0.1	/salary records
2	for A1	if A2>5000 && A2<=8000	>A1(#A2)=A2-(A2-5000)*0.03	/Tax rate level 2
3		else if A2>8000 && A2<=17000	>A1(#A2)=A2-B1-(A2-8000)*0.10	/Tax rate level 3
4		else if A2>17000 && A2<=30000	>A1(#A2)=A2-B1-C1-(A2-17000)*0.20	/Tax rate level 4

A1	Index	Member
	1	4500
	2	5970.0
	3	5485.0
	4	7910.0
	5	11510.0
	6	8360.0
	7	14210.0
	8	6940.0
	9	16810.0
	10	22410.0
	11	18410.0

8.6 Exercises

【Exercise 8.2】There is a string filename@xxxx.com.rar. Replace "@xxxx.com" with "@yyyy.net".

8.6 Exercises

【Exercise 8.2】There is a string filename@xxxx.com.rar. Replace "@xxxx.com" with "@yyyy.net".

	A	B	C
1	filename@xxxx.com.rar		/Original string
2	=[mid(A1, 1, pos@z(A1, "@")-1), "@yyyy.net", mid(A1, pos@z(A1, "."))]		/Intercept the required substring and store it in the sequence
3	for A2	>B1=B1/A3	/Traverse the string sequence and reconcatenate.

B1	Value
	filename@yyyy.net.rar

【Exercise 8.3】If you want to take the train from Handan to Shenyang, the direct ticket has been sold out and you need to transfer once. Considering the time requirement of arriving in Shenyang, some trains in three cities are available for transit: Qinhuangdao, Tangshan and Baoding. Detailed information is recorded in the sequences (data format in A1 is train number - transfer station - arrival time, and data format in B1 is train number - transfer station - departure time). List all feasible schemes (transfer train numbers and transfer station).

[[G6272, Baoding, 11:28:00], [G6272, Qinhuangdao, 14:42:00], [G6272, Tangshan, 13:46:00]] and [[G1290, Baoding, 14:16:00], [G1221, Qinhuangdao, 15:49:00], [G1221, Tangshan, 15:01:00]]

8.6 Exercises

【Exercise 8.3】If you want to take the train from Handan to Shenyang, the direct ticket has been sold out and you need to transfer once. Considering the time requirement of arriving in Shenyang, some trains in three cities are available for transit: Qinhuangdao, Tangshan and Baoding. Detailed information is recorded in the sequences (data format in A1 is train number - transfer station - arrival time, and data format in B1 is train number - transfer station - departure time). List all feasible schemes (transfer train numbers and transfer station). [[G6272, Baoding, 11:28:00], [G6272, Qinhuangdao, 14:42:00], [G6272, Tangshan, 13:46:00]] and [[G1290, Baoding, 14:16:00], [G1221, Qinhuangdao, 15:49:00], [G1221, Tangshan, 15:01:00]]

	A	B	C	D	E
1	[[G6272, Baoding, 11:28:00], [G6272, Qinhuangdao, 14:42:00], [G6272, Tangshan, 13:46:00]]	[[G1290, Baoding, 14:16:00], [G1221, Qinhuangdao, 15:49:00], [G1221, Tangshan, 15:01:00]]	[]		/Transfer information
2	for A1	for B1	if A2(2) == B2(2) && time(A2(3)) < time(B2(3))	>C1=C1|(A2(1)/"->"/B2(1)/", Station : "/A2(2))	/Match the appropriate transfer number and transfer station and store the results

C1	Index	Member
	1	G6272->G1290, Station : Baoding
	2	G6272->G1221, Station : Qinhuangdao
	3	G6272->G1221, Station : Tangshan

【Exercise 8.4】Both Martin and Helen have outstanding physics scores. They have four recent physical simulation test scores (Mike: 97, 99, 90, 86; Helen: 93, 94, 91, 96). The weights of the four tests are 0.23, 0.30, 0.21 and 0.26 according to the difficulty coefficient. Based on this, who should be sent to participate in the competition?

8.6 Exercises

【Exercise 8.4】Both Martin and Helen have outstanding physics scores. They have four recent physical simulation test scores (Mike: 97, 99, 90, 86; Helen: 93, 94, 91, 96). The weights of the four tests are 0.23, 0.30, 0.21 and 0.26 according to the difficulty coefficient. Based on this, who should be sent to participate in the competition?

	A	B	C	D	E
1	[97, 99, 90, 86]	[93, 94, 91, 96]	[0.23, 0.30, 0.21, 0.26]	0	0
2	for A1	>D1=D1+A2*C1(#A2)	/D1 is Martin's score evaluation		
3	for B1	>E1=E1+A3*C1(#A3)	/E1 is Helen's score evaluation		

Martin:

D1	Value
	93.27

Helen:

E1	Value
	93.66

From the final result, Helen has a greater advantage and should be sent to participate in the competition.

8.6 Exercises

【Exercise 8.5】There are three members in a website development team, each with different division of work: Dennis (Photoshop), Rose (HTML, JavaScript), and Jimmy (PHP, MySQL). Now Rose wants to leave and Mike, a new recruit, will take over her work. What's the latest work arrangement?

8.6 Exercises

【Exercise 8.5】There are three members in a website development team, each with different division of work: Dennis (Photoshop), Rose (HTML, JavaScript), and Jimmy (PHP, MySQL). Now Rose wants to leave and Mike, a new recruit, will take over her work. What's the latest work arrangement?

	A	B	C	D
1	[[Dennis, Photoshop], [Rose, HTML], [Jimmy, PHP], [Jimmy, MySQL], [Rose, JavaScript]]			/Work assignment information of team members
2	for A1	if A2(1) == "Rose"	>A1(#A2)(1) = "Mike"	/Replace all Rose in the list with Mike

A1	Index	Member
	1	[Dennis, Photoshop]
	2	[Mike, HTML]
	3	[Jimmy, PHP]
	4	[Jimmy, MySQL]
	5	[Mike, JavaScript]

8.6 Exercises

【Exercise 8.6】There are two sequences. The first is the names (Mike, Tom, Andy) and the second is the addresses (London, Manchester, Liverpool). The two sets of data are one-to-one corresponding. Merge the contents of the two sequences and write them to a file.

8.6 Exercises

【Exercise 8.6】There are two sequences. The first is the names (Mike, Tom, Andy) and the second is the addresses (London, Manchester, Liverpool). The two sets of data are one-to-one corresponding. Merge the contents of the two sequences and write them to a file.

	A	B	C
1	[Mike, Tom, Andy]	[London, Manchester, Liverpool]	/Define sequence A1 and B1
2	for A1	>A1(#A2)=A2/":"/B1(#A2)	/Concatenate strings in turn by position
3	>file("D:/result.txt").export(A1)		/Write A1 to a file

Result:

```
Mike:London
Tom:Manchester
Andy:Liverpool
```

【Exercise 8.7】There is an anonymous and ordered physics test result of a learning group (98, 97, 93, 92, 89, 89, 87, 83, 79, 76, 74, 74, 68, 65, 63, 62, 59, 58, 58, 52), George wants to know the ranking of his score (79) (assuming that no one else has the same score as George), how to find it quickly?

8.6 Exercises

【Exercise 8.7】There is an anonymous and ordered physics test result of a learning group (98, 97, 93, 92, 89, 89, 87, 83, 79, 76, 74, 74, 68, 65, 63, 62, 59, 58, 58, 52), George wants to know the ranking of his score (79) (assuming that no one else has the same score as George), how to find it quickly?

	A	B	C	D	E
1	[98, 97, 93, 92, 89, 89, 87, 83, 79, 76, 74, 74, 68, 65, 63, 62, 59, 58, 58, 52]	79	1	=A1.len()	/A1 is the scores of all members, B1 is the score of George, C1 is the upper limit of the query range, and D1 is the lower limit of the query range
2	for C1<=D1	=int((C1+D1)/2)	=A1(B2)		/Calculate the median location of the upper and lower limits of the range, then get the median value
3		if C2==B1	>output(B2)	break	/If the median value is equal to the value to be checked, output it
4		else if C2<B1	>D1=B2-1		/If the median value is less than the value to be checked, update the lower limit of the range
5		else	>C1=B2+1		/If the median value is greater than the value to be checked, update the upper limit of the range

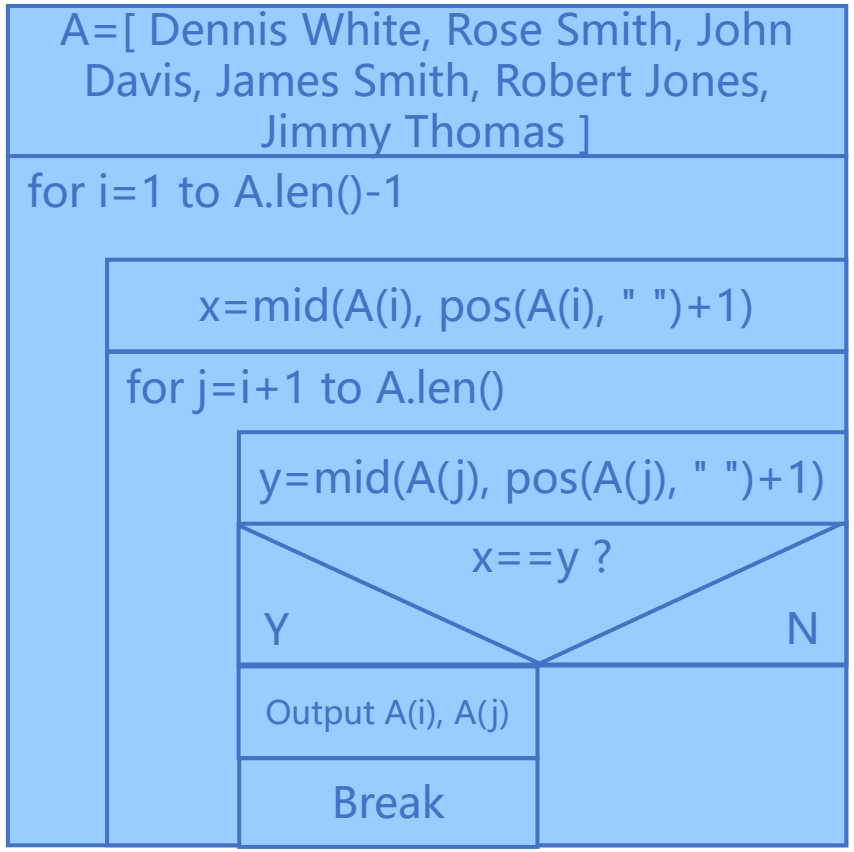
Result: 9

That is, George ranked ninth in the group.
If the code has no output, the score is not in the list.

【Exercise 8.8】Dennis White, Rose Smith, John Davis, James Smith, Robert Jones, Jimmy Thomas. Among these people, there are only two brother/sister with the same surname. Find these two people.

8.6 Exercises

【Exercise 8.8】Dennis White, Rose Smith, John Davis, James Smith, Robert Jones, Jimmy Thomas. Among these people, there are only two brother/sister with the same surname. Find these two people.



	A	B	C	D	E
1	[Dennis White, Rose Smith, John Davis, James Smith, Robert Jones, Jimmy Thomas]				
2	for A1.len()-1	=mid(A1(A2), pos(A1(A2), " ")+1)			/Get the surname in turn
3		for A2+1, A1.len()	=mid(A1(B3), pos(A1(B3), " ")+1)		/Compare backwards
4			if B2==C3	>B1=(A1(A2)/" and "/A1(B3))	/If there is the same surname, output two names
5				break A2	/Break

B1	Value
	Rose Smith and James Smith

【Exercise 8.9】In a movie, the agent has to meet the informant. There are three pre-arranged meeting places: Tokyo, London or Yaren. One day, the agent received a telegram from the informant, which contained five strings of numbers: 10001, 01010, 00100, 00100, 00100. Where does the informant suggest to meet?

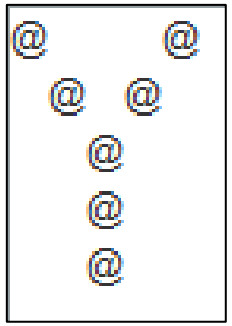
8.6 Exercises

【Exercise 8.9】In a movie, the agent has to meet the informant. There are three pre-arranged meeting places: Tokyo, London or Yaren. One day, the agent received a telegram from the informant, which contained five strings of numbers: 10001, 01010, 00100, 00100, 00100. Where does the informant suggest to meet?

Solution

These five suspicious number strings are obviously like a dot matrix. We can use the example in section 4 to show them and see the result.

Output result:



The letter "Y" is displayed, i.e. the contact location is in Yaren

	A	B	C	D
1	[[1, 0, 0, 0, 1], [0, 1, 0, 1, 0], [0, 0, 1, 0, 0], [0, 0, 1, 0, 0], [0, 0, 1, 0, 0]]			
2	for A1	=""		
3		for A2	if B3==1	>B2=B2+"@"
4			else	>B2=B2+" "
5		>output(B2)		

Chapter 9
Calculation of sequence

9.1 Set property of sequence

- Ordered property of sequence
- Sequence and set

9.2 Sequence functions

- Generation and maintenance of sequence
- String and sequence

9.3 Aggregate calculation

- Count, Sum & Average
- Ranking
- Max/Min value and median
- Other aggregate functions

9.4 Loop calculation

- Complete loop calculation with A.(x)
- Calculation by position
- Loop functions

9.5 Positioning

- Find the position of the specified member
- Find the position of a member that meets the condition
- Find the position of the Max/Min value
- Overall positioning

9.6 Select

- Find members that meet the conditions
- Select the Max/Min value
- Sorting and ranking

9.7 Calculation of multidimensional sequence

9.8 Examples of sequence calculation

9.9 Exercises

Chapter 9
Calculation of sequence

9.1 Set property of sequence

9.1.1 Ordered property of sequence

Difference between ordered and unordered sets

- The general set is unordered, that is, two sets with the same members but different orders are equal. But the **sequence is ordered**, and two sequences with the same members but different orders are not equal. In addition, members in the sequence are repeatable.
- Because the sequence is ordered, we can use **cmp(A, B)** function to compare members of A and B in the same position, and **A??B** to carry out the four basic calculations by position.
- If we only need to compare whether the members in two sequences are consistent without considering the order, we can use A.eq(B) to judge.

Ordered property

	A	B
1	=[1,2]==[2,1]	/false
2	=[1,2].eq([2,1])	/true
3	=[1,2]==[1,2]	/true

	A	B
1	=[1]==[1,1]	/false
2	=[1,1]==[1,1]	/true

9.1.1 Ordered property of sequence

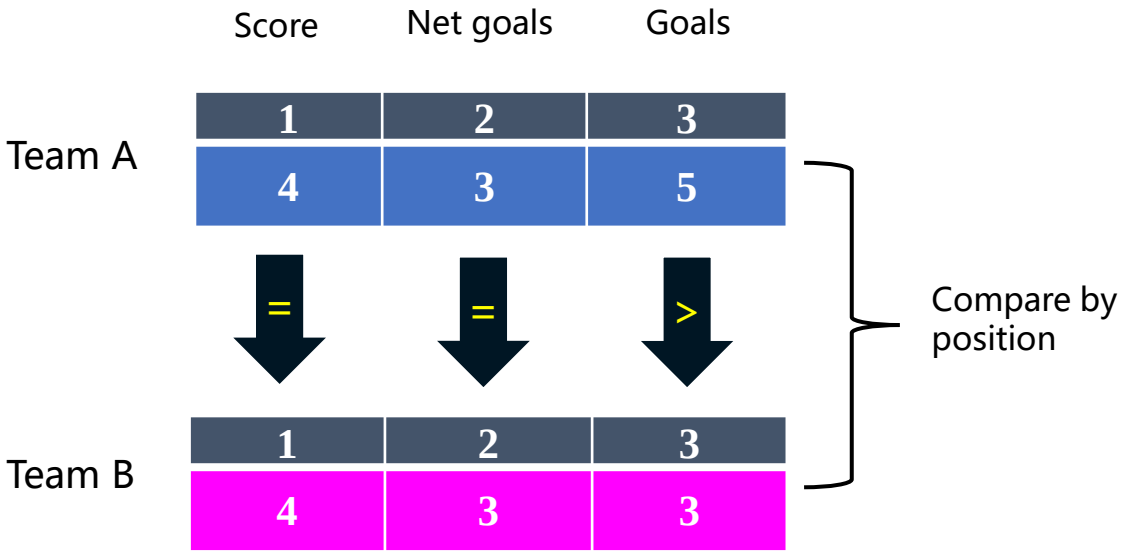
【Example 9.1】In the World Cup group match, the first condition to decide the ranking is the score of each team in all three group games. In the case of the same scores, the team with more net goals (total goals minus total goals lost) ranks higher. If the net goals are still the same, the team with more goals will be ranked higher. Now record the score, net goals and goals of each team in sequences (team A: 4 points, 3 net goals, 5 goals; Team B: 4 points, 3 net goals, 3 goals). How to quickly determine the ranking of the two teams in the group?

9.1.1 Ordered property of sequence

【Example 9.1】In the World Cup group match, the first condition to decide the ranking is the score of each team in all three group games. In the case of the same scores, the team with more net goals (total goals minus total goals lost) ranks higher. If the net goals are still the same, the team with more goals will be ranked higher. Now record the score, net goals and goals of each team in sequences (team A: 4 points, 3 net goals, 5 goals; Team B: 4 points, 3 net goals, 3 goals). How to quickly determine the ranking of the two teams in the group?

Solution

You can store the score, net goals and goals into a sequence in order, and then use the **cmp(A, B)** function to compare the two sequences by position.



	A
1	=cmp([4, 3, 5], [4, 3, 3])

Result:

A1	Value
	1

i.e.: Team A ranks higher than team B.

Note

- In sequence comparison, the values of each member in the same position will be compared in turn. When the first unequal member is encountered, 1 or - 1 will be returned according to the size, and 0 will be returned if all members are equal.
- The data type of the corresponding members must be comparable.

9.1.1 Ordered property of sequence

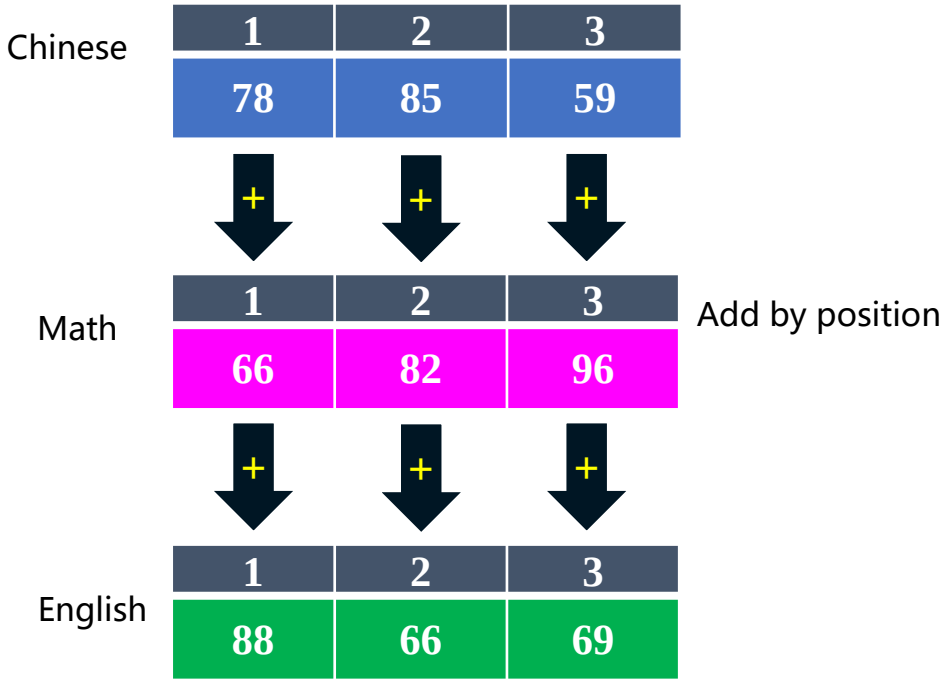
【Example 9.2】It is known that the scores of three students in a test are in turn as follows: Chinese: 78, 85, 59; Math: 66, 82, 96; English: 88, 66, 69. Find the total score of each student.

9.1.1 Ordered property of sequence

【Example 9.2】It is known that the scores of three students in a test are in turn as follows: Chinese: 78, 85, 59; Math: 66, 82, 96; English: 88, 66, 69. Find the total score of each student.

Solution

The results of each subject of 3 students can be saved in a sequence respectively, and then the total score of each student can be calculated by adding (++) by position.



	A	B	C
1	[78, 85, 59]	[66, 82, 96]	[88, 66, 69]
2	=A1++B1++C1		

A2	Index	Member
	1	232
	2	233
	3	224

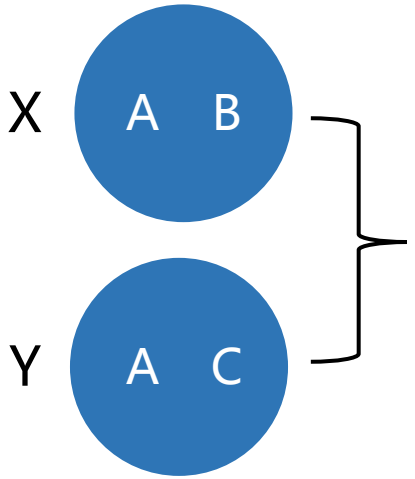
Note

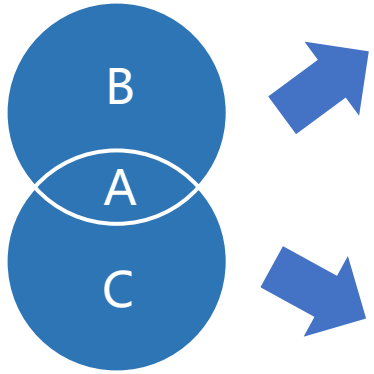
- The aligned four operations of sequences, that is, for two sequences of the same length, the members of the same position will be calculated in turn to return a new sequence.
- The aligned operations include:++, --, **, //, \\, and %%

9.1.2 Sequence and set

Set operations

- **Sequence** has the general characteristics of **set** and can be used for set operations.
- Set operations include: **intersection (&)**, **union(^)**, **difference(\)**, **concatenation(|)**, etc.

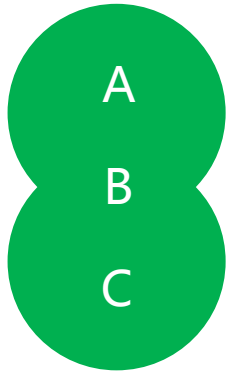


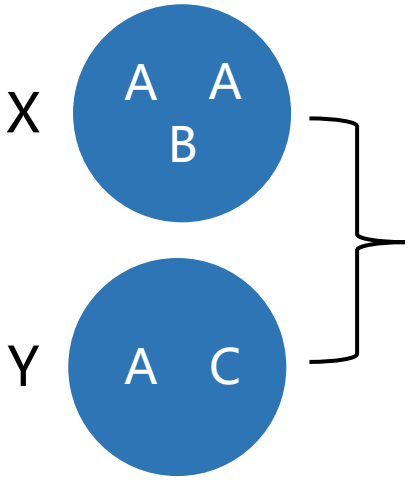


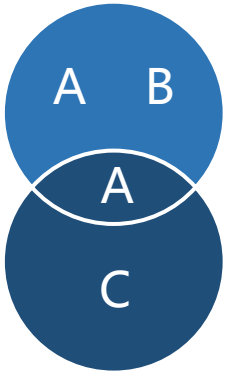
Intersection
X&Y



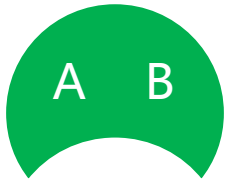
Union
X^Y

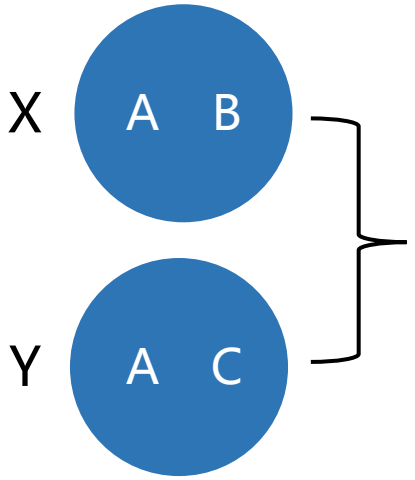


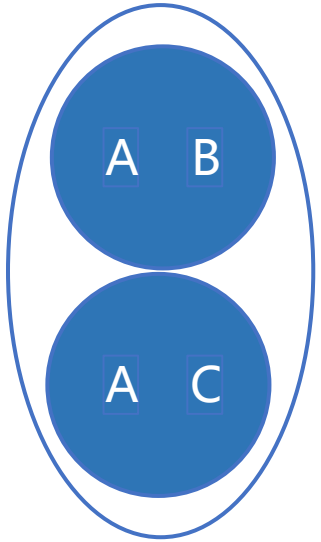




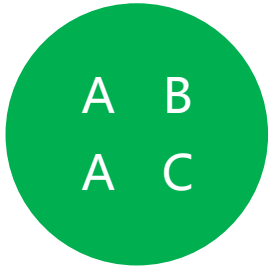
Difference
X\Y







Concatenation
X|Y



Intersection

- If the members in the left sequence appear in the right sequence, they will be added to the new sequence in order (get the same), otherwise it will be ignored.
- Each member in the right sequence can only be used for getting the same operation once, and cannot be used multiple times.

	A
1	=[4, 4, 3] ^ [1, 3, 4]
2	=[4, 3] ^ [1, 3, 4, 4]
3	=[4, 4, 3] ^ [1, 3, 4, 4]

A1	Index	Member
	1	4
	2	3

A2	Index	Member
	1	4
	2	3

A3	Index	Member
	1	4
	2	4
	3	3

【Example 9.3】There are two choices for a class to sign up for extracurricular activities interest class, writing and painting. It is known that Jimmy, Mary and Lucy have signed up for writing, and Kate, Tom and Mary have signed up for painting. Who have signed up for both of the two interest classes?

Intersection

【Example 9.3】There are two choices for a class to sign up for extracurricular activities interest class, writing and painting. It is known that Jimmy, Mary and Lucy have signed up for writing, and Kate, Tom and Mary have signed up for painting. Who have signed up for both of the two interest classes?

Solution

The registration list of each interest class is created as a sequence, and then the **intersection** of two sequences can be calculated.

	A	B	C
1	[Jimmy, Mary, Lucy]	[Kate, Tom, Mary]	/A1:Writing, B1:Painting
2	=A1^B1		/The intersection

A2	Index	Member
	1	Mary

Union

- First, the members in the left sequence are added to the new sequence in order. If the members in the right sequence appear in the left sequence, they will be ignored (deduplicated), otherwise they will be added to the new sequence in order.
- Each member in the left sequence can only be used for one deduplication operation and cannot be used multiple times.

	A
1	=[4, 3] & [3, 4]
2	=[4, 4, 3] & [3, 4]
3	=[4, 3] & [3, 4, 4]

A1	Index	Member
	1	4
	2	3

A2	Index	Member
	1	4
	2	4
	3	3

A3	Index	Member
	1	4
	2	3
	3	4

【Example 9.4】There are two choices for a class to sign up for extracurricular activities interest class, writing and painting. It is known that Jimmy, Mary and Lucy have signed up for writing, and Kate, Tom and Mary have signed up for painting. How many students have signed up for the interest classes?

【Example 9.4】There are two choices for a class to sign up for extracurricular activities interest class, writing and painting. It is known that Jimmy, Mary and Lucy have signed up for writing, and Kate, Tom and Mary have signed up for painting. How many students have signed up for the interest classes?

Solution

The registration list of each interest class is created as a sequence, the **union** of the two sequences is calculated, and then the sequence length of the union set can be calculated.

	A	B	C
1	[Jimmy, Mary, Lucy]	[Kate, Tom, Mary]	/A1:writing, B1:painting
2	=A1&B1		/Get the union
3	=A2.len()		/Get the sequence length of A2

A3	Value
	5

Difference

- If the members in the left sequence appear in the right sequence, they will be ignored, otherwise they will be added to the new sequence in order(getting the difference).
- Each member in the right sequence can only be used for one getting-the-difference operation, and cannot be used multiple times.

	A
1	=[4, 4, 3] \ [1, 4]
2	=[4, 4, 3] \ [1, 3, 4]
3	=[1, 3, 4] \ [4, 4, 3]

A1	Index	Member
	1	4
	2	3

A2	Index	Member
	1	4

A3	Index	Member
	1	1



【Example 9.5】There are two choices for a class to sign up for extracurricular activities interest class, writing and painting. It is known that Jimmy, Mary and Lucy have signed up for writing, and Kate, Tom and Mary have signed up for painting. How many people have signed up for only one interest class?

【Example 9.5】There are two choices for a class to sign up for extracurricular activities interest class, writing and painting. It is known that Jimmy, Mary and Lucy have signed up for writing, and Kate, Tom and Mary have signed up for painting. How many people have signed up for only one interest class?

Solution

The registration list of each interest class is created as a sequence, and the intersection of the two sequences is calculated, that is, the students who have signed up for both interest classes. Then we calculate the union of the two sequences, that is, all the students who have signed up for interest classes. Then calculate the **difference** set of union and intersection, that is, the students who have signed up for only one interest class. Finally, we can calculate the sequence length of the set.

	A	B	C
1	[Jimmy, Mary, Lucy]	[Kate, Tom, Mary]	/A1:writing, B1:painting
2	=A1^B1	=A1&B1	/Get the union and intersection
3	=B2\A2		/Get the difference of B2 and A2
4	=A3.len()		/Get the sequence length of A3

A4	Value
	4

Concatenation

- First, the members in the left sequence are added to the new sequence in order, and then the members in the right sequence are added to the new sequence in order. The same member can be repeated.
- You can also use non sequential "single value", which will be automatically converted to a sequence with a single member during operation.

	A
1	=[4, 3] | [1, 4]
2	=[4, 3] | 4
3	=4 | [4, 3]

A1	Index	Member
	1	4
	2	3
	3	1
	4	4

A2	Index	Member
	1	4
	2	3
	3	4

A3	Index	Member
	1	4
	2	4
	3	3

【Example 9.6】Michael and Hilda go to eat Western food. Michael orders a steak, a salad, a pizza, and Hilda orders a drumsticks, a borsch, and a steak. What dishes have they ordered?

【Example 9.6】Michael and Hilda go to eat Western food. Michael orders a steak, a salad, a pizza, and Hilda orders a drumsticks, a borsch, and a steak. What dishes have they ordered?

Solution

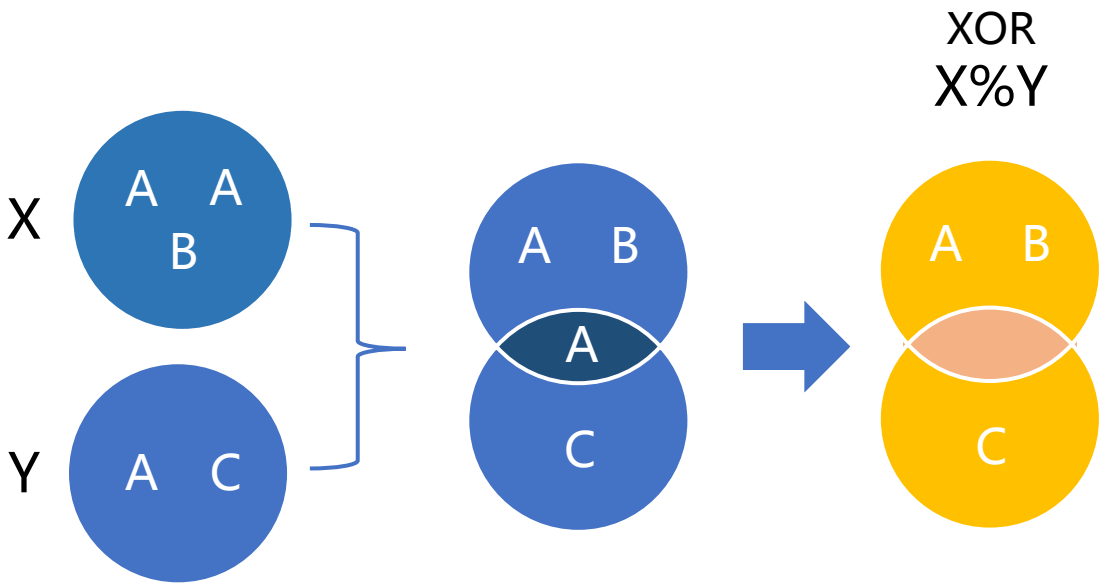
Create the dishes of Michael and Hilda as a sequence respectively. Obviously, the dishes ordered repeatedly should be calculated separately, so the **concatenation** of the two sequences should be calculated.

	A	B	C
1	[Steak, Salad, Pizza]	[Drumsticks, Borsch, Steak]	/A1:the dishes that Michael ordered, B1:the dishes that Hilda ordered
2	=A1|B1		/Get the concatenation of A1 and B1

A2	Index	Member
	1	Steak
	2	Salad
	3	Pizza
	4	Drumsticks
	5	Borsch
	6	Steak

XOR

- Members that belong to two sets but do not belong to both sets constitute an **XOR** set.
- When calculating XOR of two sequences, the members in the left sequence that do not appear in the right sequence are added to the new sequence in order; then the members in the right sequence that do not appear in the left sequence are added in turn.



【Example 9.5】There are two choices for a class to sign up for extracurricular activities interest class, writing and painting. It is known that Jimmy, Mary and Lucy have signed up for writing, and Kate, Tom and Mary have signed up for painting. How many people have signed up for only one interest class?

【Example 9.5】There are two choices for a class to sign up for extracurricular activities interest class, writing and painting. It is known that Jimmy, Mary and Lucy have signed up for writing, and Kate, Tom and Mary have signed up for painting. How many people have signed up for only one interest class?

Solution

With **XOR** operation of set, example 9.5 can be directly calculated.

	A	B	C
1	[Jimmy, Mary, Lucy]	[Kate, Tom, Mary]	/A1:writing, B1:painting
2	=A1%B1		/Get XOR of A1 and B1
3	=A2.len()		/Get the sequence length of A2

A2	Index	Member
	1	Jimmy
	2	Lucy
	3	Kate
	4	Tom

A3	Value
	4

Chapter 9
Calculation of sequence

9.2 Generation and maintenance of sequence

9.2.1 Generation and maintenance of sequence

Generation of sequence

- In SPL, the common methods of generating sequence are constant construction method and function construction method.

	A	B
1	[2, 3, 4]	/Generate a sequence of 2~4 with constant construction method, which can also be written as =[2, 3, 4]
2	=to(2, 4)	/Generate a sequence of a to b with function to(a, b) . to(n) can also be used to generate a sequence from 1 to n.
3	=["a", "b", "c", "d"].to(2, 4)	/Use the A.to() function to get the second to the fourth members from the existing sequence to form a new sequence; when the string sequence is constructed in the expression, the members need to be enclosed in quotation marks.
4	=[1, 2, 3, 4, 5].step(3, 1, 2)	/Using the A. step() function, get the first and the second of every three members to form a new sequence

A1	Index	Member
	1	2
	2	3
	3	4

A2	Index	Member
	1	2
	2	3
	3	4

A3	Index	Member
	1	b
	2	c
	3	d

A4	Index	Member
	1	1
	2	2
	3	4
	4	5

【Example 9.7】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. Prepare to send the first 3 employees to study abroad. Please show the list of dispatched employees in a sequence.

9.2.1 Generation and maintenance of sequence

【Example 9.7】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. Prepare to send the first 3 employees to study abroad. Please show the list of dispatched employees in a sequence.

Solution

The list of employees can be recorded by a sequence. For the list of dispatched employees, we only need to obtain the first three items to form a new sequence.

A.to(n) can intercept the first **n** members from the existing sequence **A** to generate a new sequence.

	A	B
1	[David, Daniel, Andrew, Ruby, Rachel, Rebecca]	/Generating employee list with constant construction method
2	=A1.to(3)	/Get the first three employees to get the dispatch list

A2	Index	Member
	1	David
	2	Daniel
	3	Andrew

9.2.1 Generation and maintenance of sequence

【Example 9.8】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. First select the first employee, and then select every other employee to study abroad. Please list the dispatched employees.

9.2.1 Generation and maintenance of sequence

【Example 9.8】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. First select the first employee, and then select every other employee to study abroad. Please list the dispatched employees.

Solution

Get one member from every other member of the initial employee sequence to generate a new sequence.

A.step(n, k_1, k_2, ...) can generate a new sequence according to the specified starting sequence numbers k_1, k_2, ... and span **n**. **When** k_1, k_2, ... are omitted, the starting sequence number is 1.

	A	B
1	[David, Daniel, Andrew, Ruby, Rachel, Rebecca]	/Generating employee list with constant construction method
2	=A1.step(2)	/Select the first and every other members

A2	Index	Member
	1	David
	2	Andrew
	3	Rachel

9.2.1 Generation and maintenance of sequence

Maintenance of sequence

- In SPL, sequence maintenance includes the insertion, modification and deletion of members in the sequence. **A.insert(), A.modify()** and **A.delete()** functions are used respectively.

	A	B
1	=[1, 2, 3].insert(0, 4)	/Insert 4 at the end
2	=[1, 4].insert(2, [2, 3])	/Insert 2 and 3 between the first and second member
3	=[1, 2, 3].modify(2, 0)	/Modify the second member
4	=[1, 2, 3].modify(2, [-2, -3])	/Modify from the second member onwards
5	=[1, 2, 3].delete(1)	/Delete the first member
6	=[1, 2, 3].delete([1, 3])	/Delete the first and third member

A1	Index	Member
	1	1
	2	2
	3	3
	4	4

A2	Index	Member
	1	1
	2	2
	3	3
	4	4

A3	Index	Member
	1	1
	2	0
	3	3

A4	Index	Member
	1	1
	2	-2
	3	-3

A5	Index	Member
	1	2
	2	3

A6	Index	Member
	1	2

9.2.1 Generation and maintenance of sequence

【Example 9.9】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. Now we have a new employee Ashley, please make a list of them.

9.2.1 Generation and maintenance of sequence

【Example 9.9】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. Now we have a new employee Ashley, please make a list of them.

Solution

Add a new member to the initial employee sequence.

A.insert(k, x) inserts member x to position k in sequence A, **x** can be a single value or a sequence; When **k** is 0, add at the end of the sequence.

	A	B
1	[David, Daniel, Andrew, Ruby, Rachel, Rebecca]	/Generating initial employee list by constant construction method
2	=A1.insert(0, "Ashley")	/Add a new member to the list

A2	Index	Member
	1	David
	2	Daniel
	3	Andrew
	4	Ruby
	5	Rachel
	6	Rebecca
	7	Ashley

Note

- **A.insert(k, x)** inserts the new member x into the k^{th} position in sequence A, and the members after the k^{th} position in the sequence will move backward. A will be modified after insertion, so the results in A1 and A2 are the same after the above code is executed.
- When adding a member at the end of the sequence, you can also use the | operator, such as changing the code in A2 to: **=A1=A1|"Ashley"**

9.2.1 Generation and maintenance of sequence

【Example 9.10】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. Now it is found that the name of the fourth employee is registered incorrectly. Actually, it should be Rudy. Please list the revised employee list.

9.2.1 Generation and maintenance of sequence

【Example 9.10】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. Now it is found that the name of the fourth employee is registered incorrectly. Actually, it should be Rudy. Please list the revised employee list.

Solution

You can modify a member at a specified position in the employee list.
A.modify(k, x) can assign values to one or more members of sequence A from the specified position k onwards.
x can be a single value or a sequence.

	A	B
1	[David, Daniel, Andrew, Ruby, Rachel, Rebecca]	/Generating initial employee list by constant construction method
2	=A1.modify(4, "Rudy")	/Modify the value of a specified member

A2	Index	Member
	1	David
	2	Daniel
	3	Andrew
	4	Rudy
	5	Rachel
	6	Rebecca

Note

- **A** will be modified after the execution of **A.modify(k, x)** , so the results in A1 and A2 are the same after the above code is executed.

【Example 9.11】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. Now, the 3rd and 5th employees need to be transferred to other departments. Please list the employees in the department after the change.

9.2.1 Generation and maintenance of sequence

【Example 9.11】The list of employees in a department is as follows: David, Daniel, Andrew, Ruby, Rachel, Rebecca. Now, the 3rd and 5th employees need to be transferred to other departments. Please list the employees in the department after the change.

Solution

Delete the members in the specified positions in the employee list.

A.delete(p) deletes members with sequence numbers in sequence **p** from sequence **A**, if only deleting one member, **p** can be an integer.

	A	B
1	[David, Daniel, Andrew, Ruby, Rachel, Rebecca]	/Generating initial employee list by constant construction method
2	>A1.delete([3, 5])	/Delete two members from the list

A1	Index	Member
	1	David
	2	Daniel
	3	Ruby
	4	Rebecca

Note

- After **A.delete(p)** deletes the members in the sequence, the position of other members will move forward. After deletion, A will be modified, so the results in A1 and A2 are the same after the above code is executed.

9.2.2 String and sequence

Mutual transformation of string and sequence

- **s.split(d)** can split the string s into a sequence through the separator d, and each character will be split into a member when there is no separator.
- **A.concat(d)** can use the separator d to concatenate the members of sequence A into a string to return. If there is no separator, it will directly concatenate the members into a string.

	A	B
1	="abc".split()	/Split into a sequence by character
2	="a, b, c".split(", ")	/Split into a sequence with comma as separator
3	="1, 1.5, dc, null".split@p(", ")	/Split with comma as seperator and automatically identify data types

A1	Index	Member
	1	a
	2	b
	3	c

A2	Index	Member
	1	a
	2	b
	3	c

A3	Index	Member
	1	1
	2	1.5
	3	dc
	4	(null)

	A	B
1	=["a", "b", "c", "d", "e"].concat()	/Each member of the sequence is concatenated into a string
2	=["a", "b", "c", "d", "e"].concat(", ")	/Sequence members are concatenated by commas

A1	Value
	abcde

A2	Value
	a,b,c,d,e

【Example 9.12】Suppose you want to fill in the blank with the word "antidisestablishmentarianism" by replacing all its vowels with "_" sign, and return it as a string.

9.2.2 String and sequence

【Example 9.12】Suppose you want to fill in the blank with the word "antidisestablishmentarianism" by replacing all its vowels with "_" sign, and return it as a string.

Solution

Use the **s.split()** function to divide the word into a string sequence of characters, replace all vowels, and then concatenate it back.

	A	B	C	D
1	="antidisestablishmentarianism".split()			/Split the word to a sequence of characters
2	for A1	if A2=="a" || A2=="e" || A2=="i" || A2=="o" || A2=="u"	>A1(#A2) = "_"	/Replace vowel character with "_" , consonant characters unchanged
3	=A1.concat()			/Concatenate the sequence of characters into a string and return

A3	Value
	_nt_d_s_st_bl_shm_nt_r__n_sm

9.2.2 String and sequence

【Example 9.13】Input an English sentence and output each English word in the sentence in turn. For example, input: The quick brown fox jumps over a lazy dog.

9.2.2 String and sequence

【Example 9.13】Input an English sentence and output each English word in the sentence in turn. For example, input: The quick brown fox jumps over a lazy dog.

Solution

Use the **s.split(d)** function to divide the sentence into a sequence of words with spaces, note that the period should be removed first.

	A	B
1	The quick brown fox jumps over a lazy dog.	/Input the sentence
2	=left(A1, len(A1)-1)	/Remove the last character dot
3	=A2.split(" ")	/Use spaces as separators to split into a sequence of words

A3	Index	Member
	1	The
	2	quick
	3	brown
	4	fox
	5	jumps
	6	over
	7	a
	8	lazy
	9	dog

9.2.2 String and sequence

【Example 9.13】Input an English sentence and output each English word in the sentence in turn. For example, input: The quick brown fox jumps over a lazy dog.

Note

- The simple function **s.words()** can be used to split the words in English sentences. In this case, the word sequence will be automatically split. There is no need to deal with period and other symbols. The separator in the middle of the words can be space or other punctuation marks.

	A	B
1	The quick brown fox jumps over a lazy dog.	/Input the sentence
2	=A1.words()	/Split the words in the sentence

A2	Index	Member
	1	The
	2	quick
	3	brown
	4	fox
	5	jumps
	6	over
	7	a
	8	lazy
	9	dog

Chapter 9
Calculation of sequence

9.3 Aggregate calculation

Aggregate calculation

- Aggregate results are counted for all members in the set. Such operations are called **aggregate operations**.
- The common aggregate calculation includes counting, sum, average, Max/Min and ranking, etc.
- The aggregate operation can be realized by the code of loop structure, but it is more convenient to use aggregate function.

Index	Member
1	1
2	4
3	9
4	16
5	25
6	36

Get SUM

	A	B	C
1	[1, 4, 9, 16, 25, 36]	0	/Input a sequence in A1, B1records the sum
2	for A1	>B1=B1+A2	/Get the SUM in loop

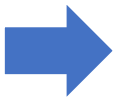
B1	Value
	91

9.3.1 Count, Sum & Average

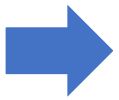
Count

- **Count** refers to the total number of members in the sequence, which is calculated by using the aggregate function **A.count()**.
- Unlike the sequence length function A.len(), the null members in the sequence are not counted.
- If the sequence contains a Boolean member false, it is not counted.

Index	Member
1	a
2	0
3	3
4	(null)
5	true
6	false



Index	Counted or not?
1	√
2	√
3	√
4	×
5	√
6	×



Count
4

Count distinct

- **Count distinct** refers to the total number of non repeating members in the sequence, which is calculated by using the aggregate function **A.icount()**.
- The same member in the sequence is calculated only once in count distinct.
- If null or false is included in the sequence, it is still ignored in count distinct.

Index	Member
1	Team Exercises
2	(null)
3	Party
4	Dinner
5	(null)
6	Party
7	Dinner
8	Party

Index	Counted or not?
1	√
2	×
3	√
4	√
5	×
6	○
7	○
8	○

Count distinct
3

9.3.1 Count, Sum & Average

【Example 9.14】A department is preparing to organize the team building activities, and receives the following suggestions from employees: Team exercises, null, Party, Dinner, null, Party, Dinner, Party. If there is no suggestion, fill in with null value. How many activities are available?

9.3.1 Count, Sum & Average

【Example 9.14】A department is preparing to organize the team building activities, and receives the following suggestions from employees: Team exercises, null, Party, Dinner, null, Party, Dinner, Party. If there is no suggestion, fill in with null value. How many activities are available?

Solution

Since the same items exist in the data, the count distinct function **A.icount()** is required to count.

	A	B
1	[Team Exercises, null, Party, Dinner, null, Party, Dinner, Party]	/Generate the sequence
2	=A1.icount()	/Count distinct

A2	Value
	3

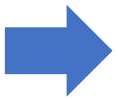
Note

- All null and false in sequence members are not counted in count distinct.

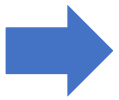
Sum

- **Sum** is to get the sum of all members in a sequence, which is calculated using the aggregate function **A.sum()**.
- In getting the sum of a sequence, non numeric members of the sequence are ignored, such as null, string, time, date, Boolean number, etc.
- If there is no member of any numeric type in the sequence, the result of A.sum() is null.

Index	Member
1	a
2	0
3	3
4	(null)
5	true
6	false



Index	Whether used to get sum?
1	×
2	√
3	√
4	×
5	×
6	×



Sum
3

9.3.1 Count, Sum & Average

【Example 9.15】It is known that the scores of several students in a certain test are as follows: 66, null, 82, 96. Someone did not take the test for some reason, so he did not get any score. Please calculate their total score and average score. Student who is absent from the test will not participate in the statistics.

9.3.1 Count, Sum & Average

【Example 9.15】It is known that the scores of several students in a certain test are as follows: 66, null, 82, 96. Someone did not take the test for some reason, so he did not get any score. Please calculate their total score and average score. Student who is absent from the test will not participate in the statistics.

Solution

Use **A.sum()** function to calculate the total score, and use total score/number of participants to get the average. Use **A.count()** to calculate the number of participants.

	A	B
1	[66, null, 82, 96]	/Generate a sequence of scores
2	=A1.sum()	/Get the total score
3	=round(A2/A1.count(), 2)	/Calculate the average score and keep 2 decimal places

A2	Value
	244

A3	Value
	81.33

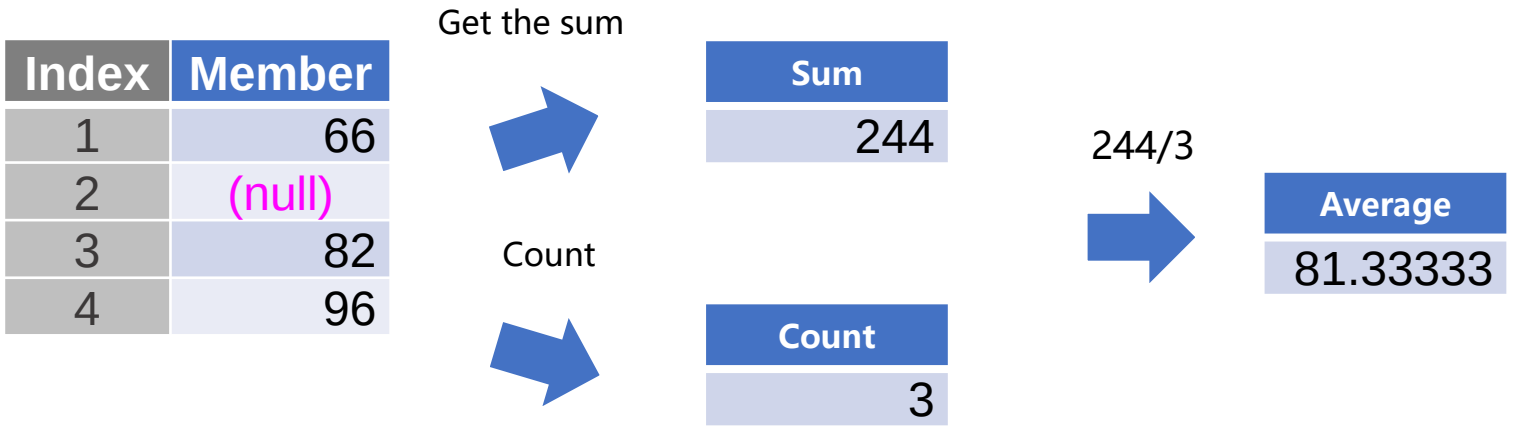
Note

- To calculate the average value, you can also use the function **A.avg()** which will be discussed below. The code in A3 can also be modified to **=round(A1.avg(), 2)**, and the result is the same. Moreover, this method of calculating the average does not need to calculate the total score and count.

9.3.1 Count, Sum & Average

Average

- The **average value** refers to the average value of the members in the sequence, which is calculated by using the aggregate function **A.avg()**.
- When calculating the average value, the null value in the sequence is not involved in the statistics.
- If the sequence contains other members of non numerical type besides null value, they are also ignored when using A.avg().



9.3.2 Ranking

Ranking

- According to the values of the members in the sequence, the ranking of each member can be calculated. By default, it is arranged from small to large, with the smallest value being the first and the largest value being the last. The ranking function **A.ranks()** can be used to calculate the ranking sequence of each member in A.
- When performing ranking calculation, the types of members in the sequence must be able to compare with each other. If there is a null value, it will be considered as the minimum.
- If you need to calculate the ranking of a value in the sequence, you can use **A.rank(y)**.

Index	Member
1	9.98
2	11.99
3	9.80
4	9.75
5	9.79
6	9.63
7	9.88
8	9.94

Ranking

Index	Ranking
1	7
2	8
3	4
4	2
5	3
6	1
7	5
8	6

9.3.2 Ranking

【Example 9.16】In a sprint race, the results of 8 competitors are as follows (seconds): 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Find their rankings in the game. If a player's score is 9.78 seconds, what is his ranking if he takes part in the competition?

9.3.2 Ranking

【Example 9.16】In a sprint race, the results of 8 competitors are as follows (seconds): 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Find their rankings in the game. If a player's score is 9.78 seconds, what is his ranking if he takes part in the competition?

Solution

Use **A.ranks()** function to calculate the ranking sequence, and use **A.rank(y)** function to calculate the ranking of a value in the sequence.

	A	B
1	[9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94]	/Generate a sequence of scores
2	=A1.ranks()	/Calculate the ranking sequence
3	=A1.rank(9.78)	/Calculate the ranking of a score

A2	Index	Member
	1	7
	2	8
	3	4
	4	2
	5	3
	6	1
	7	5
	8	6

A3	Value
	3

Note

- If there is null in the sequence, the influence of null value should be excluded when calculating the ranking.

9.3.2 Ranking

【Example 9.17】The ages of employees in a department are as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. What is the ranking of employees according to their ages in descending order? If the parallel ranking does not occupy the redundant place, what is the ranking? In data statistics, in order to eliminate the influence of parallel ranking, we often need to calculate the statistical ranking, that is, if there are k paralleled n places, then their ranking (n+(n+1)+...(n+k-1))/k, for example, there are two are tied for the second place, their ranking is (2 + 3) / 2 = 2.5. What is the statistical ranking of employees in this department?

9.3.2 Ranking

【Example 9.17】The ages of employees in a department are as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. What is the ranking of employees according to their ages in descending order? If the parallel ranking does not occupy the redundant place, what is the ranking? In data statistics, in order to eliminate the influence of parallel ranking, we often need to calculate the statistical ranking, that is, if there are k paralleled n places, then their ranking (n+(n+1)+...(n+k-1))/k, for example, there are two are tied for the second place, their ranking is (2 + 3) / 2 = 2.5. What is the statistical ranking of employees in this department?

Solution

For the race results, we are used to ascending ranking from small to large, but in many cases, we also need descending ranking from large to small. In the A.ranks() and A.rank(y) functions, you can use various options, **@z** indicates descending order, **@i** indicates that the parallel ranking does not occupy the redundant place, and **@s** represents the calculation of statistical ranking. Because of different calculation schemes, @i and @s cannot be used at the same time.

	A	B
1	[45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51]	/Generate a sequence of ages
2	=A1.ranks@z()	/Calculate the ranking sequence in descending order
3	=A1.ranks@zi()	/Calculate the ranking sequence in descending order, and the parallel ranking does not occupy the redundant place
4	=A1.ranks@zs()	/Calculate the statistical ranking sequence

A2	Index	Member
	1	4
	2	5
	3	8
	4	2
	5	5
	6	7
	7	8
	8	8
	9	2
	10	8
	11	1

A3	Index	Member
	1	3
	2	4
	3	6
	4	2
	5	4
	6	5
	7	6
	8	6
	9	2
	10	6
	11	1

A4	Index	Member
	1	4.0
	2	5.5
	3	9.5
	4	2.5
	5	5.5
	6	7.0
	7	9.5
	8	9.5
	9	2.5
	10	9.5
	11	1.0

9.3.3 Max/Min value & Median

Max/Min value

- Among all members of a sequence, the one with the largest value is called the **maximum value**, and the one with the smallest value is called the **minimum value**. The aggregate functions **A.max()** and **A.min()** can be used to calculate the maximum and minimum values respectively.
- When calculating the maximum value, the null value in the sequence is ignored.

Index	Member
1	9.98
2	11.99
3	9.80
4	9.75
5	9.79
6	9.63
7	9.88
8	9.94

Get max value

Max value
11.99

Get min value

Min value
9.63

9.3.3 Max/Min value & Median

【Example 9.18】The ages of employees in a department are as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. What is the maximum and minimum age of employees in the department?

Solution

The aggregate functions **A.max()** and **A.min()** can be used to find the maximum or minimum value.

	A	B
1	[45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51]	/Generate a sequence of ages
2	=A1.max()	/Calculate the max age
3	=A1.min()	/Calculate the min age

A2	Value
	51

A3	Value
	21

9.3.3 Max/Min value & Median

Median

- The **median** is the number in the middle of the sequence after the members are arranged in order, which is obtained by using **A.median()**.
- When calculating the median, the data types of each member in the sequence must be comparable.
- If the number of members in a sequence is even, there are two members in the middle, and their average is returned.

Index	Member
1	7
2	3
3	5
4	2
5	6
6	1
7	4

Sorting

Member
1
2
3
4
5
6
7

Median

Median
4

9.3.3 Max/Min value & Median

【Example 9.19】In a sprint race, the results of 8 competitors are as follows (seconds): 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Find the median of their results.

9.3.3 Max/Min value & Median

【Example 9.19】In a sprint race, the results of 8 competitors are as follows (seconds): 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Find the median of their results.

Solution

The aggregate function **A.median()** can be used to calculate the median.

	A	B
1	[9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94]	/Generate a sequence of scores
2	=A1.median()	/Calculate the median

A2	Value
	9.84

Note

- There are eight scores, and there are two in the middle, 9.80 in the fourth and 9.88 in the fifth. Therefore, the average value of 9.84 can be obtained by calculating the median.

9.3.4 Other aggregate functions

Null value judgment

- In aggregate calculation, you can find the first member with a non null value, using the function **A.ifn()**.
- In aggregate calculation, you can also find the first member that is neither a null value nor an empty string, using the function **A.nvl()**.

Index	Member
1	(null)
2	(null)
3	
4	(null)
5	0
6	false

Count

Index	Is the member a null value?	Is the member an empty string?
1	√	
2	√	
3	×	√
4	√	
5	×	×
6	×	×

A.ifn()

The first member with a non null value

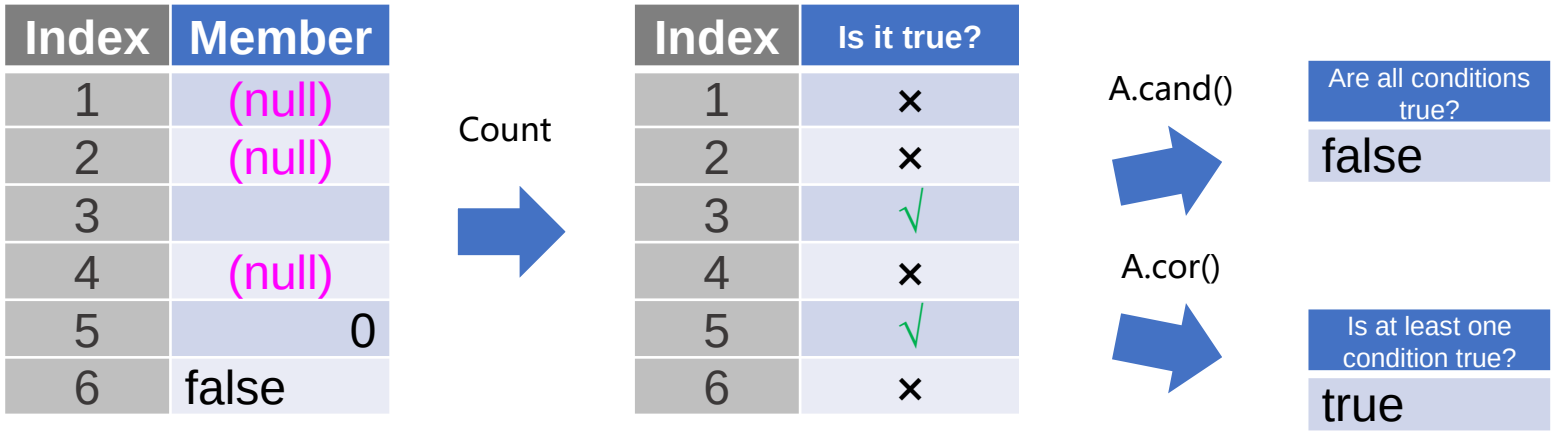
A.nvl()

The first member that is neither a null value nor an empty string
0

9.3.4 Other aggregate functions

Logic AND, OR

- In aggregate calculation, the calculation of logical AND, logical OR is usually used for Boolean value sequence to judge whether all conditions in the sequence are true, and whether at least one condition is true. **A.cand()** and **A.cor()** can be used respectively.
- In a sequence that uses the aggregate calculation of logical AND, logical OR, the members should all be Boolean values. But if there are other types of members in the sequence, null values are treated as false and other non Boolean members are treated as true.



Chapter 9
Calculation of sequence

9.4 Loop calculation

9.4.1 Complete the loop calculation with A.(x)

A.(x) and loop calculation

- In addition to for A, **A.(x)** can also be used for simple loop calculation of a sequence.
- When A.(x) is executed, each member of sequence A will be looped to calculate the expression x. In x, the current member can be referenced by the **~ symbol** to participate in the calculation, and A.(x) will return a new sequence composed of the results.

	A	B
1	[1, 2, 3]	/Original sequence
2	=A1.(~*2)	/Each member is multiplied by 2 to form a new sequence

A1	Index	Member
	1	1
	2	2
	3	3

A2	Index	Member
	1	2
	2	4
	3	6

The result of A.(x) execution used above is equivalent to the code on the right, but it is simpler.

	A	B	C
1	[1, 2, 3]	[]	/Original sequence and result sequence
2	for A1	>B1=B1|(A2*2)	/Loop through each member of the original sequence, add a calculated member to the result sequence

9.4.1 Complete the loop calculation with A.(x)

【Example 9.20】The total scores of four students in an exam are as follows: 232, 233, 224, 267. If there are three subjects in the exam, find the average score of each student.

9.4.1 Complete the loop calculation with A.(x)

【Example 9.20】The total scores of four students in an exam are as follows: 232, 233, 224, 267. If there are three subjects in the exam, find the average score of each student.

Solution

Record the test results in a sequence, and use the **A.(x)** to calculate the average score of each student according to the total score.

	A	B
1	[232, 233, 224, 267]	/Generate a sequence of total scores
2	=A1.(round(~/3, 2))	/The average score is calculated in loop with two decimal places

A2	Index	Member
	1	77.33
	2	77.67
	3	74.67
	4	89.0

Note

- **A.(x)** is equivalent to a loop calculation. x is calculated for each member of A, and the result is returned as a new sequence. The current member can be referenced by the **~ symbol** to participate in the calculation.

9.4.1 Complete the loop calculation with A.(x)

【Example 9.21】The unit price of a certain commodity is 90000 dollars, and the number of advance orders of five companies is: 25, 18, 33, 40, 9. Now the sales company carries out preferential activities. If the total amount of orders exceeds 1 million dollars, 10% discount will be given to the amount over 1 million yuan . Please calculate the total amount that each company needs to pay.

9.4.1 Complete the loop calculation with A.(x)

【Example 9.21】The unit price of a certain commodity is 90000 dollars, and the number of advance orders of five companies is: 25, 18, 33, 40, 9. Now the sales company carries out preferential activities. If the total amount of orders exceeds 1 million dollars, 10% discount will be given to the amount over 1 million yuan . Please calculate the total amount that each company needs to pay.

Solution

First use **A.(x)** to calculate the total price before discount in loop, and then use **A.(~=x)** to modify the total price to calculate the amount after discount.

	A	B	C
1	[25, 18, 33, 40, 9]	=A1.(~*9.0)	/Generate a sequence of sales quantity and calculate the price before discount.
2	=B1.()	=B1.(~)	/Calculate the price before discount in loop
3	=B1.(if(~>100, ~=(~-100)*0.9+100))		/Calculate the price after discount

B1	Index	Member
	1	212.5
	2	155.8
	3	277.3
	4	334.0
	5	81.0

A2	Index	Member
	1	212.5
	2	155.8
	3	277.3
	4	334.0
	5	81.0

B2	Index	Member
	1	225.0
	2	162.0
	3	297.0
	4	360.0
	5	81.0

A3	Index	Member
	1	212.5
	2	155.8
	3	277.3
	4	334.0
	5	(null)

Note

- When the expression x is not written, A.() returns the sequence A itself.
- As you can see from the results, the sequences in B1 and A2 are exactly the same, because the result returned in A2 is B1 itself.
- The result in B2 still keeps the data before discount. This is because when x is not an assignment statement, the result returned by A.(x) is a new sequence and has nothing to do with A.
- A3 only modifies the amount of more than 1 million dollars in the loop calculation, so the last member of the result is null.

9.4.1 Complete the loop calculation with A.(x)

【Example 9.22】Calculate the factorials of 1 ~ 6.

9.4.1 Complete the loop calculation with A.(x)

【Example 9.22】Calculate the factorials of 1 ~ 6.

Solution

In the expression x used by A.(x), the symbol # can be used to refer to the sequence number of the current member in the loop, and the symbol ~[-i] can be used to refer to the i^{th} member before the current member in the loop, that is, ~[-1] refers to the previous member in A. In SPL, in addition to relative reference to a member, the symbol ~[a, b] can also be used to refer to members within an interval. For example, ~[-3, -1] refers to the previous three members.

	A	B
1	=[0]*6	/A*n is equivalent to the concatenation set of n numbers of A, [0]*6 will get [0, 0, 0, 0, 0, 0], which is used to record the results
2	=A1.(~=if(#>1, #*~[-1], 1))	/Calculate in loop, the factorial of 1 is 1, and n! is equal to n*(n-1)!

The above code can be further simplified:

	A	B
1	=6.(~=if(#>1, #*~[-1], 1))	/Calculate in loop, the factorial of 1 is 1, and n! is equal to n*(n-1)!

A1	Index	Member
	1	1
	2	2
	3	6
	4	24
	5	120
	6	720

Note

- n.(x) is equivalent to to(n).(x), while to(n) generates a sequence of 1~n.

9.4.1 Complete the loop calculation with A.(x)

【Example 9.22】Calculate the factorials of 1 ~ 6.

Note

- In the previous solution, we need to use the previous results in the later calculation, such as the results in the sequence. When you encounter such a problem, you can use the iterate function **A.(iterate(x, a))** in the loop. In the loop expression x, the iteration value can be called with the ~~ symbol. The iteration value is the result of the expression calculated in each loop. Before the execution of the loop, the initial value of the iteration value is a.

	A	B
1	=6.(iterate(~~*~, 1))	/In iterative calculation, the initial value of factorial is 1, and each iteration result is multiplied by the value of current member

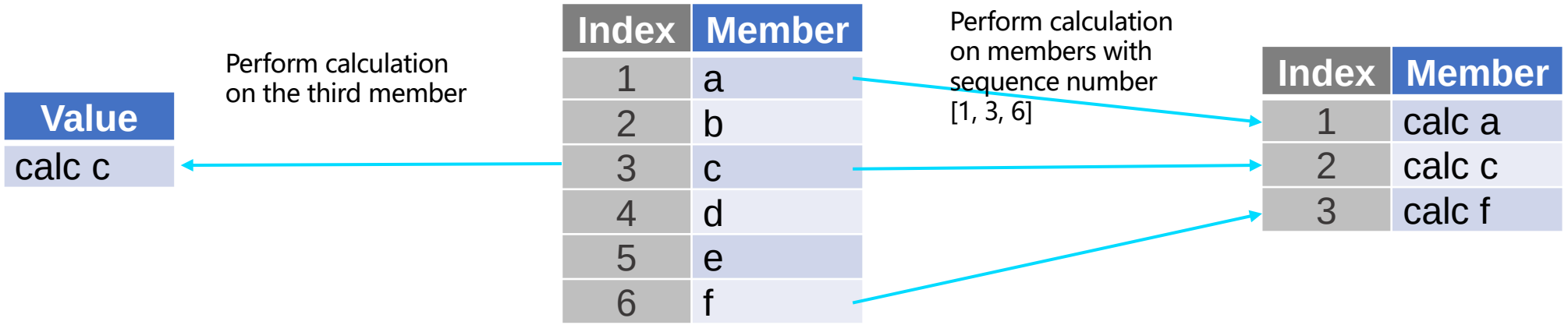
A1	Index	Member
	1	1
	2	2
	3	6
	4	24
	5	120
	6	720

Iterative functions are explained in detail in Chapter 12, section 3.

9.4.2 Calculation by position

Calculation by position

- **A.(x)** is calculated in loop for each member of the sequence. Sometimes, you only need to calculate some members of A with the specified sequence number. This kind of calculation is called **calculation by position**.
- Use **A.calc(k, x)** or **A.calc(p, x)** to calculate the k^{th} member or members with sequence numbers in sequence p respectively.



9.4.2 Calculation of specified location

【Example 9.23】According to the 1 ~ 6 factorials obtained in example 9.16, get the factorial equation of the specified number, such as: 5! = 120.

9.4.2 Calculation by position

【Example 9.23】According to the 1 ~ 6 factorials obtained in example 9.16, get the factorial equation of the specified number, such as: 5! = 120.

Solution

Based on the result of the last example, get the factorial equation by **A.calc(k, x)**.

	A	B
1	=6.(~=if(#>1, #*~[-1], 1))	/Calculate in loop, the factorial of 1 is 1, and n! is equal to n*(n-1)!
2	=A1.calc(5, #/"! = "/~)	/Calculate by position, and output the factorial equation of 5

A2	Value
	5! = 120

Note

- In calculation by position, the calculation is performed only for the member at the specified position, and the whole sequence does not have to be looped.

9.4.2 Calculation by position

【Example 9.24】The closing prices of a stock on the exchange in a week are as follows: 8.12, 8.22, 9.03, 9.53, and 9.32. Find daily closing price increase rate, excluding the first day.

9.4.2 Calculation by position

【Example 9.24】The closing prices of a stock on the exchange in a week are as follows: 8.12, 8.22, 9.03, 9.53, and 9.32. Find daily closing price increase rate, excluding the first day.

Solution

The calculation of the increase rate needs to compare the closing price with that of the previous day. After excluding the first day, **A.calc(p, x)** can be used for calculation by position.

	A	B
1	[8.12, 8.22, 9.03, 9.53, 9.32]	/Generate a sequence of closing prices
2	=A1.calc(to(2, A1.len()), round((~-~[-1])/~[-1], 3))	/Calculate the increase rate with 3 decimal places

A2	Index	Member
	1	0.012
	2	0.099
	3	0.055
	4	-0.022

Note

- Use to(2, A1.len()) to calculate the positioning sequence of 2~n, where A1.len() returns the length of the sequence.
- In the expression of calculation by position, the calculation can still be performed relative to the previous member.

9.4.3 Loop functions

Loop functions

- The general format of loop functions is **A.f(x)**, which can be calculated for each member of the sequence, and some loop statements of common patterns can be expressed by simple functions.
- The essence of loop function is a simplification of loop calculation. Therefore, for each kind of loop function, the corresponding loop calculation method can be found.
- In fact, the aggregate functions described above can also be regarded as loop functions without parameters. When used as a loop function, the parameter x can also be added. In addition, loop functions also have positioning functions that return sequence numbers of members and select functions that select all or part of members to perform calculation. These two types of functions will be introduced in separate sections later. This section will mainly explain other loop functions, including **A.run()**, **A.rank(y, x)**, **A.median(k:n, x)**, etc.

【Example 9.25】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. Find the average age of employees in the department.

9.4.3 Loop functions

【Example 9.25】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. Find the average age of employees in the department.

Solution

Age should be calculated by age(d) function according to birthday, and the average age can be calculated directly by A.avg(x) aggregation function.

	A	B
1	[1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13]	/Generate a sequence of birthdays
2	=round(A1.avg(age(~)), 2)	/Calculate average age with 2 decimal places

A2	Value
	38.33

Note

- To calculate **A.f(x)** is equivalent to calculating **A.(x)** to get sequence **B**, and then calculate **B.f()**.
- In this example, it is equivalent to calculating **A1.(age(~))** to get the age sequence, and then use A.avg() to calculate the average age. Various aggregation functions can be used in A.f(x) format, such as **A.sum(x)**, **A.count(x)** etc.

A.run(x)

- **A.run(x)** executes the expression x in A in loop and still returns A in the end.
- The loop expression x in A.run(x) needs to change A, and is generally an assignment statement, which is different from A.(x).
- A.run(x) still returns A, so it is often used in the execution cell; while A.(x) returns a new sequence consisting of the results of a loop calculation.

Price adjustment:
10% discount
for the part over
1 million dollars

Index	Member
1	225.0
2	162.0
3	297.0
4	360.0
5	81.0

A.run(x)

Index	Member
1	212.5
2	155.8
3	277.3
4	334.0
5	81.0

A.(x)

Index	Member
1	212.5
2	155.8
3	277.3
4	334.0
5	(null)

【Example 9.26】The unit price of a certain commodity is 90000 dollars, and the number of advance orders of five companies is: 25, 18, 33, 40, 9. Now the sales company carries out preferential activities. If the total amount of orders exceeds 1 million dollars, 10% discount will be given to the amount over 1 million yuan . Please calculate the total amount that each company needs to pay with A.run(x).

9.4.3 Loop functions

A.run(x)

【Example 9.26】The unit price of a certain commodity is 90000 dollars, and the number of advance orders of five companies is: 25, 18, 33, 40, 9. Now the sales company carries out preferential activities. If the total amount of orders exceeds 1 million dollars, 10% discount will be given to the amount over 1 million yuan . Please calculate the total amount that each company needs to pay with A.run(x).

Solution

First use A.(x) to calculate the total price before discount in loop, and then use A.run (~=x) to modify in loop.

	A	B	C
1	[25, 18, 33, 40, 9]		/Generate a sequence of sales quantity
2	=A1.(~*9.0)	=A2.(~)	/Calculate total price before discount in loop
3	=A2.run(if(~>100, ~=(~-100)*0.9+100))	=B2.(if(~>100, ~=(~-100)*0.9+100))	/Calculate the total price after discount

A2	Index	Member
	1	212.5
	2	155.8
	3	277.3
	4	334.0
	5	81.0

B2	Index	Member
	1	212.5
	2	155.8
	3	277.3
	4	334.0
	5	81.0

A3	Index	Member
	1	212.5
	2	155.8
	3	277.3
	4	334.0
	5	81.0

B3	Index	Member
	1	212.5
	2	155.8
	3	277.3
	4	334.0
	5	(null)

Note

- To compare the difference between A.run (x) and A.(x), the total price sequence before discount in A2 is "copied" in B2.
- In A3, A.run (x) function is used to modify the sequence in A2. In the final result, A3 and A2 are identical.
- In B3, A.(x) function is used to modify the sequence in B2, which also completes the discount modification, so B2 is the same as A2 after being modified. However, since only more than 1 million data has been modified in B3, the last member in B3 is null.

Other loop functions

- The basic format of loop function is **A.f (x)**, which is equivalent to calculating A.(x) first and then calculating B.f() in the result sequence B.
- The difference between **A.rank(y, x)** and aggregate function A.rank(y) is not to calculate the rank of y in A, but to calculate the rank of y in A.(x).
- **A.median(k:n, x)** calculates the member in the k/n position. Similarly, calculate A.(x) first, and then calculate the quantile in the results.

【Example 9.27】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. Now it is planned to arrange no less than one-third of the staff on duty. If we want to give priority to young employees on duty, how can we set the age limit for the staff on duty?

【Example 9.27】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. Now it is planned to arrange no less than one-third of the staff on duty. If we want to give priority to young employees on duty, how can we set the age limit for the staff on duty?

Solution

A.median(k:n) is used to calculate the age value at k/n position in ascending order of age. In the **A.median (k: n)** function, the data in A is divided into n groups first. If the data cannot be equally divided, the members in the front group may be one more than those in the latter group, so it can meet the condition of the lowest selected k/n employees on duty. If the data can be equally divided into n groups, the median will be the mean value of the two members on both sides of the k/n position, and there will be exactly k/n employees meeting the duty requirements.

	A	B
1	[1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13]	/Generate a sequence of birthdays
2	=A1.median(1:3, age(~))	/Find the age limit

A2	Value
	38.33

Chapter 9
Calculation of sequence

9.5 Positioning functions

9.5 Positioning functions

Positioning functions

- Finding the position of one or more specified members in a set is called **positioning**.
- When the positioning function is executed, by default, it will search from front to back starting from the first member in the sequence until the first member meeting the condition is found, and its sequence number is returned.
- When using the positioning function, if the **@z** option is added, it will search backward from back to front; if the **@a** option is added, it will return a number sequence of sequence numbers of all members that meet the condition.
- If the members in the sequence are ordered, the **@b** option can be added to do a binary search to greatly improve efficiency.
- The commonly used positioning functions include **A.pos(x), A.pselect(x), A.pmax(x), A.pmin(x), A.pseg(x), A.psort(x), A.contain(x), A.ptop()** etc.

9.5.1 Find the position of the specified member

Find the position of the specified member

- **A.pos(x)** function can be used to find the position of a specified member x in a sequence.
- When calculating A.pos(x), if x can be found in A, the sequence number of the first x is returned; if x can't be found, null is returned.
- If **@n** option is added, A.len()+1 is returned if x can't be found.

Index	Member
1	a
2	b
3	c
4	d
5	d
6	c

Find the position of c

Sequence number
3

9.5.1 Find the position of the specified member

【Example 9.28】The age of employees in a department is as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. Which employee is 25 years old? Which employee is 50 years old? Which employees are 21 years old?

9.5.1 Find the position of the specified member

【Example 9.28】The age of employees in a department is as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. Which employee is 25 years old? Which employee is 50 years old? Which employees are 21 years old?

Solution

Use **A.pos(x)** to find the sequence number of the specified member. If you need to return the sequence numbers of all members with the value of x in the sequence, you need to add the **@a** option.

	A	B
1	[45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51]	/Generate a sequence of ages
2	=A1.pos(25)	/Get the sequence number of the employee who is 25 years old
3	=A1.pos(50)	/Get the sequence number of the employee who is 50 years old
4	=A1.pos@a(21)	/Get the sequence numbers of all employees who are 21 years old

A2	Value
	6

A3	Value
	(null)

A4	Index	Member
	1	3
	2	7
	3	8
	4	10

Note

- Cannot find 50 in sequence in A3, so null is returned. In this case, if the expression in A3 is changed to =A1.pos@n(50) , since there are 11 members in A1, the result will return 12.
- The @a option is added in A4, which returns the number sequence of sequence numbers of all members whose value is 21. In calculating A.pos@a(x), if only one x is found, the sequence of one member will be returned; if x cannot be found, an empty sequence will be returned.

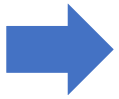
9.5.2 Find the position of a member that meets the condition

Find the position of a member that meets the condition

- **A.pselect(x)** function can be used to find the position of a member that meets the condition x.
- When calculating A.pselect(x), the sequence number of the first member that satisfies x is returned; if the member can't be found, null is returned.
- If **@n** option is added, A.len()+1 is returned if all members can't satisfy x.

Index	Member
1	1
2	2
3	6
4	24
5	120
6	720

Find position of the member whose value is greater than 100



Index
5

9.5.2 Find the position of a member that meets the condition

【Example 9.29】The age of employees in a department is as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. What are the sequence numbers of employees over 40 years old?

9.5.2 Find the position of a member that meets the condition

【Example 9.29】The age of employees in a department is as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. What are the sequence numbers of employees over 40 years old?

Solution

Use **A.pselect(x)** function to find the position of a member according to conditions. **@a** option needs to be added if the sequence numbers of all members that meet the condition need to be returned.

	A	B
1	[45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51]	/Generate a sequence of ages
2	=A1.pselect(~>40)	/Find the sequence number of the employee whose age is over 40.
3	=A1.pselect@a(~>40)	/Find the sequence numbers of all the employees whose age is over 40.

A2	Value
	1

A3	Index	Member
	1	1
	2	4
	3	9
	4	11

Note

- A2 only returns the sequence number of the first employee over 40 years old.
- The @a option is added in A3 to return the sequence of sequence numbers of all employees over 40 years old. The result required for the question is in A3.

9.5.3 Find the position of the Max/Min value

Find the position of the Max/Min value

- **A.pmax()/A.pmin()** function can be used to find the position of the maximum or minimum value.
- **A.pmax(x)** is equivalent to A.(x).pmax(). During execution, A.(x) will be calculated first and then return the position of the maximum value. It is the same with **A.pmin(x)**.
- By default, the search is from front to back. You can use @z option with both functions to search from back to front. When there are multiple records having the max or min value, position of the first-found record will be returned. You can use **@a** to return a sequence of sequence numbers of all records containing the max or min value.

Index	Member
1	9.98
2	11.99
3	9.80
4	9.75
5	9.79
6	9.63
7	9.88
8	9.94

Position of Max value → | Index |
|---|
| 2 |

Position of Min value → | Index |
|---|
| 6 |

9.5.3 Find the position of the Max/Min value

【Example 9.30】In a sprint race, the results of runners from track 1 to track 8 are as follows: 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Which track has the best performance and which track has the worst performance?

9.5.3 Find the position of the Max/Min value

【Example 9.30】In a sprint race, the results of runners from track 1 to track 8 are as follows: 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Which track has the best performance and which track has the worst performance?

Solution

A.pmax() and A.pmin() functions need to be used to find the position of the maximum and the minimum value. Note that the minimum score of the sprint race is the best performance and the maximum value is the worst performance.

	A	B
1	[9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94]	/Generate a sequence of scores
2	=A1.pmin()	/The track of the best performance
3	=A1.pmax()	/The track of the worst performance

A2	Value
	6

A3	Value
	2

9.5.3 Find the position of the Max/Min value

【Example 9.31】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. Who is the oldest employee?

9.5.3 Find the position of the Max/Min value

【Example 9.31】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. Who is the oldest employee?

Solution

We need to find the sequence number of the oldest employee, but the age needs to be calculated, which can be calculated with A.pmax(x).

	A	B
1	[1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13]	/Generate a sequence of birthdays
2	=A1.pmax(age(~))	/Get the sequence number of the oldest employee

A2	Value
	3

Note

- We can also change a solution: The oldest employee's birthday is the earliest, which can be calculated by **A.pmin()**.

9.5.4 Overall positioning

Overall positioning

- **Overall positioning functions** refer to some positioning functions that need to locate all members in the sequence. These functions themselves need to search in all members, so the @a option is not supported. The @z option is also not supported, so reverse search is not possible.
- The overall positioning includes the following functions:

  The A.pos(p) function to find the positions of multiple members in the sequence, where p is a sequence consisting of multi members;

  The A.contain(x_i, ...) function to find whether the sequence contains some members or not;

  The A.psort(x) function to calculate the number sequence of sequence numbers for each member after the sequence is sorted in ascending or descending order;

  The A.ptop(n, x) function to find the number sequence of sequence numbers for the topN members in the sequence;

  The A.pseg(x) function to find the segment of data when the members in A are ordered.

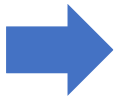
9.5.4 Overall positioning

Positioning of multi members

- **A.pos(p)** function is used to find the position sequence of multi members in the sequence.
- Different from A.pos(x), when A.pos(p) is executed, for each member in p, only the first position is returned, @a option can't be used.
- Unlike finding each member in p individually, if at least one member in p cannot be found, it will be considered that p cannot be located in A, and the A.pos(p) function will return null.

Index	Member
1	a
2	b
3	c
4	d
5	d
6	c

Find the positions of c and a



Index	Member
1	3
2	1

9.5.4 Overall positioning

【Example 9.32】An art school opens different open classes in each period, and the courses to be set in succession are: Watercolor, Cartoon, Chromatics, Sketch, Watercolor, Caricature, Oil Painting, Sketch. If student A wants to learn Sketch and Chromatics, how can he choose? If student B wants to learn Caricature and Photography, how can he choose? If student C wants to learn Sketch first and then Watercolor, how can he choose? If C wants to learn these two courses continuously, can he do it?

9.5.4 Overall positioning

【Example 9.32】An art school opens different open classes in each period, and the courses to be set in succession are: Watercolor, Cartoon, Chromatics, Sketch, Watercolor, Caricature, Oil Painting, Sketch. If student A wants to learn Sketch and Chromatics, how can he choose? If student B wants to learn Caricature and Photography, how can he choose? If student C wants to learn Sketch first and then Watercolor, how can he choose? If C wants to learn these two courses continuously, can he do it?

Solution

A.pos(p) function can be used to locate multiple members in a sequence. If you want the members in p to be continuous in A, you can add the **@c** option. When locating the members of p in A, if you need to search from front to back, that is to say, for the member whose position in p is at the back, its position in A also needs to be at the back, you can add the **@i** option, the returned sequence of sequence numbers must be monotonically increasing.

	A	B
1	[Sketch, Cartoon, Chromatics, Sketch, Watercolor, Caricature, Oil Painting, Sketch]	/Generate a sequence of courses
2	=A1.pos(["Sketch", "Chromatics"])	/the options of student A
3	=A1.pos(["Caricature", "Photography"])	/the options of student B
4	=A1.pos@i(["Sketch", "Watercolor"])	/the options of student C
5	=A1.pos@c(["Sketch", "Watercolor"])	/the options of student C

A2	Index	Member
	1	4
	2	3

A3	Value
	(null)

A4	Index	Member
	1	4
	2	5

A5	Value
	4

When locating multi members, the first position of each member is returned

Even if one member can't be located, null is returned

When using the @i option, one-way search will be performed in A, and the return sequence must be monotonically increasing.

When the @c option is used, the members in p are required to appear continuously in A, the sequence number of the first member found is returned instead of a sequence.

Whether the sequence contains some members or not

- **A.contain(x_i, ...)** function is used to find whether the sequence A contains some members or not, the return result is a Boolean value.
- When **A.contain(x_i, ...)** function is used to judge whether the sequence A contains the specified members or not, true is returned only if A contains all the specified members x_i, ...; If there are n same members in the parameters, when judging, only one member in A is considered to satisfy the condition.

Index	Member
1	a
2	b
3	c
4	d
5	d
6	c

Whether the sequence contains c and a

Whether contains
true

【Example 9.33】An art school opens different open classes in each period, and the courses to be set in succession are: Watercolor, Cartoon, Chromatics, Sketch, Watercolor, Caricature, Oil Painting, Sketch. If a student wants to learn Sketch, Chromatics and Watercolor, can he choose to study in the school?

9.5.4 Overall positioning

Whether the sequence contains some members or not

【Example 9.33】An art school opens different open classes in each period, and the courses to be set in succession are: Watercolor, Cartoon, Chromatics, Sketch, Watercolor, Caricature, Oil Painting, Sketch. If a student wants to learn Sketch, Chromatics and Watercolor, can he choose to study in the school?

Solution

To judge whether a sequence contains the specified members, **A.contain(x_i)** function can be used; **A.pos(p)** function can also be used to judge whether the result is null.

	A	B
1	[Sketch, Cartoon, Chromatics, Sketch, Watercolor, Caricature, Oil Painting, Sketch]	/Generate a sequence of courses
2	=A1.contain("Sketch", "Chromatics", "Watercolor")	/Whether the courses are included or not
3	=A1.pos(["Sketch", "Chromatics", "Watercolor"]) != null	/Whether the result of pos function is null

A2	Value
	true

A3	Value
	true

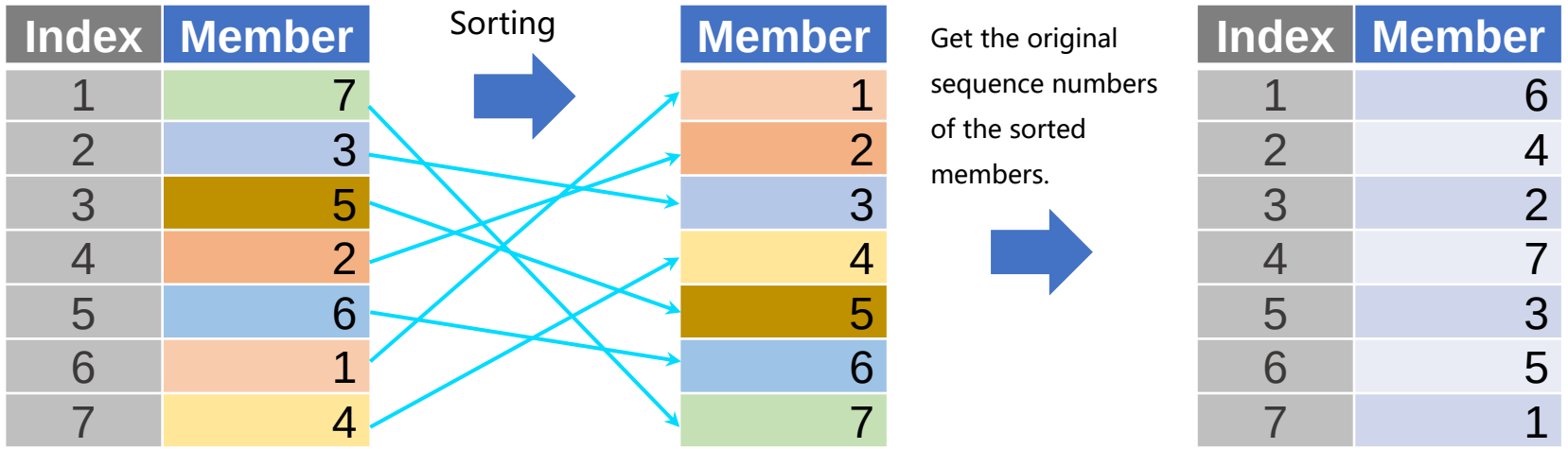
Note

- The results in A2 and A3 are identical. The parameters of the **A.contain(x_i)** function are not a sequence, but the values of each member in turn. Moreover, the effect of this function is relatively simple, and it does not support one-way search, continuous search etc.

9.5.4 Overall positioning

psort positioning

- **A.psort()** function will return a sequence of original sequence numbers of each member after sorting members in A in ascending order.
- If parameter x is added, **A.psort(x)** is equivalent to **A.(x).psort()**, calculate **A.(x)** first and then return its sequence of sequence numbers after sorting.
- **@z** option can be added to **A.psort(x)** function, **A.psort@z(x)** will return a sequence of sequence numbers after sorting **A.(x)** in descending order.



9.5.4 Overall positioning

【Example 9.34】In a sprint race, the results of runners from track 1 to track 8 are as follows: 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Find the tracks for the runners ranking from 1 to 8.

9.5.4 Overall positioning

【Example 9.34】In a sprint race, the results of runners from track 1 to track 8 are as follows: 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Find the tracks for the runners ranking from 1 to 8.

Solution

Here is to find the original sequence numbers of sorted members, A.psort() function can be used.

	A	B
1	[9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94]	/Generate a sequence of scores
2	=A1.psort()	/the tracks for the runners ranking from 1 to 8
3	=A1.psort@z()	/the tracks for the runners ranking from 8 to 1

A2	Index	Member
	1	6
	2	4
	3	5
	4	3
	5	7
	6	8
	7	1
	8	2

A3	Index	Member
	1	2
	2	1
	3	8
	4	7
	5	3
	6	5
	7	4
	8	6

Note

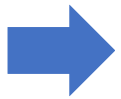
▶ The results in A2 and A3 are in the opposite order. When solving the sorting problem, we need to think about which order should be selected. For the sprint race scores, when ranking, we should take ascending order, and the code in A2 is correct.

9.5.4 Overall positioning

TopN positioning

- **A.ptop(n)** function is used to find the sequence numbers of the first n members after sorting the members in ascending order.
- If parameter x is added, **A.ptop(n, x)** is equivalent **A.(x).ptop(n)** , calculate **A.(x)** first and then return its sequence numbers of the first n members.
- **A.ptop(n, x)** function does not support @z option, but you can add a negative sign before the expression x or n to achieve the effect of sorting in descending order.

Index	Member
1	9.98
2	11.99
3	9.80
4	9.75
5	9.79
6	9.63
7	9.88
8	9.94



Index	Sequence number
1	6
2	4
3	5

9.5.4 Overall positioning

【Example 9.35】In a sprint race, the results of runners from track 1 to track 8 are as follows: 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Find the tracks for the runners ranked top3.

9.5.4 Overall positioning

【Example 9.35】In a sprint race, the results of runners from track 1 to track 8 are as follows: 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Find the tracks for the runners ranked top3.

Solution

Here is to find the original sequence numbers of sorted members, A.psort() function can be used.

	A	B
1	[9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94]	/Generate a sequence of scores
2	=A1.ptop(3)	/the tracks for the runners ranking from 1 to 3
3	=A1.psort().to(3)	/the tracks for the runners ranking from 1 to 3

A2	Index	Member
	1	6
	2	4
	3	5

A3	Index	Member
	1	6
	2	4
	3	5

Note

- The results in A2 and A3 are the same. If the top n is selected, all members can be sorted and then get the top n members. However, the calculation amount of sorting all members will be relatively large, especially when there is a large amount of data. Therefore, the algorithm in A2 is more suitable and efficient.

9.5.4 Overall positioning

【Example 9.36】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. What are the sequence numbers of the three oldest employees? What is the sequence number of the youngest employee?

9.5.4 Overall positioning

【Example 9.36】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. What are the sequence numbers of the three oldest employees? What is the sequence number of the youngest employee?

Solution

To find "3 oldest employees", you need to sort in descending order, and you need to put a negative sign before the sorting expression or n. To find the sequence number of the youngest employee, you can use function A.pmin(x) , or use A.ptop(n, x) to get the sequence number of the top one as the result.

	A	B	C
1	[1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13]		/Generate a sequence of birthdays
2	=A1.ptop(-3, age(~))	=A1.ptop(1, age(~))	/Sequence numbers of the 3 oldest employees and the youngest employee
3	=A1.pmin(age(~))	=A1.ptop@1(1, age(~))	/The youngest employee

A2	Index	Member
	1	3
	2	1
	3	2

B2	Index	Member
	1	6

A3	Value
	6

B3	Value
	6

Note

- The expression in A2 can also be written as=**A1.ptop(3, -age(~))**, and the effect is the same.
- The results in B2 and A3 are different, **A.ptop(n, x)** returns an integer sequence, and a sequence of one member is returned even if "top 1" is located; While **A.pmin(x)** returns a sequence number. If an option is added, **A.ptop@1(n, x)** can be used to locate the first or last one of A(x) , and a sequence number will be returned, as shown in B3.

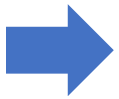
9.5.4 Overall positioning

Segment positioning

- When the members of A are ordered and used to represent a group of interval boundaries, **A.pseg(x)** function can be used to find which interval x is located in.
- When **A.pseg(x)** is used to locate the interval, each interval will adopt the left closed and right open rule by default. You can add the option **@r** to use the interval rule of left opening and right closing.

Index	Member
1	0
2	18.5
3	24
4	28

Find the segment number of 23.2



Segment number
2

When **A.pseg(x)** is used to locate the interval, each interval will adopt the left closed and right open rule by default. In this example, it is equivalent to setting the following four intervals:[0, 18.5), [18.5, 24), [24, 28), [28, +∞). 18.5≤23.2<24, so 23.2 is put into the second interval. If x<0, the returned segment number is 0.

9.5.4 Overall positioning

【Example 9.37】Body mass index (BMI) is often used to measure people's health. BMI=Weight/height^2, weight in kg, height in m. BMI < 18.5 is lean, 18.5 ≤ BMI < 24, fit, 24 ≤ BMI < 28, overweight, BMI ≥ 28, obese. If someone is 1.78m tall and weighs 80.2kg, what is his body evaluation?

9.5.4 Overall positioning

【Example 9.37】Body mass index (BMI) is often used to measure people's health. BMI=Weight/height^2, weight in kg, height in m. BMI < 18.5 is lean, 18.5 ≤ BMI < 24, fit, 24 ≤ BMI < 28, overweight, BMI ≥ 28, obese. If someone is 1.78m tall and weighs 80.2kg, what is his body evaluation?

Solution

Firstly, calculate BMI, and then use A.pseg(x) function to determine the segment number of BMI. Finally, obtain body comments from the evaluation by segment number.

	A	B	C
1	[0, 18.5, 24, 28]	[Lean, Fit, Overweight, Obese]	/Generate BMI interval sequence and evaluation sequence
2	=80.2/1.78/1.78		/Calculate BMI
3	=A1.pseg(A2)	=B1(A3)	/Locate the segment that BMI belongs to, and get the evaluation.

A2	Value
	25.31

A3	Value
	3

B3	Value
	Overweight

Chapter 9
Calculation of sequence

9.6 Selection calculation

Selection calculation

- The operation of finding one or more members in a set according to the conditions is called a **selection calculation**.
- By default, the selection calculation will search from front to back in the sequence until all the members that meet the conditions are found. Most of the selection functions will return a new sequence composed of these members, and some selection functions will return a single value by default.
- When using the selection functions, if the **@z** option is added, it will search from back to front. In this case, the order of the members in the result sequence is different from that without the option.
- If the members in the sequence are ordered, the **@b** option can be added to perform a binary search to greatly improve efficiency.
- Common selection functions include **A.select(x)**, **A.maxp(x)**, **A.minp(x)**, **A.sort(x)**, **A.top(x)** etc.

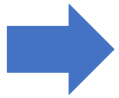
9.6.1 Find members that meet the conditions

Find members that meet the conditions

- To find all the members in the sequence that satisfy the condition x, you can use the **A.select(x)** function.
- When **A.select(x)** is calculated, a sequence consisting of all members that satisfy x is returned; if not found, an empty sequence is returned.
- If the **@1** option is added, only the first member meeting the condition x will be found and the member's single value will be returned. When using **A.select@1(x)**, if the record is not found, null is returned instead of an empty sequence.

Index	Member
1	1
2	2
3	6
4	24
5	120
6	720

Select members greater than 100



Index	Member
1	120
2	720

9.6.1 Find members that meet the conditions

【Example 9.38】The age of employees in a department is as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. Which ages are over 40? Find the results in front to back and back to front order, respectively.

9.6.1 Find members that meet the conditions

【Example 9.38】The age of employees in a department is as follows: 45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51. Which ages are over 40? Find the results in front to back and back to front order, respectively.

Solution

Find members in the sequence according to the conditions, use the A.select(x) function, and add the @z option if you want to find the result from back to front.

	A	B
1	[45, 33, 21, 49, 33, 25, 21, 21, 49, 21, 51]	/Generate a sequence of ages
2	=A1.select(~>40)	/Find the ages greater than 40
3	=A1.select@z(~>40)	/Find the ages greater than 40 from back to front

A2	Index	Member
	1	45
	2	49
	3	49
	4	51

A3	Index	Member
	1	51
	2	49
	3	49
	4	45

Note

- In the sequence of A2 and A3 results, the order of members is opposite, which is related to the order used in the search.
- When A.select(x) function is used to select members, even if only one member is found, the sequence of one member will be returned; if no qualified member is found, the empty sequence will be returned.

9.6.2 Select Max/Min value

Select Max/Min value

- To find the maximum or minimum value in a sequence, you can use the **A.maxp()** or **A.minp()** function.
- If you need to calculate the expression x for the sequence, and then find the member in A corresponding to the maximum or minimum value, use **A.maxp(x)** or **A.minp(x)** function. If you need to return the maximum or minimum value itself, you need to use **A.max(x)** and **A.min(x)**.
- **A.maxp()** or **A.minp()** function returns a single value by default. If you need to return all the maximum / minimum values, you can add the **@a** option.

Index	Member
1	9.98
2	11.99
3	9.80
4	9.75
5	9.79
6	9.63
7	9.88
8	9.94

Max value

Max value
11.99

Min value

Min value
9.63

9.6.2 Select Max/Min value

【Example 9.39】In a sprint race, the results of runners from track 1 to track 8 are as follows: 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Which score is the best and which is the worst?

9.6.2 Select Max/Min value

【Example 9.39】In a sprint race, the results of runners from track 1 to track 8 are as follows: 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Which score is the best and which is the worst?

Solution

The problem is to find the maximum and minimum value. We need to use A.maxp() and A.minp() functions. Note that the best score of the sprint race is the minimum value and the worst score is the maximum value.

	A	B
1	[9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94]	/Generate a sequence of scores
2	=A1.minp()	/Best score
3	=A1.maxp()	/Worst score

A2	Value
	9.63

A3	Value
	11.99

9.6.2 Select Max/Min value

【Example 9.40】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1995-04-11, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. What is the birthday of the youngest employee?

9.6.2 Select Max/Min value

【Example 9.40】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1995-04-11, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. What is the birthday of the youngest employee?

Solution

We need to find the date of birth corresponding to the minimum age, but we need to calculate the age. A.minp(x) can be used.

	A	B
1	[1974-11-20, 1995-04-11, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13]	/Generate a sequence of birthdays
2	=A1.minp(age(~))	/Get the birthday of the youngest employee
3	=A1.minp@a(age(~))	/Get the birthday of all the youngest employees

A2	Value
	1995-04-11

A3	Index	Member
	1	1995-04-11
	2	1995-05-13

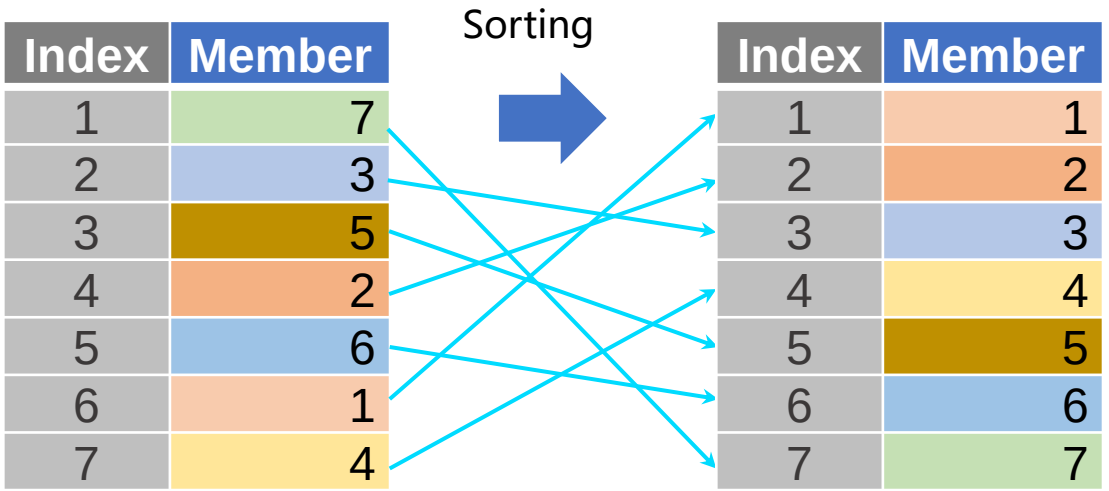
Note

- When searching for the minimum age, even if there are multiple employees with the same age, only the first birthday will be returned, even if he may not be the youngest among all people; the @a option is added in A3 to return "birthdays of all the youngest people", and the sequence will be returned.

9.6.3 Sorting & ranking

Sorting of a sequence

- **A.sort( )** function returns a new sequence sorted in ascending order.
- If the x parameter is added, A.sort(x) still uses the members in A to adjust the order to form a new sequence, but the sorting is based on A.(x).
- **@z** option can be added to **A.sort(x)** function, **A.sort@z(x)** uses the members in A to adjust the order to form a new sequence, and the sorting is based on A.(x) in descending order.



9.6.3 Sorting & ranking

【Example 9.41】In a sprint race, the results of runners from track 1 to track 8 are as follows(seconds): 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Please list the results by ranking.

9.6.3 Sorting & ranking

【Example 9.41】In a sprint race, the results of runners from track 1 to track 8 are as follows(seconds): 9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94. Please list the results by ranking.

Solution

Here you need to sort the data, and A.sort() function needs to be used.

	A	B
1	[9.98, 11.99, 9.80, 9.75, 9.79, 9.63, 9.88, 9.94]	/Generate a sequence of scores
2	=A1.sort()	/Scores in order of ranking

A2	Index	Member
	1	9.63
	2	9.75
	3	9.79
	4	9.80
	5	9.88
	6	9.94
	7	9.98
	8	11.99

Note

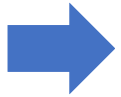
- **A.sort()** function generates a new sequence without affecting A itself. In this example, the results in A2 and A1 are different.

TopN

- **A.top(n)** function is used to find the new sequence composed of the first n members after the sequence is sorted in ascending order.
- If the x parameter is added, **A.top(n, x)** is equivalent to **A.(x).top(n)**. First calculate A.(x) and then return the sequence of the first n members of A.(x) . Note that this situation is different from the previous A.sort(x).
- **A.top(n, x)** function does not support @z option, but you can add a negative sign before n to achieve the effect of sorting in descending order.

Index	Member
1	9.98
2	11.99
3	9.80
4	9.75
5	9.79
6	9.63
7	9.88
8	9.94

Top 3



Index	Top 3
1	9.63
2	9.75
3	9.79

【Example 9.42】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. What are the ages of the three oldest employees?

9.6.3 Sorting & ranking

【Example 9.42】In a certain department, the date of birth of employees is as follows: 1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13. What are the ages of the three oldest employees?

Solution

We need to find"the ages of three oldest employees", so we need to sort in descending order and add a negative sign before n. **A.top(n, x)** function can be used.

	A	B	C
1	[1974-11-20, 1980-07-19, 1970-12-17, 1985-03-07, 1984-07-07, 1995-05-13]		/Generate a sequence of birthdays
2	=A1.top(-3, age(~))	=A1.top(3, -age(~))	/the ages of three oldest people

A2	Index	Member
	1	49
	2	45
	3	40

B2	Index	Member
	1	-49
	2	-45
	3	-40

Note

- Different from **A.ptop(n, x)**, when using the **A.top(n, x)** function, it returns a sequence of values in A. (x), rather than a sequence of sequence numbers. Therefore, the ages returned in B2 has become negative values, which is not correct.
- The option @1 can also be added to **A.top(n, x)**, and the calculation result of the first or last member of A.(x) will be returned. At this time, the corresponding n is 1 or - 1.

Chapter 9
Calculation of sequence

9.7 Calculation of multidimensional sequence

Calculation of multidimensional sequence

- When every member of a sequence is a sequence, the sequence is a **multidimensional sequence**.
- A multidimensional sequence is still a sequence, so you can use various functions of sequence.
- In a multidimensional sequence A, **A.pos@p(x)** can be used to locate the position of member x. To distinguish it from the multi member search of an ordinary sequence, you need to add the **@p** option.
- In a multidimensional sequence A , various set calculations can be performed with individual sequence members, including **A.conj(), A.union(), A.diff(), A.isect()** and **A.xunion()**.

【Example 9.43】It is known that the scores of three groups of students in a certain test are as follows: group 1: 78, 85, 59; group 2: 66, 82, 96; group 3: 88, 66, 100, 69. Find the average score of these students.

9.7 Calculation of multidimensional sequence

【Example 9.43】It is known that the scores of three groups of students in a certain test are as follows: group 1: 78, 85, 59; group 2: 66, 82, 96; group 3: 88, 66, 100, 69. Find the average score of these students.

Solution

The data here is divided into several groups, which is actually a two-dimensional sequence structure. We can use the loop function of the sequence to calculate the sum of the total score of each group to get the total score, calculate the sum of the number of students in each group to get the total number of students, and then calculate the average score. The simpler method is to use all the sequence members to calculate the concatenation, and then call the sequence function to calculate the average score.

	A	B	C
1	[[78, 85, 59], [66, 82, 96], [88, 66, 100, 69]]		/Generate a multidimensional sequence of scores
2	=A1.sum(~.sum())	=A1.sum(~.count())	/Calculate total score and total number of students
3	=A2/B2		/Calculate average
4	=A1.conj().avg()		/Combine all the students' scores into one sequence and then calculate the average score

A2	Value
	789

B2	Value
	10

A3	Value
	78.9

A4	Value
	78.9

9.7 Calculation of multidimensional sequence

【Example 9.44】The school is going to hold sports meet, including 100m, 400m, 3000m, long jump, high jump, javelin and shot put. The following is the registration of five male students. No.1: 100m, long jump; No.2: shot put; No.3: 100m, 400m, 3000m; No.4: long jump, 100m; No.5: shot put, 400m. Who will take part in both the 100m and long jump competitions? What are the events that no one registered?

Solution

The registration data here is a two-dimensional sequence structure. To find the sequence numbers of all students who have signed up for some items, you can use A.pos@ap(p) or A.pselect@a(x). To find which items that no one registered, you can first calculate the union in the registration data to get all the registered items, and then use the total item sequence and result sequence to calculate the difference; or you can add the total item sequence to the registration sequence, and calculate the difference in the two-dimensional sequence.

9.7 Calculation of multidimensional sequence

【Example 9.44】The school is going to hold sports meet, including 100m, 400m, 3000m, long jump, high jump, javelin and shot put. The following is the registration of five male students. No.1: 100m, long jump; No.2: shot put; No.3: 100m, 400m, 3000m; No.4: long jump, 100m; No.5: shot put, 400m. Who will take part in both the 100m and long jump competitions? What are the events that no one registered?

	A	B	C
1	[100m, 400m, 3000m, Long jump, High jump, Javelin, Shot put]	[[100m, Long jump], [Shot put], [100m, 400m, 3000m], [Long jump, 100m], [Shot put, 400m]]	/Generate a sequence of events and a multidimensional sequence of registration
2	=B1.pos@ap(["100m", "Long jump"])	=B1.pselect@a(~.contain("100m", "Long jump"))	/Find out who signed up for the 100m and long jump
3	=A1\B1.union()	=([A1]|B1).diff()	/Calculate the items that no one registered

A2	Index	Member
	1	1

B2	Index	Member
	1	1
	2	4

A3	Index	Member
	1	High jump
	2	Javelin

B3	Index	Member
	1	High jump
	2	Javelin

Note

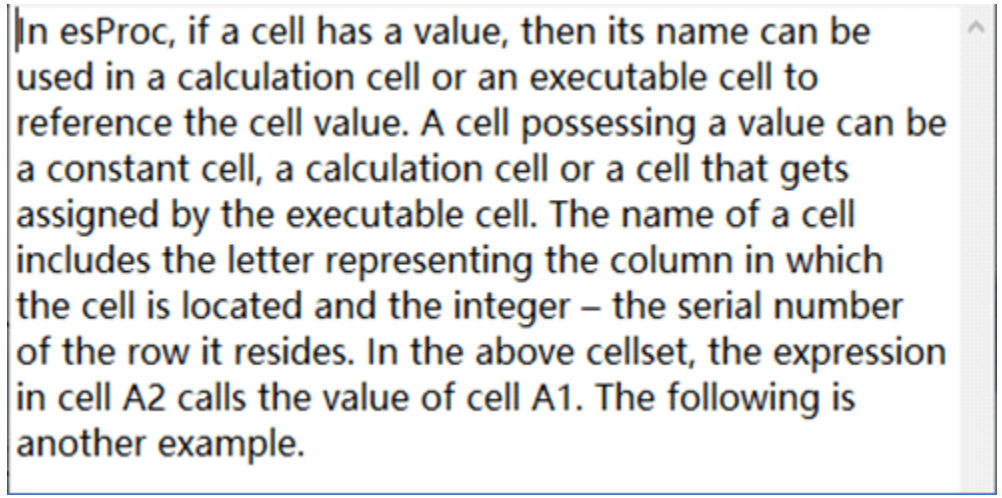
- There are some differences between the results obtained in A2 and B2. This is because although No.1 and No.4 have signed up for 100m and long jump, the order of registration is different, so **A.pos@ap(p)** can't find No.4. **A.contain(xi, ...)** is used in B2 , and the judgment ignores the order of the members in the sequence, and also considers the situation that the players sign up for more events. The calculation method in B2 is more reasonable. In addition, it should be noted that various string sequence members can be input directly as constants, and quotation marks should be added in expressions.
- The results obtained in A3 and B3 are the same. The effect of first calculating union and then calculating difference is the same as that of calculating difference after merging.

Chapter 9
Calculation of sequence

9.8 Examples of sequence calculation

9.8 Examples of sequence calculation

【Example 9.45】Read a text and count the number of word "the".



9.8 Examples of sequence calculation

【Example 9.45】Read a text and count the number of word "the".

In esProc, if a cell has a value, then its name can be used in a calculation cell or an executable cell to reference the cell value. A cell possessing a value can be a constant cell, a calculation cell or a cell that gets assigned by the executable cell. The name of a cell includes the letter representing the column in which the cell is located and the integer – the serial number of the row it resides. In the above cellset, the expression in cell A2 calls the value of cell A1. The following is another example.

Solution

Read the text to generate a string, and then extract the word sequence. Then filter out the sequence of the word "the", and finally count the number of members.

s.words() can take words out of a string to form a string sequence.

A.select(x) selects the eligible members in the sequence.

A.len() can get the number of members in the sequence.

	A	B
1	=file("text.txt").read().words()	/Split the words to form a sequence
2	=A1.select(lower(~)=="the")	/Filter out "the"
3	>output("Count="/A2.len())	/Count the members

Output result:

```
Count=13
```

Note

- In this example, the words selected in A2 will not have a null value, so it is more efficient to use A.len() instead of using A.count() to calculate. It is also necessary to exclude non empty members when using this function.

【Example 9.46】There is a group of seven judges' diving scores (7.8, 8.3, 8.1, 8.5, 8.0, 8.1, 8.0). The scoring rules are as follows: remove the highest score and the lowest score, and then multiply the total score by the action difficulty coefficient (2.7), and finally multiply by 3 / 5.

9.8 Examples of sequence calculation

【Example 9.46】There is a group of seven judges' diving scores (7.8, 8.3, 8.1, 8.5, 8.0, 8.1, 8.0). The scoring rules are as follows: remove the highest score and the lowest score, and then multiply the total score by the action difficulty coefficient (2.7), and finally multiply by 3 / 5.

Solution

According to the scoring rules, this problem can be calculated by calling SPL related functions.

A.delete(k) can delete the specified members in sequence A.

A.pmax() and **A.pmin()** can get the sequence numbers of the largest / smallest member in the sequence.

A.sum() can calculate the sum of all non empty members in the sequence.

	A	B
1	[7.8, 8.3, 8.1, 8.5, 8.0, 8.1, 8.0]	/seven judges'diving scores
2	2.7	/action difficulty coefficient
3	>A1.delete(A1.pmax())	/remove the highest score
4	>A1.delete(A1.pmin())	/remove the lowest score
5	=A1.sum()	/calculate the sum of scores
6	=A5*A2*3/5	/calculate the final score

A6	Value
	65.61

【Example 9.47】A group of eight students participated in the mathematics simulation test with the scores of 89.5, 69.5, 78, 66, 45.5, 72.5, 43.5, 59.5. Among them, Ryan got 66 points. What is his ranking in the group and the average score of the group?

9.8 Examples of sequence calculation

【Example 9.47】A group of eight students participated in the mathematics simulation test with the scores of 89.5, 69.5, 78, 66, 45.5, 72.5, 43.5, 59.5. Among them, Ryan got 66 points. What is his ranking in the group and the average score of the group?

Solution

After sorting, locate the ranking of Ryan's score in the group, and then calculate the average score.

A.sort() sorts the members in the sequence.

A.pos(x) gets the sequence number of a sequence member.

A.avg() gets the average of the non null members in the sequence.

	A	B
1	[89.5, 69.5, 78, 66, 45.5, 72.5, 43.5, 59.5]	/the scores of the group
2	66	/Ryan's score
3	=A1.sort@z()	/Sort the scores in the group in descending order
4	=A3.pos(66)	/Calculate Ryan's ranking in the group
5	=A1.avg()	/Calculate the average score

A4	Value
	5

i.e. Ryan's score ranks 5th in the group

A5	Value
	65.5

The average score of the group is 65.5

【Example 9.48】Without calling the special function of SPL, use the matrix method solve the Binary Linear Equation Group $\begin{cases} 2*x+y=5 \\ x+2*y=4 \end{cases}$

9.8 Examples of sequence calculation

【Example 9.48】Without calling the special function of SPL, use the matrix method solve the Binary Linear Equation Group $\begin{cases} 2*x+y=5 \\ x+2*y=4 \end{cases}$

Solution

Create the augmented matrix of the equation, and then the coefficient of y is changed to 0, and then the coefficient of x is changed to 1, then the value of x can be obtained. Then the obtained value of x is substituted into another equation where the coefficient of y is not 0, and the value of y can be obtained by converting the coefficient of y to 1.

	A	B
1	[[2, 1, 5], [1, 2, 4]]	/Create the augmented matrix of the equation group
2	>A1(1)=A1(1).(~*A1(2)(2))--A1(2).(~*A1(1)(2))	/eliminate y in the first equation by addition and subtraction
3	>A1(1)=A1(1).(~/A1(1)(1))	/Get the value of X in the first equation
4	>A1(2)=A1(2)--A1(1).(~*A1(2)(1))	/eliminate x in the second equation by addition and subtraction
5	>A1(2)=A1(2).(~/A1(2)(2))	/Get the value of y
6	>output("x="/A1(1)(3)/", y="/A1(2)(3))	/Output result

Output result: x=2.0, y=1.0

That is, the solution of the equation is : $\begin{cases} x=2 \\ y=1 \end{cases}$

Chapter 9
Calculation of sequence

9.9 Exercises

9.9 Exercises

【Exercise 9.1】Given all prime numbers within 30, how to quickly find all the composite numbers within 30?

【Exercise 9.2】Suppose someone goes to a certain place to work by train. The fixed schedule is: work by train on Thursday of the first week, return on Tuesday of the second week, and then work by train on Thursday of the third week, …… this goes on and on. Known: March 5, 2020 (Thursday), scheduled to go to work. Ask: from the beginning of March to the end of April, which days of departure and return tickets need to be bought?

【Exercise 9.3】Read the order amount saved in amount.txt, and count the number of orders whose amount is greater than or equal to 1000.

【Exercise 9.4】Construct the Fibonacci sequence with 50 members by recursion method and general term formula method respectively, and verify the two are completely equal. The general formula is as follows: $a_n = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n]$

【Exercise 9.5】There are 10 people who enrolled in the test, and the scores are as follows: (59, 63, 52.5, 57, 48, 63.5, 66, 32.5, 45, 55)The top 6 are admitted. What is the score of the sixth place?

【Exercise 9.6】There is a string file name "filename@www.xxxx.com.rar" , get the file name string after removing "@www.xxxx.com".

【Exercise 9.7】Use sequence to find the cyclic section of infinite cyclic decimal, taking 47 / 19 as an example.

【Exercise 9.8】Process a group of collected battery voltage data (36.4, 36.5, 36.8, 36.3, 36.5, 36.6, 36.3, 35.3, 36.4, 36.1, 36.5, 36.2, 36.7, 36.5, 36.3) by using the moving average filtering algorithm (n = 4).

9.9 Exercises

【Exercise 9.1】Given all prime numbers within 30, how to quickly find all the composite numbers within 30?

Solution

Positive integers include prime number, composite number and 1. Remove the prime number set within 30 and then remove 1 from the set of positive integers within 30, we can get the set of composite numbers within 30. We can use the difference set operation of sequence to deal with this problem.

	A	B
1	[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]	/The set of prime numbers within 30
2	=to(30)\A1\[1]	/Get the composite set by calculating difference

A2	Index	Member
	1	4
	2	6
	3	8
	4	9
	5	10
	6	12
	…	…

9.9 Exercises

【Exercise 9.2】Suppose someone goes to a certain place to work by train. The fixed schedule is: work by train on Thursday of the first week, return on Tuesday of the second week, and then work by train on Thursday of the third week, ... this goes on and on. Known: March 5, 2020 (Thursday), scheduled to go to work. Ask: from the beginning of March to the end of April, which days of departure and return tickets need to be bought?

Solution

Through simple calculation, it can be seen that there are 14 days between the departure dates, and the same is true for the return dates. The latest departure date is March 5th and the latest return date is March 10th.

The **periods()** function in SPL can obtain the date time sequence according to the specified time interval, such as using @y as the unit of year, no option as the unit of day, using @s as the unit of second, etc.

	A	B	C
1	2020-03-01	2020-04-30	
2	=periods(A1, B1, 1)		/The date sequence
3	=A2.step(14, 5)		/The departure dates
4	=A2.step(14, 10)		/The return dates

Departure

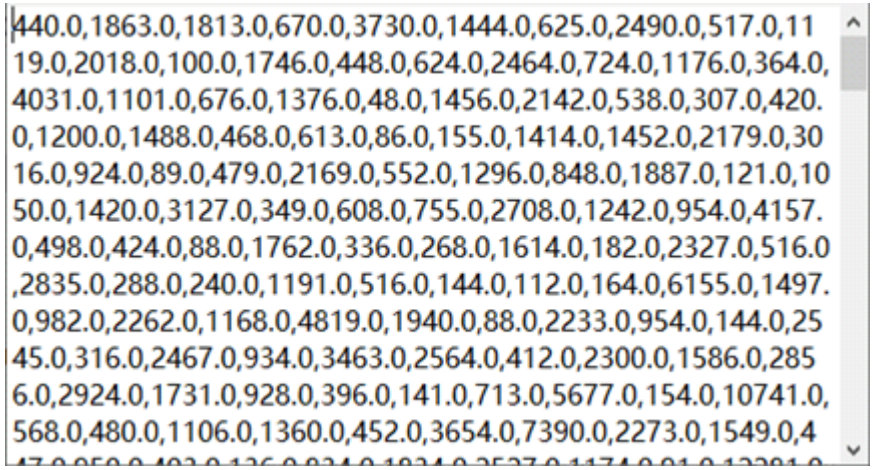
A3	Index	Member
	1	2020-03-05
	2	2020-03-19
	3	2020-04-02
	4	2020-04-16
	5	2020-04-30

Return

A4	Index	Member
	1	2020-03-10
	2	2020-03-24
	3	2020-04-07
	4	2020-04-21

9.9 Exercises

【Exercise 9.3】Read the order amount saved in amount.txt, and count the number of orders whose amount is greater than or equal to 1000.


440.0,1863.0,1813.0,670.0,3730.0,1444.0,625.0,2490.0,517.0,11
19.0,2018.0,100.0,1746.0,448.0,624.0,2464.0,724.0,1176.0,364.0,
4031.0,1101.0,676.0,1376.0,48.0,1456.0,2142.0,538.0,307.0,420.
0,1200.0,1488.0,468.0,613.0,86.0,155.0,1414.0,1452.0,2179.0,30
16.0,924.0,89.0,479.0,2169.0,552.0,1296.0,848.0,1887.0,121.0,10
50.0,1420.0,3127.0,349.0,608.0,755.0,2708.0,1242.0,954.0,4157.
0,498.0,424.0,88.0,1762.0,336.0,268.0,1614.0,182.0,2327.0,516.0
,2835.0,288.0,240.0,1191.0,516.0,144.0,112.0,164.0,6155.0,1497.
0,982.0,2262.0,1168.0,4819.0,1940.0,88.0,2233.0,954.0,144.0,25
45.0,316.0,2467.0,934.0,3463.0,2564.0,412.0,2300.0,1586.0,285
6.0,2924.0,1731.0,928.0,396.0,141.0,713.0,5677.0,154.0,10741.0,
568.0,480.0,1106.0,1360.0,452.0,3654.0,7390.0,2273.0,1549.0,4

Solution

First filter out the data whose order amount is greater than or equal to 1000, and then count the quantity.

	A	B
1	=file("amount.txt").read().split@cp()	/@p: Automatically resolve type
2	=A1.select(~>=1000)	/Keep only data greater than or equal to 1000
3	=A2.count()	/Count

A3	Value
	419

9.9 Exercises

【Exercise 9.4】Construct the Fibonacci sequence with 50 members by recursion method and general term formula method respectively, and verify the two are completely equal. The general formula is as follows: $a_n = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n]$

Solution

Generate Fibonacci sequence by two methods with sequence loop functions, and get the difference set of two sequences. If the difference set is empty, it is proved that they are completely equal.

	A	B	C
1	50	[1, 1]	
2	>A1.run(if(#>2, B1.insert(0, B1.m(-1)+B1.m(-2))))		/Generating Fibonacci sequence in B1 by recursion
3	=A1.(long(round(1/sqrt(5)*(power((1+sqrt(5))/2, ~)-power((1-sqrt(5))/2, ~)))))		/Generating Fibonacci sequence by formula method
4	=[B1, A3].diff()		/Empty sequence proves no difference

B1	Index	Member
	1	1
	2	1
	3	2
	4	3
	5	5
	6	8
	…	…

A3	Index	Member
	1	1
	2	1
	3	2
	4	3
	5	5
	6	8
	…	…

A4	Index	Member

9.9 Exercises

【Exercise 9.5】There are 10 people who enrolled in the test, and the scores are as follows: (59, 63, 52.5, 57, 48, 63.5, 66, 32.5, 45, 55)The top 6 are admitted. What is the score of the sixth place?

Solution

First sort the test scores in descending order, and then take the top six, then the score of the sixth place is the score line.

The A.top() function in SPL is suitable for getting the top n members from the original data, which is more efficient than executing A.sort() and then executing A.to().

	A	B
1	[59, 63, 52.5, 57, 48, 63.5, 66, 32.5, 45, 55]	/The sequence of test scores
2	=A1.top(-6;~)	/Get the first 6 members in descending order
3	=A2(6)	/The score of the 6th place

A3	Value
	55

9.9 Exercises

【Exercise 9.6】There is a string file name "filename@www.xxxx.com.rar" , get the file name string after removing "@www.xxxx.com".

Solution

Split the file name to a sequence of characters with **s.split(d)** function, delete the specified part and concatenate the remaining part to return.

	A	B
1	filename@www.xxxx.com.rar	
2	=A1.split()	/Split the string into a sequence of characters
3	=A2.delete(to(A2.pos@z("@"), A2.pos@z(".")-1)).concat()	/Delete the unneeded substring and concatenate the remaining part to return

A3	Value
	filename.rar

Note

- You can also use string function to do the same :=left(A1, pos@z(A1, "@")-1) or =right(A1, len(A1)-pos@z(A1, ".")+1)

9.9 Exercises

【Exercise 9.7】Use sequence to find the cyclic section of infinite cyclic decimal, taking 47 / 19 as an example.

Solution

You can use the method of comparing the remainder to determine the cyclic section of a recurring decimal.

	A	B	C	D
1	47	19		/A1 is the dividend and B1 is the divisor
2	=A1%B1			/A2 calculates the remainder to be used in the first cycle
3	for	=B5.pos(A2)		/A3 defines the infinite loop, and B3 determines the position of the remainder in this cycle
4		if B3 > 0	break	/If the remainder used in this cycle appeared in the previous cycle, break the current cycle
5		=B5 | A2		/B5 saves the remainder of each calculation
6		=B6 | int(A2*10/B1)		/B6 stores the quotient of each calculation, which is the cyclic section
7		>A2 = A2*10%B1		/A2 is updated to the remainder to be used in the next loop
8	=B6.to(B3,).concat()			/In B6, from position B3 to the end, it is a cyclic section, which is concatenated into a string

A8	Value
	473684210526315789

9.9 Exercises

【Exercise 9.8】Process a group of collected battery voltage data (36.4, 36.5, 36.8, 36.3, 36.5, 36.6, 36.3, 35.3, 36.4, 36.1, 36.5, 36.2, 36.7, 36.5, 36.3) by using the moving average filtering algorithm (n = 4).

Solution

The so-called moving average filtering is to put n sampling data into the data buffer in order (n = 4 for this problem) and calculate the arithmetic mean value. Then, each time a new data is included, the first sampled data is discarded, and then the arithmetic mean value of N data including the new data is calculated, so as to cycle until all the data are processed. In the sequence loop, the sampling data can be obtained by relative reference interval ~[a, b], and the average value can be calculated.

	A	B	C
1	[36.4, 36.5, 36.8, 36.3, 36.5, 36.6, 36.3, 35.3, 36.8, 36.4, 36.1, 36.5, 36.2, 36.7, 36.5, 36.3]	[]	/Input the collected data and record the filtering results with B1
2	>A1.run(if(#>3, B1=B1|round(~[-3, 0].avg(), 2))		/Loop through the collected data, and 4 members are used to calculate the mean value from the fourth member onward, and add the mean value to the filtering result sequence.

B1	Index	Member
	1	36.5
	2	36.53
	3	36.55
	4	36.42
	5	36.17
	6	36.25
	7	36.2
	8	36.15
	9	36.45
	10	36.3
	11	36.38
	12	36.48
	13	36.42

Chapter 10
Data table

10.1 Structured data

- Structured data
- Field and record
- Data table

10.2 Operations of table sequence

- Create/View a table sequence
- Record access
- Basic addition, deletion and modification
- Examples and extension operations

10.3 External data

- Data file
- Reading file data
- Exporting file data
- Database query

10.4 Table sequence operations

- Aggregation
- Filter

10.5 Record sequence

- The concept of record sequence
- Record sequence operations
- The difference between rseq and tseq

10.6 Exercises

Chapter 10
Data table

10.1 Structured data

10.1.1 Structured data

When we process data, we will encounter data with similar structure below.

Student score table:

	A	B	C	D	E	F
1	Index	Name	Chinese	Math	English	TotalScore
2	70605	YangLu	131	143	144	418
3	70603	WangXue	131	135	144	410
4	70609	HanLin	127	139	142	408
5	70601	ShaLongYu	123	148	136	407
6	70606	LiJianXue	126	135	140	401
7	70604	HanYuPeng	129	133	138	400

For example, Yanglu's score information is one line, each line includes index, name, Chinese score, math score and other information.

10.1.1 Structured data

Employee attendance table:

Date	Sid	Name	StartTime	EndTime	Attendance
2020-03-01	1012	Tom	08.59	18.30	Normal
2020-03-01	1015	Jack	08.52	18.00	Normal
2020-03-01	1016	Adam	08.43	18.05	Normal
…	…	…			

For example, the attendance information of employee Tom is one line, and each line includes information such as date, employee id and employee name.

Order table

No	Name	Price	Num	Details
20200501	banana	5.00	5	
20200502	apple	7.50	7	From Chengdu
20200503	cucumber	4.60	5	
20200504	orange	6.50	2	
20200505	egg	8.60	5	

For example, banana order information is one line, and each line includes serial number, commodity name, price and other information.

Visitor registration table

No	Name	Gender	IDCard	Tel	PDate	PTime	Content	Comment
1	Hando	M	xxx	123456789	2020-05-02	10:25	Client Review	

For example, the registration information of visitor Hando is one line, and each line includes serial number, name, gender and other information.

Stock table

No	Symbol	Name	LastTrade	Chg	Change
1	688123	Juchen Co., Ltd	73.30	15.23%	9.69
2	688037	Core micro	118.88	12.19%	12.92
3	002347	Tel shares	4.57	10.12%	0.42
4	601616	SGEG	3.81	10.12%	0.35
5	300165	SKYRAY	4.90	10.11%	0.45

For example, SGEG stock information is one line, and each line includes serial number, stock code, name, the latest price and other information.

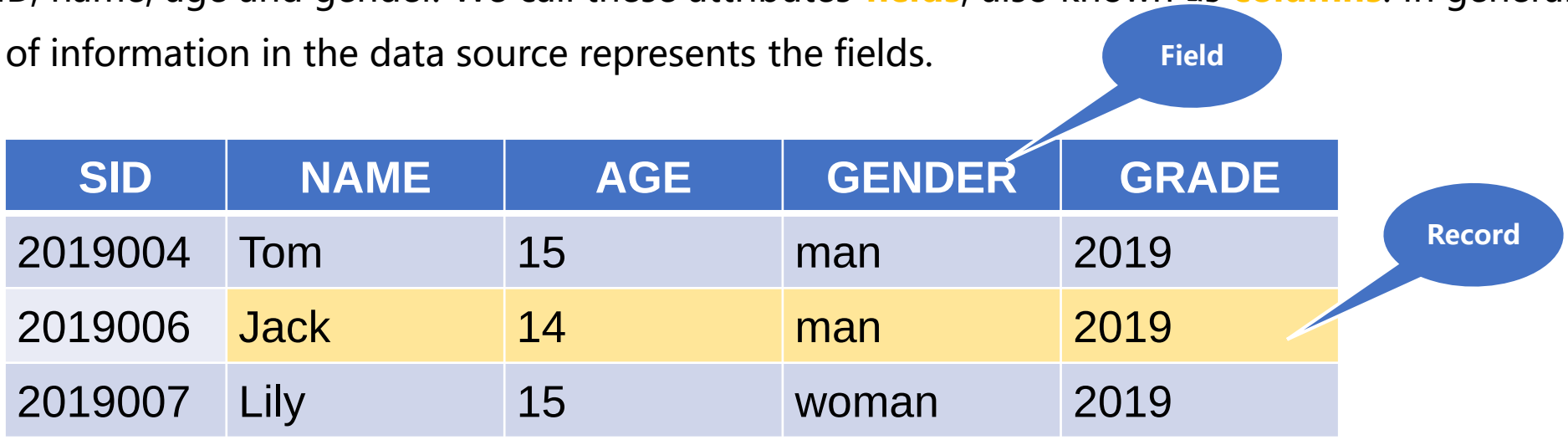
10.1.1 Structured data

From the above examples, we can see that although the contents are different, they are all composed of information line by line. Each line of information is composed of several different attributes. Data like this is **structured data**.

Index	CLASS	STUDENTID	SUBJECT	SCORE
1	Class one	1	English	84
2	Class one	1	Math	77
3	Class one	1	PE	69
4	Class one	2	English	81
5	Class one	2	Math	80

10.1.2 Field and record

Student registration form belongs to structured data, which requires to fill in attribute information such as student ID, name, age and gender. We call these attributes **fields**, also known as **columns**. In general, the first row of information in the data source represents the fields.



SID	NAME	AGE	GENDER	GRADE
2019004	Tom	15	man	2019
2019006	Jack	14	man	2019
2019007	Lily	15	woman	2019

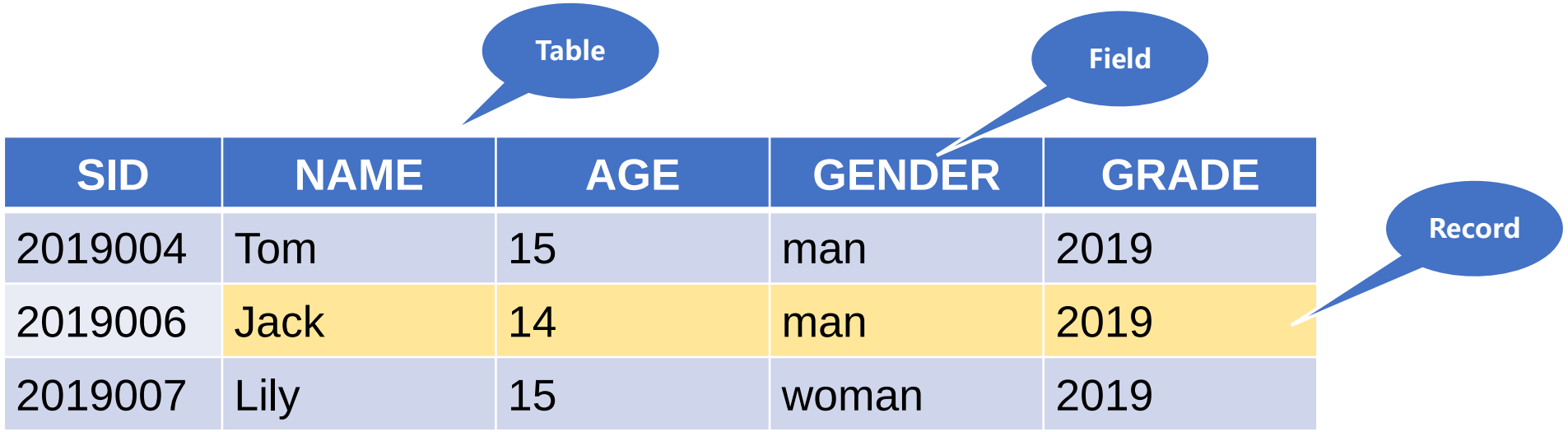
Each row in the registration form represents a data **record**, and a record consists of several field values.

That is, **rows** represent records, **columns** represent fields; each row represents a record, and each column represents a field.

This batch of data with records and fields is called **data table.**

10.1.3 Data table

The set of fields in a data table is called **data structure**, or structure for short. In the student registration form, the set of SID, name, age, gender and grade fields contained in each student information is the data structure of this data table. Sometimes we also say that data structure belongs to records, because each record also has these fields.



Data table is called **table** for short, and consists of a batch of data (records) with the same structure.
Data table is often called two-dimensional table because it is a row column two-dimensional structure. The data in the table can be determined by the row number and column number of the cell in which it is located, so as to facilitate reading and writing. For example, lily is in row 3 and column 2.

10.1.3 Data table

Let's take a look at the order table mentioned earlier.



No	Name	Price	Num	Details
20200501	banana	5.00	5	
20200502	apple	7.50	7	From Chengdu
20200503	cucumber	4.60	5	
20200504	orange	6.50	2	
20200505	egg	8.60	5	

The **fields** of the data table are No, Name, Price, Num, Details.
A **record** represents a piece of order information, for example, the field values of the record in the second row are **20200502, apple, 7.50, 7, From Chengdu**.
Generally, row and column numbers are used to represent the position of data in the table. For example, **apple** is in row 2, column 2.

10.1.3 Data table

The fields and records in the stock table

No	Symbol	Name	LastTrade	Chg	Change
1	688123	Juchen Co., Ltd	73.30	15.23%	9.69
2	688037	Core micro	118.88	12.19%	12.92
3	002347	Tel shares	4.57	10.12%	0.42
4	601616	SGEG	3.81	10.12%	0.35
5	300165	SKYRAY	4.90	10.11%	0.45

The **fields** include **No**, **Symbol**, **Name**, **LastTrade**, **Chg**, and **Change**.
A **record** represents the information of one stock. For example, the field values of the record in row 2 are **688037**, **Core micro**, **118.88**, etc.
To access data, you can use row and column number. For example, **Tel shares** is in row 3, column 3.

Chapter 10
Data table

10.2 Operations of table sequence

10.2.1 Create/View a table sequence

Before creating a data table, you should first determine what the data structure is, that is, what fields are there.

For example, the user information table consists of four fields: SID, user name, gender and age. With the structure information, you can create a table. The data table in SPL is also called **table sequence** or **tseq**, which is often used to represent the data table later.

	A	B
1	=create(SID, NAME, GENDER, AGE)	/Create user table

A1 returns an **empty table** with the structure of SID, name, gender and age. It has only structure but no record.

Index	SID	NAME	GENDER	AGE

When displayed, the index column in the table is only an auxiliary column, which represents the record sequence number in the table sequence, not the real field in the table sequence. This is the same as the index column used to display the member number in the sequence.

10.2.1 Create/View a table sequence

After creating the user information table, you can use **T.record(A)** to write sequence A data into table T in turn.

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])	/Fill in data	

[A1: D3] is the data prepared. After executing the script, the data in table A4 is as follows:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11

T.record(A): In the table sequence T, each field value of the record is appended by the member of A in turn.

10.2.2 Record access

A table sequence can be understood as a sequence of records, and its members can be accessed with **T(k)**, where k is the row position of records in the table.

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])		
5	=A4			/Access table sequence
6	=A4(1)			/Get the first record in the table
7	=A4(3)			/Get the third record in the table

A5	SID	NAME	GENDER	AGE
	1	Jack	M	12
	2	Tom	M	15
	3	July	F	11

A6	SID	NAME	GENDER	AGE
	1	Jack	M	12

A7	SID	NAME	GENDER	AGE
	3	July	F	11

10.2.2 Record access

Field is considered as the attribute of a record. In computer language, it is generally used to access the field value with the syntax of **r.F**. Sometimes it can be accessed by using the sequence number **r.#i**, where **#i** is the i^{th} field.

	A	B	C	D
1	1	Tom	M	15
2	2	July	F	11
3	=create(SID,NAME,GENDER,AGE)	>A3.record([A1:D2])		/Table sequence
4	=A3(1)			/Get the first record in the table
5	=A4.NAME			/Get the value of the field NAME in the record
6	=A4.#2			Get the value of the second field(NAME) in the record
7	=A3(1).NAME			/Gets the value of the field NAME in the first record

A3

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	11

Record

A4

SID	NAME	GENDER	AGE
1	Tom	M	15

Field value

A5

Value
Tom

The results in A6 and A7 are the same as those in A5

10.2.2 Record access

Access to a field value of the n^{th} record, such as the user name July of the third record.

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	=A3.record([A1:D3])		/Table sequence
5	=A4(3).NAME			/Access the user name of record 3
6	=A4(3).#2			/Access the user name of record 3
7	=A4(1).AGE			/Access the user Age of record 1
8	=A4(1).#4			/Access the user Age of record 1
9	=A4(1).array()			/Get the sequence of values of all fields in the first record
10	=A4(1).array(NAME, AGE)			/Get the sequence of values of the NAME and AGE fields in the first record

10.2.2 Record access

Modify the field value of the record. **r.F=x** modify the value of F field of record r to x, or **r.#i=x**, where #i is the i^{th} field of record r.

	A	B	C	D
1	1	Tom	M	15
2	2	July	F	11
3	=create(SID, NAME, GENDER, AGE)	=A3.record([A1:D2])		/Table sequence
4	=A3(1)			/Access the first record and return record A4
5	>A4.NAME="Jack"			/Modify the value of the record field NAME
6	>A4.#2="Jack"			/Modify the value of the record field NAME

A4

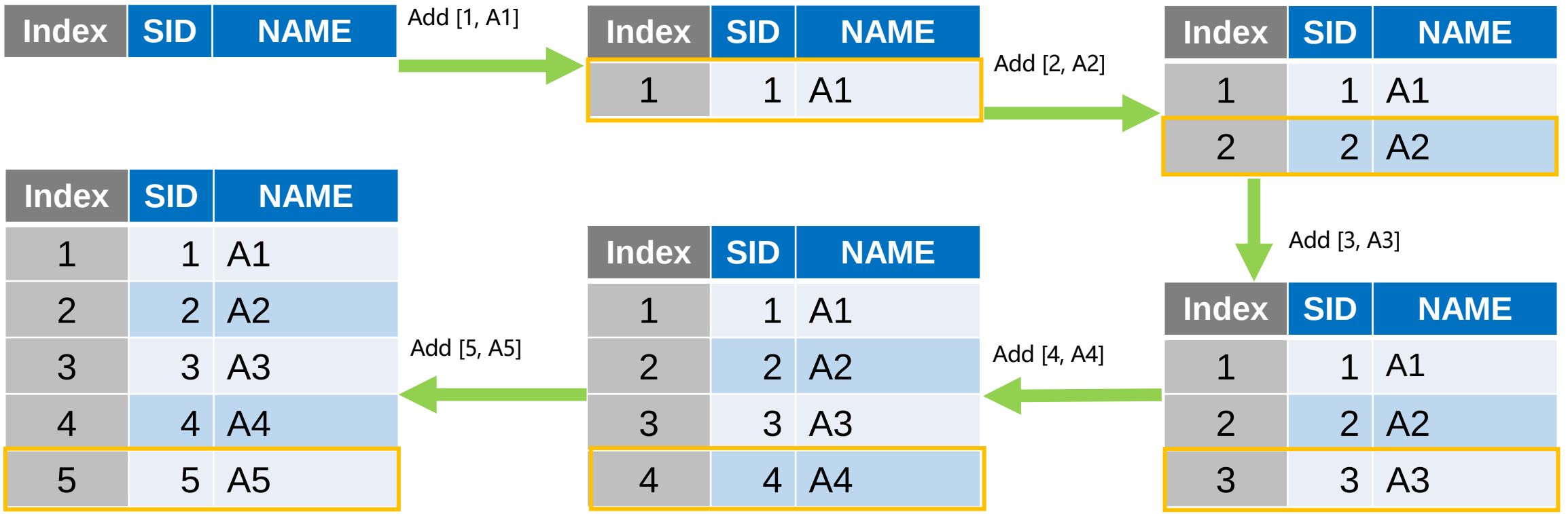
SID	NAME	GENDER	AGE
1	Tom	M	15

Modification →

A4

SID	NAME	GENDER	AGE
1	Jack	M	15

Add record: add a new record at the end of the table sequence



The newly added data is placed at the end of the table, and the data is arranged in the order of appending.

Add records

Add records. Append the newly created record to the table sequence T with **T.insert(0, x_i, ...).**

	A	B
1	=create(SID, NAME, GENDER, AGE)	/Create empty table A1
2	>A1.insert(0, 1, "Tom", "M", 15)	/Add one record
3	>A1.insert(0, 2, "July", "F", 11)	/Add one record
4	>A1.insert(0, 3, "Gail", "M", 13)	/Add one record

Index	SID	NAME	GENDER	AGE

A2 addition

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15

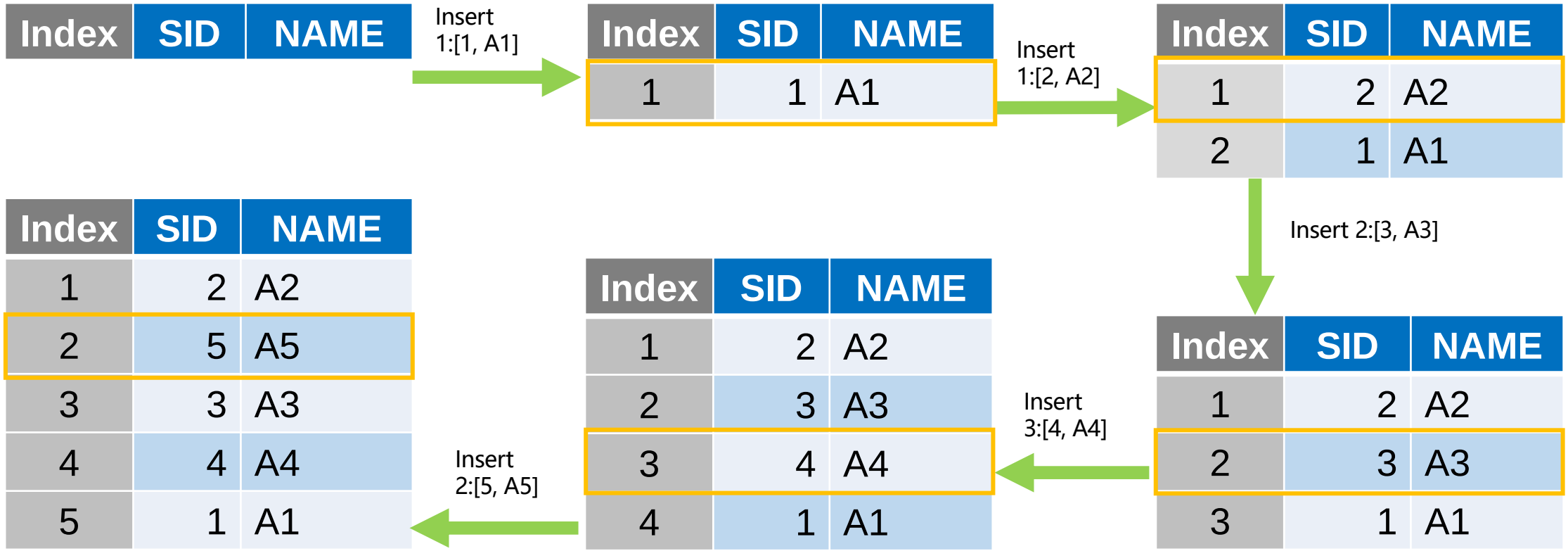
A3 addition

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	11

A4 addition

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	11
3	3	Gail	M	13

Insert record: insert the newly created record in the specified position of the table sequence, and the records after this position will be moved down accordingly.



Insert a record in the specified position, the newly added record is stored in the specified position, and the records after the specified position move down in turn.

Add records

Insert record with **T.insert(k, ...)**. Insert the newly created record before row k in the table.

	A	B
1	=create(SID, NAME, GENDER, AGE)	/Create empty table A1
2	>A1.insert(1, 1, "Tom", "M", 15)	/Add one record
3	>A1.insert(2, 2, "July", "F", 11)	/Insert record to row 2
4	>A1.insert(2, 3, "Gail", "M", 13)	/Insert record to row 2

Index	SID	NAME	GENDER	AGE

A2 insertion →

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15

A3 insertion ↓

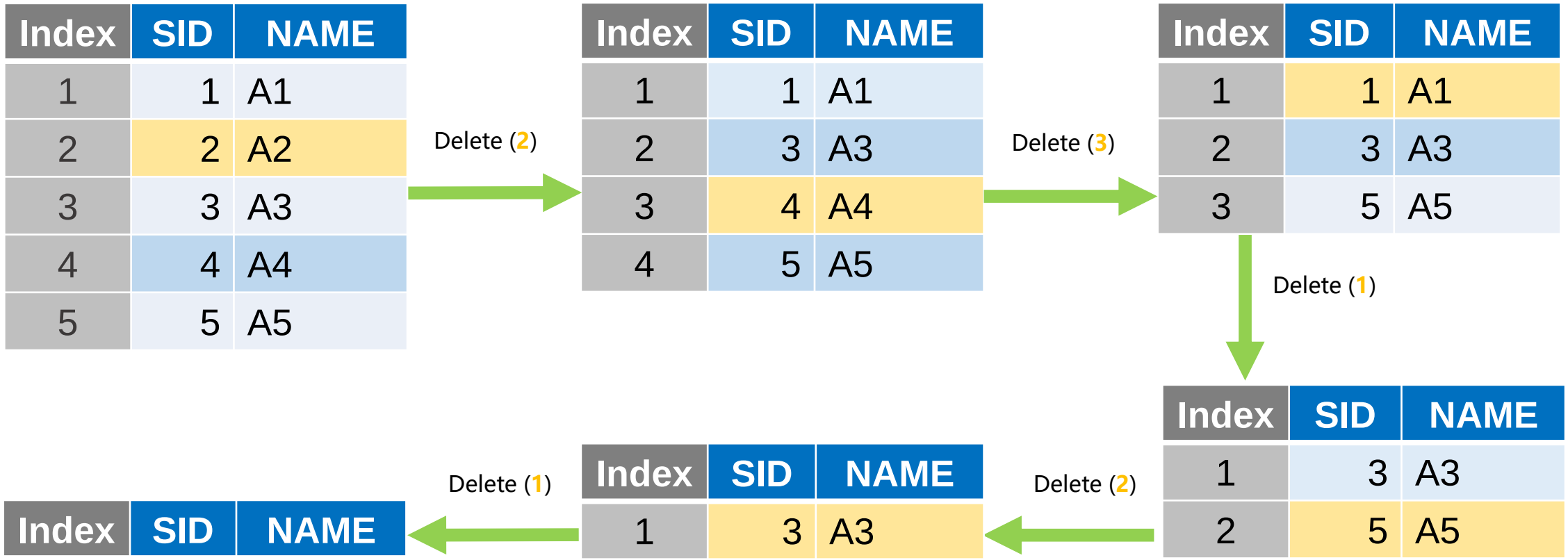
Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	11

A4 insertion ←

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	3	Gail	M	13
3	2	July	F	11

10.2.3 Basic addition, deletion and modification

Delete record: delete the record at the specified position, and the records after it will be moved up.



Pay attention to the change of record row number in the process of deleting records.

10.2.3 Basic addition, deletion and modification

Delete a record with **T.delete(k).** Delete record **k.**

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11
5	=create(SID, NAME, GENDER, AGE)	=A5.record([A1:D4])		/Table sequence
6	>A5.delete(2)	>A5.delete(1)		/A6 deletes the second record in the table, and B6 deletes the first record
7	>A5.delete(1)			/A7 deletes the first record

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

Delete (2)

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	3	July	F	11
3	4	Gail	M	11

Delete (1)

Index	SID	NAME	GENDER	AGE
1	3	July	F	11
2	4	Gail	M	11

Delete (1)

Index	SID	NAME	GENDER	AGE
1	4	Gail	M	11

10.2.3 Basic addition, deletion and modification

Modify record: modify the field value of the record with the specified row number

Index	SID	NAME
1	1	A1
2	2	A2

Modify 1:[11]

Index	SID	NAME
1	11	A1
2	3	A3

Modify the value of the first field in row 1

Index	SID	NAME
1	1	A1
2	2	A2

Modify 1:[A12:NAME]

Index	SID	NAME
1	1	A12
2	3	A3

Modify the value of NAME field in row 1

Index	SID	NAME
1	1	A1
2	2	A2

Modify 2:[30, A30]

Index	SID	NAME
1	1	A1
2	30	A30

Modify the values of fields 1 and 2 in row 2

10.2.3 Basic addition, deletion and modification

Modify record : **T.modify (k, x)** modifies the field values of row k

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	=A4.record([A1:D3])		/Fill data in the table
5	>A4.modify(2, 22)			/Modify the field value of row 2 and column 1
6	>A4.modify(3, "July33":NAME, 9:AGE)			/Modify the values of the specified fields on row 3
7	>A4.modify(1, "Jack11":NAME , "Male")			/Do modification according to the existing field order if the data has no field name

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11

A5 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	22	Tom	M	15
3	3	July	F	11

A6 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	22	Tom	M	15
3	3	July33	F	9

A7 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack11	Male	12
2	22	Tom	M	15
3	3	July33	F	9

10.2.3 Basic addition, deletion and modification

Modify record: Modify the field value of the specified record directly

SID	NAME	GENDER	AGE
1	A1	M	10

Modify (11)

SID	NAME	GENDER	AGE
11	A1	M	10

Modify the value of the first field

SID	NAME	GENDER	AGE
1	A1	M	10

Modify (A12:NAME)

SID	NAME	GENDER	AGE
1	A12	M	10

Modify the value of the specified NAME field

SID	NAME	GENDER	AGE
1	A1	M	10

Modify (A12:NAME, 15:AGE)

SID	NAME	GENDER	AGE
1	A12	M	15

Modify the values of specified fields

SID	NAME	GENDER	AGE
1	A1	M	10

Modify (30, A30, Male)

SID	NAME	GENDER	AGE
30	A30	Male	10

Do modification according to the field order if the data has no field name

SID	NAME	GENDER	AGE
1	A1	M	10

Modify (A12:NAME, Male, 15)

SID	NAME	GENDER	AGE
1	A12	Male	15

Do modification according to the order of existing field order if the data has no field name

10.2.3 Basic addition, deletion and modification

Modify record: **r.modify(x)** modifies the field value of record r

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])		/Fill data in the table
5	=A4(2)	>A5.modify(22)		/Modify the first field value of the record
6	>A5.modify("Tom22":NAME)			/Modify the value of specified field NAME of the record
7	>A5.modify(44, "Tom44", "Male")			/Do modification according to the field order if the data has no field name

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11

A5 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	22	Tom	M	15
3	3	July	F	11

A6 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	22	Tom22	M	15
3	3	July	F	11

A7 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	44	Tom44	Male	15
3	3	July	F	11

10.2.3 Basic addition, deletion and modification

Modify record: **r.F=a** or **r.#i=a** can also be used to modify records.

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])		/Table sequence
5	>A4(2).#1 = 22			/Modify the field value of row 2 and column 1
6	>A4(3).NAME="July33"			/Modify the value of NAME field in row 3
7	>A4(3).GENDER="Female"			/Modify the value of GENDER field in row 3

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11

A5 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	22	Tom	M	15
3	3	July	F	11

A6 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	22	Tom	M	15
3	3	July33	F	11

A7 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	22	Tom	M	15
3	3	July33	Female	11

10.2.4 Examples and extension operations

【Example 10.1】In addition to creating a table sequence directly with **create()**, you can also create a table with the same structure through the existing table by **T.create()** or record by **r.create()**.

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID,NAME,GENDER,AGE)	>A4.record([A1:D3])		/Fill in data
5	=A4.create()			/Create empty table sequence with the same structure as A4
6	=A4(1).create()			/Create empty table sequence with the same structure as A4(1)
7	=A6.record([A1:D3])			/Fill in data

A4	SID	NAME	GENDER	AGE
	10	Jack	M	12
	11	Tom	M	15
	12	July	F	11

A6	SID	NAME	GENDER	AGE
	10	Jack	M	12
	11	Tom	M	15
	12	July	F	11

A5	SID	NAME	GENDER	AGE

10.2.4 Examples and extension operations

【Example 10.2】Access field value through record, row and column number of a table or field name.

	A	B	C	D
1	1	Jack	M	2007-05-03
2	2	Tom	M	2004-07-12
3	3	July	F	2010-09-04
4	4	Gail	M	2010-03-08
5	=create(SID, NAME, GENDER, BIRTHDAY)	>A5.record([A1:D4])		/Initializing table sequence
6	=A5(2)			/Access record 2
7	=A6.SID	=A6.#1		/Get the value of the field SID in the record
8	=A6.NAME	=A6.#2		/Get the value of the field NAME in the record
9	=A5(2).NAME	=A5(2).#2		/Get the value of the field NAME in the second record
10	=A5(4).BIRTHDAY	=A5(4).#4		/Get the value of the field BIRTHDAY in the fourth record

10.2.4 Examples and extension operations

【Example 10.3】There is a null record phenomenon when adding records. If the specified row k in **T.insert(k, x)** is 2 or more than the number of records in the table, the record with null field values will be added.

	A	B
1	=create(SID, NAME, GENDER, AGE)	/Create empty table
2	>A1.insert(1, 1, "Tom", "M", 15)	/Add one record
3	>A1.insert(2, 2, "July", "F", 11)	/Add one record
4	>A1.insert(4, 3, "Gail", "M", 13)	/Add one record

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15

A3 insertion

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	11

A4 insertion

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	11
3	(NULL)	(NULL)	(NULL)	(NULL)
4	3	Gail	M	13

k = 4 in A4 is 2 more than the number of records in the table, and the third row of null record will appear when inserting.

10.2.4 Examples and extension operations

【Example 10.4】Insert records to table T by specified position in turn: **T.insert@r(k:[r])**

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])		/Table sequence
5	=A4.create()			/Create empty table A5 with the same structure as A4
6	=[A4(1), A4(3)]			/Get the records of row 1 and 3 in table A4
7	>A5.insert@r(2: A6)			/Starting from position 2, insert the records to table A5
8	>A5.insert@r(0: [A4(2)])			/Append the second record in A4 to A5

A4

SID	NAME	GENDER	AGE
1	Jack	M	12
2	Tom	M	15
3	July	F	11

A5

SID	NAME	GENDER	AGE
(null)	(null)	(null)	(null)
1	Jack	M	12
3	July	F	11
2	Tom	M	15

10.2.4 Examples and extension operations

【Example 10.5】Use T.insert@f(k:[r]) to insert record to table T by matching field names.

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID,NAME,GENDER,AGE)	>A4.record([A1:D3])		
5	=create(NAME,SID,AGE,GENDER)	=create(NAME, AGE)		/Create empty table A5 with the same field names as A4 but in different order
6	=A4(2)			/Read the second record
7	>A5.insert@f(1: [A6])	>B5.insert@f(1: [A6])		/Insert record to table sequence

A4	SID	NAME	GENDER	AGE
	1	Jack	M	12
	2	Tom	M	15
	3	July	F	11

A5	NAME	SID	AGE	GENDER
	Tom	2	15	M

B5	NAME	AGE
	Tom	15

When a record with the same field names as the table is inserted to the table, it will be written according to the corresponding field names, and the unmatched fields will be ignored.

10.2.4 Examples and extension operations

【Example 10.6】Use **r.modify(x)** to modify field values of record r.

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])	=A4(3)	/Access the third record
5	>C4.modify("July3":NAME, 9:AGE)			/Modify the specified field value of record
6	>C4.modify(33, "July33" , "Female")			/Do modification according to the field order if the data has no field name
7	>C4.modify("July34":NAME, "F", 11)			/Do modification according to the order of existing fields if the data has no field name

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11

A5 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July3	F	9

A6 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	33	July34	F	11

A7 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	33	July33	Female	9

10.2.4 Examples and extension operations

【Example 10.7】Use T(k).modify(x) to modify the field values of the kth record, which is equivalent to T.modify(k, x).

	A	B	C	D
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])		/Table sequence
5	>A4(3).modify("July3":NAME, 9:AGE)			/Modify the specified field values of the third record
6	>A4(3).modify(33, "July33" , "Female")			/Do modification according to the field order if the data has no field name
7	>A4(3).modify("July34":NAME, "F", 11)			/Do modification according to the order of existing fields if the data has no field name

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11

A5 modification →

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July3	F	9

A6 modification ↓

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	33	July34	F	11

← A7 modification

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	33	July33	Female	9

Chapter 10
Data table

10.3 External data

10.3.1 Data file

In the daily use of computers, we often encounter data file formats such as **CSV**, **TXT**, **XLS**.

txt format: often with tab as separator

UID	NAME	GENDER	BIRTHDATE
1	aaa	M	2012-11-12
2	bbb	M	2014-01-15
3	ddd	F	2014-07-05

First row fields

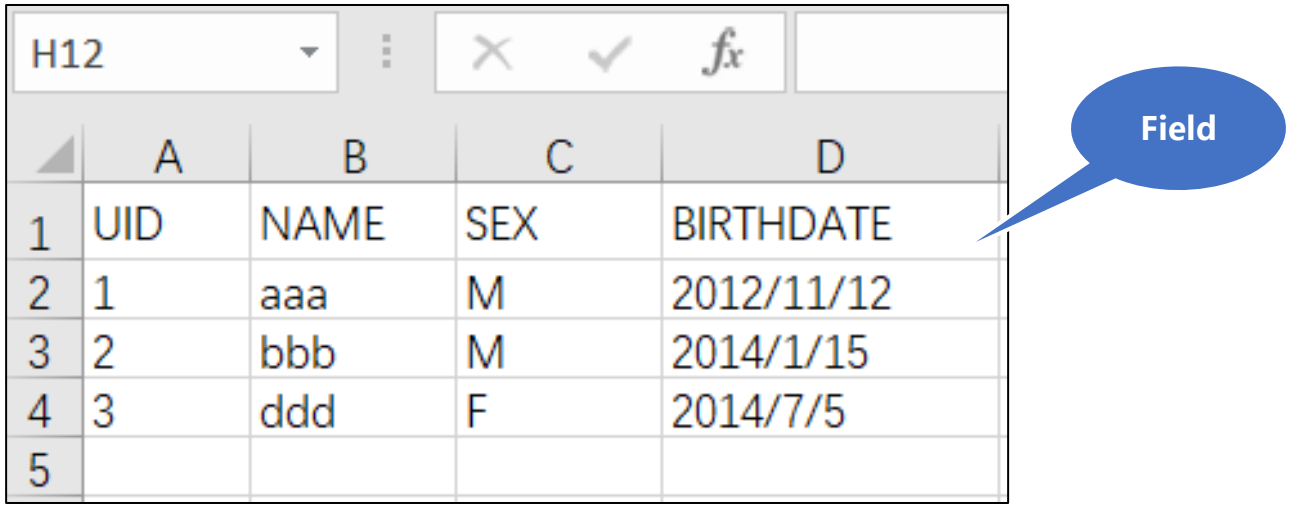
csv format : often with comma as separator

```
UID,NAME,GENDER,BIRTHDATE
1, aaa,M,2012-11-12
2, bbb,M,2014-01-15
3, ddd,F,2014-07-05
```

First row fields

10.3.1 Data file

xls, xlsx: the file format of Microsoft Excel



Data files of different file formats are important sources of table sequence.

10.3.2 Reading file data

When reading and writing file data, **f.import()** can be used for import and **f.export()** for export. The **@t** option of these two functions indicates that the first row of the file during import / export is field name. The data of emp0.txt with title in the first row is as follows:

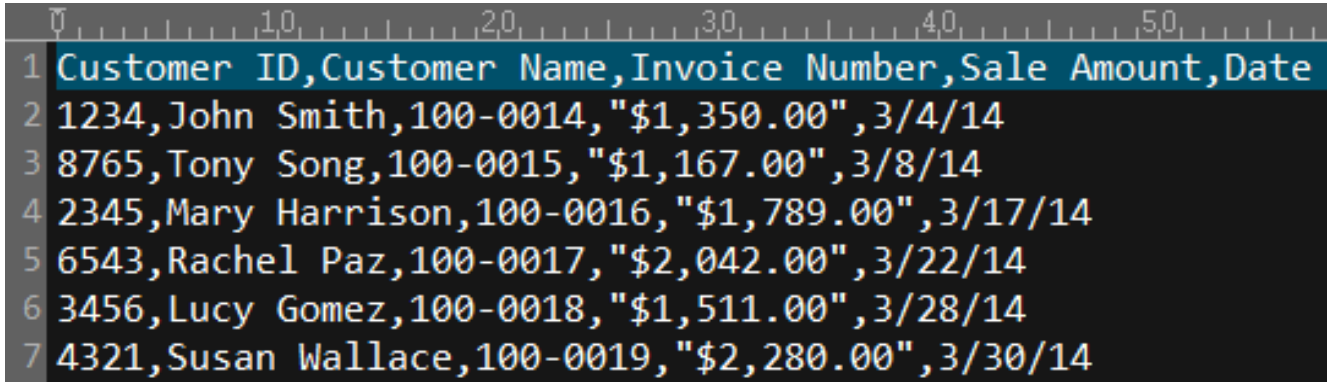
```
EID NAME  SURNAME GENDER  STATE BIRTHDAY  HIREDATE  DEPT  SALARY
201 Taylor  Brown F Georgia 1983-06-25  2004-03-01  2 5000
404 Madeline  Thomas  F Colorado  1977-01-08  2010-08-01  7 6500
203 Abigail Smith F Arizona 1972-02-20  2000-04-01  3 7000
205 Samantha  Moore F Washington  1977-10-10  2004-08-01  3 8000
402 Claire  Brown F California  1975-10-11  2010-07-01  7 7000
401 Olivia  Robinson  F Illinois  1971-04-03  2004-06-01  6 6500
210 Hannah  Johnson F Virginia  1984-07-09  2006-04-01  3 10000
398 Taylor  Smith F Pennsylvania  1986-12-13  2002-12-01  6 12000
397 Sarah Brown F Texas 1969-12-23  2005-12-01  3 6500
```

	A	B
1	=file("emp0.txt").import@t()	/Use the @t option to read the first row as the field name

Index	EID	NAME	SURNAME	GENDER	STATE	…	SALARY
1	201	Taylor	Brown	F	Georgia	…	5000
2	404	Madeline	Thomas	F	Colorado	…	6500
3	203	Abigail	Smith	F	Arizona	…	7000
…	…	…	…				

10.3.2 Reading file data

Access to the data of the CSV file. CSV file format often uses "," as the separator between data. The content of file sales.csv is as follows:



```
Customer ID,Customer Name,Invoice Number,Sale Amount,Date
1234,John Smith,100-0014,"$1,350.00",3/4/14
8765,Tony Song,100-0015,"$1,167.00",3/8/14
2345,Mary Harrison,100-0016,"$1,789.00",3/17/14
6543,Rachel Paz,100-0017,"$2,042.00",3/22/14
3456,Lucy Gomez,100-0018,"$1,511.00",3/28/14
4321,Susan Wallace,100-0019,"$2,280.00",3/30/14
```

	A	B
1	=file("sales.csv").import@tcq()	/Read csv file data

Index	Customer ID	Customer Name	Invoice Number	Sale Amount	Date
1	1234	John Smith	100-0014	$1, 350.00	3/4/14
2	8765	Tony Song	100-0015	$1, 167.00	3/8/14
3	2345	Mary Harrison	100-0016	$1, 789.00	3/17/14
…	…	…	…	…	…

The @c option in f.import@tcq() indicates using comma as separator, @q indicates that the field value is stripped if there are quotation marks.

10.3.2 Reading file data

f.xlsimport() can be used to import Excel files, f.xlsexport() to export Excel files.
Similar to f.import(), f.export(), the @t option can be used to indicate that the first row of the file during import / export is field name. The access to an Excel file is as follows:

	A	B
1	=file("book_order.xlsx").xlsimport@t()	/Read xlsx file data

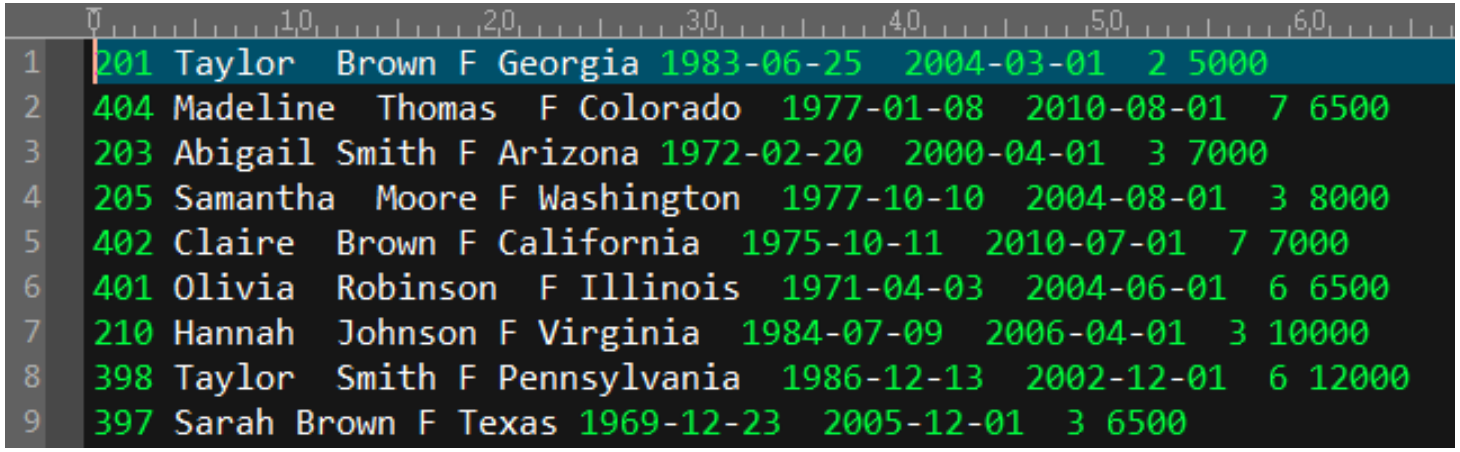
	A	B	C	D	E
1	PID	NAME	CODE	SIZE	DATE
2	9876	Daniel Farber	100-0008	1115	2013-02-02
3	8765	Laney Stone	100-0009	1367	2013-02-08
4	7654	Roger Lipney	100-0010	2135	2013-02-15
5	6543	Thomas Haines	100-0011		
6	5432	Anushka Vaz	100-0012		



Index	PID	NAME	CODE	SIZE	DATE
1	9876	Daniel Farber	100-0008	1115	2013-02-02
2	8765	Laney Stone	100-0009	1367	2013-02-08
3	7654	Roger Lipney	100-0010	2135	2013-02-15
4	6543	Thomas Haines	100-0011	1346	2013-02-17
5	5432	Anushka Vaz	100-0012	1560	2013-02-21

10.3.2 Reading file data

Read the untitled data and generate the header of the table sequence with _N to represent the field name, where n is a positive integer. The untitled data is as follows:



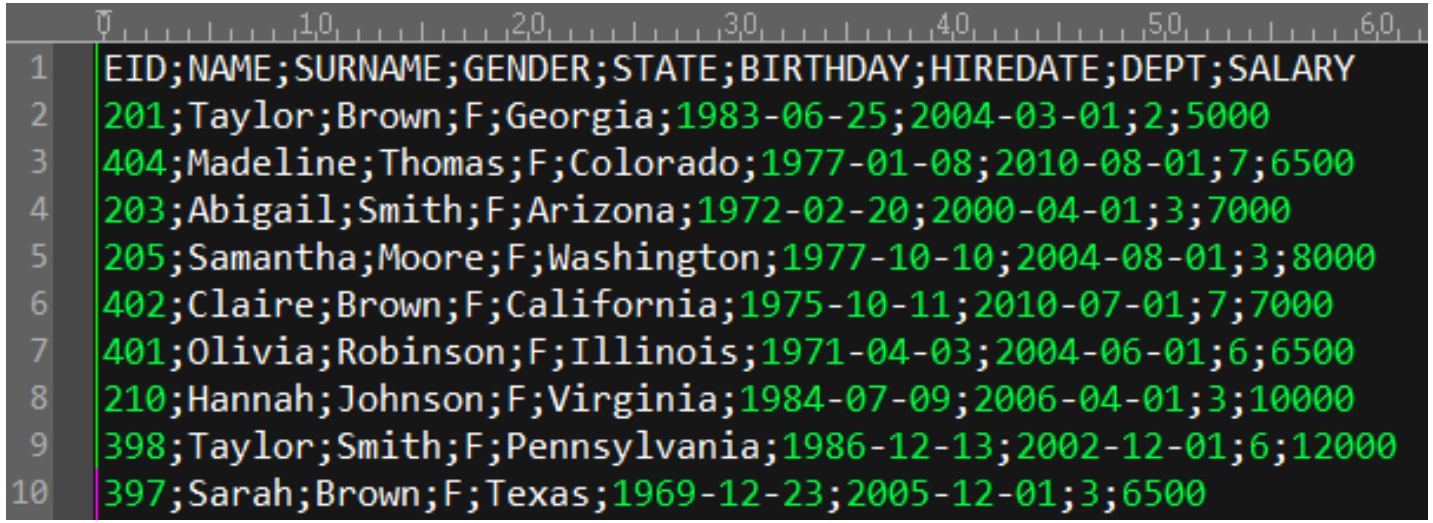
```
1  201 Taylor  Brown F Georgia 1983-06-25  2004-03-01  2 5000
2  404 Madeline  Thomas  F Colorado  1977-01-08  2010-08-01  7 6500
3  203 Abigail Smith F Arizona 1972-02-20  2000-04-01  3 7000
4  205 Samantha  Moore F Washington  1977-10-10  2004-08-01  3 8000
5  402 Claire  Brown F California  1975-10-11  2010-07-01  7 7000
6  401 Olivia  Robinson  F Illinois  1971-04-03  2004-06-01  6 6500
7  210 Hannah  Johnson F Virginia  1984-07-09  2006-04-01  3 10000
8  398 Taylor  Smith F Pennsylvania  1986-12-13  2002-12-01  6 12000
9  397 Sarah Brown F Texas 1969-12-23  2005-12-01  3 6500
```

	A	B
1	=file("emp0.txt").import()	/Read untitled text data

Index	_1	_2	_3	_4	_5	…	_9
1	201	Taylor	Brown	F	Georgia	…	5000
2	404	Madeline	Thomas	F	Colorado	…	6500
3	203	Abigail	Smith	F	Arizona	…	7000
4	205	Samantha	Moore	F	Washington	…	8000
…	…	…	…	…	…	…	…

10.3.2 Reading file data

Separator: specific separator for text data can be specified when the data is loaded. Using ";" as the separator between text data, the data of emp0b.txt is as follows:



```
EID;NAME;SURNAME;GENDER;STATE;BIRTHDAY;HIREDATE;DEPT;SALARY
201;Taylor;Brown;F;Georgia;1983-06-25;2004-03-01;2;5000
404;Madeline;Thomas;F;Colorado;1977-01-08;2010-08-01;7;6500
203;Abigail;Smith;F;Arizona;1972-02-20;2000-04-01;3;7000
205;Samantha;Moore;F;Washington;1977-10-10;2004-08-01;3;8000
402;Claire;Brown;F;California;1975-10-11;2010-07-01;7;7000
401;Olivia;Robinson;F;Illinois;1971-04-03;2004-06-01;6;6500
210;Hannah;Johnson;F;Virginia;1984-07-09;2006-04-01;3;10000
398;Taylor;Smith;F;Pennsylvania;1986-12-13;2002-12-01;6;12000
397;Sarah;Brown;F;Texas;1969-12-23;2005-12-01;3;6500
```

	A	B
1	=file("emp0b.txt") .import@t(;, ";")	/Read in data using the semicolon ';' as separator

The returned content is similar to the previous one and will not be listed again.

In **f.import@t(fn_1, fn_2, ...;k:n, s)**, **fn_1, fn_2, ...** represent the fields that need to be imported, and all fields are imported when omitted; **k:n** represents that dividing the data into n segments and only read the k^{th} segment, and all file is read in when omitted; **s** represents the separator, and semicolon is used here.

10.3.2 Reading file data

Part of the data: Sometimes you don't need to load all the data, you can read part of it by the fields.

	A	B
1	=file("emp.txt")	
2	=A1.import@t(EID, NAME, BIRTHDAY, SALARY)	=A1.import@t(#1, #2, #6, #9)

Index	EID	NAME	BIRTHDAY	SALARY
1	1	Rebecca	1974-11-20	7000
2	2	Ashley	1980-07-19	11000
3	3	Rachel	1970-12-17	9000
4	4	Emily	1985-03-07	7000
…	…	…	…	…

When reading data, you can use the field name directly or specify the field sequence number **#i**. The corresponding fields of #i here are EID, NAME , BIRTHDAY, SALARY, so the data returned in A2 and B2 are the same.

10.3.2 Reading file data

Special characters in the field name: if there are space, $, comma and other special characters in the middle of the field name, you need to add single quotation marks ' ' to the field name to read the data of the specified fields.

```
Customer ID,Customer Name,Invoice Number,Sale Amount,Date
1234,John Smith,100-0014,"$1,350.00",3/4/14
8765,Tony Song,100-0015,"$1,167.00",3/8/14
2345,Mary Harrison,100-0016,"$1,789.00",3/17/14
6543,Rachel Paz,100-0017,"$2,042.00",3/22/14
3456,Lucy Gomez,100-0018,"$1,511.00",3/28/14
4321,Susan Wallace,100-0019,"$2,280.00",3/30/14
```

	A	B
1	=file("sales.csv").import@tcq('Customer ID', 'Customer Name', 'Invoice Number', 'Sale Amount')	/Read data of specified fields

Index	Customer ID	Customer Name	Invoice Number	Sale Amount
1	1234	John Smith	100-0014	$1, 350.00
2	8765	Tony Song	100-0015	$1, 167.00
3	2345	Mary Harrison	100-0016	$1, 789.00
…	…	…	…	…

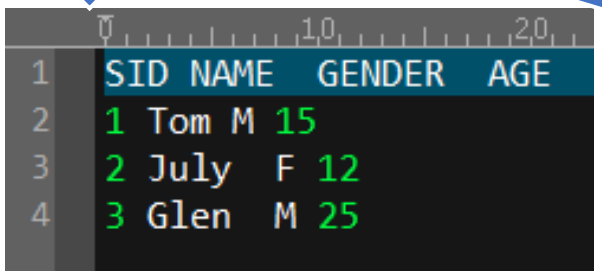
10.3.3 Exporting file data

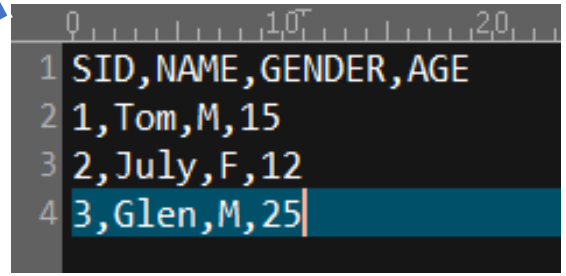
Write table sequence to txt and csv files, and save it to local file.

	A	B	C	D
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	25
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])	/Create table sequence, fill in data	
5	>file("txt/user_100.txt").export@t(A4)		/Save data in the table as user_100.txt	
6	>file("txt/user_200.csv").export@tc(A4)		/Save data in the table as user_200.csv	

Data in A4:

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	25





f.export@t() saves the exported data in the file, where @t means to write the field names as the first record to the file. By default, tab is used as the separator, and @c is used to indicate that the comma is used as the separator.

10.3.3 Exporting file data

Export the modified data to a file.

	A	B	C	D
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	25
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])	/Table sequence	
5	>A4(1).GENDER="Male"		/Modify the value of GENDER field in the first record	
6	>A4(2).GENDER="Female"		/Modify the value of GENDER field in the second record	
7	>A4(3).GENDER="Male"		/Modify the value of GENDER field in the third record	
8	>file("txt/user_101.txt").export@t(A4)		/Save the data in the table as user_101.txt	

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	

Modification

Index	SID	NAME	GENDER	AGE
1	1	Tom	Male	15
2	2	July	Female	12
3	3	Glen	Male	25

A8 export

```
SID NAME  GENDER  AGE
1 Tom Male  15
2 July  Female  12
3 Glen  Male  25
```

10.3.3 Exporting file data

Export the added data to a file.

	A	B	C	D
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	25
4	=create(SID, NAME, GENDER, AGE)	>A4.record([A1:D3])	/Table sequence	
5	>A4.insert(0, 4, "Tina", "F":GENDER, 10)		/Insert one record	
6	>A4.insert(0, 5, "Alisa", "F", 11)		/Insert one record	
7	>file("txt/user_102.txt").export@t(A4)		/Save the data in the table as user_102.txt	

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	
3	3	Glen	M	

A5, A6 insertion

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	25
4	4	Tina	F	10
5	5	Alisa	F	11

A7 export

```
SID NAME  GENDER  AGE
1 Tom M 15
2 July  F 12
3 Glen  M 25
4 Tina  F 10
5 Alisa F 11
```

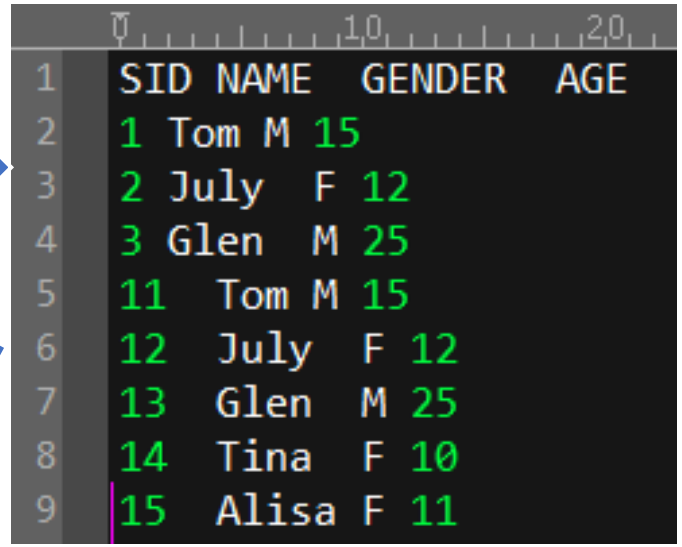
10.3.3 Exporting file data

Merge files: **f.export@a(T)** appends data with the same structure to an existing file.

	A	B
1	=file("txt/user_102.txt").import@t()	/Import data to table sequence
2	>file("txt/user_101.txt").export@ta(A1)	/Save the table data by appending to the existing file user_101.txt

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	25

Index	SID	NAME	GENDER	AGE
1	11	Tom	M	15
2	12	July	F	12
3	13	Glen	M	25
4	14	Tina	F	10
5	15	Alisa	F	11



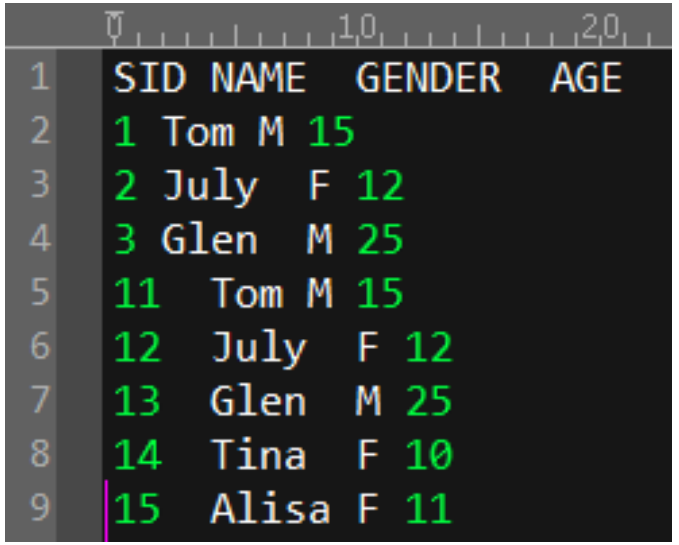
10.3.3 Exporting file data

Merge files and export: Merge data first and then export.

	A	B	C
1	=file("txt/user_100.txt").import@t()		
2	=file("txt/user_102.txt").import@t()		
3	for A2	>A1.insert@r(0:[A3])	/Merge the data from table A2 into table A1
4	>file("txt/user_103.txt").export@t(A1)		/Save the data in the table as user_103.txt

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	25

Index	SID	NAME	GENDER	AGE
1	11	Tom	M	15
2	12	July	F	12
3	13	Glen	M	25
4	14	Tina	F	10
5	15	Alisa	F	11



10.3.3 Exporting file data

Split file and export: split the data of the file according to conditions first, and then export to different files.

	A	B	C
1	=file("txt/user_103.txt").import@t()		
2	=A1.create()	=A1.create()	
3	for A1	if A3.SID<10	
4			=A2.insert@r(0:[A3])
5		else	
6			=B2.insert@r(0:[A3])
7	>file("txt/user_110.txt").export@t(A2)		
8	>file("txt/user_111.txt").export@t(B2)		

Index	SID	NAME	GENDER	AGE
1	1	Tom	M	15
2	2	July	F	12
3	3	Glen	M	25
4	11	Tom	M	15
5	12	July	F	12
6	13	Glen	M	25
7	14	Tina	F	10
8	15	Alisa	F	11

A7 export

```
SID NAME  GENDER  AGE
1 Tom M 15
2 July  F 12
3 Glen  M 25
```

A8 export

```
SID NAME  GENDER  AGE
11  Tom M 15
12  July  F 12
13  Glen  M 25
14  Tina  F 10
15  Alisa F 11
```

10.3.4 Database query

Relational database is the most important storage place of structured data. Many structured data we operate comes from and will be written back to relational database.

SQL(Structured Query Language) is a kind of language for managing data in relational database.

SQL can be used to extract data from relational database:

```
mysql> select * from emp where eid<10;
+------+----------+---------+--------+------------+------------+------------+-----------+--------+--------------+
| EID  | NAME     | SURNAME | GENDER | STATE      | BIRTHDAY   | HIREDATE   | DEPT      | SALARY | IDENTITYCARD |
+------+----------+---------+--------+------------+------------+------------+-----------+--------+--------------+
|    1 | Rebecca  | Moore   | F      | California | 1974-11-20 | 2005-03-11 | R&D       |   7000 | 1230001      |
|    2 | Ashley   | Wilson  | F      | New York   | 1980-07-19 | 2008-03-16 | Finance   |  11000 | 1230002      |
|    3 | Rachel   | Johnson | F      | New Mexico | 1970-12-17 | 2010-12-01 | Sales     |   9000 | 1230003      |
|    4 | Emily    | Smith   | F      | Texas      | 1985-03-07 | 2006-08-15 | HR        |   7000 | 1230004      |
|    5 | Ashley   | Smith   | F      | Texas      | 1975-05-13 | 2004-07-30 | R&D       |  16000 | 1230005      |
|    6 | Matthew  | Johnson | M      | California | 1984-07-07 | 2005-07-07 | Sales     |  11000 | 1230006      |
```

The knowledge of relational database is relatively professional, and there are a lot of reference books, so this document will not repeat it.

Chapter 10
Data table

10.4 Table sequence operations

Table sequence operations

- A table sequence is also a sequence. You can use the sequence functions described in Chapter 8, such as using for T to loop through each record in the table sequence.
- Similar to sequence, when looping through a table sequence, we can calculate aggregate results for all members, such as count, sum, average, Max/Min etc. Only that for a sequence, we calculate the aggregate result for all members, but for a table sequence, we often calculate the aggregate value of a certain field for all records.
- When looping through a table sequence, you may also need to filter the records to select the qualified records to add to a new table sequence with the same structure.

10.4.1 Aggregation

Count: get the number of users. Accumulate the number of users in the table by loop.

	A	B	C	D
1	=file("txt/user_1.txt").import@t()	0		/B1 counts all users
2	>iM=0			/iM counts the male members
3	for A1	>B1=B1+1		/Accumulate the number of users
4		if A3.GENDER=="M"		/Judge whether the member is male or not
5			>iM=iM+1	/Accumulate the number of male users
6	>output("count=" / B1/"; iM=" / iM)			/Output the number of users and male users

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

Output: count = 4; iM=3

A3 executes the for loop, looping through each user. During the loop, B3 accumulates the number of users and C5 accumulates the number of male users until the end of the loop.

In fact, the table sequence is also a **sequence composed of records**. Therefore, the **for T** loop of a table sequence is essentially the same as the for A loop of a sequence introduced in Chapter 8. Only that in the table sequence, each member is a record. For example, in the code above, when the loop is executed, the values in the loop main cell A3 are records in turn. In the loop calculation of the table sequence, it is usually necessary to calculate the values of a certain field in the records.

10.4.1 Aggregation

Sum: get the sum of male user age. In the loop, calculate the age of each male user, and then accumulate the age.

	A	B	C	D
1	=file("txt/user2.txt").import@t()	0		/B1accumulates the ages
2	for A1	if A2.GENDER=="M"		/Judge whether the member is male or not
3			=age(A2.BIRTHDAY)	/Calculate the age of the user
4			>B1=B1+C3	/Accumulate age
5	>output("sum=" / B1)			/Output the total age

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	M	2007-05-03
2	2	Tom	M	2004-07-12
3	3	July	F	2010-09-04
4	4	Gail	M	2010-03-08

Output: sum=39

In the loop, the age is calculated according to the user's birth date, and then the age is accumulated.

10.4.1 Aggregation

Average: The following is to get the average age of users. Get the sum of ages of all users and the number of users, and then divide the cumulative age by the number of users to calculate the average age.

	A	B	C
1	0	0	/B1accumulates the ages
2	=file("txt/user2.txt").import@t()		/Read the text file data
3	for A2	=age(A3.BIRTHDAY)	/Calculate age
4		>B1=B1+B3	/Accumulate age
5		>A1=A1+1	/Accumulate number of users
6	=round(B1/A1, 2)		/Calculate average age
7	>output("avg="/A6)		/Output average age

Output: avg=12.0

for T loops through the records of each user, first calculate the age according to the user's birth date, and then accumulate the user's age and the number of users. After the loop ends, divide the cumulative age by the number of users to calculate the average age, use round() function to round the result and keep two decimal places.

10.4.1 Aggregation

Max/Min: Calculate the maximum and minimum ages of users.

	A	B	C	D
1	>iMax=0	>iMin=99		/Initializing the maximum and minimum age parameters
2	=file("txt/user_2.txt").import@t()			/Read the text file data
3	for A2	=age(A3.BIRTHDAY)		/Calculate the age of the user
4		if B3>iMax	>iMax=B3	/Get the larger age by comparison
5		if B3<iMin	>iMin=B3	/Get the smaller age by comparison
6	>output("Min="/iMin /"; Max="/iMax)			/Output the max/min ages

Output: Min=9; Max=16

In the for T loop, first calculate the age according to the user's birth date, then get the larger age and smaller age by comparison and store them in the variables iMax and iMin respectively. After the loop ends, the maximum and minimum ages of the user are obtained.

10.4.1 Aggregation

Top N: Calculate the Top 5 largest age values.

Sequence B1 is used to store the Top 5 largest age values, and all initialization values are 0.
When looping through the user records, the outer loop calculates the age according to the date of birth. The inner loop uses it to compare with the members in sequence B1. If the age is older than the one in B1, the age is inserted at this position, the last value is deleted, and break the inner loop; This is repeated until the end of the loop, and the Top 5 largest age values are calculated.

	A	B	C	D	E
1	=file("txt/emp.txt").import@t()	[0, 0, 0, 0, 0]			/Initializing the result sequence
2	for A1	=age(A2.BIRTHDAY)			/Calculate age
3		for 5	if B2>B1(B3)	>B1.insert(B3, B2)	/Insert the larger value
4				>B1.delete(6)	/Delete the last value
5				break	

10.4.2 Filter

Save records that meet the condition to a table sequence. For records with the same structure, **T.insert@r(0:[r])** can be used.

	A	B	C
1	=file("txt/employee.txt").import@t()	=A1.create()	/Generate a table sequence with the same structure
2	for A1	if A2.BIRTHDAY>date("1987-08-01")	/Find records with a birth date greater than 1987-08-01
3			>B1.insert@r(0:[A2])

C3: Insert eligible records into B1

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	…
1	1	Rebecca	Moore	F	California	1974-11-20	…
2	2	Ashley	Wilson	F	New York	1980-07-19	…
3	3	Rachel	Johnson	F			
4	4	Emily	Smith	F			
5	5	Ashley	Smith	F			
…	…	…	…				

B1 table sequence

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	…
1	89	Emily	Johnson	F	Michigan	1987-12-09	…
2	241	Samantha	Davis	F	Georgia	1987-12-04	…
3	393	Hannah	Johnson	F	Florida	1987-09-06	…
4	437	Zachary	Williams	M	Tennessee	1987-12-01	…
5	472	Hannah	Brown	F	California	1987-12-27	…

10.4.2 Filter

Save records that meet the conditions to a table sequence. For records with the same fields as the table sequence, **T.insert@f(0:[r])** can be used to write the values corresponding to the matched fields to the table sequence:

	A	B	C
1	=file("txt/employee.txt").import@t()	=create(EID, NAME, STATE, BIRTHDAY)	/B1: the table sequence to store records after filtering
2	for A1	if A2.BIRTHDAY>date("1987-01-01") && A2.STATE=="California"	/Find records a birth date greater than 1987-01-01 and the state as California
3			>B1.insert@f(0:[A2])

C3: Insert the records that meet both the given conditions into table B1

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	…
1	1	Rebecca	Moore	F	California	1974-11-20	…
2	2	Ashley	Wilson	F	New York	1980-07-19	
3	3	Rachel	Johnson	F	New Mexico		
4	4	Emily	Smith	F	Texas		
5	5	Ashley	Smith	F	Texas		
…	…	…	…				

B1 table sequence

Index	EID	NAME	STATE	BIRTHDAY
1	119	Madison	North Caro…	1987-05-10
2	123	Ashley	Indiana	1987-02-09
3	151	Samantha	California	1987-02-06
…	…	…	…	…

10.4.2 Filter

Use expression to filter. Find the records whose NAME field starts with "Ca" :

	A	B	C
1	=file("txt/employee.txt").import@t()	=A1.create()	/Generate a new table sequence with the same structure
2	for A1	if left(A2.NAME, 2)=="Ca"	/Find records whose NAME field starts with "Ca"
3			>B1.insert@r(0: [A2])

B2: The function left(A2.NAME, 2) intercepts the substring of the first two characters of the user name. If the substring is equal to "Ca", the record is appended to the sequence B1.

Index	EID	NAME	SURNAME	GENDER	STATE	…	SALARY
1	1	Rebecca	Moore	F	California	…	7000
2	2	Ashley	Wilson	F	New York	…	11000
3	3	Rachel	Johnson	F	New Mexico	…	9000
4	4	Emily	Smith	F	Texas	…	7000
…	…	…	…				

B1 data:

Index	EID	NAME	SURNAME	GENDER	STATE	…	SALARY
1	67	Cameron	Jones	M	New York		5000
2	220	Caleb	Smith	M	New York		7000
3	289	Cameron	Johnson	M	California		7000

Chapter 10
Data table

10.5 Record sequence

10.5.1 The concept of record sequence

In the last example of the previous section: find the records whose NAME field starts with "Ca". We can write the code in another way:

	A	B	C
1	=file("txt/employee.txt").import@t()	[]	/Define empty sequence B1
2	for A1	if left(A2.NAME, 2)=="Ca"	/Find records whose NAME field starts with "Ca"
3			>B1=B1|A2

B2: Obtain the substring of the first two characters of the user name. If the substring is equal to "Ca", the record is appended to the sequence B1.

Index	EID	NAME	SURNAME	GENDER	STATE	…	SALARY
1	67	Cameron	Jones	M	New York		5000
2	220	Caleb	Smith	M	New York		7000
3	289	Cameron	Johnson	M	California		7000

In B1, we use empty sequence instead of A1.create() to create an empty table sequence. In C3, we add members to the sequence instead of using the insertion function of the table sequence. The result is actually a sequence composed of records, rather than the new table sequence in the previous section. However, when viewing the values of B1, the display effect is similar to that of the table sequence.

10.5.1 The concept of record sequence

As you can see from this example, the sequence composed of multiple records looks like a table sequence.

The table sequence (**TSeq**) itself is also a sequence of records, and these records can be used to form a new sequence, which we call **record sequence** (**RSeq**). In other words, **record sequence** is **the sequence composed of records**, and the sequence composed of records of the same table sequence is called **pure record sequence**.

The table sequence has data structure, while **ordinary record sequence has no data structure**. However, the record sequence with all records of the **same data structure** (such as pure record sequence) can be displayed by using the **data structure of records**. A table sequence processes data through a data structure, and the results that are not processed by a data structure are no longer table sequence. For example, the concatenation result of the previous page is a record sequence rather than a table sequence.

In fact, the table sequence is also a sequence composed of all records, so in the operations, the table sequence can also be regarded as a **special record sequence**.

10.5.1 The concept of record sequence

Record sequence is the sequence composed of records.

The following shows the composition of a record sequence, whose data members are records from a table sequence:

Table sequence A data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

1:[2, Tom, M, 15]
2:[1, Jack, M, 12]
3:[4, Gail, M, 11]

Record sequence data

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	1	Jack	M	12
3	4	Gail	M	11

Extract records A([2, 1, 4]) from a table sequence to form a record sequence.

10.5.1 The concept of record sequence

Records are directly taken from the table sequence to form a record sequence.

	A	B
1	=file("txt/user_1.txt").import@t()	/Import data, return a table sequence
2	=A1([2, 1, 4])	/The second, the first and the fourth records in the table sequence are used to form the record sequence

Table sequence A1 data:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

Record sequence A2 data:

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	1	Jack	M	12
3	4	Gail	M	11

10.5.1 The concept of record sequence

Influence of data modification: copy the data of the table sequence to the new table sequence and record sequence, and then modify the records of the source table sequence to observe the impact on the new table sequence and record sequence. The operation is as follows:

	A	B	C
1	=file("txt/user_4.txt").import@t()		/Import data
2	=A1.create()	[]	/Generate new table sequence A2, define empty sequence B2
3	for A1	>A2.insert@r(0:[A3])	/Insert record to table sequence A2
4		>B2=B2|A3	/Concatenate records to record sequence B2
5	=A1(2).NAME="July33"		/Modify the NAME value of the second record of the source table sequence

Table sequence A2 data:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	3	July	F	11



Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	3	July	F	11

Record sequence B2 data:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	3	July	F	11



Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	3	July33	F	11

The new table sequence in A2 only copies the structure of A1. When A.insert() is used, a **new record** is generated and has nothing to do with the record in A1. Therefore, modification of A1 will not affect A2. The NAME value of the second record in record sequence B2 becomes July33, because the record sequence does not store the entity record, but the **record object**. Therefore, the modification of the record in A1 will change the record in record sequence B2.

10.5.1 The concept of record sequence

When a record is modified, all variables in memory that refer to the same record will also change. This is the **reference property** of the record sequence.

	A	B
1	=file("txt/user_1.txt").import@t()	/Table sequence
2	=[A1(4), A1(1), A1(3), A1(2), A1(1)]	/Construct a record sequence with repeated records
3	>A1.modify(1, 10, "JACKER")	/Modify the first record of the table sequence
4	=A2(2)==A2(5)	/true is returned

A1	SID	NAME	GENDER	AGE
	1	Jack	M	12
	2	Tom	M	15
	3	July	F	11
	4	Gail	M	13

A3 modification →

A1	SID	NAME	GENDER	AGE
	10	JACKER	M	12
	2	Tom	M	15
	3	July	F	11
	4	Gail	M	13

A2	SID	NAME	GENDER	AGE
	4	Gail	M	13
	1	Jack	M	12
	3	July	F	11
	2	Tom	M	15
	1	Jack	M	12

A3 modification →

A2	SID	NAME	GENDER	AGE
	4	Gail	M	13
	10	JACKER	M	12
	3	July	F	11
	2	Tom	M	15
	10	JACKER	M	12

10.5.1 The concept of record sequence

Records of a record sequence can also come from multiple table sequences.

Table sequence A data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

Table sequence B data

Index	SID	NAME	GENDER	AGE
1	13	Glen	M	25
2	14	Tina	F	10
3	15	Alisa	F	11

Record sequence data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	13	Glen	M	25
5	14	Tina	F	10
6	15	Alisa	F	11

10.5.1 The concept of record sequence

Even from multiple table sequences of different structures :

Table sequence A data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

Table sequence B data

Index	SID	NAME	GENDER	BIRTHDAY
1	13	Glen	M	1985-02-06
2	14	Tina	F	2010-04-08
3	15	Alisa	F	2005-06-03

Record sequence data

Index	SID	NAME	GENDER	AGE	BIRTHDAY
1	1	Jack	M	12	(null)
2	2	Tom	M	15	(null)
3	3	July	F	11	(null)
4	13	Glen	M	(null)	1985-02-06
5	14	Tina	F	(null)	2010-04-08
6	15	Alisa	F	(null)	2005-06-03

This is just to **demonstrate** the data in the record sequence. If the data structure of the records in the record sequence is different, actually it **can not** be displayed by the data structure at the same time as the table sequence.

10.5.1 The concept of record sequence

Data member repeatability, the same record can **repeatedly** appear in the record sequence:

Table sequence data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

1:[1, Jack, M, 12]
2:[2, Tom, M, 15]
3:[3, July, F, 11]
4:[2, Tom, M, 15]
5:[2, Tom, M, 15]
6:[1, Jack, M, 12]
7:[1, Jack, M, 12]

Record sequence data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	2	Tom	M	15
5	2	Tom	M	15
6	1	Jack	M	12
7	1	Jack	M	12

10.5.1 The concept of record sequence

Record sequence data can come from different table sequences, not only refers to the records directly from the table sequence, but also refers to the records indirectly from the table sequence, that is, the records from the record sequence.

	A	B	C
1	=file("txt/user_1.txt").import@t()		/Table sequence
2	=file("txt/user_3.txt").import@t()		/Table sequence
3	=[A1(4), A1(1), A1(3), A2(1)]	=[A1(1), A1(3), A2(2), A2(2), A3(4)]	/Construct different record sequences

Table sequence A1

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	13

Table sequence A2

Index	SID	NAME	GENDER	AGE
1	20	Liky	F	9
2	21	Lucy	F	7
3	22	Mark	M	13

Record sequence A3

Index	SID	NAME	GENDER	AGE
1	4	Gail	M	13
2	1	Jack	M	12
3	3	July	F	11
4	20	Liky	F	9

Record sequence B3

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	3	July	F	11
3	21	Lucy	F	7
4	21	Lucy	F	7
5	20	Liky	F	9

10.5.2 Record sequence operations

Generation of a record sequence: concatenate records to a defined empty sequence or construct from records.

	A	B
1	=file("txt/user_1.txt").import@t()	/Table sequence
2	[]	/Define an empty sequence
3	>A2=A2|A1(2)	/Concatenate one record
4	>A2=A2|A1([1, 3])	/Concatenate multi records
5	=[A2(2), A2(1), A2(3), A2(3)]	/Construct new record sequence from records of the record sequence
6	=A2([2, 1, 3, 3])	/Obtain records by sequence number sequence from a record sequence to construct a new record sequence
7	=A1.m([2, 3, 4])	/A.m(), obtain records by sequence number from a table sequence to construct a record sequence

A2 record sequence:

Index	Member

A3 concatenation

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15

A4 concatenation

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	1	Jack	M	12
3	3	July	F	11

The record sequence of A5 and A6:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	3	July	F	11

The record sequence of A7:

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	3	July	F	11
3	4	Gail	M	11

10.5.2 Record sequence operations

Table sequence and record sequence are sequences composed of records. For the **generation of record sequence(RSeq),** it is sometimes more convenient to use the function **A.to().**

	A	B
1	=file("txt/user_1.txt").import@t()	/Import data
2	=A1.to()	/The record sequence composed of all records in table sequence A1
3	=A1.to(2)	/Get the first two records to construct a record sequence
4	=A1.to(2, 4)	/A record sequence composed of records from row 2 to row 4 of the table
5	=A1.to(, 3)	/A record sequence composed of records from row 1 to row 3 of the table

A2:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

A3:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15

A4:

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	3	July	F	11
3	4	Gail	M	11

A5:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11

10.5.2 Record sequence operations

As a **sequence**, record sequence(RSeq) can be accessed, updated and deleted just like sequence.

	A	B	C
1	=file("txt/user_1.txt").import@t()	0	/Table sequence
2	=[A1(1), A1(4)]		/Construct a record sequence with records
3	=A1.to()		/Construct a record sequence with A.to()
4	=A3(1)		/Access the first record
5	=A3.m(1)		/Access the first record
6	=A3.m(-1)		/Access the last record
7	>A2(2)=A4		/Record update
8	>A2.insert(0, A4)		/Insert record
9	>A2.delete(1)		/Delete record
10	=A2|A3		/Concatenation
11	=A2&A3		/Union
12	=A2^A3		/Intersection
13	for A3	>B1=B1+1	/Traversal access

10.5.2 Record sequence operations

The record sequence itself has no data structure, but its members are records. Therefore, the field values of records can also be accessed by **r.F** or **r.#i**, which is similar to the access of table sequence records.

	A	B	C
1	=file("txt/user_1.txt").import@t()		
2	=A1.to()		/Generate a record sequence
3	=A2(1)		/Access the first record of the record sequence
4	=A3.NAME	=A3.#2	/Access the value of NAME
5	=A2(1).NAME	=A2(1).#2	/Access the value of NAME
6	>A2(1).NAME="Jacker"	>A2(1).#2="Jacker"	/Modify the value of NAME
7	>A2(1).GENDER="Male"	>A2(1).#3="Male"	/Modify the value of GENDER

10.5.2 Record sequence operations

The **loop operation** of a record sequence is similar to that of a table sequence. For example: count.

	A	B	C	D
1	=file("txt/user_1.txt").import@t()	0		/B1 counts all users
2	=A1.to()	>iM=0		/iM counts male users
3	for A2	>B1=B1+1		/Accumulate number of users
4		if A3.GENDER=="M"		/Judge whether it is male or not
5			>iM=iM+1	/Accumulate number of male users
6	>output("count=" / B1/"; iM=" / iM)			/Output number of users and number of male users

Sum: get the sum of ages of all male users.

	A	B	C	D
1	=file("txt/user2.txt").import@t()	=A1.to()	0	/C1 accumulates the sum of ages
2	for B1	if A2.GENDER=="M"		/Judge whether it is male or not
3			=age(A2.BIRTHDAY)	/Calculate age
4			>C1=C1+C3	/Accumulate age
5	=output("sum=" / C1)			/Output the total age

Record sequence is also sequence composed of records, and the loop of record sequence is **the same** as that of table sequence.

10.5.2 Record sequence operations

Average: The average age of users is calculated below.

	A	B	C
1	0	0	/B1 accumulates age
2	=file("txt/user2.txt").import@t()	=A2.to()	/Generate a record sequence through table sequence
3	for B2	=age(A3.BIRTHDAY)	/Calculate age
4		>B1=B1+B3	/Accumulate age
5		>A1=A1+1	/Accumulate number of users
6	=round(B1/A1, 2)		/Calculate average age
7	=output("avg="/A6)		/Output average age

Max/Min: Calculate the Max/Min age of users.

	A	B	C	D
1	>iMax=0	>iMin=99		/Initializing the maximum and minimum age parameters
2	=file("txt/user_2.txt").import@t()	=A2.to()		/Generate a record sequence through table sequence
3	for B2	=age(A3.BIRTHDAY)		/Calculate age
4		if B3>iMax	>iMax=B3	/Get the larger age by comparison
5		if B3<iMin	>iMin=B3	/Get the smaller age by comparison
6	=output("Min="/iMin /"; Max="/iMax)			/Output the Max and Min age

10.5.2 Record sequence operations

Conditional filtering: the records that meet the conditions are stored in the record sequence.

	A	B	C
1	=file("txt/employee.txt").import@t()	[]	/B1 is an empty sequence for storing filtered records
2	for A1	if A2.BIRTHDAY>date("1987-01-01") && A2.STATE=="California"	/Find records with a birth date greater than January 1, 1987 and a state name of California
3			>B1=B1|A2

Use expression to filter: find records with user names beginning with "Ca":

	A	B	C
1	=file("txt/emp.txt").import@t()	[]	/Define an empty sequence
2	for A1	if left(A2.NAME, 2)=="Ca"	/Find records with user names starting with "Ca"
3			>B1=B1|A2

Store the filtered records to a record sequence. The following uses like() to find records with user names ending in "son":

	A	B	C
1	=file("txt/emp.txt").import@t()	[]	/Define an empty sequence
2	for A1	if like(A2.NAME, "*son")	/Find records with user names ending in "son"
3			>B1=B1|A2

10.5.2 Record sequence operations

TopN, calculate the Top5 largest age value.

When the record sequence is initialized, the record within the first 5 records with the largest birth date, that is, the youngest, is taken as the member of B1 record sequence.
when looping through user records, insert the records with smaller birth dates into record sequence B1 in ascending order, and delete the last record; repeat this process until the end of the loop, and the top five largest age values are calculated.

	A	B	C	D	E
1	=file("txt/emp.txt").import @t()	=A1([4, 4, 4, 4, 4])			/Construct record sequence with user records
2	for A1	for 5	if A2.BIRTHDAY<B1(B2).BIRTHDAY	>B1.insert(B2, A2)	/Insert the record with smaller birth date
3				>B1.delete(6)	/Delete the last record
4				break	

Through aggregate operations and conditional filtering, we can see that record sequence is similar to table sequence in usage.

10.5.2 Record sequence operations

Record sequence can perform set operation. The data of the following two record sequences are from the same table sequence, of which [2, 3] records are common, and then the operations of concatenation, intersection, union and difference are performed.

	A	B	C
1	=file("txt/user_1.txt").import@t()		/Import data
2	=A1([1, 2, 3])	=A1([2, 3, 4])	/Generate record sequence
3	=A2|B2		/Concatenation of record sequences, return all records in the set
4	=A2^B2		/Intersection of record sequences, return common records
5	=A2&B2		/Union of record sequences, return deduplicated records
6	=A2\B2		/Difference between record sequences, records in A2 but not in B2

A2|B2:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	2	Tom	M	15
5	3	July	F	11
6	4	Gail	M	11

A2\B2:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12

A2^B2:

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	3	July	F	11

A2&B2:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

10.5.2 Record sequence operations

The record sequences from different table sequences are used for intersection operation. The next two record sequences come from two table sequences with the same data.

	A	B	C
1	=file("txt/user_1.txt").import@t()	=A1.create()	/Create new table sequence B1
2	>B1.insert@r(0:A1)		/Copy data to new table sequence B1
3	=A1([1, 2, 3])	=B1([1, 2, 3])	/Generate two record sequences
4	=A3^B3		/Intersection of record sequences, return null

A3^B3 data:

Index	Member

A4: The intersection of two record sequences returns null.

The members of different table sequences are always different, even if their fields and values are the same, so the intersection operation of record sequences from different table sequences always returns null. Only the record sequences from the same table sequence performing set operation is meaningful.

10.5.2 Record sequence operations

The record sequence intersection "^" is used to find the records that meet the filtering conditions. The following is to query the data of sales over 20000 in 2015.

	A	B	C	D
1	=file("txt/sales.txt").import@t()	[]	[]	/Import data
2	for A1	if A2.AMOUNT>20000	>B1=B1|A2	/Store the data with sales over 20000 to record sequence B1
3		if A2.ORDERDATE>=date("2015-01-01") && A2.ORDERDATE<=date("2015-12-31")	>C1=C1|A2	/Store the data in 2015 to record sequence C1
4	=B1^C1			/Calculate common records

A4 data:

Index	ORDERID	CLIENT	SELLERID	AMOUNT	ORDERDATE
1	790	HANAR	13	28200.0	2015-01-04 15:28:05
2	800	EGU	20	29400.0	2015-01-10 15:28:05
3	802	QUICK	8	28600.0	2015-01-13 15:28:05
4	806	MIP	12	20900.0	2015-01-14 15:28:05
…	…	…	…		

Through the loop, the sales orders over 20000 and the sales orders in 2015 are respectively stored in record sequence B1 and C1. After the loop ends, the intersection of B1 and C1 is calculated, which is the required result.

10.5.2 Record sequence operations

if you want to compare whether the field values of the records in different table sequences are the same, how to do it?

To find the records with duplicate field values in table A and table B, we can compare the record values with **cmp(r_1, r_2)** to find the same records. Take a record from A and compare it with the records in B in turn, if it is equal, the record will be stored in the record sequence; repeat this process to find out the records with the same field values.

	A	B	C	D	E
1	=file("txt/sales.txt") .import@t()	=A1.create()	=A1.create()		/Import data
2	for A1	if A2.AMOUNT>20000	>B1.insert@r(0:[A2])		/Store the data with sales over 20000 to table sequence B1
3		if A2.ORDERDATE >= date("2015-01-01") && A2.ORDERDATE <= date("2015-12-31")	>C1.insert@r(0:[A2])		/Store the data in 2015 to table sequence C1
4	for B1	for C1	if cmp(A4, B4)==0	>D1=D1|A4	break

The calculation results are identical with the previous record sequence intersection operation.

cmp(r_1, r_2): when two records r_1 and r_2 are used to compare, if all field values of the two records are exactly the same, return 0.

10.5.3 The difference between rseq and tseq

The table sequence has a data structure, but **the record sequence itself has no data structure**, and some operations of the table sequence are not supported:

	A	B
1	=file("txt/user_1.txt").import@t()	/Import text data
2	=A1.to()	/Generate record sequence
3	>B1.insert(2:A1(3))	/Insert record
4	/=B1.insert(1, 11:SID)	/not supported
5	>A1.modify(2, "Tom22":NAME)	/Modify the NAME field of row2 of table sequence to Tom22
6	>A2(1).NAME="Jacker"	/Modify the NAME field of row1 of record sequence to Jacker
7	>A1.delete(1)	/Deleting the first record in the table sequence does not affect the record sequence data
8	>A2.delete(3)	/Delete the third record in the record sequence

1. When adding a record in a record sequence, A.insert(k, x:F), A.insert(k:A, x:F), A.insert@r(k:A), A.insert@f(k:A) **are not supported**.
2. When modifying data in a record sequence, A.modify(k, x:F), A.modify(k:A, x :F), A.modify@r, A.modify@f **are not supported**.
3. Deleting records in the table sequence **does not affect** the data in the record sequence.
4. Record sequence **can not** use A.record() to fill in data.

10.5.3 The difference between rseq and tseq

Since record sequence is also a kind of sequence, non record data can be added to record sequence. If it is still regarded as record sequence to operate, there may be errors.

	A	B	C
1	=file("txt/user_1.txt").import@t()		/Table sequence
2	=A1.to()		/generate a record sequence
3	>A2.insert(1, 11)	=A2(1)	/Insert non record data
4	/=B3.NAME	/=B3.#2	/Access NAME value
5	=A2(2).NAME	=A2(2).#2	/Access NAME value

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	1	Jack	M	12
3	3	July	F	11

A3 inserts data

Index	Member
1	11
2	[2, Tom, M, …]
3	[1, Jack, M, …]
4	[3, July, F…]

B3 has no field information and cannot be accessed as a record sequence. For record sequences, try to avoid operations like this.

10.5.3 The difference between rseq and tseq

Record sequence can be made up of records from table sequences of different structures, but only limited calculations can be done.

The following calculates the salaries of personnel from two departments A and B. The table structure provided by department B has an extra "allowance" field.

Since the salary field is available in both table sequences, you can first concatenate the data of the table sequences into a record sequence, and then traverse the record sequence records to accumulate salaries

	A	B	C
1	=file("txt/emp_c1.txt").import@t()		/Load emp_c1 data
2	=file("txt/emp_c2.txt").import@t()		/Load emp_c2 data
3	=A1|A2	0	/Concatenate data to get a record sequence
4	for A3	>B3=B3+A4.SALARY	/Accumulate salary
5	>output("Total=" / B3)		/Output total salary

Output: Total=23200

10.5.3 The difference between rseq and tseq

Compare the table sequence and record sequence for **intersection** and **union**.

The following two record sequence data is from the same table sequence, and then the record sequence data is stored in the two new table sequences respectively, and then the intersection "^" and union "&" operations are performed.

	A	B	C
1	=file("txt/user_1.txt").import@t()		/Import data
2	=A1([1, 2, 3])	=A1([2, 3, 4])	/Generate record sequences
3	=A1.create()	>A3.insert@r(0:A2)	/Generate a table sequence with data from A2
4	=A1.create()	>A4.insert@r(0:B2)	/Generate a table sequence with data from B2
5	=A3^A4		/Intersection between table sequences always returns null
6	=A3&A4		/Union of table sequences is equivalent to concatenation
7	=A2^B2		/Intersection between record sequences returns the common records
8	=A2&B2		/Union of record sequences returns the union

A2, A3 data:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11

B2, A4 data:

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	3	July	F	11
3	4	Gail	M	11

10.5.3 The difference between rseq and tseq

Table sequence set operations:

A3^A4 data:

Index	Member

The **intersection** of different table sequences is a null sequence.

A3&A4 data:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	2	Tom	M	15
5	3	July	F	11
6	4	Gail	M	11

A table sequence **union** is all the members of two tables, and the result is equal to the concatenation.

This shows that the records in different table sequences are always different, regardless of whether the fields and field values are the same.

Record sequence set operations:

A2^B2 data:

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	15
2	3	July	F	11

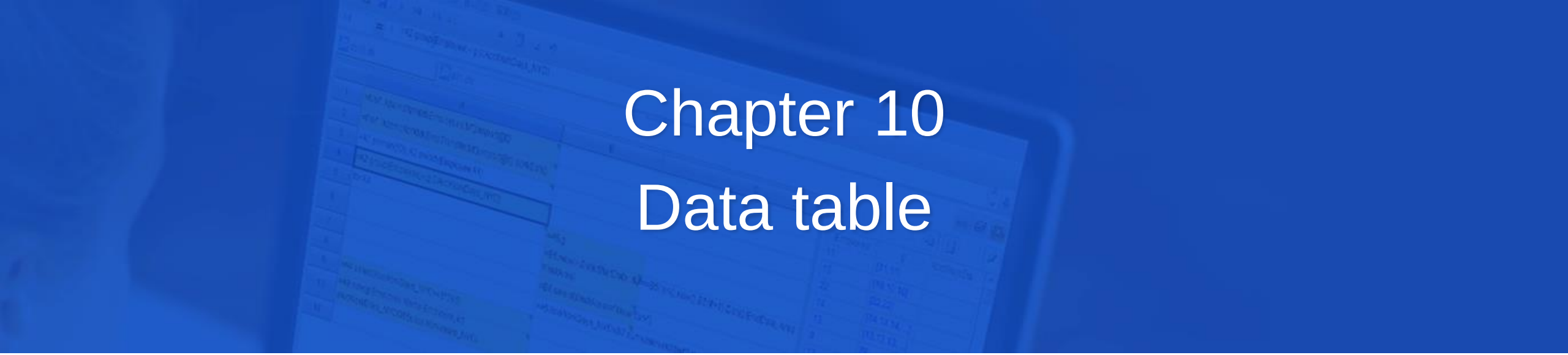
The record sequence **intersection** is the common member of two sets.

A2&B2 data:

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

A record sequence **union** is the members of two sets after removing duplicate data.

Notes for set operations of record sequences:
1. Record sequences have the same structure.
2. The record sequence data comes from the same table sequence.



Chapter 10
Data table

10.6 Exercises

10.6 Exercises

【Exercise 10.1】Calculate the amount of shopping expenses. The amount of each commodity is the unit multiplied by the quantity, and then the total amount of the commodities is the total consumption. Here is the shopping list(Using data from foods2.txt):

Index	ID	NAME	PLACE	UNIT	UNITPRICE	NUM
1	1	Apple	MI	1	1.69	7.0
2	2	Banana	FL	1	0.69	6.5
3	3	Cucumber	CA	3	0.77	4.5
4	4	Onion	CA	1	0.99	4.0
5	5	Orange	FL	2	4.99	10.0

	A	B	C
1	=file("foods2.txt").import@t()	0	/B1 is commodity amount variable
2	for A1	>B1=B1+A2.UNITPRICE*A2.NUM	/Accumulate commodity amount
3	>output("pay = " / B1)		/Output total payment

Output:pay = 73.64

A2 loops through each commodity. In the loop, B2 calculates the unit price of the commodity multiplied by the quantity of the commodity, that is, the amount of the commodity, and adds up to the amount of the commodity. Until the end of the loop, the cumulative amount of commodity is the total consumption.

10.6 Exercises

【Exercise 10.2】Delete the records with GENDER="M" in the table(Using data from txt/user_1.txt).

	A	B	C	D
1	=file("txt/user_1.txt").import@t()	=A1.count()		/Table sequence
2	for B1, 1	if A1(A2).GENDER=="M"	>A1.delete(A2)	/Delete records

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

After deletion

Index	SID	NAME	GENDER	AGE
1	3	July	F	11

The for loop traverses the records to find the records with GENDER="M", and then delete them. Considering that deletion affects the order of table data, reverse order deletion is adopted.

10.6 Exercises

【Exercise 10.3】Find the youngest user(Using data from txt/employee.txt).

	A	B	C	D
1	=file("txt/employee.txt").import@t()	=A1(1).BIRTHDAY		/C1 stores the record with the smallest age
2	for A1	=A2.BIRTHDAY		/Calculate age
3		if (B1<B2)	>C1=A2	/Store the record with the smaller age
4			>B1=B2	/Store the smaller value
5	>output("user="/C1.NAME/";AGE ="/ age(C1.BIRTHDAY))			

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	…
1	1	Rebecca	Moore	F	California	1974-11-20	…
2	2	Ashley	Wilson	F	New York	1980-07-19	…
3	3	Rachel	Johnson	F	New Mexico	1970-12-17	…
…	…	…	…				

Output: user=Emily; AGE=32

By comparing the birth date in the loop, we can find the younger user, and store the corresponding record at the same time, so as to know which user is the youngest at the end of the loop. If there are multiple users of the same age, only the first one will be returned.

10.6 Exercises

【Exercise 10.4】The difference between T.insert@r(k:A) and T.insert@f(k:A) (Using data from txt/user_5.txt).

	A	B	C
1	=file("txt/user_5.txt").import@t()		
2	=A1.create()	=create(NAME, SID, AGE, GENDER)	=create(NAME, SID, AGE, DEPT)
3	>A2.insert@r(0: A1)	>B2.insert@f(0: A1)	>C2.insert@f(0: A1)
4	/Data replication between same structures	/The same field names; the field order may be different	/The same field names; the number of fields may be different

Table A2 data:

Index	SID	NAME	GENDER	AGE
1	10	Jack	M	
2	11	Tom	M	
3	12	July	F	

Table B2 data:

Index	NAME	SID	AGE	GEND
1	Jack	10	12	M
2	Tom	11	15	M
3	July	12	11	F

Table C2 data:

Index	NAME	SID	AGE	DEPT
1	Jack	10	12	(null)
2	Tom	11	15	(null)
3	July	12	11	(null)

When inserting records with the same fields but different field order, T.insert@f(k:A) is more convenient, it automatically assigns the record data of A to the new record of T according to the same fields; while T.insert@r(k:A) assigns the record data of A to the new record of T according to the field order.

10.6 Exercises

【Exercise 10.5】The records with different conditions are modified accordingly. (Using data from foods.txt)

The price after discount is modified according to different unit field values:

	A	B	C
1	=file("foods.txt").import@t()		/Load text file data
2	for A1	if A2.UNIT==1	>A2.UNITPRICE=round(A2.UNITPRICE*(1-0.30), 2)
3		else if A2.UNIT==2	>A2.UNITPRICE=round(A2.UNITPRICE*(1-0.25), 2)
4		else	>A2.UNITPRICE=round(A2.UNITPRICE*(1-0.20), 2)

Index	ID	NAME	PLACE	UNIT	UNITPRICE	SELLERS
1	1	Apple	MI	1	1.69	Ashley Williams
2	2	Banana	FL	1	0.69	Andrew Miller
3	3	Cucumber	CA	3	0.77	Christina Williams
4	4	Onion	CA	1		
5	5	Orange	FL	2		

Price modification

Index	ID	NAME	PLACE	UNIT	UNITPRICE	SELLERS
1	1	Apple	MI	1	1.18	Ashley Williams
2	2	Banana	FL	1	0.48	Andrew Miller
3	3	Cucumber	CA	3	0.62	Christina Williams
4	4	Onion	CA	1	0.69	Christina Williams
5	5	Orange	FL	2	3.74	Ashley Williams

The complex calculation logic can be realized by dealing with different conditions in the loop body.

10.6 Exercises

【Exercise 10.6】Calculate the sum of the data in the score field in the table. (Using data from myscore.txt)

The data type of score is string. When getting the sum, the string should be split by comma, and then the cumulative values should be calculated.

Index	DeviceID	Month	Score
1	1050	1	73, 85
2	1020	2	66, 73, 102, 140
3	1012	3	81, 72, 63, 68

	A	B	C	D
1	=file("myscore.txt").import@tq()	0		/B1 is used to store data value
2	for A1	=A2.Score.split@cp()		/Split string into array
3		for B2	>B1=B1+B3	/Accumulate data values
4	>output("total="/B1)			/Output sum

Output: total=823

To get the sum of the data in the score field, the number string of each record is divided into an integer sequence, B3 traverses each member of the sequence B2, and C3 accumulates the data.

10.6 Exercises

【Exercise 10.7】The problem of merging multiple lines of a field(Using data from ident.xlsx).

The "cause analysis" field of an identification detail sheet is composed of three lines (Fault phenomenon, Diagnostic results, Maintenance plan and effect), which needs to be merged. The data is as follows:

	A	B	C	D	E	F	G	H
1	Details of identification sheet							
2	No	Num	Cause analysis	Code	Flag	Assembly plant	Price	Pdate
3	2408-B190094	1	Fault phenomenon: A1◦ Diagnostic results: B1◦ Maintenance plan and effect: C1◦	E123	12JS160T-1707140 FAST 18H20 009C 18H170209	E123	849.23	2020-4-10
4	1746-B200001	1	Fault phenomenon: A2◦	E123	FAST GEAR 18A161BXY	0KF	1494.55	2020-3-18
5			Diagnostic results: B2◦					
6			Maintenance plan and effect: C2◦					
7	2030-B200049	1	Fault phenomenon: A3◦	fst	FAST 19F02	E123	1962.09	2020-4-3
8			Diagnostic results: B3◦					
9			Maintenance plan and effect: C3◦					
10	2137-B200028	1	Fault phenomenon: A4◦	E123	JS130T-1701180 FAST GEAR 18L07	E123	916.24	2020-4-7
11			Diagnostic results: B4◦					
12			Maintenance plan and effect: C4◦					
13	2408-B190148	1	Fault phenomenon: A5◦	E123	12JSD160T-1707140-2 FAST 17C30	E123	1899.49	2020-3-31
14			Diagnostic results: B5◦					
15			Maintenance plan and effect: C5◦					
16								

Analysis: the result is to merge the field "cause analysis" into the adjacent record. Therefore, when traversing the data, if the field No of the record is not empty, the record r will be inserted into the table. If it is empty, the "cause analysis" field will be merged with the "cause analysis" field of the last record r.

10.6 Exercises

【Exercise 10.7】The problem of merging multiple lines of a field(Using data from ident.xlsx).

	A	B	C	D
1	=file("ident.xlsx").xlsimport@t(;, 2)	=A1.create()		/Import data
2	for A1	if A2.No==null		/Judge whether it is null
3			>C1.'Cause analysis'=C1.'Cause analysis'+"; "+A2.'Cause analysis'	/Concatenate the record value into a string and modify the record
4		else		
5			>B1.insert@r(0:[A2])	/Append data to table B1
6			=C1=B1(B1.len())	/Save the newly added record to C1
7	=file("ident.xlsx").xlsexport@t(B1;"merge")			/Output data to excel file

A1: The 2 in f .xlsimport@t(;, 2) represents reading from row 2.
A7: f.xlsexport@t(B1; "merge") indicates that the data in B1 is stored in an Excel file named "merge".

	A	B	C	D	E	F	G	H
1	No	Num	Cause analysis	Code	Flag	Assembly p	Price	Pdate
2	2408-B19C	1	Fault phenomenon: A1。 Diagnostic results： B1。 Maint	E123	12JS160T-	E123	849.23	2020-04-10
3	1746-B20C	1	Fault phenomenon: A2。 ; Diagnostic results： B2。 ; Mai	E123	FAST GEAR	0KF	1494.55	2020-03-18
4	2030-B20C	1	Fault phenomenon: A3。 ; Diagnostic results： B3。 ; Mai	fst	FAST 19FC	E123	1962.09	2020-04-03
5	2137-B20C	1	Fault phenomenon: A4。 ; Diagnostic results： B4。 ; Mai	E123	JS130T-17	E123	916.24	2020-04-07
6								

10.6 Exercises

【Exercise 10.8】Merge multiple data files with the same structure.

In the users directory, there are multiple data files with the same structure in the form of user_#i.txt that need to be merged (#i is the serial number).

	A	B	C	D
1	[]			/Define an empty sequence
2	for 10	="users/user_"+string(A2)+".txt"		/Concatenate to a string users/user_#i.txt
3		if file(B2).exists()	=file(B2).import@t()	/If the file exists, its data is loaded
4			>A1=A1|C3	/Concatenate data into A1
5	>file("users.txt").export@t(A1)			/Output table data to a file users.txt

B3: judge whether the file exists, C3: load file data, C4: merge file data.

Index	SID	NAME	GENDER	AGE
1	10	Jack10	M	19
2	11	Lucy11	F	30
3	12	Tom12	M	25

Index	SID	NAME	GENDER	AGE
1	20	Jack20	M	29
2	21	Lucy21	F	30
3	22	Tom22	M	25

Merge data

Index	SID	NAME	GENDER	AGE
1	10	Jack10	M	19
2	11	Lucy11	F	30
3	12	Tom12	M	25
4	20	Jack20	M	29
…	…	…	…	…

Chapter 11 Calculation of table

11.1 Aggregation

11.2 Filter and positioning

- Filter
- Positioning
- Calculation of specified location

11.3 Aligned calculation

- Modification
- Generation
- Addition

11.4 Sorting and ranking

- Sorting
- Sorting and positioning
- Ranking

11.5 Max/Min value

- Max/Min value
- TOP

11.6 Exercises

Chapter 11
Calculation of table

11.1 Aggregation

11.1 Aggregation

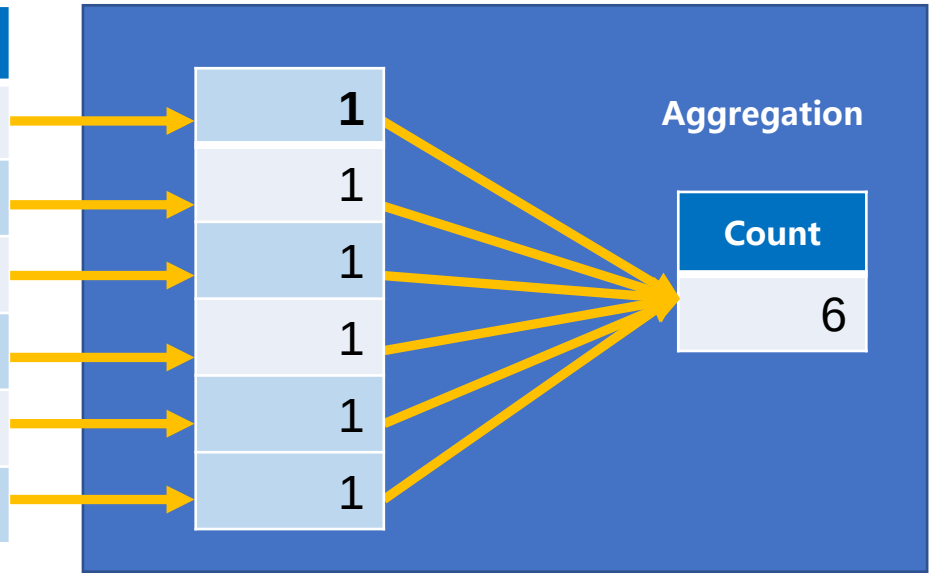
In the previous chapter, we implemented the aggregate operation with loop statements. Of course, this is no problem, but it is troublesome.

Table sequence is similar to sequence, it can be operated as a set, and aggregate can be realized by functions. As the sequence operations introduced in Chapter 9, aggregate operations can also be performed on table sequences or record sequences with aggregate functions, such as **T.count(x), T.sum(x), T.avg(x), T.min(x), T.max(x)**, etc.

It should be noted that when using aggregate functions for table sequences or record sequences, the aggregate expression **x** usually performs aggregate calculation for the values of **a field or certain fields.**

Count: Count() function counts the table records.

Index	SID	NAME
1	1	A1
2	2	A2
3	3	A3
4	4	A4
5	(null)	(null)
6	5	A5



When count() counts the table records, the **empty records** whose fields are all null are also counted.

11.1 Aggregation

【Example 11.1】**Count:** Find the number of records in the user table. (Using data from user_101.txt)

Solution:

A.count() calculates the number of records in the table, including the empty records.

	A	B
1	=file("txt/user_101.txt").import@t()	/Import data
2	=A1.count()	/Count records

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11
5	(null)	(null)	(null)	(null)
6	(null)	(null)	(null)	(null)
7	7	Liky	F	10

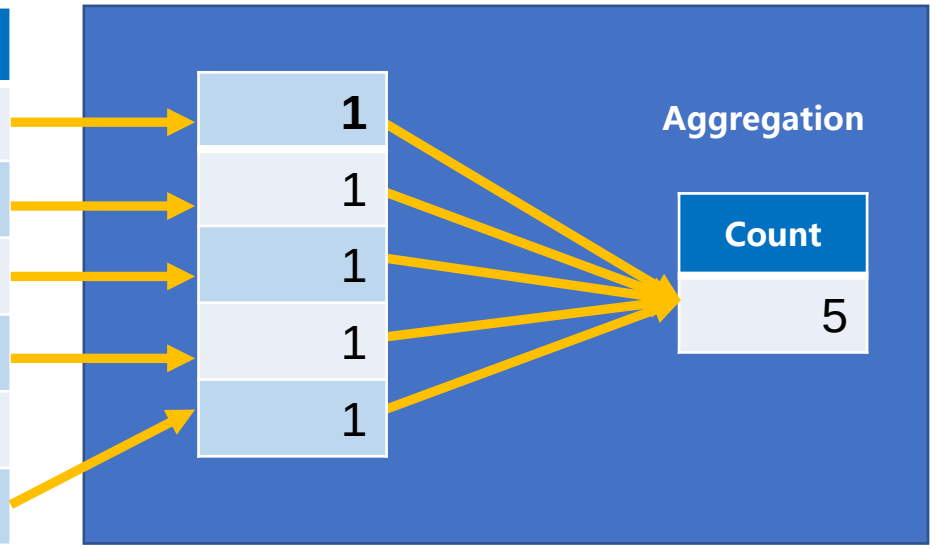
A2	Value
	7

There are seven records, including two empty records.

When counting with A.count(), empty records are included.

Count: Count(x) counts the field values.

Index	SID	NAME
1	1	A1
2	2	A2
3	3	A3
4	4	A4
5	(null)	(null)
6	5	A5



When count(NAME) counts the values of NAME field, null values are ignored and not included.

11.1 Aggregation

【Example 11. 2】**Count:** Find the number of users in the user table. (Using data from user_101.txt)

Solution:
Find the number of non null user names. **A.count(x)** calculates the number of records with non null result of expression x.

	A	B
1	=file("txt/user_101.txt").import@t()	/Import data
2	=A1.count(NAME)	/Count the number of users

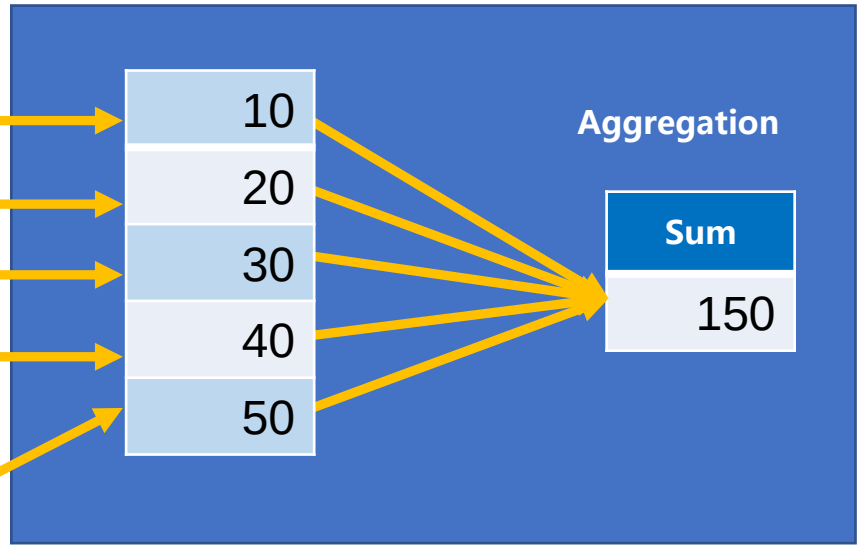
Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11
5	(null)	(null)	(null)	(null)
6	(null)	(null)	(null)	(null)
7	7	Liky	F	10

A2	Value
	5

The number of records in the table is 7, and the number of users is 5, excluding 2 null values. A.count(x) calculates the number of records whose result of expression x is not null.

Sum: sum(x) adds the field values cumulatively.

Index	SID	SCORE
1	1	10
2	2	20
3	3	30
4	4	40
5	(null)	(null)
6	5	50



When sum(SCORE) calculates the sum of values in the SCORE field, null values are ignored and excluded.

11.1 Aggregation

【Example 11. 3】**Sum:** Get the sum of user ages. (Using data from user_101.txt)

Solution:
Accumulate the AGE values in the records. **A.sum(x)** calculates the sum of values of x column in table A.

	A	B
1	=file("txt/user_101.txt").import@t()	/Import table
2	=A1.sum(AGE)	/Calculate the sum of AGE values

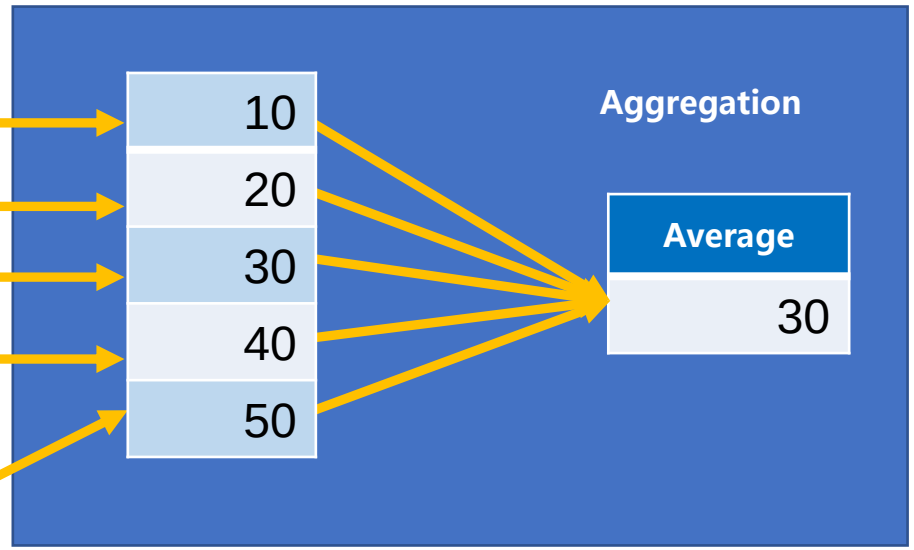
Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11
5	(null)	(null)	(null)	(null)
6	(null)	(null)	(null)	(null)
7	7	Liky	F	10

A2	Value
	49

A.sum(x) ignores the null values.

Average: avg(x) calculates the average value of field values.

Index	SID	SCORE
1	1	10
2	2	20
3	3	30
4	4	40
5	(null)	(null)
6	5	50



When avg(SCORE) calculates the average of scores, the null field value is not included.

11.1 Aggregation

【Example 11. 4】**Average:** Find the average age of users. (Using data from user_101.txt)

Solution:
A.avg(x) calculates the average value of column x in table A.

	A	B
1	=file("txt/user_101.txt").import@t()	/Import data
2	=A1.avg(AGE)	/Calculate average age

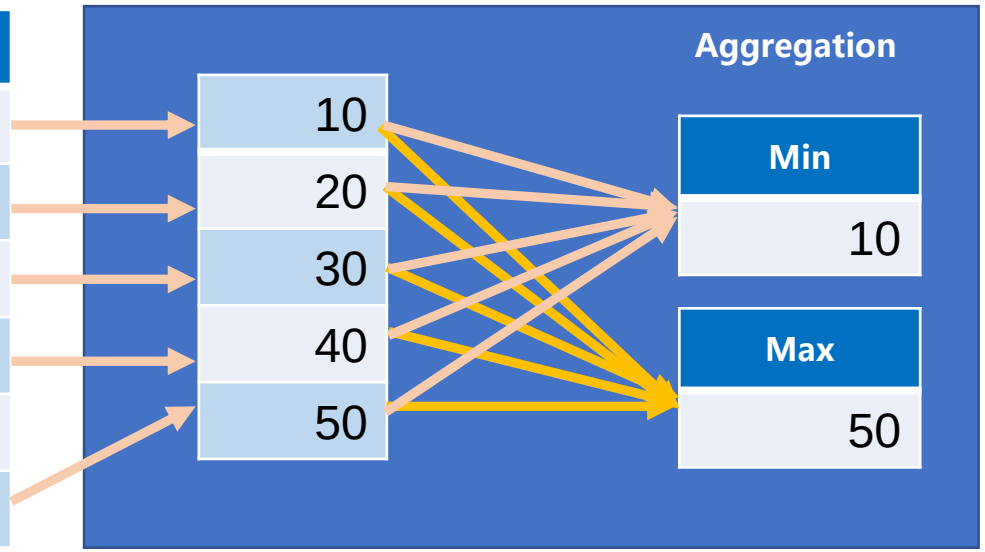
Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11
5	(null)	(null)	(null)	(null)
6	(null)	(null)	(null)	(null)
7	7	Liky	F	10

A2	Value
	11.8

Output the average value: **T.avg(x)**
Calculate the average age of users, and the effect of empty records is ignored.

Max/Min: Get the Max value from field values, and get the Min value from field values.

Index	SID	SCORE
1	1	10
2	2	20
3	3	30
4	4	40
5	(null)	(null)
6	5	50



When calculating the maximum value and minimum value of SCORE, the null value is ignored.

【Example 11. 5】**Max/Min value**: Find the maximum (minimum) value of the user's age. (Using data from user_101.txt)

Solution:
A.max(x) gets the maximum value of column x in table A. **A.min(x)** gets the minimum value of column x in table A.

	A	B
1	=file("txt/user_101.txt").import@t()	/Read the text file data
2	=A2.max(AGE)	/Calculate the maximum age
3	=A2.min(AGE)	/Calculate the minimum age

A2	Value
	15

A3	Value
	10

Chapter 11
Calculation of table

11.2 Filter and positioning

11.2.1 Filter

Filter using **relational operators** >, >=, ==, <=, < etc.

【Example 11.6】Query the employee whose name is Ashley.(Using data from emp.txt)

Solution:
A.select(x) can be used to filter the table. If the value of expression x is not false or null, it meets the filtering criteria. The @1 option returns the first record that meets the criteria, ignoring x returns all records. The use of A.select (x) in table sequence or record sequence is the same as in sequences.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.select(NAME=="Ashley")	/Select records with NAME==Ashley
3	=A1.select@1(NAME=="Ashley")	/Select the first record with NAME==Ashley

A2	Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
	1	2	Ashley	Wilson	F	New York	1980-07-19	2008-03-16	Finance	11000
	2	5	Ashley	Smith	F	Texas	1975-05-13	2004-07-30	R&D	16000
	3	68	Ashley	Moore	F	New York	1980-05-09	2007-04-01	Sales	10000
	4	80	Ashley	Johnson	F	Illinois	1974-10-20	2004-09-01	Marketing	7000
	…	…	…	…	…	…	…	…	…	…

A3	Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
	1	2	Ashley	Wilson	F	New York	1980-07-19	2008-03-16	Finance	11000

Note that the result in A2 is a **record sequence** of records, not a new table sequence.

11.2.1 Filter

Filter using **logical operators** &&, ||, ! Etc.

【Example 11.7】Query the employee whose name is Ashley and whose surname is Smith.(Using data from emp.txt)

Solution:
A.select(x) supports multiple conditions.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.select(NAME=="Ashley" && SURNAME== "Smith")	/Select records that meet multi conditions

A2

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	5	Ashley	Smith	F	Texas	1975-05-13	2004-07-30	R&D	16000
2	123	Ashley	Smith	F	Indiana	1987-02-09	2011-03-01	Sales	5000
3	163	Ashley	Smith	F	Kentucky	1979-12-26	2010-12-01	HR	5000
4	339	Ashley	Smith	F	Missouri	1980-04-17	2001-11-01	Production	7000
5	376	Ashley	Smith	F	Michigan	1969-08-07	2000-07-01	Marketing	7000

11.2.1 Filter

Other **expressions** can also be used as filter conditions, such as string matching queries.

【Example 11.8】Query the employees whose state name starts with "New" and has worked for more than 10 years. (Using data from emp.txt)

Solution:
Find the state name starting with "New" by string matching, and the length of service can be calculated by age function.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.select(left(STATE, 3)=="New" && age(HIREDATE)>10)	/Select the employees whose state name starts with "New" and has worked for more than 10 years.

A2

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	2	Ashley	Wilson	F	New York	1980-07-19	2008-03-16	Finance	11000
2	12	Jessica	Davis	F	New York	1980-09-11	2008-09-11	Sales	7000
3	15	Alexis	Smith	F	New York	1983-07-10	2006-07-10	Sales	8000
4	25	Sarah	Davis	F	New York	1976-11-25	2006-11-25	Marketing	12000
5	38	Matthew	Johnson	M	New York	1972-11-20	2002-11-20	R&D	6000
…	…	…	…	…	…	…	…	…	…

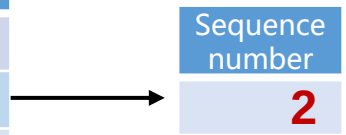
11.2.2 Positioning

Sometimes, we don't just care about the record that meets the condition, but more about the position of the first record that meets the condition.

Similar to locating members in a sequence described in Chapter 9, you can use the positioning functions of a sequence to locate records in a table sequence or record sequence.

Find the sequence number of the first record with AVAILABLE=1 in the data table.

Index	SID	AVAILABLE
1	101	0
2	102	1
3	103	1
4	104	0
5	105	0
6	106	1
7	107	1
8	108	0



Return sequence number of the record

When a record satisfying the condition is found, the search is terminated and the corresponding **sequence number** is returned immediately.
Just like when we put things into the automatic storage cabinet, we only need to return an empty cabinet.

11.2.2 Positioning

【Example 11.9】Find employees whose hire date is earlier than Sydney. (Using data from emp.txt)

Solution:

If you locate the position of a specified employee in the ordered table, the records before this position are all the records to be found.

A.pselect(x) finds the sequence number of the first record that meets the search condition x.

	A	B
1	=file("txt/emp3.txt").import@t().sort(HIREDATE)	/Load data and sort by HIREDATE
2	=A1.pselect(NAME:"Sydney")	/Return the sequence number of the first record that satisfies the condition
3	=A1.to(A2-1)	/Find employee records

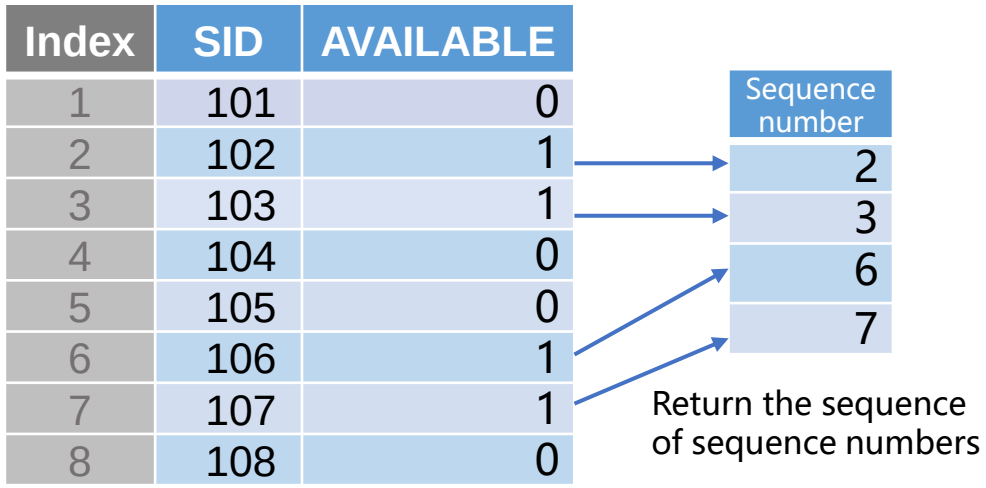
A3

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	18	Jonathan	Moore	M	Florida	1971-03-07	2000-03-07	Administration	7000
2	203	Abigail	Smith	F	Arizona	1972-02-20	2000-04-01	Sales	7000
3	487	Hannah	Brown	F	Pennsylvania	1980-08-27	2000-05-01	Marketing	5000
4	425	Joseph	Thomas	M	Illinois	1971-07-02	2000-05-01	Sales	7000
5	226	William	Smith	M	New York	1976-01-23	2000-05-01	Sales	6500
…	…	…	…	…	…	…	…	…	…

11.2.2 Positioning

When querying data, find the **sequence numbers** of **all** records that meet the conditions.

Find the sequence numbers of all the records with AVAILABLE=1 in the data table.



Index	SID	AVAILABLE
1	101	0
2	102	1
3	103	1
4	104	0
5	105	0
6	106	1
7	107	1
8	108	0

Find the sequence numbers of all records that meet the conditions, and return the sequence composed of sequence numbers.
For example, you want to know which automatic storage cabinets are empty.

11.2.2 Positioning

【Example 11.10】Find the biggest up and down amount in those days when the closing price of the stock 000062 is higher than 10? (Using data from stockrecords.txt)

Solution:

A.pselect() can find the records whose closing price is greater than 10, and then use **A.max(x)** to get the record with the largest increase or decrease from the filtered records. Use **A.pselect@a(x)** to find the sequence composed of all the sequence numbers that meet the conditions, which is the same as the positioning function of the sequence in Chapter 9.

	A	B
1	=file("txt/stockrecords.txt").import@tq().select(STOCKID=="000062") .sort(DATE)	/Load data
2	=A1.pselect@a(CLOSING>10)	/Return the sequence numbers of all records whose closing price is greater than 10
3	=A2.max(if (~>1, A1(~).CLOSING-A1(~-1).CLOSING, 0))	/Return the max up/down value

A2

Index	Member
1	34
2	76
3	77
4	79
5	80
…	…

A3

Value
1.1600000000000001

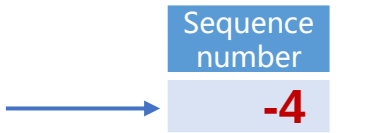
A.pselect@a(x) returns a sequence of record sequence numbers, A.max(x) gets the maximum value difference of adjacent records.

11.2.2 Positioning

When querying data in an **ordered** data table, if the data is not found, return the **opposite number** of data insertion position that meets the conditions.

In the data table **sorted** by score, find the record with SCORE=100.

Index	SID	SCORE
1	101	50
2	102	84
3	103	90
4	104	143
5	105	147
6	106	150
7	107	154
8	108	183



Return sequence number

The existence of the record is judged by the result. If not, the data can be inserted into the ordered data table.

Note: in this case, it is required that the records are ordered, otherwise the returned result is incorrect.

11.2.2 Positioning

【Example 11.11】Store the new book BOOK110 with BOOKID BK_1010 in an ordered data table. (Using data from books.txt)

Solution:
Search for book with BOOKID BK_1010 in a book table sorted by BOOKID, if not found, the return value is the opposite number of insertion position, which can be implemented by **A.pselect@s(x)**, and then insert the data according to the returned sequence number.

	A	B	C
1	=file("txt/books.txt").import@t()		/Import data
2	=A1.pselect@s(BOOKID:"BK_1010")		/Find the position of new book
3	if (A2<0)	>A1.insert(-A2, "BK_1010", "BOOK110")	/Insert the record to the table sequence

A1

Index	BOOKID	NAME
1	BK_1001	BOOK101
2	BK_1002	BOOK122
3	BK_1003	BOOK103
4	BK_1005	BOOK114
5	BK_1008	BOOK115
6	BK_1011	BOOK136



A1

Index	BOOKID	NAME
1	BK_1001	BOOK101
2	BK_1002	BOOK122
3	BK_1003	BOOK103
4	BK_1005	BOOK114
5	BK_1008	BOOK115
6	BK_1010	BOOK110
7	BK_1011	BOOK136

A2

Value
-6

A.pselect(x) returns -6, the opposite number is the insertion position. The table sequence is still ordered.

11.2.2 Positioning

After having the record sequence number, the relevant calculations can done by the sequence numbers.

【Example 11.12】Find the increase/decrease amount of the closing price of stock 000062 on the day that the closing price is the highest. (Using data from stockrecords.txt)

Solution:
Similar to the use in the sequence, A.pselect() can be used to find the record position of the highest stock price, and then A.calc() can be used to calculate the rise/fall amount of the stock.
A.calc(k, x) calculates expression **x** at the position of the record with sequence number **k**. **k** is the record sequence number, **x** is the expression, **~** can be used to represent the current record, **~.F** or **F** refers to the value of **F** field of current record, **~.F[-i]** or **F[-i]** refers to the value of **F** field of the previous **i^{th}** record.

	A	B
1	=file("txt/stockrecords.txt").import@tq().select(STOCKID="000062"). sort(DATE)	=A1.max(CLOSING)
2	=A1.pselect(CLOSING==B1)	/Return the sequence number of the record with the highest closing price
3	=A1.calc(A2, CLOSING - CLOSING[-1])	/Calculate the increase/decrease at the position of specified sequence number

A3	Value
	1.1300000000000008

11.2.2 Positioning

After having the sequence of record sequence numbers, the relevant calculations can done by the sequence.

【Example 11.13】Find the increase/decrease amount in those days when the closing price of stock 000062 is higher than 10. (Using data from stockrecords.txt)

Solution:
A.pselect() can find records with stock price greater than 10, and then use **A.calc()** to calculate the rise/fall amount of the stock from the filtered records. In **A.calc(p, x)**, p is the sequence of record sequence numbers, and x is the calculation expression. The expression is calculated for the specified records, and the calculation results are returned.

	A	B
1	=file("txt/stockrecords.txt").import@tq().select(STOCKID =="000062")	/Load data
2	=A1.pselect@a(CLOSING>10)	/Return the sequence number of all records whose stock price is greater than 10
3	=A1.calc(A2, CLOSING - CLOSING[-1])	/Calculate the amount of increase/decrease by record sequence numbers

A3	Index	Member
	1	0.1899999999999995
	2	0.8400000000000016
	3	0.1999999999999993
	4	0.8599999999999994
	5	0.9100000000000001
	…	…

A.calc() can locate and calculate a single record or multiple records through different types of parameters.

Chapter 11
Calculation of table

11.3 Aligned calculation

11.3.1 Modification

Modify M to Male and F to Female in field GENDER. The implementation of SPL script for loop is as follows:

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	M	2007-05-03
2	2	Tom	M	2004-07-12
3	3	July	F	2010-09-04
4	4	Gail	M	2010-03-08



Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	Male	2007-05-03
2	2	Tom	Male	2004-07-12
3	3	July	Female	2010-09-04
4	4	Gail	Male	2010-03-08

	A	B	C
1	=file("txt/user_2.txt").import@t()		/Import data
2	for A1	if A2.GENDER=="M"	>A2.GENDER="Male"
3		else	>A2.GENDER="Female"

To modify the field values of a column, the for loop mode requires multiple lines of code, and it is much easier to write with the function **A.run(x)**.

11.3.1 Modification

【Example 11.14】Modify M to Male and F to Female in field GENDER. (Using data from user_2.txt)

Solution:
For each record in the table, it can be modified by **A.run(x)** function.
A.run(x) calculates the expression x for each record in the table, modifies the current record with the calculation result, and finally returns the modified **A**.

	A	B
1	=file("txt/user_2.txt").import@t()	/Import data
2	>A1.run(GENDER=if (GENDER=="M", "Male" , "Female"))	/Modify "F", "M" to corresponding "Female", "Male"

A1

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	M	2007-05-03
2	2	Tom	M	2004-07-12
3	3	July	F	2010-09-04
4	4	Gail	M	2010-03-08



A1

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	Male	2007-05-03
2	2	Tom	Male	2004-07-12
3	3	July	Female	2010-09-04
4	4	Gail	Male	2010-03-08

Note: The expression in **A.run(x)** x is assignment expression.

11.3.1 Modification

Modify the values of multiple fields at the same time.

【Example 11.15】Modify the GENDER field of user table, and convert the date of birth into age. (Using data from user_2.txt)

Solution:
When there are multiple fields in the table that need to be modified, that is to calculate multiple expressions, which can be realized by **A.run(x_1, x_2, ...x_i)** function.
Similar to A.run(x), **A.run(x_1, x_2, ...x_i)** is for multiple expressions x_i, and "~" can be used to refer to the current record.

	A	B
1	=file("txt/user_2.txt").import@t()	/Import file data
2	>A1.run(GENDER=if (GENDER=="M", "Male" , "Female"), BIRTHDAY=age(BIRTHDAY))	/Calculate multiple expressions

A1

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	M	2007-05-03
2	2	Tom	M	2004-07-12
3	3	July	F	2010-09-04
4	4	Gail	M	2010-03-08

A2修改

A1

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	Male	13
2	2	Tom	Male	16
3	3	July	Female	9
4	4	Gail	Male	10

11.3.2 Generation

When the data of the specified field needs to be obtained, such as:

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	M	2007-05-03
2	2	Tom	M	2004-07-12
3	3	July	F	2010-09-04
4	4	Gail	M	2010-03-08



Index	Member
1	2007-05-03
2	2004-07-12
3	2010-09-04
4	2010-03-08

When accessing BIRTHDAY field data or calculating age by birth date, the for loop method is as follows:

	A	B	C
1	=file("txt/user_2.txt").import@t()		/A2 is used to store birthday values
2	[]	[]	/B2 is used to store age values
3	for A1	>A2=A2|A3. BIRTHDAY	/store BIRTHDAY values in loop
4	for A1	>B2=B2|age(A4. BIRTHDAY)	/store age values in loop

There is a simpler method than for loop, whether it is to get the field values directly or to calculate the field values.

11.3.2 Generation

【Example 11.16】Get the data of the BIRTHDAY field in the table and calculate the users' ages. (Using data from user_2.txt)

Solution:

When obtaining table column data, such as the BIRTHDAY field data, it can be obtained through **A.(x)**, and the use of **A.(x)** is the same as that in the sequence in Chapter 9.

When used in a table sequence or record sequence, A.(x) calculates the expression x for each record in the table and returns a sequence of calculated results. When x is null, all data in table A is returned; x is generally a field name or an expression composed of field names. In the expression, ~ can be used to reference the current record.

	A	B
1	=file("txt/user_2.txt").import@t()	/Import data
2	=A1.(BIRTHDAY)	/Get BIRTHDAY data
3	=A1.(age(BIRTHDAY))	/Calculate age through birthday

A1

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	M	2007-05-03
2	2	Tom	M	2004-07-12
3	3	July	F	2010-09-04
4	4	Gail	M	2010-03-08

A2

Index	Member
1	2007-05-03
2	2004-07-12
3	2010-09-04
4	2010-03-08

A3

Index	Member
1	13
2	16
3	9
4	10

11.3.2 Generation

Create a new table sequence based on the data in the table.

For example, extract the name and birthday field data in the table.

Original data table

Index	SID	NAME	GENDER	BIRTHDAY
1	1	Jack	M	2007-05-03
2	2	Tom	M	2004-07-12
3	3	July	F	2010-09-04
4	4	Gail	M	2010-03-08

New data table

Index	NAME	BIRTHDAY
1	Jack	2007-05-03
2	Tom	2004-07-12
3	July	2010-09-04
4	Gail	2010-03-08

11.3.2 Generation

【Example 11.17】Extract the data of EID, NAME, SURNAME, DEPT fields from the employee table. (Using data from emp.txt)

Solution:

A.new(x_i:F_i, ...) function can be used to construct a new table sequence with the specified fields in the table.

A.new(x_i:F_i, ...) generates a new table sequence with the same number of records as table A, x_i is the expression and the calculated results are the field values. F_i is the field name. If omitted, the identifier resolved in x_i is used.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.new(EID, NAME, SURNAME, DEPT)	/Generate a new table sequence with fields: EID, NAME, etc.

A2

Index	EID	NAME	SURNAME	DEPT
1	1	Rebecca	Moore	R&D
2	2	Ashley	Wilson	Finance
3	3	Rachel	Johnson	Sales
4	4	Emily	Smith	HR
5	5	Ashley	Smith	R&D
...	...	...		

A.new(x_i:F_i, ...), if F_i is omitted, the expression string of x_i is used as the new field name.

11.3.2 Generation

Computed columns can be used to construct a new table sequence from an existing table.

【Example 11.18】Get data from employee table to construct a new table, concatenate NAME and SURNAME to get the FULLNAME, calculate user age, store it in a new field age, and read #4 column (gender). (Using data from emp.txt)

Solution:

Using the fields and calculated columns of the existing table to construct a new table sequence, we can **A.new(x_i:F_i, ...)** function to implement. In **A.new(x_i:F_i, ...)**, x_i is an expression and the results are field values, when the parameter x_i uses #i, it represents the i^{th} column, and the original column name is used.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.new(EID, NAME+" "+SURNAME:FULLNAME, age(BIRTHDAY):AGE, #4)	/Generate a new table sequence with field: EID, FULLNAME etc. #4 represents the GENDER field.

A2	Index	EID	FULLNAME	AGE	GENDER
	1	1	Rebecca Moore	45	F
	2	2	Ashley Wilson	39	F
	3	3	Rachel Johnson	49	F
	4	4	Emily Smith	35	F
	5	5	Ashley Smith	45	F
	...	...	...	...	...

11.3.2 Generation

Sometimes the calculation is done after the new table is generated, as needed.

【Example 11.19】After getting EID, NAME, SURNAME, BIRTHDAY field data from the employee table, calculate the age field. (Using data from emp.txt)

Solution:

In the process of constructing a new table sequence with **A.new(x_i:F_i, ...)**, a new field AGE without values can be added. **A.run(x)** calculates age according to the date of birth and stores it in AGE.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.new(EID, NAME+" "+SURNAME:FULLNAME, BIRTHDAY, AGE)	/Generate a new table sequence with field: EID, FULLNAME etc. (The AGE field is empty)
3	>A2.run(AGE=age(BIRTHDAY))	/Calculate age and set it to table sequence.

A2	Index	EID	FULLNAME	BIRTHDAY	AGE
	1	1	Rebecca Moore	1974-11-20	45
	2	2	Ashley Wilson	1980-07-19	39
	3	3	Rachel Johnson	1970-12-17	49
	4	4	Emily Smith	1985-03-07	35
	5	5	Ashley Smith	1975-05-13	45
	…	…	…	…	…

11.3.3 Addition

New fields can be appended to an existing data table to store the data as needed.

The scores of each student is recorded in the score table. If you want to know the sum of each student's scores and need further calculation, you can add a field to record it. For example, add score field to record the sum of scores.

Index	SID	Name	English	Math	History	Score
1	101	Tom	50	65	73	188
2	102	July	80	98	66	244
3	103	Jack	68	88	81	237
4	104	Mark	86	78	84	248
5	105	Megan	87	94	85	266

11.3.3 Addition

【Example 11.20】Check each student's score and total score from the score table. (Using data from score_2.txt)

Solution:
There are scores of each student in each subject in the score table. To add a new field to store the total score, **A.derive()** function can be used.
A.derive(x_i:F_i, ...) adds new fields to construct a new table sequence on the basis of copying table A, x_i is an expression, and the calculation result is the field values; F_i is the new field name, F_i cannot have the same name as the original fields of A.

	A	B
1	=file("txt/score_2.txt").import@t()	/Load text data and return a table sequence
2	=A1.derive(English+Math+History: Score)	/Add Score field to record the total score

A2	Index	SID	Name	English	Math	History	Score
	1	101	Tom	50	65	73	188
	2	102	July	80	98	66	244
	3	103	Jack	68	88	81	237
	4	104	Mark	86	78	84	248
	5	105	Megan	87	94	85	266

11.3.3 Addition

As needed, sometimes new fields are added when a new table is created, and the data is written to the new fields later.

【Example 11.21】Find the sum and average of each student's scores in each subject. (Using data from score_2.txt)

Solution:

A.derive() can be used to add two fields, and use A.run(x) to calculate the total score and the average score, and then saves them into these two fields.

	A	B
1	=file("txt/score_2.txt").import@t()	/Load text data and return a table sequence
2	=A1.derive(TScore, AvgScore)	/Add TScore, AvgScore fields
3	>A2.run(TScore=English+Math+History, AvgScore=round(TScore/3, 2))	/Save the total score and average score to the new fields.

A2	Index	SID	Name	English	Math	History	TScore	AvgScore
	1	101	Tom	50	65	73	188	62.67
	2	102	July	80	98	66	244	81.33
	3	103	Jack	68	88	81	237	79.0
	4	104	Mark	86	78	84	248	82.67
	5	105	Megan	87	94	85	266	88.67

A3: TScore is used to store the total score of each record; round() keeps the average score to two decimal places.

Chapter 11
Calculation of table

11.4 Sorting and ranking

11.4.1 Sorting

Sorting is to adjust a set of unordered records into ordered records.

Field is often used to sort for data tables. When sorting, first determine which field or fields to use, and sort in **ascending** or **descending** order. By default the sorting is in **ascending order**. This is different from the sorting of sequence introduced in Chapter 9. In the table sequence, **records** are sorted. Sorting is usually based on the values of a field or the calculation result of an expression. The result returned after sorting is the **record sequence** obtained by adjusting the order of records.

Table sequence data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

ascending order : sort from small to large;
descending order : sort from large to small

To sort by NAME is to sort by the values of **ASCII** (in ascending order)

The record sequence after sorting by NAME in **ascending** order

Index	SID	NAME↑	GENDER	AGE
1	4	Gail	M	11
2	1	Jack	M	12
3	3	July	F	11
4	2	Tom	M	15

The record sequence after sorting by NAME in **descending** order

Index	SID	NAME↓	GENDER	AGE
1	2	Tom	M	15
2	3	July	F	11
3	1	Jack	M	12
4	4	Gail	M	11

11.4.1 Sorting

【Example 11.22】Sort the employee table by BIRTHDAY. (Using data from emp.txt)

Solution:
A.sort (x) sorts table records by the given field.
x is an expression, sorting records in ascending order by x, -x indicates descending order; option **@z** indicates reverse order.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.sort(BIRTHDAY)	/Sort by BIRTHDAY in ascending order
3	=A1.sort@z(BIRTHDAY)	/Sort by BIRTHDAY in descending order
4	=A1.sort(-BIRTHDAY)	/Sort by BIRTHDAY in descending order

11.4.1 Sorting

【Example 11.22】Sort the employee table by BIRTHDAY. (Using data from emp.txt)

A2

Sort by BIRTHDAY in **ascending** order

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY↑	HIREDATE	DEPT	SALARY
1	296	Olivia	Johnson	F	Ohio	1968-11-05	2006-11-01	Production	10000
2	444	Alexis	Johnson	F	New Hampshire	1968-11-12	2010-12-01	Marketing	6500
3	440	Nicholas	Smith	M	Illinois	1968-11-24	2008-07-01	Marketing	8000
4	158	Luis	Johnson	M	Ohio	1968-11-27	2003-02-01	Technology	8000
5	331	Tyler	Williams	M	Florida	1968-12-23	2001-09-01	Sales	6500
…	…	…	…	…	…	…	…	…	…

A3

Sort by BIRTHDAY in **descending** order

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY↓	HIREDATE	DEPT	SALARY
1	472	Hannah	Brown	F	California	1987-12-27	2008-10-01	Sales	10000
2	89	Emily	Johnson	F	Michigan	1987-12-09	2010-02-01	Technology	7000
3	241	Samantha	Davis	F	Georgia	1987-12-04	2007-01-01	Finance	5000
4	437	Zachary	Williams	M	Tennessee	1987-12-01	2009-02-01	Marketing	6500
5	393	Hannah	Johnson	F	Florida	1987-09-06	2006-01-01	Sales	5000
…	…	…	…	…	…	…	…	…	…

11.4.1 Sorting

For **multi field sorting**, compare the first field when sorting. If it is sorted in ascending order, the one with larger field value is at the back, and the one with smaller field value is in the front. If it is sorted descending order, the order is reversed; if the field values are equal, then compare the second field, and the sorting rule of the second field is the same as that of the first field, and so on for subsequent fields.

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	11
4	4	Gail	M	11

The record sequence after sorting by AGE in ascending order and by GENDER in ascending order

Index	SID	NAME	GENDER↑	AGE↑
1	3	July	F	11
2	4	Gail	M	11
3	1	Jack	M	12
4	2	Tom	M	15

The record sequence after sorting by AGE in ascending order and by GENDER in descending order

Index	SID	NAME	GENDER↓	AGE↑
1	4	Gail	M	11
2	3	July	F	11
3	1	Jack	M	12
4	2	Tom	M	15

11.4.1 Sorting

【Example11.23】Sort the employee table by NAME in ascending order and by SURNAME in descending order. (Using data from emp.txt)

Solution:
A.sort(x_1, x_2, ...x_i) supports sorting by **multi fields**, and the sorting method for each field can be specified independently.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.sort(NAME, -SURNAME)	/Sort by NAME in ascending order and by SURNAME in descending order

A2 The record sequence after sorting NAME by NAME in ascending order and by SURNAME in descending order

Index	EID	NAME↑	SURNAME↓	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	180	Abigail	Smith	F	New York	1972-09-19	2007-05-01	HR	5000
2	203	Abigail	Smith	F	Arizona	1972-02-20	2000-04-01	Sales	7000
3	274	Abigail	Smith	F	Illinois	1978-04-05	2003-08-01	Marketing	5000
4	410	Abigail	Smith	F	Tennessee	1984-06-28	2004-04-01	Marketing	10000
5	311	Abigail	Johnson	F	Florida	1983-09-11	2003-08-01	Production	6500
…	…	…	…	…	…	…	…	…	…

11.4.1 Sorting

In addition to sort by specified fields, you can also sort by the values of a calculated expression.

【Example 11.24】Sort the employee table by the month and day of BIRTHDAY field. (Using data from emp.txt)

Solution:
Get month and day from BIRTHDAY. In A.sort(x_i, ...), x_i is a number of expressions, which are calculated to get the month and day, and then sorted by them.

	A	B
1	=file("txt/emp.txt").import@t()	/Load data
2	=A1.sort(month(BIRTHDAY), day(BIRTHDAY))	/Sort by month and day of BIRTHDAY in ascending order

A2 Sort by month and day of BIRTHDAY in ascending order

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY↑	HIREDATE	DEPT	SALARY
1	172	Brandon	Brown	M	Pennsylvania	1980-01-01	2002-01-01	HR	6500
2	334	Matthew	Jones	M	California	1977-01-01	2000-12-01	Sales	8000
3	449	Jessica	Jones	F	California	1986-01-02	2006-05-01	Production	5000
4	207	Jacob	Smith	M	Tennessee	1972-01-03	2010-12-01	Sales	5000
5	294	Madison	Johnson	F	Michigan	1986-01-04	2010-06-01	Production	10000
…	…	…	…	…	…	…	…	…	…

11.4.1 Sorting

The **null value sorting**: when sorting by a specified field, if there is a null value in the field, the record with the null value can be put at the beginning or end.

Table sequence data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	(null)
4	4	Gail	M	11

Sort by AGE in ascending order(the record with null value at the end of the record sequence)

Index	SID	NAME	GENDER	AGE↑
1	4	Gail	M	11
2	1	Jack	M	12
3	2	Tom	M	15
4	3	July	F	(null)

Sort by AGE in ascending order(the record with null value at the beginning of the record sequence)

Index	SID	NAME	GENDER	AGE↑
1	3	July	F	(null)
2	4	Gail	M	11
3	1	Jack	M	12
4	2	Tom	M	15

When sorting in ascending order, the position of the record with null value is shown in the above.
Similarly, in the case of sorting in descending order, there is a similar situation, which will not be repeated here.

11.4.1 Sorting

【Example 11.25】Sort by AGE in ascending order, and check the record with null value at the beginning or end. (Using data from user_12.txt)

Solution:
A.sort(x) sorts table by the field, -x represents sorting in descending order, the record with null value is put at the beginning by default; with **@0** option the record with null value is put at the end.

	A	B
1	=file("txt/user_12.txt").import@t()	/Load text data and return a table sequence
2	=A1.sort(AGE)	/Sort by age in ascending order, and the record with null value is at the beginning
3	=A1.sort@0(AGE)	/Sort by age in ascending order, and the record with null value is at the end

A1

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	(null)
3	3	July	F	14
4	4	Gail	M	11
5	5	Liky	F	10

A2

Index	SID	NAME	GENDER	AGE
1	2	Tom	M	(null)
2	5	Liky	F	10
3	4	Gail	M	11
4	1	Jack	M	12
5	3	July	F	14

A3

Index	SID	NAME	GENDER	AGE
1	5	Liky	F	10
2	4	Gail	M	11
3	1	Jack	M	12
4	3	July	F	14
5	2	Tom	M	(null)

11.4.2 Sorting and positioning

Get the sequence numbers in the original record sequence of sorted members. The following demonstrates the original sequence numbers of sorted letter sequence and sorted records by AGE.

The sequence numbers in the original sequence of sorted letters

Char	a	c	e	g	f	d	b
Sequence number	1	2	3	4	5	6	7
	After sorting						
Char	a	b	c	d	e	f	g
Sequence number	1	7	2	6	3	5	4

Table data

Index	SID	NAME	GENDER	AGE
1	1	Jack	M	12
2	2	Tom	M	15
3	3	July	F	13
4	4	Gail	M	11
5	5	Liky	F	10

sorting →

Sort by AGE

Index	SID	NAME	GENDER	AGE↑
1	5	Liky	F	10
2	4	Gail	M	11
3	1	Jack	M	12
4	3	July	F	13
5	2	Tom	M	15

Return sequence numbers →

Original sequence Index

Original Index
5
4
1
3
2

11.4.2 Sorting and positioning

【Example 11.26】Find the sequence numbers of employees from Ohio State sorted by DEPT and HIREDATE in the original table. (Using data from emp.txt)

Solution:
Get the sequence numbers of employees in Ohio State in the original sequence after sorting, and then generate a new table sequence according to the sequence numbers, and add the sequence numbers to the new table sequence. In **A.psort(x)**, x is an expression to obtain the sequence numbers of sorted records in the original sequence.

	A	B
1	=file("txt/emp.txt").import@t().select(STATE=="Ohio")	/Load text data and return a record sequence
2	=A1.psort(DEPT, HIREDATE)	/Get the sequence numbers of records sorted by DEPT and HIREDATE in ascending order
3	=A1(A2).derive(A2(#): ROW)	/Append the sequence numbers in A2 to the new table sequence

A3

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY	ROW
1	219	Emily	Johnson	F	Ohio	1986-10-25	2007-09-01	Finance	6500	6
2	108	William	Brown	M	Ohio	1978-03-25	2000-10-01	Marketing	6500	2
3	494	Emily	Johnson	F	Ohio	1975-08-12	2001-12-01	Marketing	8000	22
4	488	Hannah	Smith	F	Ohio	1973-04-06	2002-06-01	Marketing	8000	20
5	268	Jasmine	Johnson	F	Ohio	1977-01-09	2008-10-01	Marketing	10000	8
…	…	…	…	…	…	…	…	…	…	…

A3: **A(p)** generates a table sequence according to the sequence numbers in A2, and **A.derive()** adds a field ROW to record the corresponding sequence numbers.

11.4.3 Ranking

Now various kinds of rankings are popular, such as student score ranking list, pop song ranking list, automobile sales monthly ranking list, etc. through the ranking list, we can understand the **ranking** situation. For example, through the monthly automobile sales ranking list, we know that in April 2020, the sales volume of Volkswagen Bora cars was 23731, ranking the sixth. Just like sorting, ranking can be in ascending and descending orders. Usually, the default ranking is in ascending order.

Rank by age in ascending order

Index	SID	NAME	AGE	Rank↑
1	1	Jack	12	3
2	2	Tom	15	4
3	3	July	11	1
4	4	Gail	11	1

Rank by age in descending order

Index	SID	NAME	AGE	Rank↓
1	1	Jack	12	2
2	2	Tom	15	1
3	3	July	11	3
4	4	Gail	11	3

Note: To demonstrate the ranking, a description field is added on the basis of the source table sequence, which is marked in orange to indicate the ranking order of data records. In the following case, the source table sequence will not be listed separately.

11.4.3 Ranking

【Example 11.27】View the ascending and descending ranking of subject 1 scores. (Using data from scores.txt)

Solution:
A.ranks(x) can be used to rank the scores. **A.ranks(x)** calculates the ranking of x value corresponding to each record, and the default is in ascending order; **-x** indicates ranking of x in **descending** order; option **@z** indicates reverse order.

	A	B
1	=file("txt/scores.txt").import@t().select(Subject:1)	/Query subject 1
2	=A1.ranks(Score)	/Return ranking of score(from small to large)
3	=A1.ranks@z(Score)	/Return ranking of score(from large to small)
4	=A1.ranks(-Score)	/Return ranking of score(from large to small)

A1 Data table of scores

Index	Student	Subject	Score
1	10001	1	71
2	10002	1	87
3	10003	1	88
4	10004	1	53
5	10005	1	91
6	10006	1	92
…	…	…	…

A2 Ranking in ascending order

Index	Member↑
1	31
2	69
3	70
4	2
5	83
6	84
…	…

A3 Ranking in descending order

Index	Member↓
1	69
2	32
3	28
4	99
5	18
6	16
…	…

11.4.3 Ranking

Ranking and dense ranking

In normal ranking, if a certain rank n is tied up with m, the next following one is n + m. As shown on the left side of the figure, the user table is ranked in ascending order by age. July and Gail are tied for the second place, then Jack will be ranked fourth; while in dense ranking, the next one is n + 1, as shown on the right side of the figure, Jack ranks third. Generally, the ranking without special explanation is normal, such as the one mentioned above.

Ranking in ascending order by age

Index	SID	NAME	AGE	Rank↑
1	1	Jack	12	4
2	2	Tom	15	5
3	3	July	11	2
4	4	Gail	11	2
5	5	Liky	10	1

Dense ranking in ascending order by age

Index	SID	NAME	AGE	Rank↑
1	1	Jack	12	3
2	2	Tom	15	4
3	3	July	11	2
4	4	Gail	11	2
5	5	Liky	10	1

Ranking in descending order by age

Index	SID	NAME	AGE	Rank↓
1	1	Jack	12	2
2	2	Tom	15	1
3	3	July	11	3
4	4	Gail	11	3
5	5	Liky	10	5

Dense ranking in descending order by age

Index	SID	NAME	AGE	Rank↓
1	1	Jack	12	2
2	2	Tom	15	1
3	3	July	11	3
4	4	Gail	11	3
5	5	Liky	10	4

11.4.3 Ranking

【Example 11.28】The ascending and descending order dense ranking of commodity sales record table. (Using data from goods.txt)

Solution:
ranks@i(x) achieves dense ranking of x, which can be divided into ascending and descending order.

	A	B
1	=file("txt/goods.txt").import@t()	/Query subject 1
2	=A1.ranks@i(SALE)	/Dense ranking in ascending order
3	=A1.ranks@i(-SALE)	/Dense ranking in descending order

Data table of sales

A1

Index	SID	PDATE	SALE
1	101	2019-03-01	260
2	102	2019-03-02	184
3	103	2019-03-03	328
4	104	2019-03-04	143
5	105	2019-03-05	210
6	106	2019-03-06	328
7	107	2019-03-07	254
8	108	2019-03-08	239

Dense ranking in ascending order

A2

Index	Member↑
1	6
2	2
3	7
4	1
5	3
6	7
7	5
8	4

Dense ranking in descending order

A3

Index	Member↓
1	2
2	6
3	1
4	7
5	5
6	1
7	3
8	4

11.4.3 Ranking

In addition to ranking and dense ranking, there is also **average ranking**. In normal ranking, if a certain rank n is tied up with m, the rank of m members in average ranking is n+(m-1) / 2.

Ranking in ascending order by age

Index	SID	NAME	AGE	Rank
1	1	Jack	12	4
2	2	Tom	15	5
3	3	July	11	2
4	4	Gail	11	2
5	5	Liky	10	1

Average ranking by age

Index	SID	NAME	AGE	Rank
1	1	Jack	12	4.0
2	2	Tom	15	5.0
3	3	July	11	2.5
4	4	Gail	11	2.5
5	5	Liky	10	1.0

As shown in the figure above, July and Gail are tied for the second place in the normal ranking, while in the average ranking, the ranking is 2+(2-1)/2 = 2.5.

11.4.3 Ranking

【Example 11.29】Find the ranking of subject 1 scores under the average ranking. (Using data from scores.txt)

Solution:
ranks@s(x) calculates the ranking of the x value of each record under the average ranking, which may not be an integer.

	A	B
1	=file("txt/scores.txt").import@t().select(Subject:1)	/Import subject 1 scores
2	=A1.ranks@s(Score)	/Calculate the ranking under the average ranking

A1 Record sequence of subject 1 scores

Index	Student	Subject	Score
1	10001	1	71
2	10002	1	87
3	10003	1	88
4	10004	1	53
5	10005	1	91
6	10006	1	92
7	10007	1	69
8	10008	1	76
…	…	…	…

A2 Average ranking

Index	Member
1	31.5
2	69.0
3	71.5
4	2.0
5	83.0
6	84.5
7	27.0
8	48.0
…	…

11.4.3 Ranking

Sometimes we want to know the ranking of a given data in the ranking list. For example, the ranking of English score 90 in the class.

Table sequence data

Index	SID	NAME	SCORE	ASCENDING↑	DESCENDING↓
1	1	Jack	80	1	5
2	2	Tom	85	2	3
3	3	July	91	4	2
4	4	Gail	85	2	3
5	5	Liky	95	5	1

Ascending ranking of 90 in the table

Index	Member
1	4

Descending ranking of 90 in the table

Index	Member
1	3

Dense ranking of 90 in the table

Index	Member
1	3

Average ranking of 85 in the table

Index	Member
1	2.5

11.4.3 Ranking

【Example 11.30】In the score table, find the number of people whose score in subject 1 is higher than 90 points, and the number of people whose score in subject 1 is less than 90 points. (Using data from scores.txt)

Solution:
A.rank(y, x) calculates the ranking of y value among the x values of all records. The default is the ranking in ascending order, and the option **@z** is in reverse order.

	A	B	C
1	=file("txt/scores.txt").import@t().select(Subject:1)		/Query subject 1
2	=A1.rank(90, Score)	=A2-1	/Return the ranking by score (from small to large)
3	=A1.rank@z(90, Score)	=A3-1	/Return the ranking by score (from large to small)

A1	Index	Student	Subject	Score
	1	10001	1	71
	2	10002	1	87
	3	10003	1	88
	4	10004	1	53
	5	10005	1	91
	6	10006	1	92
	…	…	…	…

the number of people with score less than 90 points

B2	Value
	79

the number of people with score higher than 90 points

B3	Value
	18

11.4.3 Ranking

【Example 11.31】Find the ranking of each student's score in subject 1. (Using data from scores.txt)

Solution:
Construct a new table to store the data of student and score fields, including two new fields "Ascending" and "descending". Get the ranking in ascending order and descending order by **A.ranks()** and use **A.run()** to write the ranking result to the corresponding fields.

	A	B	C
1	=file("txt/scores.txt").import@t().select(Subject:1)	=A1.new(Student, Score, Ascending, Descending)	/B1 creates a new table sequence to store the ranking information
2	=A1.ranks(Score)	=A1.ranks@z(Score)	/Rank by score in ascending order and descending order respectively
3	>B1.run(Ascending=A2(#), Descending=B2(#))		/Write the ranking info to table sequence

B1	Index	Student	Score	Ascending↑	Descending↓
	1	10001	71	31	69
	2	10002	87	69	32
	3	10003	88	70	28
	4	10004	53	2	99
	5	10005	91	83	18
	6	10006	92	84	16
	…	…	…	…	…

11.4.3 Ranking

Similarly, **rank(x, y)** can be used to find the ranking of each record in the table sequence.

【Example 11.32】Find the ranking of each student's score in subject 1. (Using data from scores.txt)

Solution:
Construct a new table to store the data of student and score fields, including two new fields "Ascending" and "descending". Get the ranking in ascending order and descending order by **A.rank(x, y)** and use **A.run()** to write the ranking result to the corresponding fields.

	A	B
1	=file("txt/scores.txt").import@t().select(Subject:1)	/Import the scores data of subject 1
2	=A1.new(Student, Score, Ascending, Descending)	/create a new table sequence to store the ranking information
3	>A2.run(Ascending=A2.rank(~.Score, Score), Descending=A2.rank@z(~.Score, Score))	/Calculate the ranking of the score of the current record in all A2 scores

A2	Index	Student	Score	Ascending↑	Descending↓
	1	10001	71	31	69
	2	10002	87	69	32
	3	10003	88	70	28
	4	10004	53	2	99
	5	10005	91	83	18
	…	…	…	…	…

~.Score in A3 refers to the score value of the current record in A2 from "**A2.run(...)**".
The Score after it refers to the Score column in A2 from "**A2.rank(...)**".

Chapter 11
Calculation of table

11.5 Max/Min value

11.5.1 Max/Min value

Max/Min value: Find the maximum or minimum value from the data table.

To find the maximum value in a set, you will often encounter the following situations: :
A. find the maximum **value**
B. find the **sequence number** of the record where the maximum value is located.
C. find the **record** where the maximum value is located.

In the same way, the minimum value will be similar to the above several cases.

Index	Student	Score
1	10047	94
2	10004	96
3	10026	72
4	10037	89
5	10073	94
6	10068	71
7	10054	86

Results of the highest score under the above three operations:

A returns value:

Value
96

B returns sequence number:

Sequence number
2

C returns record:

Student	Score
10004	96

11.5.1 Max/Min value

A. Find maximum **value**

【Example 11.33】Find the highest score of subject 1. (Using data from scores.txt)

Solution:
Find the maximum value from the score field. **A.max(x)** calculates the maximum value of x values from all records in sequence A. To find the maximum value in a table sequence or record sequence is similar to finding the maximum value in a sequence in Chapter 9.

	A	B
1	=file("txt/scores.txt").import@t().select(Subject==1)	/Load subject 1 score data
2	=A1.max(Score)	/99, return the maximum value

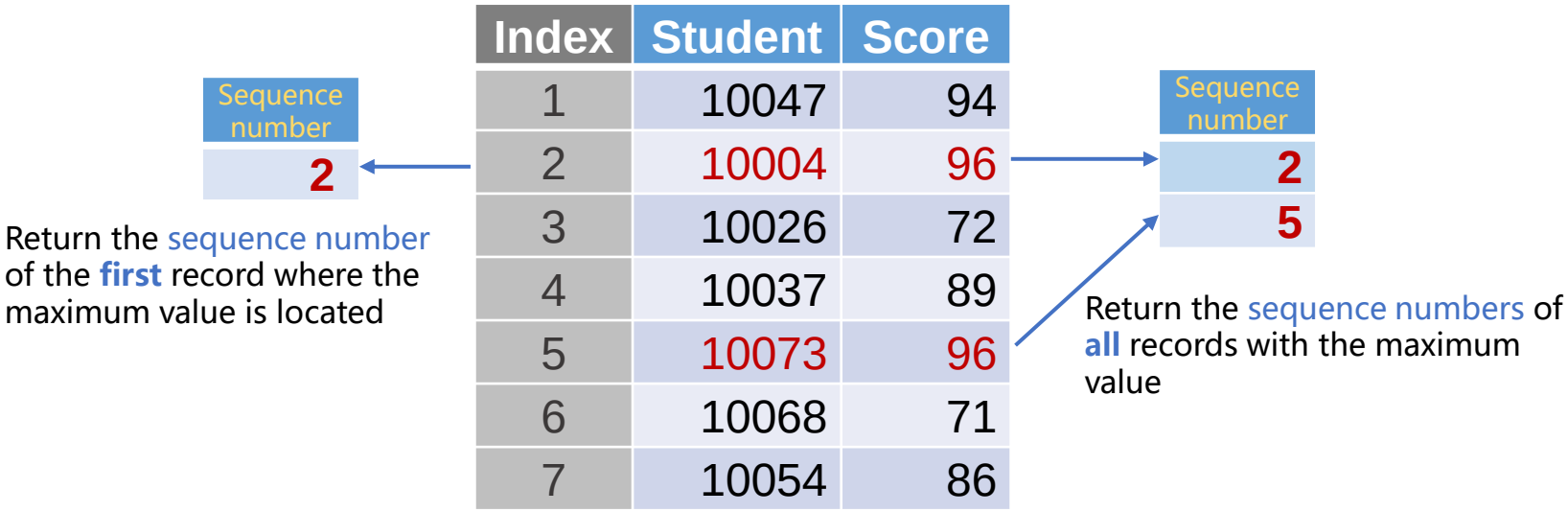
Return the max value:

A2	Value
	99

11.5.1 Max/Min value

B. find the **sequence number** of the record where the maximum value is located.

Find the sequence number of the record where highest score is located.



11.5.1 Max/Min value

【Example 11.34】Based on the daily sales record table of goods, calculate the increase rate of the records with the largest sales volume. (Using data from goods.txt)

Solution:

Because the sales records are sorted by date, **A.pmax (x)** can obtain the record sequence number with the largest sales volume, get the record corresponding to the last sales through the sequence number, and then calculate the increase rate. **A.pmax(x)** calculates the sequence number of the record where the maximum value of x is located. By default, it returns the sequence number of the first record with the maximum value, and the option @a returns the sequence numbers of all the records with the maximum value.

	A	B	C
1	=file("txt/goods.txt").import@t()	=create(SID, SALE, RATE)	/Loading sales data in May
2	=A1.pmax@a(SALE)		/Get the sequence numbers of all records with the largest sales volume
3	=A1.calc(A2, (SALE/SALE[-1]-1))		/Calculate growth rate
4	for A2	>B1.insert(0, A1(A4).SID, A1(A4).SALE, round(100*A3(A2.pos(A4)), 2)/"%")	/The relevant data is saved as a table sequence, and round() keeps the growth rate to two decimal places

A1

Index	SID	PDATE	SALE
1	101	2019-03-01	260
2	102	2019-03-02	184
3	103	2019-03-03	328
4	104	2019-03-04	143
5	105	2019-03-05	210
…	…	…	…

A2

Index	Member
1	3
2	6

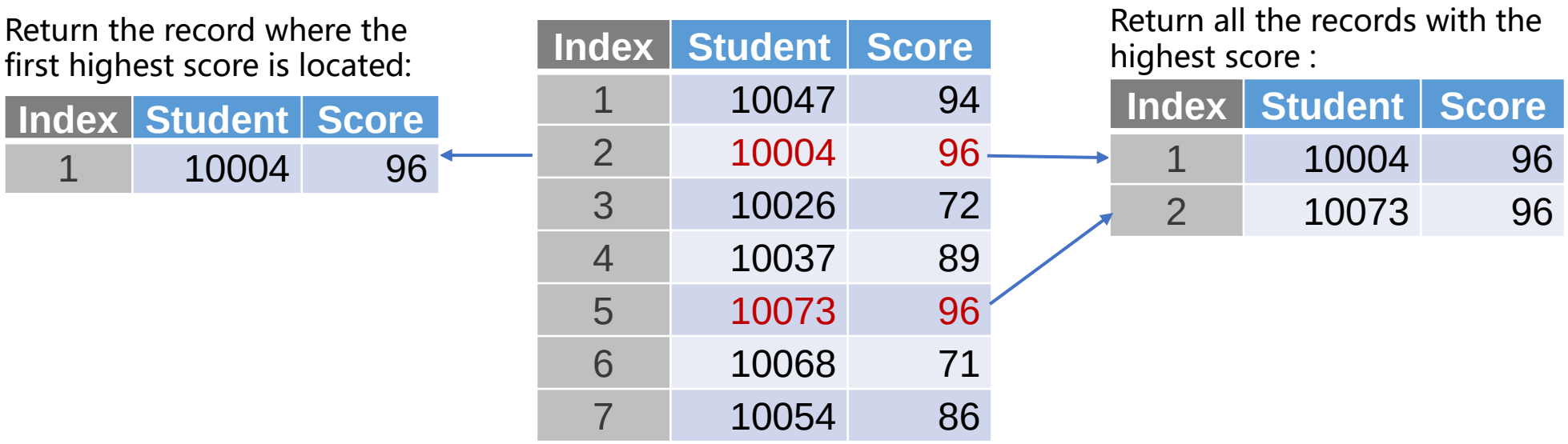
B1

Index	SID	SALE	RATE
1	103	328	78.26%
2	106	328	56.19%

11.5.1 Max/Min value

C. find the **record** where the maximum value is located.

Find the record where the highest score is located:



11.5.1 Max/Min value

【Example 11.35】Find the student IDs of the best score in subject 1. (Using data from scores.txt)

Solution:
Use **A.maxp()** to get the record with the highest score. **A.maxp (x)** finds the record where the maximum value of x is located. By default, the record with the first maximum value of x is returned. The option **@a** returns all the records with the maximum value of x.

	A	B
1	=file("txt/scores.txt").import@t().select(Subject:1)	/Load subject 1 score data
2	=A1.maxp@a(Score)	/Return all the records with the highest score
3	=A2.(Student)	/Return the student IDs of the best score

A1

Index	Student	Subject	Score
1	10001	1	71
2	10002	1	87
3	10003	1	88
4	10004	1	53
5	10005	1	91
…	…	…	…

The records with the highest score

A2

Index	Student	Subject	Score
1	10054	1	99
2	10066	1	99
3	10068	1	99
4	10087	1	99

A3

Index	Member
1	10054
2	10066
3	10068
4	10087

11.5.1 Max/Min value

To find the **minimum** value is similar to finding the maximum value:

A. A.min(x) finds the minimum value of x.
B. A.pmin(x) finds the sequence number of the record where the minimum value of x is located.
C. A.minp(x) finds the record where the minimum value of x is located.

Find the lowest score in the table:

	A	B
1	=file("zh/txt/scores.txt").import@t().select(Subject==1)	/Load subject 1 score data
2	=A1.min(Score)	/Return the lowest score
3	=A1.pmin(Score)	/Return the sequence number of the first record with the lowest score
4	=A1.minp(Score)	/Return the first record with the lowest score

11.5.2 TOP

TOP-N: Sometimes we don't care about the ranking of all members, especially when the amount of data is large, we usually only focus on the top (last) N members.

To find the maximum N members in a set, you will also encounter the following situations:
A. find the values of the maximum n members.
B. Find the records corresponding to the maximum n members.
C. Find the sequence numbers of the maximum n members.

In the same way, finding the minimum n members will be similar to the above situations.

Index	Student	Score
1	10047	94
2	10004	96
3	10026	72
4	10037	89
5	10073	94
6	10068	71
7	10054	86

The results of the top three best scores under the above three operations:

A returns values:

Value
96
94
94

B returns a record sequence of records

Index	Student	Score
1	10004	96
2	10047	94
3	10073	94

C returns sequence numbers:

Sequence number
2
1
5

11.5.2 TOP

A. find the **values** of the maximum(minimum) n members in the set.

After sorting by score field, the first five score values are obtained:

Top 5 worst scores

Index	Score↑
1	71
2	72
3	86
4	86
5	89

Original data table

Index	Student	Score
1	10047	94
2	10004	96
3	10026	72
4	10037	89
5	10073	94
6	10068	71
7	10054	86
8	10034	86

Top 5 best scores

Index	Score↓
1	96
2	94
3	94
4	89
5	86

When getting the **largest top n** values, they are sorted in **descending** order; when getting the **smallest top n** values, they are sorted in **ascending** order.

11.5.2 TOP

【Example 11.36】Find top 5 scores in subject 1. (Using data from scores.txt)

Solution:
Use **A.top(n, x)** to find top 5 scores.
A.top(n, x), n=0, return null. n>0, calculate the smallest n x values of all records. n<0, calculate the largest n x values of all records. n can't be omitted; if x is omitted, it is interpreted as ~.

	A	B
1	=file("txt/scores.txt").import@t().select(Subject==1)	/Load subject 1 score data
2	=A1.top(-5, Score)	/Return the top 5 best scores

Best scores

A2	Index	Member↓
	1	99
	2	99
	3	99
	4	99
	5	98

Note: When there are multiple identical maximum values, all of them are not necessarily returned. In the above example, if the number of score 99 exceeds 5, only the first 5 are returned.

11.5.2 TOP

B. Find the **records** corresponding to the maximum n members in the set.

After sorting by score field, the first five records are obtained:

Records of top 5 worst scores

Index	Student	Score↑
1	10068	71
2	10026	72
3	10034	86
4	10054	86
5	10037	89

Original data table

Index	Student	Score
1	10047	94
2	10004	96
3	10026	72
4	10037	89
5	10073	94
6	10068	71
7	10054	86
8	10034	86

Records of top 5 best scores

Index	Student	Score↓
1	10004	96
2	10047	94
3	10073	94
4	10037	89
5	10054	86

When getting the records of the **largest** top n values, they are sorted in **descending** order. When getting the records of the **smallest** top n values, they are sorted in **ascending** order.

11.5.2 TOP

【Example 11.37】Find the records of top 5 best scores and worst scores in subject 1 respectively. (Using data from scores.txt)

Solution:
A.top(n; x) can be used to obtain the records corresponding to the top n members.
A.top(n; x) is similar to A.top(n, x), but the value of x is not returned, but the corresponding **record**.

	A	B
1	=file("txt/scores.txt").import@t().select(Subject==1)	/Load subject 1 score data
2	=A1.top(5; Score)	/Return the records with top 5 worst scores
3	=A1.top(-5; Score)	/Return the records with top 5 best scores

records with top 5 **worst** scores:

A2

Index	Student	Subject	Score↑
1	10047	1	50
2	10004	1	53
3	10026	1	57
4	10037	1	60
5	10073	1	60

records with top 5 **best** scores:

A3

Index	Student	Subject	Score↓
1	10066	1	99
2	10087	1	99
3	10068	1	99
4	10054	1	99
5	10034	1	98

11.5.2 TOP

C. Find the **sequence numbers** of the largest (smallest) n members in the set so that they can be used for other calculations.

After sorting by Score field, the sequence numbers of the first five records are obtained:

Sequence numbers of records with top 5 worst scores

Student	Score↑	ROW
10068	71	6
10026	72	3
10034	86	8
10054	86	7
10037	89	4

Original data table

Index	Student	Score
1	10047	94
2	10004	96
3	10026	72
4	10037	89
5	10073	94
6	10068	71
7	10054	86
8	10034	86

Sequence numbers of records with top 5 best scores

ROW	Student	Score↓
2	10004	96
1	10047	94
5	10073	94
4	10037	89
8	10054	86

When getting the sequence numbers of records with the **largest** top n values, they are sorted in **descending** order.
When getting the sequence numbers of records with the **smallest** top n values, they are sorted in **ascending** order.

11.5.2 TOP

【Example 11.38】Based on the daily sales records of goods, calculate the increments of the top three sales compared with that of the previous day. (Using data from goods.txt)

Solution:
Obtain the sequence numbers of the first three records with the largest sales volume, and then calculate the increments compared with the previous day through the sequence number. To get the record sequence number, use **ptop()**.
A.ptop(n, x) is similar to A.top(n, x), but does not return the x value, but the record sequence number corresponding to the x value.

	A	B
1	=file("txt/goods.txt").import@t().sort(PDATE)	/Import data
2	=A1.ptop(-3, SALE)	/Get the sequence numbers of the first three records with the largest sales volume
3	=A1.calc(A2, (SALE-SALE[-1]))	/Calculate increments

Sequence numbers of records with top3 sales volume

A2	Index	Member
	1	6
	2	3
	3	1

Sales increments

A3	Index	Member
	1	118
	2	144
	3	260

11.5.2 TOP

Multi Max/Min records: There may be more than one maximum (or minimum) value in the records, and all maximum (or minimum) values need to be listed.

Based on the score table, the records with Max/Min value are obtained:

Records with Min score

Index	Student	Score↑
1	10068	71
2	10034	71

Original data table

Index	Student	Score
1	10047	94
2	10004	96
3	10026	72
4	10037	89
5	10073	96
6	10068	71
7	10054	86
8	10034	71

Records with Max score

Index	Student	Score↓
1	10004	96
2	10073	96

For example, the records with the best (worst) score of a subject are more than one.

11.5.2 TOP

【Example 11.39】List the records with the best and worst score of subject 1 respectively. (Using data from scores.txt)

Solution:
A.top(n; x) returns **all the records** corresponding to the minimum value or maximum value when n is ±1.

	A	B
1	=file("txt/scores.txt").import@t().select(Subject==1)	/Load subject 1 score data
2	=A1.top(1; Score)	/Return the records with the worst score
3	=A1.top(-1; Score)	/Return the records with the best score

Worst score data :

A2

Index	Student	Subject	Score↑
1	10047	1	50

Best score data :

A3

Index	Student	Subject	Score↓
1	10066	1	99
2	10087	1	99
3	10068	1	99
4	10054	1	99

Chapter 11
Calculation of table

11.6 Exercises

11.6 Exercises

【Exercise 11.1】Find the number of employees with more than 10 years of service in HR department. (Using data from emp.txt)

Solution:
The length of service can be obtained according to the HIREDATE, and **A.count(x)** can filter the data and count the data according to the conditions.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.count(DEPT=="HR" && age(HIREDATE)>10)	/Find the number of employees with more than 10 years of service in HR department

Output number of employees:

A2	Value
	14

11.6 Exercises

【Exercise 11.2】Find employees who joined the company when less than 20 years old. (Using data from emp.txt)

Solution:
Find employees with the difference between hire date and birth date less than 20 years. **A.select(x)** supports expression queries.

	A	B
1	=file("txt/emp.txt").import@t()	/Load text data and return a table sequence
2	=A1.select(elapse@y(BIRTHDAY, 20)>=HIREDATE)	/Filter

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY
1	3	Rachel	Johnson	F	New Mexico	1970-12-17	2010-12-01	Sales	9000
2	4	Emily	Smith	F	Texas	1985-03-07	2006-08-15	HR	7000
3	6	Matthew	Johnson	M	California	1984-07-07	2005-07-07	Sales	11000
4	7	Alexis	Smith	F	Illinois	1972-08-16	2002-08-16	Sales	9000
5	9	Victoria	Davis	F	Texas	1983-12-07	2009-12-07	HR	3000
…	…	…	…	…	…	…	…	…	…

Use **elapse@y(F, n)** to calculate the date after n years of the birth date.

11.6 Exercises

【Exercise 11.3】Based on the daily sales record table, find the daily sales records when the cumulative sales volume reaches 1000. (Using data from goods.txt)

Solution:
Accumulate the sales data in the table. If the cumulative sales volume exceeds 1000, it will be stopped. Then, the records constituting the sales volume of 1000 will be obtained according to the returned sequence numbers.
A.pselect(x) is used to implement the condition.

	A	B
1	=file("txt/goods.txt").import@t()	0
2	=A1.pselect((B1=B1+SALE, B1>1000))	/The return value is 6
3	=A1.to(A2)	/Get the first 6 records

A2

Index	SID	PDATE	SALE
1	101	2019-03-01	260
2	102	2019-03-02	184
3	103	2019-03-03	328
4	104	2019-03-04	143
5	105	2019-03-05	210
6	106	2019-03-06	328
7	107	2019-03-07	254
8	108	2019-03-08	239

A3

Index	SID	PDATE	SALE
1	101	2019-03-01	260
2	102	2019-03-02	184
3	103	2019-03-03	328
4	104	2019-03-04	143
5	105	2019-03-05	210

A2: if the cumulative sales volume reaches 1000, the current record sequence number will be returned.

11.6 Exercises

【Exercise 11.4】Number the clothes according to the style number. The expected result is shown in the No field in the figure below. (Using data from clothing.txt)

ModelNumber	ColorCode	SizeCode	Num	No
LN02960	1020	004	1	1
LN02960	1020	005	2	1
LN02960	1020	006	2	1
LN02960	1020	007	2	1
LN02960	1020	008	3	1
LN02960	1020	010	2	1
LN02960	1020	012	1	1
LN02960	1020	014	1	1
LN03194	1051	003	1	2
LN03194	1051	004	1	2
LN03194	1051	005	2	2
LN03194	1051	006	2	2
LN03194	1051	007	2	2
LN03194	1051	008	2	2
LN03194	1051	010	2	2
LN03194	1051	009	1	2
LN03184	1051	003	1	3
LN03184	1051	004	1	3
LN03184	1051	005	2	3

Solution:
Clothes are numbered according to the ModelNumber order. Because the styleNo is repeated, we need to sort them by ModelNumber first, and then use **A.ranks@i(x)** to do a dense ranking to get the number of each style clothes

	A	B
1	=file("clothing.txt").import@tq().sort(ModelNumber)	/Load text data, sort by StyleNo and return
2	=A1.derive(A1.ranks@i(ModelNumber)(#):No)	/Write the ranking to a new table after dense ranking

11.6 Exercises

【Exercise 11.5】Sort the score table by subject and score. (Using data from scores.txt)

Solution:
Use **A.sort(x_i, ...)** to sort records by subject in ascending order and by score in descending order.

	A	B
1	=file("txt/scores.txt").import@t()	/Load score data
2	=A1.sort(Subject, -Score)	/Sort records by subject in ascending order and by score in descending order

A1

Index	Student	Subject	Score
1	10001	1	71
2	10001	2	81
3	10001	3	81
4	10001	4	83
5	10001	5	77
6	10002	1	87
7	10002	2	67
8	10002	3	79
9	10002	4	96
…	…	…	…

A2 sorting

A2

Index	Student	Subject	Score
1	10005	1	91
2	10003	1	88
3	10002	1	87
4	10001	1	71
5	10004	1	53
6	10003	2	97
7	10001	2	81
8	10004	2	79
9	10002	2	67
…	…	…	…

11.6 Exercises

【Exercise 11.6】Based on the product table, for the same NO, the PID is increased in turn every five records, and the PID is also increased when NO is different. The expected result is shown in the PID field below. (Using data from product.txt)

NAME	NO	PID	cnt
P3C0001	90560	P3CA001	0
P3C0002	90560	P3CA001	1
P3C0003	90560	P3CA001	2
P3C0004	90560	P3CA001	3
P3C0005	90560	P3CA001	4
P3C0006	90560	P3CA002	0
P3C0007	90560	P3CA002	1
P3C0008	90560	P3CA002	2
P3C0009	90560	P3CA002	3
P3C0010	90560	P3CA002	4
P3C0011	90560	P3CA003	0
P3C0012	90560	P3CA003	1
P3C0013	90561	P3CA004	0
P3C0014	90561	P3CA004	1
P3C0015	90561	P3CA004	2
P3C0016	90561	P3CA004	3
P3C0017	90561	P3CA004	4
P3C0018	90561	P3CA005	0
P3C0019	90561	P3CA005	1

Solution:
Generate the PID field: the records with the same NO are grouped into one group for every five records, and at last, less than five records are divided into a group. Therefore, the cnt value is calculated in turn during processing. When cnt is 0, the cumulative count is carried out, and the string composed of count and "P3CA" is saved as a new PID.
A.derive() can be used to add PID field to create a new table, and A.run() can investigate the change of NO and store the incremental count into PID.

	A	B
1	=file("product.txt").import@tq().sort(NO)	/Load text data, sort it and return
2	=A1.derive(PID)	/Append PID field to new table
3	>cnt=0, idx=0	
4	>A2.run(a= (if(NO==NO[-1], (cnt=cnt+1)%5, cnt=0)), PID=format("P3CA%03d", (if (a==0, idx=idx+1, idx))))	/Calculate remainder count

Note: During debugging, we can use derive() to add an auxiliary TMP field to store the temporary variable a in A3, which is helpful to observe and analyze problems.

11.6 Exercises

【Exercise 11.6】Based on the product table, for the same NO, the PID is increased in turn every five records, and the PID is also increased when NO is different. The expected result is shown in the PID field below. (Using data from product.txt)

In the following, A4 is decomposed into two steps, and TMP field is added to illustrate how to obtain PID value.

NAME	NO	TMP	PID
P3C0001	90560	0	P3CA001
P3C0002	90560	1	P3CA001
P3C0003	90560	2	P3CA001
P3C0004	90560	3	P3CA001
P3C0005	90560	4	P3CA001
P3C0006	90560	0	P3CA002
P3C0007	90560	1	P3CA002
P3C0008	90560	2	P3CA002
P3C0009	90560	3	P3CA002
P3C0010	90560	4	P3CA002
P3C0011	90560	0	P3CA003
P3C0012	90560	1	P3CA003
P3C0013	90561	0	P3CA004
P3C0014	90561	1	P3CA004
P3C0015	90561	2	P3CA004
P3C0016	90561	3	P3CA004
P3C0017	90561	4	P3CA004
P3C0018	90561	0	P3CA005
P3C0019	90561	1	P3CA005

2	=A1.derive(TMP, PID)	/Append fields PID and TMP
3	>cnt=0, idx=0	/cnt counts, idx counts
4	>A2.run(TMP= (if(NO==NO[-1], (cnt=cnt+1)%5, cnt=0)))	/Calculate remainder count
5	>A2.run(PID=format("P3CA%03d", (if (TMP==0, idx=idx+1, idx))))	

Decompose A4 into two steps:

1. calculate the remainder of 5. In A4, when the adjacent NO is the same, the count variable cnt is added by 1, and cnt is written into TMP by the remainder of 5. When NO is different, cnt is reset to 0 and written into TMP. The result of calculating the remainder is shown in the TMP column.

2. Form a string. In A5, PID data is generated according to TMP value. When the TMP value is 0, the count variable IDX is accumulated by 1, otherwise the IDX value remains unchanged, IDX and "P3CA" form a new string to be written to PID. The result is shown in the PID column.

11.6 Exercises

【Exercise 11.7】The hygiene performance of each class is as follows. Rank the hygiene score of each line from high to low and the add the class names. (Using data from hygiene.txt)

Date	Class1	Class2	Class3	Class4	Class5
2020-01-02	89	82	95	93	96
2020-01-03	93	90	98	93	97
2020-01-04	95	94	95	99	92
2020-01-05	97	89	93	93	95
2020-01-06	94	96	95	94	91



Date	No1	No2	No3	No4	No5
2020-01-02	Class5(96)	Class3(95)	Class4(93)	Class1(89)	Class2(82)
2020-01-03	Class3(98)	Class5(97)	Class1(93)	Class4(93)	Class2(90)
2020-01-04	Class4(99)	Class1(95)	Class3(95)	Class2(94)	Class5(92)
2020-01-05	Class1(97)	Class5(95)	Class3(93)	Class4(93)	Class2(89)
2020-01-06	Class2(96)	Class3(95)	Class1(94)	Class4(94)	Class5(91)

Solution:
1. Get the class fields from the table sequence.
2. Combine scores with corresponding class fields to form sequence A.
3. Merge the date and sequence A and write it into a new table sequence.

11.6 Exercises

【Exercise 11.7】The hygiene performance of each class is as follows. Rank the hygiene score of each line from high to low and the add the class names. (Using data from hygiene.txt)

	A	B
1	=file("hygiene.txt").import@t().sort(Date)	/Load text data
2	=A1.fname().to(2,)	/Get the class fields in the table
3	=A1.((a=~.array().to(2,), b=a.psort@z(), b.(A2(~)/ "(" / a(~) / ")")))	/Combine score and class
4	=A3.conj(A1(#).Date|~)	/Concatenate date and data in A3
5	=create(Date, No1, No2, No3, No4, No5).record(A4)	/Fill the table sequence

A2

Index	Member
1	Class1
2	Class2
3	Class3
4	Class4
5	Class5

A3

Index	Member
1	[Class5(96), Class3(95), Class4(93), …]
2	[Class3(98), Class5(98), Class1(93), …]
3	[Class4(99), Class1(95), Class3(95), …]
4	[Class1(97), Class5(95), Class3(93), …]
5	[Class2(96), Class3(95), Class1(94), …]

A4

Index	Member
1	2020-01-02
2	Class5(96)
3	Class3(95)
4	Class4(93)
…	…

A5

Index	Date	No1	…
1	2020-01-02	Class5(96)	…
2	2020-01-03	Class3(98)	…
3	2020-01-04	Class4(99)	…
4	2020-01-05	Class1(97)	…
5	2020-01-06	Class2(96)	…

Note: In A3, a is the score sequence of the record, b is the sequence numbers of sorted scores in descending order. Then the score is combined with the class information by the sequence number.

11.6 Exercises

【Exercise 11.8】For the five orders with the largest sales amount of sales ID 10, find the order amount increase compared with the previous order.

Solution:
A.ptop() finds the record sequence numbers of the five orders with the largest sales amount. The returned sequence numbers can be used to obtain the records of the corresponding orders and the corresponding previous orders, so as to make corresponding calculations for the two records. A.(x) calculates the sequence number of the previous order record, and A.derive() stores the increment of the order.

	A	B	C
1	=file("txt/sales.txt").import@t().select(SELLERID==10)	=A1.sort(ORDERDATE)	/Load text data and return a record sequence
2	=B1.ptop(-5; AMOUNT)	=B1(A2)	/Find the sequence numbers of the five orders with the largest sales amount
3	=A2.(B1.m(~-1).AMOUNT)		/Find the previous order record by sequence number
4	=B2.derive(A3(#): PREMOUNT, AMOUNT-PREMOUNT: DIFF)		/Write relevant records to new table sequence

A4

Index	ORDERID	CLIENT	SELLERID	AMOUNT	ORDERDATE	PREMOUNT	DIFF
1	29	QHHW	10	29500.0	2012-12-05	21800.0	7700.0
2	361	HANAR	10	28500.0	2013-11-04	4900.0	23600.0
3	259	EGU	10	25600.0	2013-07-18	23700.0	1900.0
4	433	DNEDL	10	24200.0	2014-01-07	5390.0	18810.0
5	182	VILJX	10	23700.0	2013-05-05	7448.0	16252.0

Note: Through the sequence numbers of records, adjacent data can be obtained and the changes between them can be understood.

11.6 Exercises

【Exercise 11.9】Find the records of stock 600005 with maximum continuous rising days. (Using data from txt/stockrecords.txt)

Solution:
According to the daily stock price difference, use **A.pmax()** to find the position of the maximum consecutive rising days, and calculate the maximum days. **A.to()** obtains the records that constitute the data.

	A	B
1	=file("txt/stockrecords.txt").import@tq().select(STOCKID=="600005").sort(DATE)	/Load text data, sort by date, return a record sequence
2	>z=0, max=0	/Max records the maximum continuous rising days and z records the current continuous rising days
3	=A1.pmax(if(CLOSING-CLOSING[-1]>0,(z=z+1,if (z>max, max=z)), (if (z>max, max=z), z=0)))	/Calculate the max continuous rising days and return the sequence number of the max continuous rising days
4	=A1.to(A3-max, A3)	/Get the records

A5

Index	STOCKID	DATE	CLOSING
1	600005	2015-11-20	19.98
2	600005	2015-11-23	20.69
3	600005	2015-11-24	22.5
4	600005	2015-11-25	23.4
…	…	…	…

Note: In A3, the second **if** calculates the current continuous rise days, and the third **if** calculates the maximum continuous rising days. If there is no third if calculation, if the maximum continuous rise occurs in the last one of A1, it is impossible to set z to max.

Chapter 12
Order-related calculations

12.1 Ordered set

- The concept of ordered set
- Access by sequence number

12.2 Iterative function

12.3 Positioning calculations

- Locate members
- Locate max/min value
- Get the sequence numbers of members that meet the conditions

12.4 Sorting and alignment

- Sorting
- Alignment

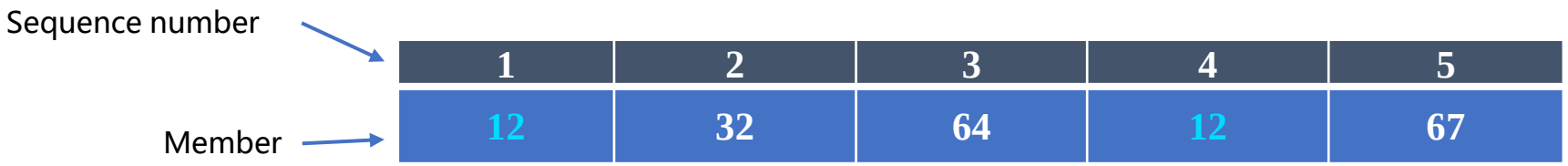
12.5 Exercises

Chapter 12
Order-related calculations

12.1 Ordered set

12.1.1 The concept of ordered set

An **ordered set** is a set of ordered data. It is equivalent to array or linked list in some programming languages. Members of an ordered set can be repeated.



12.1.1 The concept of ordered set

The difference between an **ordered set** and a **set (unordered)** is as follows:

- Members of an ordered set can be repeated, and members can be accessed by their sequence numbers. (note that in an ordered set the positions of members are ordered, but not necessarily sorted.)

- Members of a set (unordered) are not repeatable, can only be traversed, and cannot be accessed by sequence numbers.

12.1.2 Access by sequence number

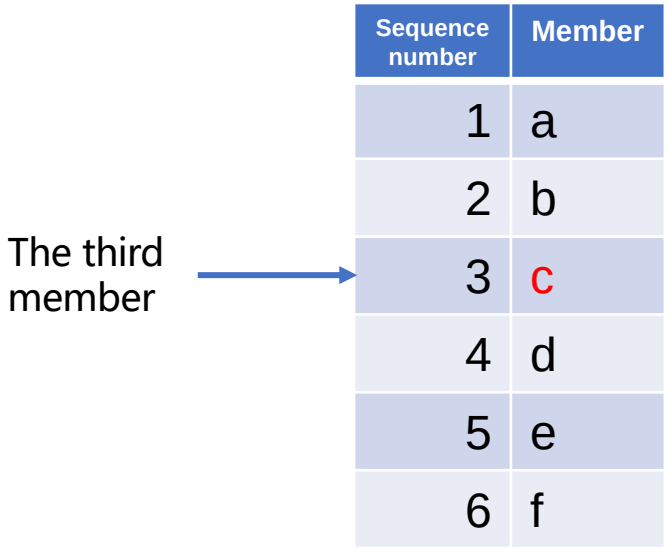
Access by sequence number refers to accessing members of an ordered set by their sequence numbers (index subscript).

What are the common types of sequence number access?

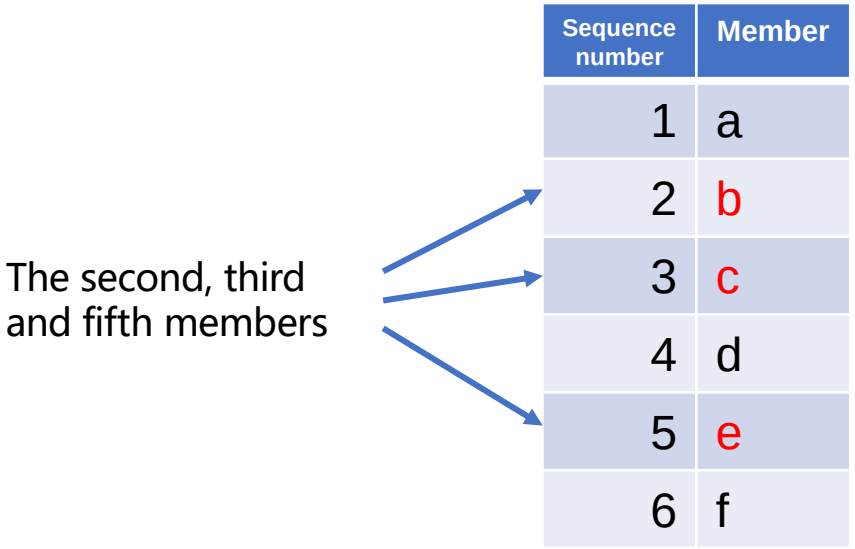
- Access by sequence number directly
- Access by sequence number range
- Fixed span access
- Average segment access

Direct sequence number access refers to accessing the members of an ordered set directly through a single or multiple sequence numbers.

- **Access with single sequence number**



- **Access with multi sequence numbers**





【Example 2.1】Query the closing prices of the first trading day and the last trading day of the Shanghai stock index in January 2019. The complete data of closing prices in January is as follows:

[2465.291, 2464.3628, 2514.8682, 2533.0887, 2526.4622, 2544.3449, 2535.0985, 2553.8313, 2535.7653, 2570.3449, 2570.4221, 2559.637, 2596.0056, 2610.5094, 2579.7041, 2581.0041, 2591.6935, 2601.7234, 2596.9767, 2594.2536, 2575.5756, 2584.5724]

【Example 2.1】Query the closing prices of the first trading day and the last trading day of the Shanghai stock index in January 2019. The complete data of closing prices in January is as follows:

1	2	3	…	20	21	22
2465.291	2464.3628	2514.8682	…	2594.2536	2575.5756	2584.5724

Solution

Sequence numbers can be used directly to access members of an ordered set.

	A	B	C	D
1	[2465.291, 2464.3628, …, 2575.5756, 2584.5724]			/Constant sequence of closing prices in January 2019
2	=A1(1)	=A1.m(1)	/A negative sequence number indicates in reverse order	/Access the first member
3	=A1(22)	=A1.m(22)	=A1.m(-1)	/Access the 22th(last) member
4	=A1([1, 22])	=A1.m([1, 22])	=A1.m([1, -1])	/Access by multi sequence numbers, the first and the 22th(last) members

Execution result:

2465.291

2584.5724

[2465.291, 2584.5724]

A1 defines a **constant sequence**. Sequence is an ordered set of data members. Sequence members are separated by commas and enclosed in brackets. Due to the space, the constant sequence in A1 has been partly omitted, and the complete data has been given in the question.

Access by sequence number directly

- **Single sequence number access**: To access the i^{th} member of ordered set A, A(i) or A.m(i) can be used.
- **Multi ssequence number access**: To access the $[i_1, i_2, ..., i_n]$ members of ordered set A, A($[i_1, i_2, ..., i_n]$) or A.m($[i_1, i_2, ..., i_n]$) can be used.
- A.m() function supports negative sequence numbers, indicating counting backwards. For example, A.m(-1) means the last member, and A.m([-1, -2]) means the last and second to last members.

$[i_1, i_2, ..., i_n]$ indicates the sequence composed of sequence numbers.

Sequence number range access refers to accessing members of an ordered set through sequence number interval.

Members 2 to 5

Sequence number	Member
1	a
2	b
3	c
4	d
5	e
6	f

The last one to three members

Sequence number	Member
1	a
2	b
3	c
4	d
5	e
6	f

【Example 12.2】Query the closing prices of the first three trading days and the last three trading days of the Shanghai stock index in January 2019. The complete data of closing prices in January is as follows:

[2465.291, 2464.3628, 2514.8682, 2533.0887, 2526.4622, 2544.3449, 2535.0985, 2553.8313, 2535.7653, 2570.3449, 2570.4221, 2559.637, 2596.0056, 2610.5094, 2579.7041, 2581.0041, 2591.6935, 2601.7234, 2596.9767, 2594.2536, 2575.5756, 2584.5724]

【Example 12.2】Query the closing prices of the first three trading days and the last three trading days of the Shanghai stock index in January 2019. The complete data of closing prices in January is as follows:

1	2	3	…	20	21	22
2465.291	2464.3628	2514.8682	…	2594.2536	2575.5756	2584.5724

Solution

Sequence number range can be used to access members of an ordered set.

	A	B	C	D
1	[2465.291, 2464.3628, …, 2575.5756, 2584.5724]			/Constant sequence of closing prices in January 2019
2	=A1.to(3)	=A1.m(1:3)	=A1.to(1, 3)	/Access the first three members
3	=A1.to(-3)	=A1.m(-3:-1)		/Access the last three members
4	=A1.m(1:3, -3:-1)			/Access the first three and last three members

Execution result:

[2465.291, 2464.3628, 2514.8682]

[2594.2536, 2575.5756, 2584.5724]

[2465.291, 2464.3628, 2514.8682, 2594.2536, 2575.5756, 2584.5724]

Query the closing prices from the 3rd to 5th trading days:

	A	B	C
2	=A1.to(3, 5)	=A1.m(3:5)	/Access members of interval 3 to 5
3	=A1.to(5, 3)		/Access members of interval 5 to 3, the order of members is opposite to that of A2

[2514.8682, 2533.0887, 2526.4622]

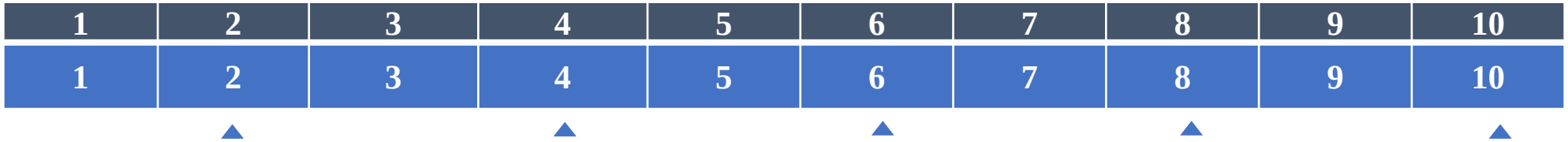
[2526.4622, 2533.0887, 2514.8682]

Access by sequence number range:

- First n / last n: for the first n members of the ordered set A, you can use **A.to(n)** to access; for the last n members, you can use **A.to (-n)** to access.
- Single interval access: for the a^{th} to b^{th} members of an ordered set A,
  When a<=b, we can use: **A.m(a:b)** and **A.to(a, b)**;
  When a>=b, we can also use: **A.m(a:b)** and **A.to(a, b)**, but the sequence number order of the returned members decreases from a to b.
- Multi interval access: **A.m()** function can be used to access members with multi sequence number intervals. For example: A.m(a:b, [c, d], e, ...) .

Fixed span access refers to accessing members according to the specified starting sequence number and fixed span.

For example, if you want to get the even numbers between 1 and 10, start with the second member, and get the numbers with a span of 2, and get [2, 4, 6, 8, 10]:



【Example 12.3】For a sequence of even numbers from 2 to 100, get the members of multiple of 5.

12.1.2 Access by sequence number

【Example 12.3】For a sequence of even numbers from 2 to 100, get the members of multiple of 5.

1	…	5	…	10	…	15	…	50
2	…	10	…	20	…	30	…	100

Solution

Access members by a fixed **span** to get multiples.

	A	B
1	=to(100).step(2, 2)	/Get members starting from the second member and with a span of 2
2	=A1.step(5, 5)	/Get members starting from the fifth member and with a span of 5

Execution result:

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, …, 100]

[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

Note

- **to(n)** function is used to create a sequence from 1 to n. to(100).step(2, 2) can be simplified to 100.step(2, 2).
- To access members of an ordered set A starting from sequence number k and with a fixed span m, **A.step(m, k)** function can be used.

Average segment access is to segment an ordered set evenly and access one segment of it.

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

Segment 1 Segment 2

When the ordered set can not be evenly segmented, the members number of subsequent segments will be 1 less than that of the previous segment:

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

Segment 1 Segment 2 Segment 3



【Example 12.4】For a sequence of even numbers from 2 to 100, get the members of the first fifth and the last fifth.

12.1.2 Access by sequence number

【Example 12.4】For a sequence of even numbers from 2 to 100, get the members of the first fifth and the last fifth.

1	…	10	11	…	40	41	…	50
2	…	20	22	…	80	82	…	100

Solution

The method of average segmentation can be used to obtain the members of an interval in an ordered set.

	A	B
1	=100.step(2, 2)	/The sequence of even numbers from 2 to 100
2	=A1.to@z(1, 5)	/Divide the ordered set evenly into 5 segments, and get the first segment
3	=A1.to@z(5, 5)	/Divide the ordered set evenly into 5 segments, and get the last segment

Execution result:

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

[82, 84, 86, 88, 90, 92, 94, 96, 98, 100]

Note

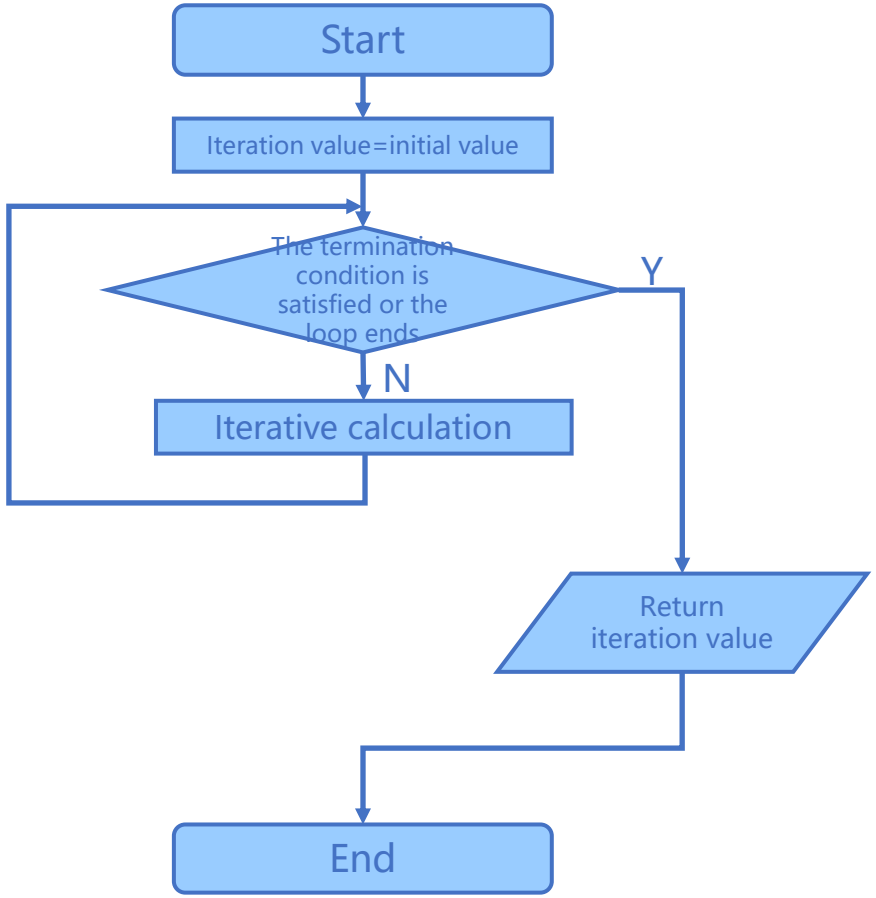
- **A.to@z(i, n)** function can be used to divide the ordered set A evenly into n segments and get the i^{th} segment.

Chapter 12
Order-related calculations

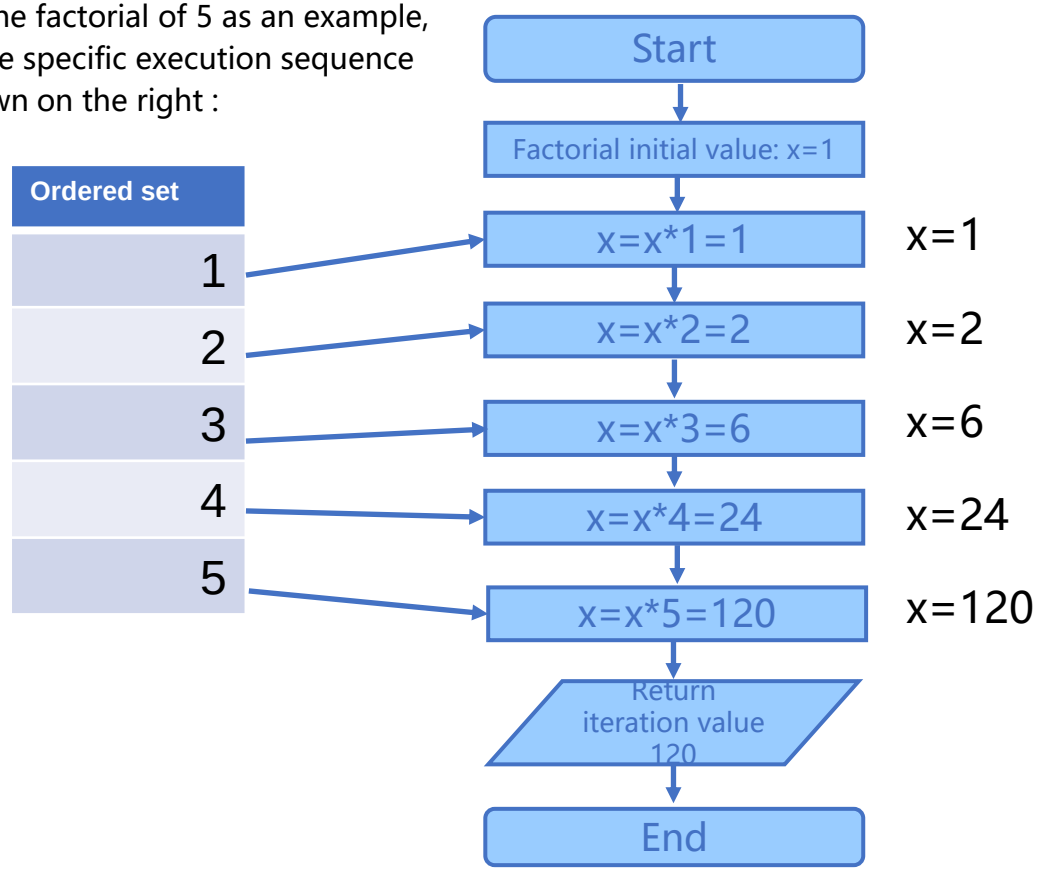
12.2 Iterative function

12.2 Iterative function

Iteration is the repetition of operations until certain conditions are met. The function used to implement iterative calculation is called **iterative function**.



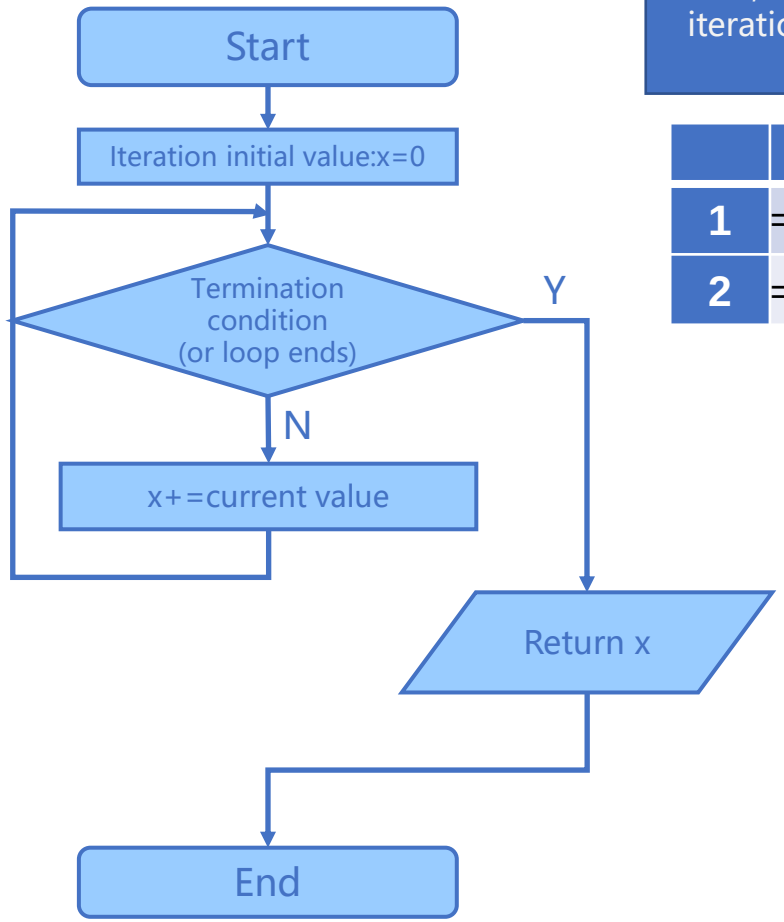
Take the factorial of 5 as an example, and the specific execution sequence is shown on the right :



【Example 12.5】Get the sum of arithmetic sequence: 1+3+5+7+9+13+…+99.

12.2 Iterative function

【Example 12.5】Get the sum of arithmetic sequence: 1+3+5+7+9+13+…+99.



Solution

First, use span fetching to create a sequence of odd numbers within 100. Then use the iteration function to calculate the sum of the iteration value and the current number in turn.

	A	B
1	=100.step(2, 1)	/Fetching member by span, get a sequence of odd numbers within 100
2	=A1.iterate(~~+~, 0)	/Iteration calculation accumulates the current value. The initial value is 0.

A2
2500

Function **A.iterate(x, a, c)** is used for iteration calculation.

The parameters and return value of A.iterate(x, a, c)

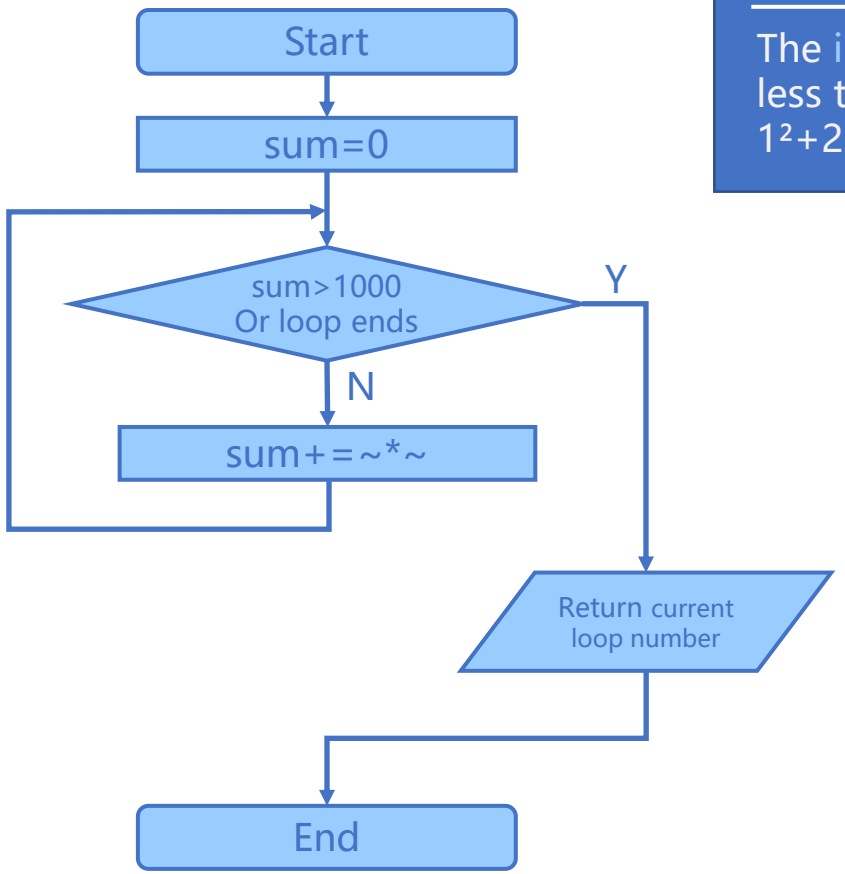
- **x**: An iterative expression containing ~~.
- **a**: Iteration initial value.
- **c**: The conditional expression of iteration termination. If omitted, it iterates to the end of the loop.
- ~~: Return value.

When the function is executed, the initial value is a before the iteration starts, and it is assigned to the iteration value ~~ . c is judged first each time. If it is false and the loop of A is not completed, then x is calculated and ~~ is used to save the result. If the iteration is completed or c is true, return ~~.

【Example 12.6】Find n so that the sum of squares from 1 to n is just over 1000.

12.2 Iterative function

【Example 12.6】Find n so that the sum of squares from 1 to n is just over 1000.



Solution

The initial value of iteration is 0. The iteration is repeated from 1 to 1000 (n must be less than 1000), and the square of the current value is accumulated successively. I.e. $1^2+2^2+...+n^2$, when the result is greater than 1000, the iteration is terminated.

	A	B
1	>sum=0	/Define the variable sum, which represents the sum of squares.
2	=1000.iterate((sum+=power(~,2), ~~=#), 0, sum>1000)	/Accumulate the sum of squares of each member repeatedly. End when the total is more than 1000, and return the current loop number.

A2
14

Note

- The returned value here is the sequence number, not the sum of squares. Therefore, another variable sum is defined to calculate the sum of squares and assign the current loop number # to ~~.

Chapter 12
Order-related calculations

12.3 Positioning calculations

12.3.1 Positioning calculations

Positioning calculation is to obtain the sequence numbers of members in an ordered set according to specified conditions.

What are the common types of Positioning calculations?

- Locate a member
- Locate the Max/Min value
- Get the sequence numbers of the members that satisfy the conditions

12.3.1 Positioning calculations

The previous section describes how to access **members** of an ordered set by **sequence numbers**. Locating members is the opposite, looking for **sequence numbers** by comparing **members** in an ordered set.

Locate an employee whose name is "Emily".

Emily's sequence number is 4

ID	NAME	BIRTHDAY	ENTRYDATE	DEPT	SALARY
1	Rebecca	1974/11/20	2005/03/11	R&D	7000
2	Ashley	1980/07/19	2008/03/16	Finance	11000
3	Rachel	1970/12/17	2010/12/01	Sales	9000
4	Emily	1985/03/07	2006/08/15	HR	7000
5	Ashley	1975/05/13	2004/07/30	R&D	16000

Compare each employee's name with "Emily" in turn, and return the sequence number if they are the same.

12.3.1 Positioning calculations

【Example 12.7】Find the sequence numbers of the score of 100. The scores are as follows:

[94, 97, 100, 96, 96, 92, 99, 87, 82, 92, 87, 89, 97, 90, 99, 95, 93, 100, 96, 98, 88, 90, 90, 88, 89, 97, 99, 93, 100, 82]

12.3.1 Positioning calculations

【Example 12.7】Find the sequence numbers of the score of 100.

1	2	3	…	28	29	30
94	97	100	…	93	100	82

Solution

Locate the member in the ordered set, compare the member and value 100 in turn, and return sequence number if the values are equal.

	A	B
1	[94,97,100,96,96,92,99,87,82,92,87,89,97,90,99,95,93,100,96,98,88,90,90,88,89,97,99,93,100,82]	/Constant sequence of scores
2	=A1.pos(100)	/Use pos() function to locate the sequence number

Execution result:

3

You can locate members from back to front:

	A	B
2	=A1.pos@z(100)	/Use @z option with pos() function to locate from back to front

29

There may be more than one 100:

	A	B
2	=A1.pos@a(100)	/Use the pos()@a function to locate the sequence numbers of all 100 points.

[3, 18, 29]

How to locate members:

- For an ordered set A, to locate member x, you can use the **A.pos(x)** function to get the sequence number of member x in the ordered set.
- Members of an ordered set can be repeated, and multiple sequence numbers may be returned when locating a member. **A.pos@a(x)** returns the sequence numbers of all x's. By default, only the **first** one is returned.
- By default, to locate a member is from front to back, and **A.pos@z(x)** can be used to locate in the reverse order.

12.3.1 Positioning calculations

Locating max / min value refers to looking for the sequence number where the max / min value of members is located.

Locate the employees with the highest and lowest salary:

ID	NAME	BIRTHDAY	ENTRYDATE	DEPT	SALARY
1	Rebecca	1974/11/20	2005/03/11	R&D	7000
2	Ashley	1980/07/19	2008/03/16	Finance	11000
3	Rachel	1970/12/17	2010/12/01	Sales	9000
4	Emily	1985/03/07	2006/08/15	HR	6500
5	Ashley	1975/05/13	2004/07/30	R&D	16000

The sequence number of the lowest salary is 4

The sequence number of the highest salary is 5

12.3.1 Positioning calculations

【Example 12.8】For the highest and lowest closing prices of the Shanghai Composite Index in 2019, calculate the growth rate. The complete data is as follows:

[2465.291, 2464.3628, 2514.8682, 2533.0887, 2526.4622, 2544.3449, 2535.0985, 2553.8313, 2535.7653, 2570.3449, 2570.4221, 2559.637, 2596.0056, 2610.5094, 2579.7041, 2581.0041, 2591.6935, 2601.7234, 2596.9767, 2594.2536, 2575.5756, 2584.5724, 2618.2323, 2653.896, 2671.8937, 2721.0683, 2719.6999, 2682.385, 2754.3566, 2755.6459, 2761.2189, 2751.8012, 2804.2262, 2961.2834, 2941.5163, 2953.8242, 2940.9538, 2994.005, 3027.5755, 3054.2466, 3102.0991, 3106.4179, 2969.8614, 3026.9925, 3060.3073, 3026.9513, 2990.685, 3021.7512, 3096.4176, 3090.975, 3090.6403, 3101.455, 3104.1487, 3043.0313, 2997.0954, 3022.7196, 2994.9434, 3090.758, 3170.361, 3176.8227, 3216.2952, 3246.5714, 3244.8103, 3239.6639, 3241.9299, 3189.9619, 3188.6256, 3177.7866, 3253.5978, 3263.1179, 3250.2012, 3270.7973, 3215.0426, 3198.5938, 3201.6137, 3123.8261, 3086.3989, 3062.4985, 3078.3389, 2906.464, 2926.3908, 2893.7567, 2850.9537, 2939.2106, 2903.7131, 2883.6107, 2938.6781, 2955.711, 2882.2962, 2870.6046, 2905.9688, 2891.7046, 2852.5153, 2852.9948, 2892.3782, 2909.9106, 2914.6964, 2905.8054, 2898.6961, 2890.0809, 2862.2803, 2861.4181, 2827.7978, 2852.1302, 2925.7162, 2909.3796, 2910.7406, 2881.9743, 2887.6221, 2890.158, 2917.8029, 2987.1186, 3001.9802, 3008.1479, 2982.0738, 2976.2837, 2996.7926, 2978.8784, 3044.9028, 3043.9428, 3015.2633, 3005.2488, 3011.0588, 2933.3634, 2928.2295, 2915.3022, 2917.7609, 2930.5462, 2942.1853, 2937.6168, 2931.6926, 2901.1766, 2924.2006, 2886.974, 2899.9446, 2923.2775, 2937.3601, 2944.541, 2941.007, 2952.3383, 2932.5058, 2908.766, 2867.8376, 2821.4957, 2777.5559, 2768.6795, 2794.5523, 2774.7532, 2814.9944, 2797.2596, 2808.9146, 2815.7976, 2823.8238, 2883.096, 2880.0012, 2880.3307, 2883.4358, 2897.4253, 2863.5673, 2902.1932, 2893.7564, 2890.9192, 2886.2365, 2924.1063, 2930.1538, 2957.4116, 2985.8648, 2999.6013, 3024.7388, 3021.2024, 3008.8118, 3031.2351, 3030.7544, 2978.1178, 2985.6586, 2999.2789, 3006.4467, 2977.0767, 2985.3406, 2955.4325, 2929.0875, 2932.167, 2905.1892, 2913.5704, 2924.8566, 2947.7106, 2973.6558, 3007.8834, 2991.0459, 2978.7124, 2977.3342, 2938.1413, 2939.6179, 2954.3799, 2941.6188, 2940.9215, 2954.9327, 2980.0498, 2954.176, 2939.3209, 2929.0561, 2958.1992, 2975.4919, 2991.5622, 2978.5955, 2978.7144, 2964.1849, 2909.9746, 2914.8232, 2905.2409, 2909.8697, 2891.3431, 2909.2002, 2933.9908, 2911.0534, 2903.6379, 2885.2884, 2906.1688, 2907.0602, 2903.1947, 2889.6934, 2871.9813, 2875.8072, 2884.6988, 2878.1157, 2899.4685, 2912.0136, 2914.4775, 2917.3157, 2924.4173, 2915.6983, 2967.6764, 2984.3925, 3022.4219, 3017.0444, 3017.0658, 3004.9376, 2962.7513, 2982.6806, 2981.8805, 3007.3546, 3005.0355, 3040.0239, 3050.124]

12.3.1 Positioning calculations

【Example 12.8】For the highest and lowest closing prices of the Shanghai Composite Index in 2019, calculate the growth rate.

1	2	3	…	242	243	244
2465.291	2464.3628	2514.8682	…	3005.0355	3040.0239	3050.124

Solution

First locate the sequence number of the maximum / minimum value, and then extract the closing price of the current day and the previous day according to the sequence number, and calculate the growth rate of the current day.

	A	B
1	[2465.291, 2464.3628, …, 3040.0239, 3050.124]	/The constant sequence of closing price of Shanghai stock index in 2019 (see the example for all data)
2	=A1.pmax()	/Use pmax() function to locate the sequence number of the highest closing price.
3	=if(A2>1, A1(A2)/A1(A2-1), 1.0)	/Use the sequence number of the highest point to calculate the growth rate of the highest point and the previous day. If the first day's closing price is the highest, the growth rate is regarded as 1.0
4	=A1.pmin()	/Use pmin() function to locate the sequence number of the lowest closing price
5	=if(A4>1, A1(A4)/A1(A4-1), 1.0)	/Use the sequence number of the lowest point to calculate the growth rate of the lowest point and the previous day. If the first day's closing price is the lowest, the growth rate is regarded as 1.0

Execution result:

72
1.0063
2
0.9996

12.3.1 Positioning calculations

How to locate max/min value:

- For ordered set A, **A.pmax()** function locates the maximum value, and **A.pmin()** function locates the minimum value.
- There may be multiple maximum / minimum values, **A.pmax@a()/A.pmin@a()** can return the sequence numbers of all maximum / minimum values. By default, only the first one is returned.
- By default, to locate the max/min value is from front to back, and **A.pmax@z()/A.pmin@z()** can be used to locate in the reverse order.

Get the sequence numbers of members that meet the conditions. It means that the members of the ordered set are calculated according to the specified conditions, and the sequence numbers of members whose result is true are returned.

Locate employees with salary over 10000.

ID	NAME	BIRTHDAY	ENTRYDATE	DEPT	SALARY
1	Rebecca	1974/11/20	2005/03/11	R&D	7000
2	Ashley	1980/07/19	2008/03/16	Finance	11000
3	Rachel	1970/12/17	2010/12/01	Sales	9000
4	Emily	1985/03/07	2006/08/15	HR	6500
5	Ashley	1975/05/13	2004/07/30	R&D	16000

The sequence numbers of employees with salary over 10000 are [2, 5]

12.3.1 Positioning calculations

【Example 12.9】For the trading days when the closing price of the SSEC index rises more than 80 points from the opening price in 2019, find the growth rate of the trading amount compared with the previous day.(Using data from SSEC.txt)

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
…	…	…	…

12.3.1 Positioning calculations

Get the sequence numbers of members that meet the conditions

【Example 12.9】For the trading days when the closing price of the SSEC index rises more than 80 points from the opening price in 2019, find the growth rate of the trading amount compared with the previous day.(Using data from SSEC.txt)

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
…	…	…	…

Solution

First obtain the sequence numbers of members that meet the conditions (Close-Open>80), and then get the trading amount of the current day and the previous day according to the sequence number, and calculate the increase of the current day.

	A	B
1	=file("SSEC.txt").import@t()	/Import SSEC data
2	=A1.select(year(Date)==2019).sort(Date)	/Select records of 2019 and sort by Date in ascending order
3	=A2.pselect@a(Close-Open>80)	/Use pselect@a() function to locate all the sequence numbers that meet the condition
4	=A3.new(A2(~).Date:Date, if(~>1, A2(~).Amount/A2(~-1).Amount, 1.0):Increase)	/Use A.new() to create a new table sequence. Loop through the located sequence numbers, calculate the amount increase compared with the previous day. For the first day, the increase is regarded as 1.0

A4

Date	Increase
2019/02/25	1.758491
2019/03/29	1.334483
2019/04/16	1.011236

12.3.1 Positioning calculations

Get the sequence numbers of members that meet the conditions

How to get sequence numbers of members that meet the conditions:

- For ordered set A, the **A.pselect(x)** function can be used to obtain the sequence numbers of members satisfying the condition x.
- There may be more than one member satisfying the condition x, **A.pselect@a(x)** returns all sequence numbers that satisfy the condition x. By default, only the first one is returned.
- By default, to get the sequence numbers of members satisfying the condition is locating from front to back, and **A.pselect@z(x)** can be used to locate in the reverse order.

Chapter 12
Order-related calculations

12.4 Sorting and alignment

12.4.1 Sorting

The process of arranging the members of an ordered set according to the key words in a certain way is called **sorting**. Its purpose is to adjust a sequence of "unordered" data into the sequence of "ordered" data.

Employee table

ID	NAME	SALARY
1	Rebecca	7000
2	Ashley	11000
3	Rachel	9000
4	Emily	6500

Sort by salary in ascending order

ID	NAME	SALARY
4	Emily	6500
1	Rebecca	7000
3	Rachel	9000
2	Ashley	11000

Sort by salary in descending order

ID	NAME	SALARY
2	Ashley	11000
3	Rachel	9000
1	Rebecca	7000
4	Emily	6500

12.4.1 Sorting

【Example 12.10】Sort the Shanghai index by trading amount in ascending order. (Using data from SSEC.txt)

Date	Amount
2019/12/31	2.27E11
2019/12/30	2.67E11
2019/12/27	2.58E11
2019/12/26	1.96E11
…	…

12.4.1 Sorting

【Example 12.10】Sort the Shanghai index by trading amount in ascending order. (Using data from SSEC.txt)

Solution

Sort the ordered set by the specified field (transaction amount) in ascending order.

	A	B
1	=file("SSEC.txt").import@t()	/Import data file
2	=A1.sort(Amount)	/Use sort() function to sort by Amount in ascending order

Date	Amount
2019/12/31	2.27E11
2019/12/30	2.67E11
2019/12/27	2.58E11
2019/12/26	1.96E11
…	…

A2

Date	Amount
2019/01/02	9.76E10
2019/01/30	1.01E11
2019/01/23	1.05E11
…	…

Note

- The **sort(x)** function can perform sorting according to the specified condition x. By default, it is sorted in ascending order, and can be reversed by using the **@z** option.

12.4.1 Sorting

Sometimes, we don't want to change the original order of the ordered set, just want to get the sequence numbers after sorting.

The employee table is ordered by name. If you do not change the employee table, find the original sequence numbers of records sorted by salary in ascending order.

NAME	SALARY
Ashley	11000
Emily	6500
Rachel	9000
Rebecca	7000



Sequence numbers sorted by salary
2
4
3
1

The sequence number of the record ranked 1st is 2

The sequence number of the record ranked 2nd is 4

The sequence number of the record ranked 3rd is 3

The sequence number of the record ranked 4th is 1

12.4.1 Sorting

【Example 12.11】Get the employees' names in ascending order of salary. (Using data from EmployeeSample.txt)

ID	Name	BirthDay	EntryDate	Department	Salary
1	Rebecca	1974/11/20	2005/03/11	6	7000
2	Ashley	1980/07/19	2008/03/16	2	11000
3	Rachel	1970/12/17	2010/12/01	7	9000

12.4.1 Sorting

【Example 12.11】Get the employees' names in ascending order of salary. (Using data from EmployeeSample.txt)

Solution

Obtain the original sequence numbers of the employee in ascending order according to the salary, and then obtain the names from the employee table according to the original sequence numbers.

ID	Name	Salary
1	Rebecca	7000
2	Ashley	11000
3	Rachel	9000

	A	B
1	=file("EmployeeSample.txt").import@t()	/Import employee table from text file
2	=A1.psort(Salary)	/Use psort() function to find the sequence numbers of records sorted by salary in ascending order
3	=A1.(Name)(A2)	/Get the names through the sequence numbers from employee table

A3	Name
	Rebecca
	Rachel
	Ashley

Note

- **psort()** function gets the original sequence numbers of sorted members, and it does not change the order of the sequence. The psort() function also supports the **@z** option for reverse order.

12.4.1 Alignment

Alignment is the process of arranging the members of an ordered set in a specified order.

Comparison between sorting and alignment

- Sorting can only arrange data in ascending or descending order.
- If you need to sort the data in a specified order, use **alignment**.

Query the work log in the order of Monday, Tuesday, Wednesday, Thursday and Friday.

Weekday	DailyRecord
Wednesday	Log1
Monday	Log2
Friday	Log3
Tuesday	Log4
Thursday	Log5

→

Weekday	DailyRecord
Monday	Log2
Tuesday	Log4
Wednesday	Log1
Thursday	Log5
Friday	Log3

12.4.1 Sorting

【Example 12.12】Query the sales amount in 2019 in the order of months(January, February, ..., December)(Using data from MonthSales.txt)

Year	Month	Amount
2019	December	3063.0
2019	November	3868.6
2019	October	2713.5
…	…	…

12.4.1 Sorting

【Example 12.12】Query the sales amount in 2019 in the order of months(January, February, ..., December)(Using data from MonthSales.txt)

Year	Month	Amount
2019	December	3063.0
2019	November	3868.6
2019	October	2713.5
…	…	…

Solution

To query in a fixed order, sorting can't be used, and you need to use the alignment function.

	A	B
1	=file("MonthSales.txt").import@t()	/Import sales data
2	=A1.select(Year==2019)	/Select data in 2019
3	[January,February,March,April,May,June,July,August,September,October,November,December]	/Define the constant sequence from Jan to Dec
4	=A2.align@s(A3, Month)	/Use align@s() function to align the Month of sales table to A3

A4

Year	Month	Amount
2019	January	3730.0
2019	February	1444.8
2019	March	2018.6
…	…	…

Note

- **P.align@s(A, y)** function aligns the y values of sequence P to the order of sequence A.

Chapter 12
Order-related calculations

12.5 Exercises

12.5 Exercises

【Exercise 12.1】What is the third member and second to last member in the prime sequence [2, 3, 5, 7, 11, 13, 17, 19]?

【Exercise 12.1】What is the third member and second to last member in the prime sequence [2, 3, 5, 7, 11, 13, 17, 19]?

Solution

Sequence numbers can be used directly to access single or multiple members of an ordered set.

	A	B
1	[2, 3, 5, 7, 11, 13, 17, 19]	/Define a sequence of constants
2	=A1(3)	/Access the third member, A1.m(3) can also be used
3	=A1.m(-2)	/Access the second to last member
4	=A1.m(3, -2)	/Access the third member and second to last member at the same time

Execution result:

```
5
17
[5, 17]
```

【 Exercise 12.2】Query on which day in 2014 that the sales plan of 150000 was completed.(Using data from Sales.txt)

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
…	…	…	…	…

12.5 Exercises

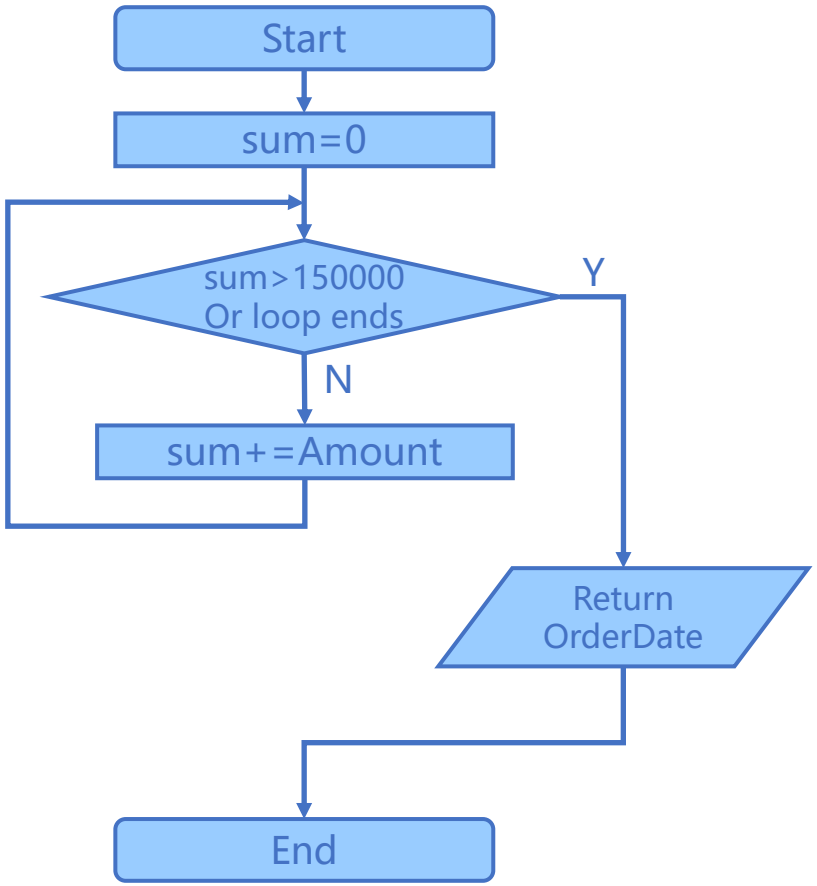
【Exercise 12.2】Query on which day in 2014 that the sales plan of 150000 was completed.(Using data from Sales.txt)

OrderID	Customer	SellerId	OrderDate	Amount
10400	EASTC	1	2014/01/01	3063.0
10401	HANAR	1	2014/01/01	3868.6
10402	ERNSH	8	2014/01/02	2713.5
10403	ERNSH	4	2014/01/03	1005.9
…	…	…	…	…

Solution

Use iteration to calculate the daily cumulative sales, and terminate the iteration when it exceeds 150000.

	A	B
1	=file("Sales.txt").import@t()	/Import sales data
2	=A1.select(year(OrderDate)==2014).sort(OrderDate)	/Select data in 2014 and sort it by OrderDate in ascending order
3	=sum=0, A2.iterate((sum+=Amount, ~~=OrderDate), 0, sum>150000)	/Define the sum variable to represent cumulative sales. If the total sales amount exceeds 150000, it will be terminated and the order date will be returned.



Output result: 2014/03/25

12.5 Exercises

【Exercise 12.3】Find the name of the customer whose customer ID is AROUT. (Using data from Customer.txt)

ID	Name	City	Postcode	Tel	Fax
ALFKI	SanChuan	Tianjin	343567	(030) 30074321	(030) 30765452
ANATR	DongNan	Tianjin	234575	(030) 35554729	(030) 35553744
ANTON	TanShenHang	Shijiazhuang	985060	(0321) 5553932	(null)
AROUT	GuoDing	Shenzhen	890879	(0571) 45557788	(0571) 45556750
…	…	…	…	…	…

12.5 Exercises

【Exercise 12.3】Find the name of the customer whose customer ID is AROUT. (Using data from Customer.txt)

ID	Name	City	Postcode	Tel	Fax
ALFKI	SanChuan	Tianjin	343567	(030) 30074321	(030) 30765452
ANATR	DongNan	Tianjin	234575	(030) 35554729	(030) 35553744
ANTON	TanShenHang	Shijiazhuang	985060	(0321) 5553932	(null)
AROUT	Guoding Co., Ltd	Shenzhen	890879	(0571) 45557788	(0571) 45556750
…	…	…	…	…	…

Solution

Locate the member, find the sequence number of the record with customer ID of AROUT, and then access its name.

	A	B
1	=file("Customer.txt").import@t()	/Import customer table data
2	=A1.(ID).pos("AROUT")	/Use pos() function to locate the sequence number of record
3	=A1(A2).Name	/Access customer table member by sequence number and get its name

Output result: Guoding Co., Ltd

Note

- The result of A1(A2) is a record. **Record.Field** can be used to access the field value of a record.

12.5 Exercises

【Exercise 12.3】Find the name of the customer whose customer ID is AROUT. (Using data from Customer.txt)

ID	Name	City	Postcode	Tel	Fax
ALFKI	SanChuan	Tianjin	343567	(030) 30074321	(030) 30765452
ANATR	DongNan	Tianjin	234575	(030) 35554729	(030) 35553744
ANTON	TanShenHang	Shijiazhuang	985060	(0321) 5553932	(null)
AROUT	Guoding Co., Ltd	Shenzhen	890879	(0571) 45557788	(0571) 45556750
…	…	…	…	…	…

Out

Solution

Another solution is to select the record with customer ID of AROUT, and then access its name.

	A	B
1	=file("Customer.txt").import@t()	/Import customer table data
2	=A1.select@1(ID:"AROUT")	/Use select@1() function to find the first record that meets the condition
3	=A2.Name	/Get the name

Output result: Guoding Co., Ltd

Note

- When using **select@1()**, the result is a record.

12.5 Exercises

【Exercise 12.4】For the trading day in 2019 when the closing price of SSEC index increases the highest compared with the opening price, find the increase rate of trading amount compared with the previous day. (Using data from SSEC.txt)

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
…	…	…	…

12.5 Exercises

【Exercise 12.4】For the trading day in 2019 when the closing price of SSEC index increases the highest compared with the opening price, find the increase rate of trading amount compared with the previous day. (Using data from SSEC.txt)

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
…	…	…	…

Solution

Find the sequence number in an ordered set according to the maximum value of (closing price - opening price). Then according to the sequence number, find the trading amount of the current day and the previous day to calculate the increase rate.

	A	B
1	=file("SSEC.txt").import@t()	/Import SSEC index data
2	=A1.select(year(Date)==2019).sort(Date)	/Select data in 2019 and sort it by Date in ascending order
3	=A2.pmax(Close-Open)	/Use the pmax() function locate the sequence number whose closing price increases the most than the opening price.
4	=if(A3>1, A2(A3).Amount/A2(A3-1).Amount, 1.0)	/Get the trading amount of the day with the highest growth and the previous day through the sequence number, and calculate the increase rate.

A3
34

A4
1.75849

12.5 Exercises

【Exercise 12.5】For the trading days in 2019 when the closing price of SSEC index rises by more than 3, find the trading amount increase compared with the previous day. (Using data from SSEC.txt)

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
…	…	…	…

12.5 Exercises

【Exercise 12.5】For the trading days in 2019 when the closing price of SSEC index rises by more than 3, find the trading amount increase compared with the previous day. (Using data from SSEC.txt)

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
…	…	…	…

Solution

Find the sequence numbers in an ordered set according to the specified condition, then according to the sequence numbers, find the trading amount of the current day and the previous day to calculate the increase rate.

	A	B
1	=file("SSEC.txt").import@t()	/Import SSEC index data
2	=A1.select(year(Date)==2019).sort(Date)	/Select data in 2019 and sort it by Date in ascending order
3	=A2.pselect@a(Close/Close[-1]>1.03)	/use pselect@a() function to locate all the sequence numbers whose closing prices have increased by more than 1.03 compared with the previous day.
4	=A3.new(A2(~).Date:Date, if(~>1, A2(~).Amount/A2(~-1).Amount, 1.0):Increase)	/Create a table sequence. Loop through the located sequence numbers, calculate the trading amount increase of these trading days compared with the previous day

A3	Member
	34
	58
	84

A4	Date	Increase
	2019/02/25	1.75849
	2019/03/29	1.33448
	2019/05/10	1.39086

【Exercise 12.6】Query the medal list of Olympic Games and arrange the records in descending order according to game number and ranking.(Using data from Olympic.txt)

Game	Nation	Gold	Silver	Copper
1	USA	11	7	1
1	Greece	10	19	17
1	Germany	7	5	2
1	France	5	4	2
…	…	…	…	…

12.5 Exercises

【Exercise 12.6】Query the medal list of Olympic Games and arrange the records in descending order according to game number and ranking.(Using data from Olympic.txt)

Game	Nation	Gold	Silver	Copper
1	USA	11	7	1
1	Greece	10	19	17
1	Germany	7	5	2
1	France	5	4	2
…	…	…	…	…

Solution

Sort by game number and ranking in descending order. The order of ranking should compare the number of gold, silver and bronze medals in turn.

	A	B
1	=file("Olympic.txt").import@t()	/Import data
2	=A1.sort@z(Game, 1000000*Gold+1000*Silver+Copper)	/Use the sort() function to sort by the number of sessions and total scores, and the @z option represents the reverse order

A2	Game	Nation	Gold	Silver	Copper
	30	USA	46	29	29
	30	China	38	27	23
	30	UK	29	17	19
	…	…	…	…	…

【Exercise 12.7】List product names in descending order of unit price. (Using data from ProdctPrice.txt)

ID	Name	SUP_ID	Category	Price
1	Apple	2	1	18.5
2	Milk	1	1	21.0
3	Tomato	1	2	12.0
4	Salt	2	2	15.0

12.5 Exercises

【Exercise 12.7】List product names in descending order of unit price. (Using data from ProdctPrice.txt)

Solution

First get the sequence numbers of records sorted by unit price in descending order, and then get the names from the table sequence according to the sequence numbers.

ID	Name	Price
1	Apple	18.5
2	Milk	21.0
3	Tomato	12.0
4	Salt	15.0

	A	B
1	=file("Product.txt").import@t()	/Import product table data
2	=A1.psort@z(Price)	/Use the @z option with psort() function to get the descending sequence numbers of unit price
3	=A1.(Name)(A2)	/Get the names from the product table according to the unit price sequence numbers.

A3	NAME
	Milk
	Apple
	Salt
	Tomato

Chapter 13
Grouping calculations

13.1 Group aggregation

- Group aggregation
- Calculate unique values

13.2 The essence of grouping

- Grouped subsets
- Group aggregation vs. grouping

13.3 Non equivalent grouping

- Equivalent grouping and non equivalent grouping
- Alignment grouping
- Enumeration grouping

13.4 Ordered grouping

- Order of grouping
- Ordered grouping

13.5 Exercises

Chapter 13
Grouping calculations

13.1 Group aggregation

13.1.1 Group aggregation

Sometimes we need to make statistics to data by different types, which requires group aggregation. **Group aggregation** refers to grouping data according to certain rules, and then do aggregate operation for each group.

Calculate the total score of each student:

SID	Subject	Score
1	Math	93
2	Math	97
1	Chinese	79
2	Chinese	95

Grouping

SID	Subject	Score
1	Math	93
1	Chinese	79

SID	Subject	Score
2	Math	97
2	Chinese	95

Aggregation

SID	TotalScore
1	172
2	192

13.1.1 Group aggregation

【Example 13.1】Calculate the total sales of each customer in 2014. (Using data from Sales.txt)

OrderID	Customer	OrderDate	Amount
10400	EASTC	2014/01/01	3063.0
10401	RATTC	2014/01/01	3868.6
10402	ERNSH	2014/01/02	2713.5
…	…	…	…

13.1.1 Group aggregation

【Example 13.1】Calculate the total sales of each customer in 2014. (Using data from Sales.txt)

Solution

The **groups** function of SPL can realize group aggregation. Function **A.groups(x:F, …;y:G, …)** performs grouping and aggregation by one or more fields / expressions, and forms a new table sequence with F, … G, … as fields, where y is the aggregate function.

	A	B
1	=file("Sales.txt").import@t() .select(year(Date)==2014)	/Import the sales table from the file and select the sales record of 2014
2	=A1.groups(Customer; sum(Amount):Amount)	/Group and aggregate the total sales of each customer

A2	Customer	Amount
	ANATR	1129.75
	…	…

Note

- The function **sum()** is used to get the sum, and can be used for aggregate operation in **A.groups()** function.

13.1.1 Group aggregation

【Example 13.2】Calculate the highest salary, lowest salary and average age of employees in each department. (Using data from Employee.txt)

ID	Name	Department	Birthday	Salary
1	Rebecca	6	1974/11/20	7000
2	Ashley	2	1980/07/19	11000
3	Rachel	7	1970/12/17	9000
…	…	…	…	…

13.1.1 Group aggregation

【Example 13.2】Calculate the highest salary, lowest salary and average age of employees in each department. (Using data from Employee.txt)

Solution

Group the employee table by department, and calculate the maximum salary, minimum salary and average age of each department. The **A.groups()** function can be used for group aggregation, and there can be multiple aggregate functions and aggregate fields.

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=A1.groups(Department; max(Salary):MaxSalary, min(Salary):MinSalary, round(avg(age(Birthday)), 1):AvgAge)	/Group and calculate the maximum salary, minimum salary and average age of each department

A2	Department	MaxSalary	MinSalary	AvgAge
	1	16000	5000	44.0
	2	11000	5000	39.3
	3	12000	3000	41.4
	…	…	…	…

Note

- The function **avg()** is used to calculate the average value, the function **max()** is used to get the maximum value, and the function **min()** is used to get the minimum value, which can all be used in **A.groups()** function for aggregate operations.

【Example 13.3】Calculate the number of excellent students (more than 85 points) and the median score of each subject in each class. (Using data from Score.txt)

CLASS	STUDENTID	SUBJECT	SCORE
1	1	English	84
1	1	Math	77
1	1	Chinese	69
2	1	English	96
2	1	Math	63
2	1	Chinese	81
1	2	English	81
1	2	Math	80
1	2	Chinese	97
…	…	…	…

13.1.1 Group aggregation

【Example 13.3】Calculate the number of excellent students (more than 85 points) and the median score of each subject in each class. (Using data from Score.txt)

Solution

Group the score table by two fields: class and subject, and count the number of excellent students and calculate the median. **A.groups()** function can be used for group aggregation, and there can be multiple grouping fields / expressions and aggregate functions.

	A	B
1	=file("Score2.txt").import@t()	/Import the score table from file
2	=A1.groups(CLASS,SUBJECT; count(SCORE>85):ExcellentCount, median(,SCORE):Median)	/Group by class and subject, count the number of excellent students and calculate the median score

A2

CLASS	SUBJECT	ExcellentCount	Median
1	Chinese	3	79
1	English	3	75
1	Math	5	75
…	…	…	…

Note

- The function **count()** is used to count and the function **median(, x)** is used to calculate the median, which can all be used in groups function for aggregate operations.

13.1.1 Group aggregation

There are two key factors in group aggregation operation: Grouping fields and aggregation values. The result of the group aggregation operation is a table sequence composed of grouping fields and aggregation values.



Class	Subject	ExcellentCount	Median
1	English	3	75
1	Math	5	75
1	Chinese	3	79
2	English	4	76
2	Math	6	79
2	Chinese	2	77
…	…	…	…

Grouping fields (possibly more than one) **are unique** in a group aggregation result.

Each class and subject determines a **unique** record.

【Example 13.4】Count the frequencies of customers who ranked in the top three of monthly sales in 2014. (Using data from Sales.txt)

OrderID	Customer	OrderDate	Amount
10400	EASTC	2014/01/01	3063.0
10401	RATTC	2014/01/01	3868.6
10402	ERNSH	2014/01/02	2713.5
…	…	…	…

13.1.1 Group aggregation

【Example 13.4】Count the frequencies of customers who ranked in the top three of monthly sales in 2014. (Using data from Sales.txt)

Solution

Group the sales table by the month of the order date, and calculate the total sales amount of each month. **A.groups()** function can be used for group aggregation.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=A1.select(year(OrderDate)==2014)	/Select the 2014 records from the sales table
3	=A2.groups(month(OrderDate):Month; top(-3;Amount):Top3)	/Group and get the top three sales of each month
4	=A3.conj(Top3).groups(Customer; count(~):Count)	/Group again and get the frequencies of each customer

Note

- Function **top(n; x)** gets the first n/ last n records of value x, when n is a positive number, get the last n records of value x, when n is a negative number, get the first **|n|** records of value x . It can be used in **A.groups()** function for aggregate operation.

13.1.1 Group aggregation

【Example 13.4】Count the frequencies of customers who ranked in the top three of monthly sales in 2014. (Using data from Sales.txt)

A3	Month	Top3
	1	[[10424,MEREP,2014/01/23,11493.2], …]
	2	[[10440,HANAR,2014/02/10,10793.1], …]
	3	[[10479,HANAR,2014/03/19,10495.6], …]
	…	…

OrderID	Customer	OrderDate	Amount
10424	MEREP	2014/01/23	11493.2
10417	SIMOB	2014/01/16	11283.2
10430	ERNSH	2014/01/30	5796.0

Grouping field

A4	Customer	Count
	ERNSH	4
	FOLIG	1
	FOLKO	1
	…	…

In the result table sequence of group aggregation, the **value of grouping field is unique**. There are no duplicate months in A3 and no duplicate customers in A4.

13.1.1 Group aggregation

Take【Example13.1】as an example, take a look at the actual calculation process of the groups function. Calculate the total sales of each customer in 2014. (Using data from Sales.txt)

OrderID	Customer	OrderDate	Amount
10400	EASTC	2014/01/01	3063.0
10401	RATTC	2014/01/01	3868.6
10402	ERNSH	2014/01/02	2713.5
…	…	…	…

13.1.1 Group aggregation

【Example13.1】Calculate the total sales of each customer in 2014. (Using data from Sales.txt)

Solution

The function **groups** calculates the SUM during group aggregation. First, create an empty table sequence to store the aggregation results. Loop through each record of the data table, search for the grouping field value in the aggregation result. If it cannot be found, create a new group; if it is found, perform the SUM calculation. Finally, sort by the grouping field.

Customer	Amount
EASTC	3063.0
RATTC	3868.6
…	…
RATTC	3674.0
…	…

Customer	Amount

Create an empty table sequence to store aggregation results.

Customer	Amount
EASTC	3063.0

Create a new group when **EASTC** cannot be found

Customer	Amount
EASTC	3063.0
RATTC	3868.6

Create a new group when **RATTC** cannot be found

Customer	Amount
EASTC	3063.0
RATTC	7, 542.6

Accumulate sales when **RATTC** is found

13.1.1 Group aggregation

【Example13.1】Calculate the total sales of each customer in 2014. (Using data from Sales.txt)

Solution

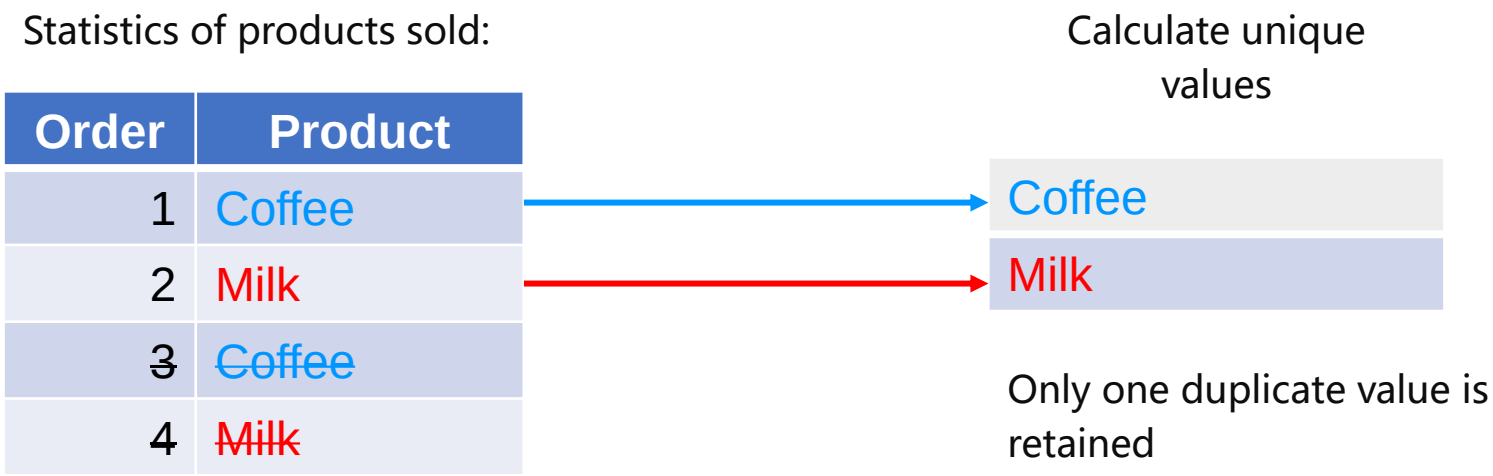
The calculation principle of function **groups** can be simulated by loop function (take **sum** as an example). SPL script is as follows:

	A	B	C	D
1	=file("Sales.txt").import@t().select(year(OrderDate)==2014)			/Import the sales table from the file and select the sales records of 2014
2	=create(Customer, Amount).keys(Customer)			/Create an empty table sequence and set customer as the primary key
3	for A2	=A2.find(A3.Customer)		/Loop through the sales table, find the current customer in the aggregation result
4		if B3==null	>A2.insert(0, A3.Customer, A3.Amount)	/If it's not found, create a new group
5		else	>B3.Amount+=A3.Amount	/If it's found, calculate the SUM
6	=A2.sort(Customer)			/Sort by customer

A6	Customer	Amount
	ANATR	1129.75
	…	…

13.1.2 Calculate unique values

As mentioned earlier, the result set of group aggregation is a table sequence composed of **grouping fields** and **aggregation values**. In the result table sequence, the **values of grouping fields are unique**, so the **deduplication of field values** can be realized by grouping without aggregation values.



13.1.2 Calculate unique values

【Example 13.5】Check the states and Departments of the employees. (Using data from Employee.txt)

ID	Name	State	Department
1	Rebecca	California	6
2	Ashley	New York	2
3	Rachel	New Mexico	7
…	…	…	…

13.1.2 Calculate unique values

【Example 13.5】Check the states and Departments of the employees. (Using data from Employee.txt)

Solution

As mentioned earlier, grouping without aggregation values can get unique values. Function **A.groups(x:F, …)** can be used to calculate unique values.

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=A1.groups(State, Department)	/Use the groups function to calculate unique values

A2	State	Department
	Alabama	4
	Alabama	7
	Arizona	5
	Arizona	7
	Arizona	8
	Arkansas	6
	…	…

Note

- The result set of function **groups** to calculate unique values is a table sequence composed of grouping fields / expressions. It is similar to the result set of group aggregation, except that there is no aggregation values.

13.1.2 Calculate unique values

【Example13.6】Query customers with sales records in 2014. (Using data from Sales.txt)

OrderID	Customer	OrderDate	Amount
10400	ANATR	2014/01/01	3063.0
10401	ERNSH	2014/01/01	3868.6
10402	ANATR	2014/01/02	2713.5
…	…	…	…

13.1.2 Calculate unique values

【Example13.6】Query customers with sales records in 2014. (Using data from Sales.txt)

Solution

Get the unique values of a single field. **A.groups(x)** function can be used to get the **unique values** of **x**.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=A1.select(year(OrderDate)==2014)	/Select sales records in 2014
3	=A2.groups(Customer)	/Use the groups function to calculate the unique values of the customer field

A3	Customer
	ANATR
	ANTON
	ERNSH
	…

Note

- The result returned by the function **groups** is **always** a table sequence. When calculating the unique values of a single field, the table sequence of the single field is returned, not the sequence.

13.1.2 Calculate unique values

【Example13.6】Query customers with sales records in 2014. (Using data from Sales.txt)

Solution

There is often **only one** grouping field to calculate the unique values. It is not convenient for groups to return a table sequence of a single field, so it is simplified to **id**. **A.id(x_i, ...)** function of SPL can be used to get the unique values for each x_i .

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=A1.select(year(OrderDate)==2014)	/Select sales records in 2014
3	=A2.id(Customer)	**/Use the ID function to calculate the unique values of the customer field**

A3	Member
	ANATR
	ANTON
	ERNSH
	…

Note

- When there is only one x_i, function **A.id(x_i, ...)** returns a sequence; when there are multiple x_i, it returns a sequence composed of sequences.

13.1.2 Calculate unique values

Take 【Example13.5】as an example to see the return values of ID function with multi fields. Check the states and Departments of the employees. (Using data from Employee.txt)

Solution

When x_i ... are multiple fields, **A.id(x_i, ...)** returns a **sequence** composed of unique value sequences.

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=A1.id(State, Department)	**/Use id function to calculate the unique values**

A2	Member
	[California,New York,New Mexico,...]
	[6,2,7,3,4,5,1,8]

Member
California
New York
…

Member
6
2
…

13.1.2 Calculate unique values

【Example 13.7】Count the number of customers per month in 2014. (Using data from Sales.txt)

OrderID	Customer	OrderDate	Amount
10400	ANATR	2014/01/01	3063.0
10401	ERNSH	2014/01/01	3868.6
10402	ANATR	2014/01/02	2713.5
…	…	…	…

13.1.2 Calculate unique values

【Example 13.7】Count the number of customers per month in 2014. (Using data from Sales.txt)

Solution

Group the sales table by month and count the number of unique customer values.
A.groups() function can be used to do group aggregation.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=A1.select(year(OrderDate)==2014)	/Select sales records in 2014
3	=A2.groups(month(OrderDate):Month; icount(Customer):Count)	/Group and count the number of customers per month

A3	Month	Count
	1	27
	2	22
	3	24
	…	…

Note

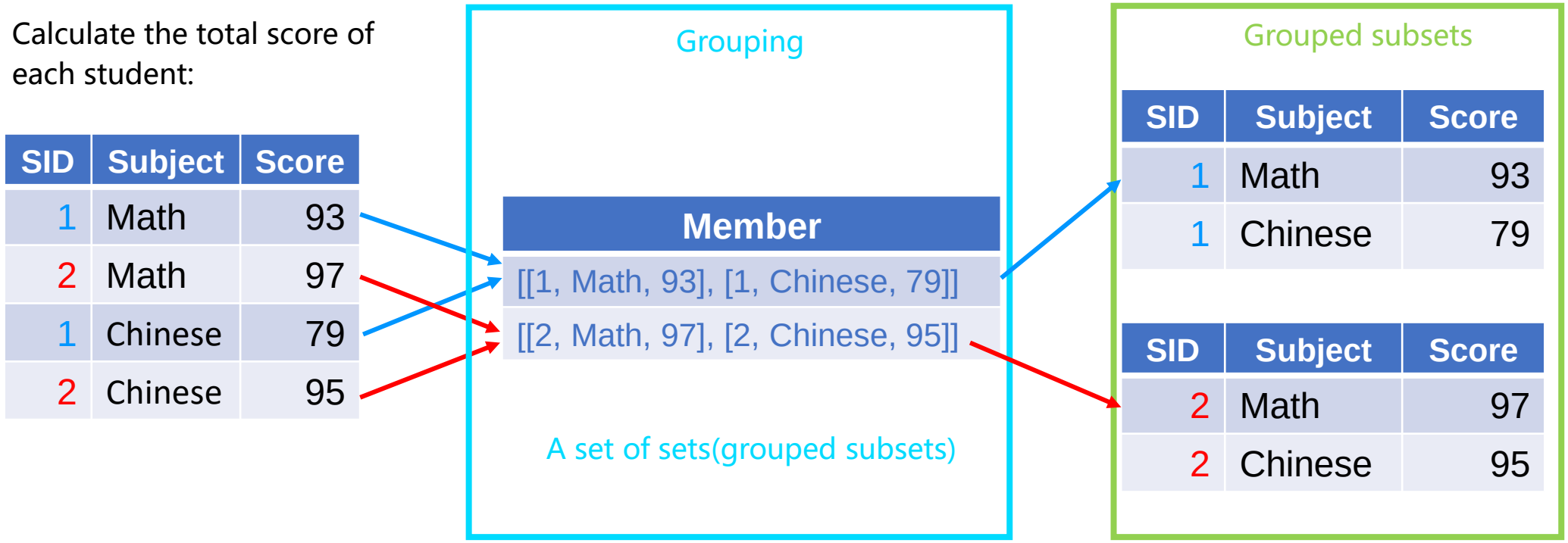
- The function **icount()** is used to count the number of unique values of an expression, and can be used for aggregate operation in **groups** function.

Chapter 13
Grouping calculations

13.2 The essence of grouping

13.2.1 Grouped subsets

The essence of grouping operation is to divide a set into several subsets according to certain rules, that is to say, the return value should be a set composed of sets. For each member set, we call it a **grouped subset**.



13.2.1 Grouped subsets

In general, people are more interested in the aggregate value of the subset, so the grouping operation is often accompanied by the further aggregation calculation of the subset. However, there are still times when we are more interested in these grouped subsets than aggregate values.

Check the scores of students whose score difference of two subjects is more than 10:

SID	Subject	Score
1	Math	93
2	Math	97
1	Chinese	79
2	Chinese	95

Grouping

Member
[[1, Math, 93], [1, Chinese, 79]]
[[2, Math, 97], [2, Chinese, 95]]

Grouped subset satisfying conditions

SID	Subject	Score
1	Math	93
1	Chinese	79

Group aggregation can only get aggregate results, but **cannot** get the details of the **grouped subsets**.

13.2.1 Grouped subsets

【Example 13.8】Find employees with the same date of birth. (Using data from Employee.txt)

ID	Name	Birthday	Salary
1	Rebecca	1974/11/20	7000
2	Ashley	1980/07/19	11000
3	Rachel	1970/12/17	9000
…	…	…	…

13.2.1 Grouped subsets

【Example 13.8】Find employees with the same date of birth. (Using data from Employee.txt)

Solution

What we are concerned about here is not the aggregation result, but the grouped subsets, so we need to use grouping. Function **A.group(x_i, ...)** is used to do equivalence grouping and sorting by the values of x_i, the return value is the sequence of sequences (grouped subsets).

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=A1.group(Birthday)	/Use the group function to group by date of birth without aggregation
3	=A2.select(~.len()>1).conj()	/Select employees with the same date of birth

A2

Member
[296, Olivia, 1968/11/05, 10000]
[[67, Cameron, 1970/03/14, 5000], [115, Gabriel, 1970/03/14, 7000]]
[32, Andrew, 1971/08/27, 11000]
…

ID	Name	Birthday	Salary
296	Olivia	1968/11/05	10000

ID	Name	Birthday	Salary
67	Cameron	1970/03/14	5000
115	Gabriel	1970/03/14	7000

A3

ID	Name	Birthday	Salary
67	Cameron	1970/03/14	5000
115	Gabriel	1970/03/14	7000
32	Andrew	1971/08/27	5000
47	Elizabeth	1971/08/27	6500
…	…	…	…

13.2.1 Grouped subsets

【Example 13.9】Calculate the number of people in each department, and then calculate the average age of employees in departments with more than 50 people.(Using data from Employee.txt)

ID	Name	Birthday	Department
1	Rebecca	1974/11/20	6
2	Ashley	1980/07/19	2
3	Rachel	1970/12/17	7
…	…	…	…

13.2.1 Grouped subsets

【Example 13.9】Calculate the number of people in each department, and then calculate the average age of employees in departments with more than 50 people.(Using data from Employee.txt)

Solution

The **grouping** itself is a complex operation, and the grouped subsets can be retained and reused to improve the operation efficiency. In the second aggregation, the first grouped subsets can be used to continue the calculation without grouping again.

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=A1.group (Department)	/Use the group function to group by department
3	=A2.new(Department, count(~):Count)	/Calculate the number of people in each department
4	=A2.select(~.count()>50).new(Department, ~.avg(age(Birthday)):Age)	/Select the subsets with more than 50 members and calculate the average age

A2	Member
	[[18,Jonathan,1971/03/07,1],…]
	…

ID	Name	Birthday	Department
18	Jonathan	1971/03/07	1
…	…	…	…

A3	Department	Count
	1	4
	2	24
	…	…

A4	Department	Age
	4	42.56
	5	41.33
	7	41.95

13.2.1 Grouped subsets

Take 【Example 13.8】as an example to see the grouping process of group function. Find employees with the same date of birth. (Using data from Employee.txt)

ID	Name	Birthday	Salary
1	Rebecca	1974/11/20	7000
2	Ashley	1980/07/19	11000
3	Rachel	1970/12/17	9000
…	…	…	…

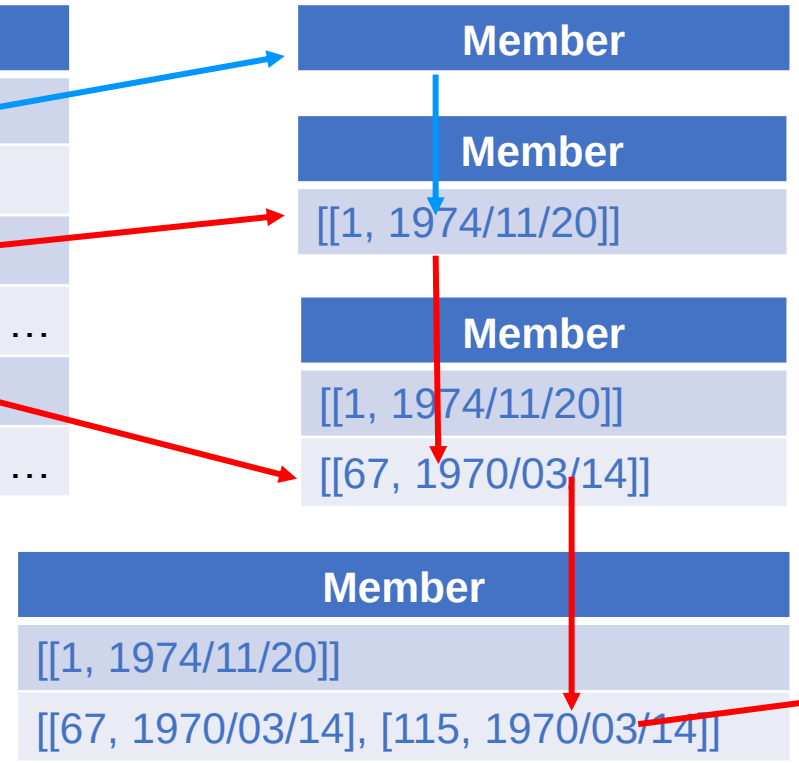
13.2.1 Grouped subsets

【Example 13.8】Find employees with the same date of birth. (Using data from Employee.txt)

Solution

The function **group** performs grouping by the grouping field/expression. First, create an empty sequence to store the grouping results. Loop through each record of the data table, search for the grouping field value in the grouping result. If it cannot be found, create a new group; if it is found, append record to the positioned group. Finally, sort by the grouping field.

ID	Birthday
1	1974/11/20
…	…
67	1970/03/14
…	…
115	1970/03/14
…	…



Create an empty sequence to store the grouping result

Create a new group when **1974/11/20** can't be found

Create a new group when 1970/03/14 can't be found

Append record to the group when 1970/03/14 is found

ID	Birthday
67	1970/03/14
115	1970/03/14

13.2.1 Grouped subsets

【Example 13.8】Find employees with the same date of birth. (Using data from Employee.txt)

Solution

Loop function can be used to simulate the calculation principle of function group. SPL script is as follows:

	A	B	C	D	E
1	=file("Employee.txt").import@t()				/Import employee table from file
2	[]				/Create an empty sequence to store the grouping result
3	for A1	if A2.len()==0	>A2=[[A3]]		/Loop through the employee table, insert the first record to the empty sequence
4		else	=A2.(Birthday).pos(A3.Birthday)		/Locate Birthday value in the result set
5			if C4==null	>A2=A2|[[A3]]	/Create a new group if it is not found
6			else	>A2(C4)=A2(C4)|A3	/Append the record to the group if it is found
7	=A2.sort(Birthday)				/Sort by Birthday
8	=A7.select(~.len()>1).conj()				/Select employees with the same date of birth

13.2.1 Grouped subsets

【Example 13.8】Find employees with the same date of birth. (Using data from Employee.txt)

A2

Member
[296, Olivia, 1968/11/05, 10000]
[[67, Cameron, 1970/03/14, 5000], [115, Gabriel, 1970/03/14, 7000]]
[32, Andrew, 1971/08/27, 11000]
…

ID	Name	Birthday	Salary
296	Olivia	1968/11/05	10000

ID	Name	Birthday	Salary
67	Cameron	1970/03/14	5000
115	Gabriel	1970/03/14	7000

A8

ID	Name	Birthday	Salary
67	Cameron	1970/03/14	5000
115	Gabriel	1970/03/14	7000
87	Sarah	1972/02/16	5000
355	Lauren	1972/02/16	6500
…	…	…	…

13.2.2 Group aggregation vs. grouping

【Example 13.10】Calculate the total score and average score of each student in each class. (Using data from Score.txt)

CLASS	STUDENTID	SUBJECT	SCORE
1	1	English	84
1	1	Math	77
1	1	Chinese	69
2	1	English	96
2	1	Math	63
2	1	Chinese	81
1	2	English	81
1	2	Math	80
1	2	Chinese	97
…	…	…	…

13.2.2 Group aggregation vs. grouping

【Example 13.10】Calculate the total score and average score of each student in each class. (Using data from Score.txt)

Solution

Group by CLASS and STUDENTID first, and then perform aggregate calculations for the grouped subsets.

	A	B
1	=file("Score.txt").import@t()	/Import score table from file
2	=A1.group(CLASS, STUDENTID)	/Use the group function to group by CLASS and STUDENTID
3	=A2.new(CLASS, STUDENTID, ~.sum(SCORE):TotalScore, round(~.avg(SCORE), 2):AvgScore)	/Calculate the total score and average score of each student

A2

Member
[[1,1,English,84], …]
[[1,2,English,81], …]
[[1,3,English,75], …]
…

CLASS	STUDENTID	Subject	Score
1	1	English	84
1	1	Math	77
1	1	Chinese	69

A3

CLASS	STUDENTID	TotalScore	AvgScore
1	1	230	76.67
1	2	258	86.0
1	3	228	76.0
…	…	…	…

13.2.2 Group aggregation vs. grouping

【Example 13.10】Calculate the total score and average score of each student in each class. (Using data from Score.txt)

Solution

Group function with aggregate functions can be used to simplify. Function **A.group(x:F, ...;y:G, ...)** groups sequence **A** by **x, ...** first, and then perform **y** operation.

	A	B
1	=file("Score.txt").import@t()	/Import score table from file
2	=A1.group(CLASS, STUDENTID; ~.sum(SCORE): TotalScore, round(~.avg(SCORE), 2):AvgScore)	**/Use the group function to group by CLASS and STUDENTID, and calculate the total score and average score of each student**

A2	CLASS	STUDENTID	TotalScore	AvgScore
	1	1	230	76.67
	1	2	258	86.0
	1	3	228	76.0
	…	…	…	…

13.2.2 Group aggregation vs. grouping

【Example 13.10】Calculate the total score and average score of each student in each class. (Using data from Score.txt)

Solution

Using the group aggregation function **A.groups(x:F, ...;y:G, ...)** can also solve the problem. The expression of the function **groups** is very close to that of **group**. The difference is that y is limited to several aggregate expressions of cumulative calculation. We have learned that the actual execution process of function **groups** is cumulative, which does not generate **grouping subsets** and is more efficient.

	A	B
1	=file("Score.txt").import@t()	/Import score table from file
2	=A1.groups(CLASS, STUDENTID; sum(SCORE):TotalScore, avg(SCORE):AvgScore)	**/Use the groups function to group by CLASS and STUDENTID, and calculate the total score and average score of each student**

A2	CLASS	STUDENTID	TotalScore	AvgScore
	1	1	230	76.67
	1	2	258	86
	1	3	228	76
	…	…	…	…

13.2.2 Group aggregation vs. grouping

【Example 13.11】Query the employees whose salary exceeds the Department average salary. (Using data from Employee.txt)

ID	Name	Department	Salary
1	Rebecca	6	7000
2	Ashley	2	11000
3	Rachel	7	9000
…	…	…	…

13.2.2 Group aggregation vs. grouping

【Example 13.11】Query the employees whose salary exceeds the Department average salary. (Using data from Employee.txt)

Solution

This problem can not be solved by using groups directly. It needs to do complex calculations for each subset after grouping. The **y** in function **A.group(x:F, ...;y:G, ...)** is not limited to **aggregate functions**, but can be any **sequence** functions.

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=A1.group(Department; (a=avg(Salary), ~.select(Salary>a)):High)	/Use the group function to group by department, calculate the average salary of each department, and select the employees whose salary is higher than the average.
3	=A2.conj(High)	/Concatenate the selected records

A3	ID	Name	Department	Salary
	20	Alexis	1	16000
	42	Michael	1	12000
	51	Madison	3	9000
	…	…	…	…

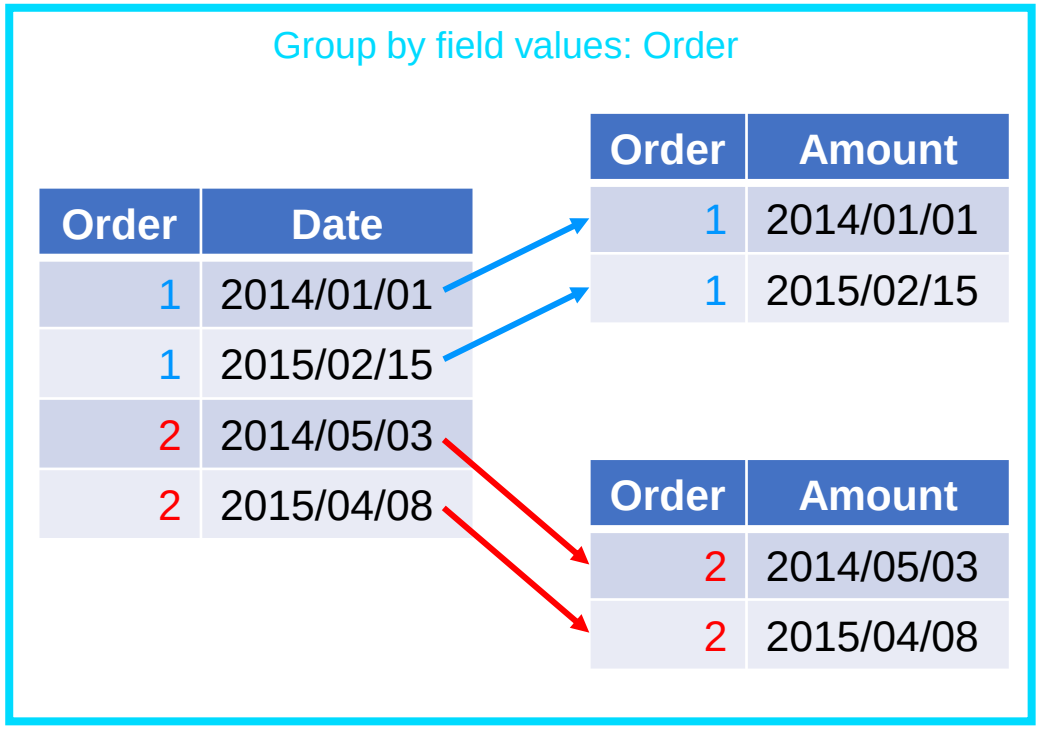
Chapter 13
Grouping calculations

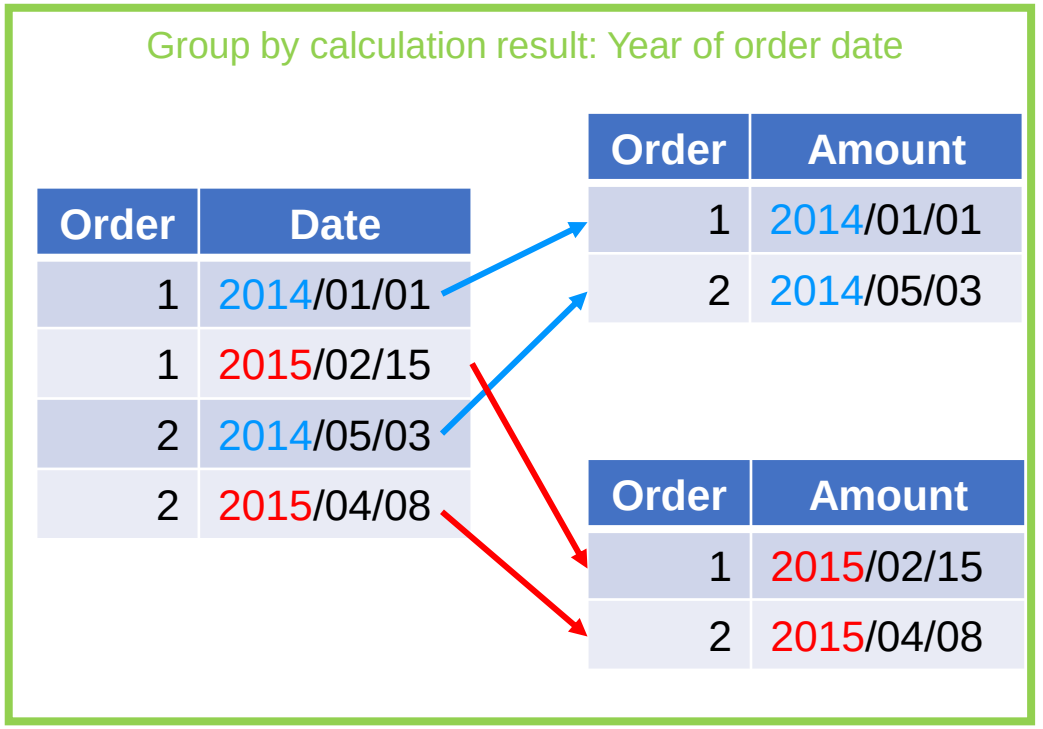
13.3 Non equivalent grouping

13.3.1 Equivalent grouping and non equivalent grouping

The grouping problems we have encountered before have a common feature: grouping is based on the **same** field value or expression result, and this kind of grouping is called **equivalent grouping**, which is the most common grouping form. Equivalent grouping has the characteristics of complete division:

1. There is no null subset in the grouping result.
2. Any member of the original set belongs to and only belongs to one subset.





13.3.1 Equivalent grouping and non equivalent grouping

We learned equivalent grouping and complete division. Is there any **non equivalent grouping** and **incomplete division**? yes. There are also incomplete divisions with the following characteristics:

1. The grouping results contain null subsets.
2. The grouping results do not contain all the original set members.
3. The same member is divided into two groups.

Group by municipality, first-tier city, second-tier city, third-tier city:

1. No members are divided to third tier cities

2. Shaoguan (the fourth tier city) is not included in the grouping results

3. Beijing is divided to both municipalities directly under the central government and first tier cities

ID	City
1	Beijing
2	Guangzhou
3	Dalian
4	Shaoguan

Municipalities

ID	City
1	Beijing

First-tier cities

ID	City
1	Beijing
2	Guangzhou

Second-tier cities

ID	Name
3	Dalian

13.3.2 Alignment grouping

Compared with a benchmark set, a field or expression of the members of the set to be grouped is compared with the members of the benchmark set. If the comparison result is the same, the members are divided into the same group, and the number of groups is the same as the number of members of the benchmark set. This grouping is called alignment grouping. Alignment grouping may have null groups, or members not assigned to any of the groups.

Alignment grouping: align the initial letter of Name in the order of A, B, C:

ID	Name
1	Christian
2	Rudy
3	Andrew
4	Claire

Name
A
B
C

Member
[3, Andrew]
[]
[[1, Christian], [4, Claire]]

B null group

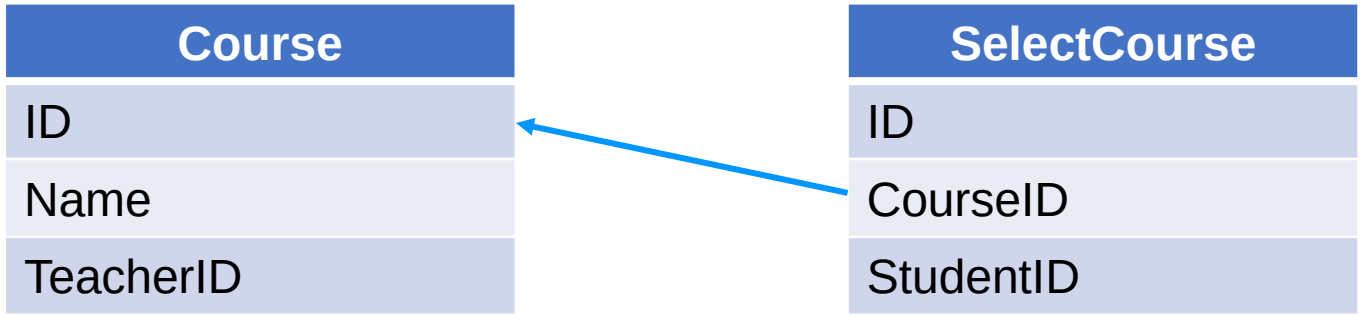
Others
[2, Rudy]

R not assigned

13.3.2 Alignment grouping

Each group keeps only one record

【Example 13.12】Query which courses are not selected according to the order of *Course* table. (The data of *Course* table and *SelectCourse* table are from Course.txt and SelectCourse.txt respectively)



13.3.2 Alignment grouping

Each group keeps only one record

【Example 13.12】Query which courses are not selected according to the order of *Course* table. (The data of *Course* table and *SelectCourse* table are from Course.txt and SelectCourse.txt respectively)

Solution

Group records in *SelectCourse* table according to the order of the course table. The null groups are the courses that no students selected, and **alignment grouping** should be used. The **P.align(A:x, y)** function of SPL aligns the records of P to the field x of A through field y. By default, each group selects one matching member.

	A	B
1	=file("Course.txt").import@t()	/Import course table from file
2	=file("SelectCourse.txt").import@t()	/Import SelectCourse table from file
3	=A2.align(A1:ID, CourseID)	/Align SelectCourse table by Course.ID and select one matching member for each group
4	=A1(A3.pos@a(null))	/Select records of unselected courses (value is null) in the Course table

A3	Member
	(null)
	[13, 2, 7]
	…
	[6, 9, 57]
	(null)

Only care about whether the group is null or not, and do not care about the members of each group. Each group can return one matching member.

ID	CourseID	StudentID
13	2	7

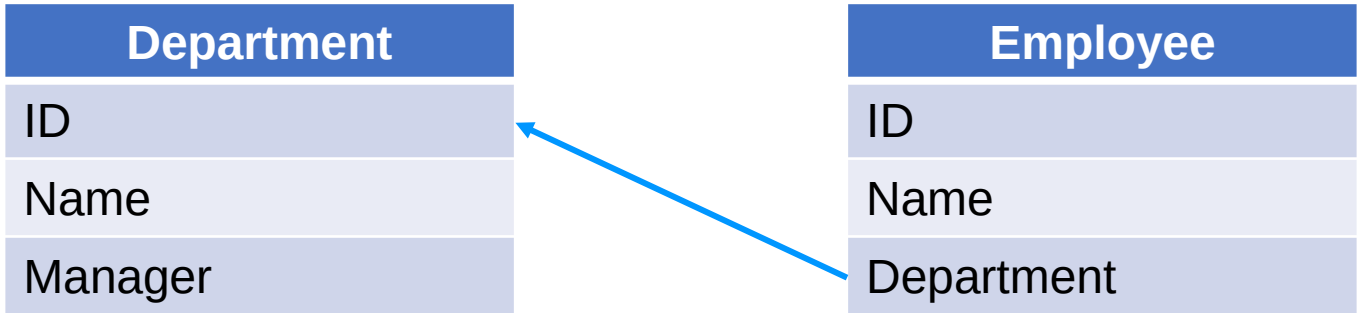
Null group

A4	ID	NAME	TeacherID
	1	Environmental protection and sustainable development	5
	10	Music appreciation	18

13.3.2 Alignment grouping

Each group selects all matching members

【Example 13.13】Calculate the number of employees in each department by the department order in *DEPARTMENT* table. (The data of *Department* table and *Employee* table are from Department.txt and Employee.txt respectively)



13.3.2 Alignment grouping

Each group selects all matching members

【Example 13.13】Calculate the number of employees in each department by the department order in *DEPARTMENT* table. (The data of *Department* table and *Employee* table are from Department.txt and Employee.txt respectively)

Solution

The alignment grouping can be used. Different from the previous section, we care about the number of matching members in each group, so all matching members need to be selected. The **@a** option of the **P.align (A:x, y)** function returns all matching members in each group.

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=file("Department.txt").import@t()	/Import department table from file
3	=A1.align@a(A2:ID, Department)	/Align and group EMPLOYEE table by DEPARTMENT.ID; @a option returns all matching records for each group
4	=A3.new(Department, ~.count():Count)	/Count the number of employees in each department

A3

Member
[[18,Jonathan,1,...], [20,Alexis, 1,...], …]
[[1,Rebecca,6,...],[5,Ashley,6,...],...]
[[3,Rachel,7,...],[6,Matthew,7,...],...]
…

Return all matching records in each group

Name	Department
Rebecca	6
Ashley	6
Ryan	6
…	…

A4

Department	Count
1	4
2	24
3	19
…	…

【Example 13.14】Calculate the total sales amount of each working day by the order of Monday, Tuesday, weekend, Thursday, Friday, and put the sales amount of weekends in a new group. (Using data from DailySales.txt)

Week	Day	Amount
1	Sunday	2018.2
1	Monday	3063.0
1	Tuesday	3868.6
1	Wednesday	2713.5
1	Thursday	1005.9
1	Friday	1675.0
1	Saturday	400.0
2	Sunday	2123.2
…	…	…

13.3.2 Alignment grouping

【Example 13.14】Calculate the total sales amount of each working day by the order of Monday, Tuesday, weekend, Thursday, Friday, and put the sales amount of weekends in a new group. (Using data from DailySales.txt)

Solution

In this example, in addition to selecting all matching members for each group, the unmatched members should be stored in a new group. The **@n** option of the **P.align (A:x, y)** function creates a new group at the end of the benchmark set to hold mismatched members. The benchmark set here is a sequence, and the parameter x can be omitted.

	A	B
1	=file("DailySales.txt").import@t()	/Import sales table from file
2	[Monday, Tuesday, Wednesday, Thursday, Friday]	/Create a sequence of constants from Monday to Friday
3	=A1.align@an(A2,Day)	/Align and group the sales table by the weekday sequence in A2, return all matching members for each group and put the unmatched members in a new group, i.e. the weekend group
4	=A3.new(if(#>A2.len(),"Weekend",Day):Weekday,~.sum(Amount):Amount)	/Calculate the total sales amount of each group and generate a new table sequence. The last group is renamed weekend, otherwise it is the day of the week of the first record in this group.

13.3.2 Alignment grouping

Put unmatched members in a new group

【Example 13.14】Calculate the total sales amount of each working day by the order of Monday, Tuesday, weekend, Thursday, Friday, and put the sales amount of weekends in a new group. (Using data from DailySales.txt)

A3	Member
	[[1, Monday, 3063.0], [2, Monday, 1194.0], …]
	[[1, Tuesday, 3868.6], [2, Tuesday, 1622.4], …]
	[[1, Wednesday, 2713.5], [2, Wednesday, 319.2], …]
	[[1, Thursday, 1005.9], [2, Thursday, 802.0], …]
	[[1, Friday, 1675.0], [2, Friday, 1208.5], …]
	[[1, Saturday, 400.0], [2, Sunday, 2018.2], …]

A4	Weekday	Amount
	Monday	107206.26
	Tuesday	81944.78
	Wednesday	113883.85
	Thursday	108168.7
	Friday	88865.28
	Weekend	223319.88

Week	Day	Amount
1	Sunday	2018.2
1	Saturday	400.0
2	Sunday	2123.2
2	Saturday	372.0
…	…	…

Mismatched members are stored in the last group

13.3.3 Enumeration grouping

Enumeration grouping means : A group of enumeration conditions are specified in advance, and the members of the set to be grouped are used as parameters to calculate the conditions. The members satisfying the condition are all divided into a subset corresponding to the condition, and the subsets of the result set correspond to the specified conditions one by one.

The grouping field is Age:

ID	Name	Age
1	Rebecca	45
2	Ashley	39
3	Rachel	29
4	Emily	35

Enumeration conditions

Subject
? < 30
? < 40

Enumeration grouping

Member
[3, Rachel, 29]
[[2, Ashley, 39], [4, Emily, 35]]

ID	Name	Age
2	Ashley	39
4	Emily	35

Thinking:

(1) 45 does not meet the conditions, how to deal with unmatched members?

(2) 29 not only meets the age < 30 condition, but also meets the age < 40 condition. Is it repeatedly assigned to two groups?

13.3.3 Enumeration grouping

Each member is only put in the first matching group

【Example 13.15】The following is the urban population table of the major cities in China. Classify the cities by population. (Using data from UrbanPopulation.txt)

ID	City	Population	Province
1	Shanghai	12286274	Shanghai
2	Beijing	9931140	Beijing
3	Chongqing	7421420	Chongqing
4	Guangzhou	7240465	Guangdong
5	Hong Kong	7010000	Hong Kong Special Administrative Region
…	…	…	…

Mega cities have a population of more than 2 million, super cities have a population of 1-2 million, big cities have a population of 500000-1000000, and others are small and medium-sized cities.

13.3.3 Enumeration grouping

Each member is only put in the first matching group

【Example 13.15】The following is the urban population table of the major cities in China. Classify the cities by population. (Using data from UrbanPopulation.txt)

Solution

According to the relationship between population and city size, the constant sequence of conditions is created. The function **P.enum(E, y)** calculates the expression y for each member in sequence P in turn. Taking the calculation result as a parameter, if a certain conditional expression in E holds, the record is assigned to the corresponding group. By default, each member is only stored in the first matching group.

ID	City	Population	Province
1	Shanghai	12286274	Shanghai
…	…	…	…

Member
?>2000000
?>1000000 && ?<=2000000
?>500000 && ?<=1000000
?<=500000

Define the city grouping conditions according to the rules.

The conditions can also be simplified, and arranged in the descending order of lower limit population of each group. Each member is only stored in the first matching group.

ID	City	Population	Province
1	Shanghai	12286274	Shanghai
…	…	…	…

Member
?>2000000
?>1000000
?>500000
?<=500000

Although Shanghai satisfies multiple conditions, it will only be assigned to the first matching group, namely the mega city group.

13.3.3 Enumeration grouping

Each member is only put in the first matching group

【Example 13.15】The following is the urban population table of the major cities in China. Classify the cities by population. (Using data from UrbanPopulation.txt)

	A	B
1	=file("UrbanPopulation.txt").import@t()	/Import Urban population table from file
2	[?>2000000,?>1000000,?>500000,?<=500000	/Create the constant sequence of grouping conditions
3	=A1.enum(A2, Population)	/Enumeration grouping of the urban population table according to the conditional sequence of A2. By default, each member is stored in the first matching group

A3

Member
[[1, Shanghai, 12286274, ...], [2, Beijing, 9931140, ...], …]
[[28, Changsha, 1965282, ...], [29, Nanchang, 1900817, ...], …]
[[69, Huainan, 974026, ...], [70, Haikou, 967336, ...], …]
[]

ID	City	Population	Province
1	Shanghai	12286274	Shanghai
2	Beijing	9931140	Beijing
3	Chongqing	7421420	Chongqing

【Example 13.16】Group employees by age interval [<35, 35 to 49, other], and calculate average salary. (Using data from Employee.txt)

ID	Name	Birthday	Salary
1	Rebecca	1974/11/20	7000
2	Ashley	1980/07/19	11000
3	Rachel	1970/12/17	9000
…	…	…	…

13.3.3 Enumeration grouping

Put unmatched members in a new group

【Example 13.16】Group employees by age interval [<35, 35 to 49, other], and calculate average salary. (Using data from Employee.txt)

Solution

The grouping condition may not match all the members. In this case, the unmatched members can be put in a new group. The **@n** option of the function **P.enum (E, y)** creates a new group at the end of the result set to hold unmatched members.

ID	Name	Birthday	Salary
1	Rebecca	1974/11/20	7000
2	Ashley	1980/07/19	11000
3	Rachel	1970/12/17	9000
4	Emily	1986/03/07	7000
…	…	…	…

Calculate age according to birthday

Age
46
40
50
34
…

Member
?<35
?<50

Enumeration grouping of age by conditions

An employee aged 50 does not meet any grouping conditions and is assigned to a new group.

13.3.3 Enumeration grouping

Put unmatched members in a new group

【Example 13.16】Group employees by age interval [<35, 35 to 49, other], and calculate average salary. (Using data from Employee.txt)

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	[?<35, ?<50]	/Create a sequence of constants for grouping conditions
3	=A1.enum@n(A2, age(Birthday))	/First, calculate the age according to the birthday. Then do enumeration grouping by age, and put the unmatched members in a new group
4	=A3.new(if(#>A2.len(),"Other",A2(#)):Age, round(~.avg(Salary), 2):AvgSalary)	/Calculate the average salary of each group, and set the name of the last group as other

A3	Member
	[[64,Nathan,1986/09/29,5000], [73,Justin,1986/11/03,10000], …]
	[[1,Rebecca,1974/11/20,7000],[2,Ashley,1980/07/19,11000], …]
	[[67,Cameron,1970/03/14,5000],[74,Kayla,1969/07/06,5000], …]

ID	Name	Birthday	Salary
64	Nathan	1986/09/29	5000
37	Justin	1986/11/03	10000
…	…	…	…

A4	Age	AvgSalary
	?<35	7080.36
	?<50	7454.88
	Other	7191.18

13.3.3 Enumeration grouping

【Example 13.17】The following is the table of urban GDP. Calculate the per capita GDP of municipalities, first tier cities and second tier cities respectively. (Using data from GDP.txt; GDP unit: 100 million yuan, population unit: 10000 people)

ID	City	GDP	Population
1	Shanghai	32679	2418
2	Beijing	30320	2171
3	Shenzhen	24691	1253
4	Guangzhou	23000	1450
5	Chongqing	20363	3372
…	…	…	…

13.3.3 Enumeration grouping

【Example 13.17】The following is the table of urban GDP. Calculate the per capita GDP of municipalities, first tier cities and second tier cities respectively. (Using data from GDP.txt; GDP unit: 100 million yuan, population unit: 10000 people)

Solution

Note that there may be repeated members during grouping. For example, Beijing is both a first tier city and a municipality directly under the central government. The **@r** option of the function **P.enum(E, y)** is used for repeated enumeration, that is, a record of P is put in multi matching groups.

When making statistics on municipalities and first tier cities, Beijing should be included to both and is divided to two groups repeatedly.

ID	City	GDP	Population
…	…	…	…
2	Beijing	30320	2171
…	…	…	…

Member
municipalities
first-tier cities
second-tier cities

13.3.3 Enumeration grouping

【Example 13.17】The following is the table of urban GDP. Calculate the per capita GDP of municipalities, first tier cities and second tier cities respectively. (Using data from GDP.txt; GDP unit: 100 million yuan, population unit: 10000 people)

	A	B
1	=file("GDP.txt").import@t()	/Import GDP table from file
2	[Beijing,Shanghai,Tianjin,Chongqing]	/Define constant sequence of municipalities
3	[Beijing,Shanghai,Guangzhou,Shenzhen]	/Define constant sequence of first-tier cities
4	[Chengdu,Hangzhou,Chongqing,Wuhan,Xian,Suzhou,Tianjing,Nanjing,Changsha,Zhengzhou,Dongguan,Qingdao,Shenyang,Ningbo,Kunming]	/Define constant sequence of second-tier cities
5	[A2.pos(?)>0,A3.pos(?)>0, A4.pos(?)>0]	/Define constant sequence of enumeration conditions
6	=A1.enum@r(A5,City)	/Enumeration grouping by city
7	=A6.new(["Municipality","First-tier","Second-tier"](#):Area,~.sum(GDP)/~.sum(Population)*10000:CapitaGDP)	/Calculate the per capita GDP of each group

A6	Member
	[[1,Shanghai,32679,2418],[1,Beijing,30320,2171],...]
	[[1,Shanghai,32679,2418],[1,Beijing,30320,2171],...]
	[[5,Chongqing,20363,3372],[6,Tianjin,18809,1557],...]

ID	City	GDP	Population
1	Shanghai	32679	2418
…	…	…	…

A7	Area	CapitaGDP
	Municipality	107345.03
	First-tier	151796.49
	Second-tier	106040.57

Chapter 13
Grouping calculations

13.4 Order of grouping

13.4.1 Order of grouping

Previously, when performing **grouping / group aggregation**, by default, the grouping result was sorted by grouping field in ascending order. But sometimes we don't care about the **order** of the result set, and we **don't need to sort** the result set.

Group by subject:

SID	Subject	Score
1	Math	93
2	Math	97
1	Chinese	79
2	Chinese	95

Not sorted

Subject	AvgScore
Math	95
Chinese	87

Sorted

(by subject in ascending order)

Subject	AvgScore
Chinese	87
Math	95

13.4.1 Order of grouping

【Example 13.18】Query the top two sales records of each customer's orders in 2014. (Using data from Sales.txt)

OrderID	Customer	OrderDate	Amount
10400	EASTC	2014/01/01	3063.0
10401	RATTC	2014/01/01	3868.6
10402	ERNSH	2014/01/02	2713.5
10403	ERNSH	2014/01/03	1005.9
10404	EASTC	2014/01/03	1675.0
10405	RATTC	2014/01/06	400.0
10406	EASTC	2014/01/07	2018.2
10407	ERNSH	2014/01/07	1194.0
…	…	…	…

13.4.1 Order of grouping

【Example 13.18】Query the top two sales records of each customer's orders in 2014. (Using data from Sales.txt)

Solution

Group the sales table by customer, and select the top two sales records for each grouped subset. The **@u** option of **A.group(xi, …)** function is used when sorting is not required after grouping, and sorting is not performed for grouping field.

OrderID	Customer	OrderDate	Amount
10400	EASTC	2014/01/01	3063.0
10401	RATTC	2014/01/01	3868.6
10402	ERNSH	2014/01/02	2713.5
10403	ERNSH	2014/01/03	1005.9
10404	EASTC	2014/01/03	1675.0
10405	RATTC	2014/01/06	400.0
10406	EASTC	2014/01/07	2018.2
10407	ERNSH	2014/01/07	1194.0
…	…	…	…

Keep original order

Member
[[10400, EASTC, 2014/01/01, 3063.0], …]
[[10401, RATTC, 2014/01/01, 3868.6], …]
[[10402, EASTC, 2014/01/02, 2713.5], …]

OrderID	Customer	OrderDate	Amount
10400	EASTC	2014/01/01	3063.0
10404	EASTC	2014/01/03	1675.0
10406	EASTC	2014/01/07	2018.2

13.4.1 Order of grouping

【Example 13.18】Query the top two sales records of each customer's orders in 2014. (Using data from Sales.txt)

	A	B
1	=file("Sales.txt").import@t().select(year(OrderDate)==2014)	/Import the sales table from the file and select the sales records of 2014
2	=A1.group@u(Customer)	/Use the group function to group by customer and do not sort after grouping
3	=A2.conj(~.top(-2; Amount))	/Select the top two sales records for each customer

A2	Member
	[[10400, EASTC, 2014/01/01, 3063.0], …]
	[[10401, RATTC, 2014/01/01, 3868.6], …]
	…

A3	OrderID	Customer	OrderDate	Amount
	10400	EASTC	2014/01/01	3063.0
	10406	EASTC	2014/01/07	2018.2
	…	…	…	…

Note

- In this example, the sales table is ordered by date. When the @u option is added, the result set is not sorted. If there is no option, the result set will be sorted by default, that is, in the order of customer names.

13.4.1 Order of grouping

【Example 13.19】Calculate the starting and ending dates of each shift for each employee (e.g. Emily is on duty from March 1, 2018 to March 3, 2018), and the duty list is in ascending order by date. (Using data from Duty.txt)

Date	Name
2018/03/01	Emily
2018/03/02	Emily
2018/03/03	Emily
2018/03/04	Johnson
2018/03/05	Ashley
2018/03/06	Emily
2018/03/07	Emily
2018/03/08	Ashley
2018/03/09	Ashley
2018/03/10	Ashley
…	…

13.4.1 Order of grouping

【Example 13.19】Calculate the starting and ending dates of each shift for each employee (e.g. Emily is on duty from March 1, 2018 to March 3, 2018), and the duty list is in ascending order by date. (Using data from Duty.txt)

Solution

Put the consecutive employees into a group and start a new group when the value of name field changes. The **@o** option of **A.group(xi, ...)** function only compares the grouping field value with the adjacent value, and the result set is no longer sorted.

Start a new group when name changes

Date	Name
2018/03/01	Emily
2018/03/02	Emily
2018/03/03	Emily
2018/03/04	Johnson
2018/03/05	Emily
…	…

Date	Name
2018/03/01	Emily
2018/03/02	Emily
2018/03/03	Emily

Date	Name
2018/03/04	Johnson

Date	Name
2018/03/05	Emily

13.4.1 Order of grouping

【Example 13.19】Calculate the starting and ending dates of each shift for each employee (e.g. Emily is on duty from March 1, 2018 to March 3, 2018), and the duty list is in ascending order by date. (Using data from Duty.txt)

	A	B
1	=file("Duty.txt").import@t()	/Import duty table from file
2	=A1.group@o(Name)	/Use the @o option with group function, start a new group when the name changes
3	=A2.new(Name,~(1).Date:Begin,~.m(-1).Date:End)	/Because the dates are ordered, the first date in each group is the start date and the last date is the end date.

A2	Member
	[[2018/03/01, Emily], [2018/03/02, Emily], …]
	[[2018/03/04, Johnson]]
	[[2018/03/05, Emily]]
	…

Date	Name
2018/03/01	Emily
2018/03/02	Emily
2018/03/03	Emily

A3	Name	Begin	End
	Emily	2018/03/01	2018/03/03
	Johnson	2018/03/04	2018/03/04
	Emily	2018/03/05	2018/03/05
	…	…	…

13.4.2 Ordered grouping

For an ordered set, does the order of the original data set have significance when grouping? The answer is yes. Next, we will learn about three kinds of ordered grouping:

- **Group by sequence number**

  Group by the calculated sequence numbers.

- **Group by value change**

  Start a new group when the value of grouping field changes.

- **Group by condition change**

  Start a new group when the result of grouping condition is true.

13.4.2 Ordered grouping

Group by sequence number

【Example 13.20】Organize the log file into structured data (a table sequence with the fields of USERID, UNAME, IP, TIME, URL, BROWSER, LOCATION, MODULE). (Using data from log.txt)

The format of the existing log is as follows:

The first line is IP, TIME, GET, URL, BROWSER;

The second line MODULE;

The third line USERID, UNAME, LOCATION

```
10.10.10.143        2013-04-01 21:14:44        GET        /p/pt301/index.jsp  Mozilla/6.0
#module:production#
47356    Jessica    Chicago
10.10.2.76          2013-04-01 21:18:50        GET        /h/homepage.jsp     Chrome/35
#module:homepage#
419      Jacob      Houston
10.10.54.218        2013-04-01 21:25:19        GET        /p/pt27/index.jsp   Mozilla/6.0
#module:production#
3464     Madison    Detroit
10.10.10.145        2013-04-01 21:30:02        GET        /u/userlist/showlist.jsp          Mozilla/6.0
#module:usercenter#
432442   Phoenix    San Jose
10.2.1.242          2013-04-01 21:30:15        GET        /p/pt271/index.jsp  Mozilla/6.0
#module:production#
3435567  Megan      San Jose
```

13.4.2 Ordered grouping

【Example 13.20】Organize the log file into structured data (a table sequence with the fields of USERID, UNAME, IP, TIME, URL, BROWSER, LOCATION, MODULE). (Using data from log.txt)

Solution

Divide the log into a group every three lines. Then, split the members of each group according to "\t" and generate structured data by position. The function **A.group(x)** puts records with the same calculated result of x to a group.

(sequence number - 1) divided by 3, records with the same result are put in the same group:

Index	Member
1	10.10.10.143 2013-04-01 21:14:44 GET /p/pt301/index.jsp Mozilla/6.0
2	#module:production#
3	47356 Jessica Chicago
4	10.10.2.76 2013-04-01 21:18:50 GET /h/homepage.jsp Chrome/35
5	#module:homepage#
6	419 Jacob Houston
…	…

Result is 0

Index	Member
1	10.10.10.143 2013-04-01 21:14:44 GET /p/pt301/index.jsp Mozilla/6.0
2	#module:production#
3	47356 Jessica Chicago

Result is 1

Index	Member
1	10.10.2.76 2013-04-01 21:18:50 GET /h/homepage.jsp Chrome/35
2	#module:homepage#
3	419 Jacob Houston

13.4.2 Ordered grouping

Group by sequence number

【Example 13.20】Organize the log file into structured data (a table sequence with the fields of USERID, UNAME, IP, TIME, URL, BROWSER, LOCATION, MODULE). (Using data from log.txt)

	A	B
1	=file("log.txt").read@n()	/Read a sequence of strings by line from file
2	=A1.group((#-1)\3)	/Use the group function to group every three rows
3	=A2.(~.conj(~.split("\t")))	/Each group splits each row by "\t" and concatenates the content into a sequence
4	=A3.new(~(7):USERID, ~(8):UNAME, ~(1):IP, ~(2):TIME, ~(4):URL, ~(5):BROWSER, ~(9):LOCATION, left(~(6).array("\:")(2), -1):MODULE)	/Generate structured data

A2

Member
[[10.10.10.143 2013-04-01 21:14:44 GET /p/pt301/index.jsp Mozilla/6.0], …]
[[10.10.2.76 2013-04-01 21:18:50 GET /h/homepage.jsp Chrome/35], …]

Member
10.10.10.143 2013-04-01 21:14:44 GET …
#module:production#
47356 Jessica Chicago

A4

USERID	UNAME	IP	TIME	URL	BROWSER	LOCATION	MODULE
47356	Jessica	10.10.10.143	2013-04-01 21:14:44	/p/pt301/index.jsp	Mozilla/6.0	Chicago	production
419	Jacob	10.10.2.76	2013-04-01 21:18:50	/h/homepage.jsp	Chrome/35	Houston	homepage



【Example 13.21】Which country has been ranked the first place in the Olympic medal list continuously for the most times and What are the games? (Using data from Olympic.txt)

Game	Nation	Gold	Silver	Copper
30	USA	46	29	29
30	China	38	27	23
30	UK	29	17	19
30	Russia	24	26	32
30	Korea	13	8	7
30	Germany	11	19	14
30	France	11	11	12
30	Italy	8	9	11
30	Hungary	8	4	5
30	Australia	7	16	12
29	China	51	21	28
…	…	…	…	…

13.4.2 Ordered grouping

Group by value change

【Example 13.21】Which country has been ranked the first place in the Olympic medal list continuously for the most times and What are the games? (Using data from Olympic.txt)

Solution

Perform **value change grouping** to data in the order of sequence numbers, and start a new group when the field value changes.

Start a new group when the value of Nation field changes

Game	Nation	Gold	Silver	Copper
30	USA	46	29	29
29	China	51	21	28
28	USA	35	39	29
27	USA	37	24	31
26	USA	44	32	25
…	…	…	…	…

Game	Nation	Gold	Silver	Copper
30	USA	46	29	29

Game	Nation	Gold	Silver	Copper
29	China	51	21	28

Game	Nation	Gold	Silver	Copper
28	USA	35	39	29
27	USA	37	24	31
26	USA	44	32	25

13.4.2 Ordered grouping

Group by value change

【Example 13.21】Which country has been ranked the first place in the Olympic medal list continuously for the most times and What are the games? (Using data from Olympic.txt)

	A	B
1	=file("Olympic.txt").import@t()	/Import Olympic table from file
2	=A1.group@o1(Game).group@o(Nation)	/Reserve only the first place for each game. Use the @o option with the group function to start a new group when the value of Nation field changes
3	=A2.maxp(~.len())	/Select the records of the country with the max number of consecutive champions

A2	Member
	[[30,USA,46,29,29]]
	[[29,China,51,21,28]]
	[[28,USA,35,39,29],[27,USA,37,24,31],...]
	[[25,Joint team (CIS),45,38,29]]
	…

Game	Nation	Gold	Silver	Copper
28	USA	35	39	29
27	USA	37	24	31
26	USA	44	32	25

A3	Game	Nation	Gold	Silver	Copper
	10	USA	41	32	30
	9	USA	22	18	16
	8	USA	45	27	27
	7	USA	41	27	28

【Example 13.22】What's the maximum number of continuous rising days of SSEC index in 2019? (Using data from SSEC.txt; the index of the first trading day is regarded as rising)

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
…	…	…	…

13.4.2 Ordered grouping

Group by condition change

【Example 13.22】What's the maximum number of continuous rising days of SSEC index in 2019? (Using data from SSEC.txt; the index of the first trading day is regarded as rising)

Solution

Start a new group when condition (closing price <= previous day's closing price) changes. The **@i** option of function **A.group(x)** starts a new group if the result of x is true, where x is a bool expression. Option @i can only have one expression x.

Start a new group when closing price does not rise

Date	Open	Close	Amount
2019/01/02	2497.8805	2465.291	9.76E10
2019/01/03	2461.7829	2464.3628	1.07E11
2019/01/04	2446.0193	2514.8682	1.39E11
2019/01/07	2528.6987	2533.0887	1.46E11
2019/01/08	2530.3001	2526.4622	1.23E11
2019/01/09	2536.4173	2544.3449	1.61E11
2019/01/10	2543.8534	2535.0985	1.33E11
…	…	…	…

Date	Open	Close	Amount
2019/01/02	2497.8805	2465.291	9.76E10

Date	Open	Close	Amount
2019/01/03	2461.7829	2464.3628	1.07E11
2019/01/04	2446.0193	2514.8682	1.39E11
2019/01/07	2528.6987	2533.0887	1.46E11

Date	Open	Close	Amount
2019/01/08	2530.3001	2526.4622	1.23E11
2019/01/09	2536.4173	2544.3449	1.61E11

13.4.2 Ordered grouping

Group by condition change

【Example 13.22】What's the maximum number of continuous rising days of SSEC index in 2019? (Using data from SSEC.txt; the index of the first trading day is regarded as rising)

	A	B
1	=file("SSEC.txt").import@t()	/Import SSEC table from file
2	=A1.select(year(Date)==2019).sort(Date)	/Select data in 2019 and sort it by date
3	=A2.group@i(Close<=Close[-1])	/Use the @i option with group function to start a new group when condition changes(the closing price of the day doesn't rise). Thus the records of each group are records of continuous rising.
4	=A3.max(~.len())	/Calculate the maximum days

A3	Member
	[[2019/01/02,2497.8805,2465.291,9.76E10]]
	[[2019/01/03,2461.7829,2464.3628,1.07E11], …]
	[[2019/01/08,2530.3001,2526.4622,1.23E11], ...]
	…

Date	Open	Close	Amount
2019/01/03	2461.7829	2464.3628	1.07E11
2019/01/04	2446.0193	2514.8682	1.39E11
2019/01/07	2528.6987	2533.0887	1.46E11

A4	Value
	7

Chapter 13
Grouping calculations

13.5 Exercises

13.5 Exercises

【Exercise 13.1】Calculate the total score of each student in each class. (Using data from Score.txt)

CLASS	STUDENTID	SUBJECT	SCORE
1	1	English	84
1	1	Math	77
1	1	Chinese	69
2	1	English	96
2	1	Math	63
2	1	Chinese	81
1	2	English	81
1	2	Math	80
1	2	Chinese	97
…	…	…	…

13.5 Exercises

【Exercise 13.1】Calculate the total score of each student in each class. (Using data from Score.txt)

Solution

Use **A.groups()** function to group by CLASS and STUDENTID fields and calculate sum of scores.

	A	B
1	=file("Score.txt").import@t()	/Import score table from file
2	=A1.groups(CLASS, STUDENTID; sum(SCORE):TotalScore)	/Calculate the total score of each student in each class by group aggregation

A2	CLASS	STUDENTID	TotalScore
	1	1	230
	1	2	258
	1	3	228
	1	4	240
	…	…	…

【Exercise 13.2】Calculate the number of sales of each product in 2014. (Using data from Sales.txt)

OrderID	ProductID	OrderDate	Amount
10400	27	2014/01/01	3063.0
10401	17	2014/01/01	3868.6
10402	70	2014/01/02	2713.5
…	…	…	…

13.5 Exercises

【Exercise 13.2】Calculate the number of sales of each product in 2014. (Using data from Sales.txt)

Solution

Use **A.groups()** function to perform group aggregation and count the numbers.

	A	B
1	=file("Sales.txt").import@t().select(year(OrderDate)==2014)	/Import sales table from file and select the 2014 records
2	=A1.groups(ProductID; count(~):Count)	/Group and count the sales number of each product

A2	ProductID	Count
	1	7
	2	4
	3	7
	…	…

【Exercise 13.3】What are the customers in 2014. (Using data from Sales.txt)

OrderID	Customer	OrderDate	Amount
10400	EASTC	2014/01/01	3063.0
10401	HANAR	2014/01/01	3868.6
10402	ERNSH	2014/01/02	2713.5
…	…	…	…

13.5 Exercises

【Exercise 13.3】What are the customers in 2014. (Using data from Sales.txt)

Solution

Function **A.id()** can be used to deduplicate.

	A	B
1	=file("Sales.txt").import@t().select(year(OrderDate)==2014)	/Import sales table from file and select the 2014 records
2	=A1.id(Customer)	/Use the ID function to deduplicate the customers and return the unique values of the customers

A2	Member
	ANATR
	ANTON
	AROUT
	…

【Exercise 13.4】Check the scores of students whose total score is higher than 240. (Using data from Score.txt)

CLASS	STUDENTID	SUBJECT	SCORE
1	1	English	84
1	1	Math	77
1	1	Chinese	69
…	…	…	…

13.5 Exercises

【Exercise 13.4】Check the scores of students whose total score is higher than 240. (Using data from Score.txt)

Solution

Use function **A.group()** to group by STUDENTID, and select the subsets that meet the condition(total score is higher than 240).

	A	B
1	=file("Score.txt").import@t()	/Import score table from file
2	=A1.group(STUDENTID)	/Group by STUDENTID
3	=A2.select(~.sum(SCORE)>240).conj()	/Select the subsets with total score higher than 240 and concatenate them

A2

Member
[[1,1,English,84],[1,1,Math,77],[1,1, Chinese,69]]
[[1,2,English,97],[1,2,Math,80],[1,2,Chinese,81]]
[[1,3,English,75],[1,3,Math,86],[1,3,Chinese,67]]
…

A3

CLASS	STUDENTID	SUBJECT	SCORE
1	2	English	97
1	2	Math	80
1	2	Chinese	81
…	…	…	…

【Exercise 13.5】Calculate the login times of each user within three days of the last login. (Using data from LoginLog.txt)

ID	UserID	IP	LoginTime
1	1	37.17.184.11	2012/05/09 09:01:10
2	3	61.134.201.1	2012/05/09 09:02:43
3	7	124.114.171.101	2012/05/09 09:03:18
…	…	…	…

13.5 Exercises

【Exercise 13.5】Calculate the login times of each user within three days of the last login. (Using data from LoginLog.txt)

Solution

Use the **A.group()** function to group by user ID, find the last login time in each grouped subset, and then calculate the number of logins in the previous three days.

	A	B
1	=file("LoginLog.txt").import@t()	/Import log file from file
2	=A1.group(UserID)	/Group by UserID
3	=A2.new(UserID, ~.max(LoginTime):Last, ~.count(interval(LoginTime, Last)<=3):Count)	/Calculate the last login time of each user, and count the number of records within 3 days of the last login time.

A2	Member
	[[1, 1, 37.17.184.11, 2012/05/09 09:01:10], ...]
	[[4, 2, 183.202.48.25, 2012/05/09 09:05:15], ...]
	…

ID	UserID	IP	LoginTime
4	2	183.202.48.25	2012/05/09 09:05:15
…	…	…	…

A3	UserID	Last	Count
	1	2012/05/09 09:12:41	2
	2	2012/05/09 09:09:41	2
	…	…	…

【Exercise 13.6】Find all the scores of the top three students in English. (Using data from Score.txt)

CLASS	STUDENTID	SUBJECT	SCORE
1	1	English	84
1	1	Math	77
1	1	Chinese	69
…	…	…	…

13.5 Exercises

【Exercise 13.6】Find all the scores of the top three students in English. (Using data from Score.txt)

Solution

Find the top three students in English first, and then perform alignment grouping according to this order. The **@a** option can be used with the **P.align(A, y)** function to return all matching members.

	A	B
1	=file("Score.txt").import@t()	/Import score table from file
2	=A1.select(SUBJECT:"English").top(-3; SCORE)	/Select the records with top3 English scores
3	=A2.(STUDENTID)	/Select the STUDENTIDs of the top3 English scores
4	=A1.align@a(A3, STUDENTID).conj()	/Perform alignment grouping by STUDENTID

A3	Member
	4
	9
	6

A4	CLASS	STUDENTID	SUBJECT	SCORE
	1	4	English	96
	1	4	Math	63
	1	4	Chinese	81
	…	…	…	…

13.5 Exercises

【Exercise 13.7】Find the number of excellent, good, normal and fail students respectively according to the average score. (Using data from Score.txt)

CLASS	STUDENTID	SUBJECT	SCORE
1	1	English	84
1	1	Math	77
1	1	Chinese	69
…	…	…	…

85-100 is excellent, 75-84 is good, 60-74 is normal, and below 60 is fail

13.5 Exercises

【Exercise 13.7】Find the number of excellent, good, normal and fail students respectively according to the average score. (Using data from Score.txt)

Solution

Perform group aggregation first to calculate the average score of each student, and then perform enumeration grouping according to conditions. The **@n** option is used with **P.enum(E, y)** function to create a new group at the end of the result set to store unmatched members.

	A	B
1	=file("Score.txt").import@t()	/Import score table from file
2	[?>=85,?>=75,?>=60]	/Create a constant sequence of grouping conditions
3	=A1.groups(STUDENTID; avg(SCORE):AvgScore)	/Perform group aggregation to calculate the average score of each student
4	=A3.enum@n(A2,AvgScore)	/Perform enumeration grouping by the average score according to A2 conditions
5	=A4.new(["Excellent","Good","Normal","Fail"](#):Level, ~.len():Count)	/Count the number of members in each group

A3	STUDENTID	AvgScore
	1	76.67
	2	86.0
	3	76.0
	…	…

A4	Member
	[2]
	[1,3,4,6,7,9,10]
	[5,8,11,12,13,14]
	[]

A5	Level	Count
	Excellent	1
	Good	7
	Normal	6
	Fail	0

【Exercise 13.8】Count the maximum number of continuous empty seats in the cinema. (Using data from Cinema.txt)

Row	Col	Empty
1	1	YES
1	2	YES
1	3	NO
…	…	…

13.5 Exercises

【Exercise 13.8】Count the maximum number of continuous empty seats in the cinema. (Using data from Cinema.txt)

Solution

It's a matter of ordered grouping. Use the **@o** option of **A.group()** function to only compare the adjacent records. When the row number changes (those in the front and back rows are not adjacent), or when there is a change in the value of Empty field, start a new group.

	A	B
1	=file("Cinema.txt").import@t()	/Import cinema table from file
2	=A1.group@o(Row, Empty; ~.count():Count)	/Use the @o option of the group function to start a new group when the value changes
3	=A2.select(Empty:"YES").max(Count)	/Select the subsets of empty seats and calculate the max number of members

A2	Row	Empty	Count
	1	YES	2
	1	NO	4
	1	YES	3
	…	…	…

A3	Value
	7

【Exercise 13.9】Find the longest consecutive days for both the closing price and trading amount of SSEC index to rise in 2019. (Using data from SSEC.txt; the first trading day is regarded as rising in both closing price and amount)

Date	Open	Close	Amount
2019/12/31	3036.3858	3050.124	2.27E11
2019/12/30	2998.1689	3040.0239	2.67E11
2019/12/27	3006.8517	3005.0355	2.58E11
2019/12/26	2981.2485	3007.3546	1.96E11
…	…	…	…

13.5 Exercises

【Exercise 13.9】Find the longest consecutive days for both the closing price and trading amount of SSEC index to rise in 2019. (Using data from SSEC.txt; the first trading day is regarded as rising in both closing price and amount)

Solution

It's a matter of ordered grouping. Use the **@i** option of the **A.group()** function to start a new group when conditions change (closing price does not rise or trading amount does not rise).

	A	B
1	=file("SSEC.txt").import@t()	/Import SSEC index table from file
2	=A1.select(year(Date)==2019).sort(Date)	/Select the records of 2019 and sort by date
3	=A2.group@i(Close<=Close[-1] || Amount<=Amount[-1])	/Use the @i option with the group function to start a new group when condition changes
4	=A3.max(~.len())	/Calculate the maximum number of members, that is, the maximum number of days

A3	Member
	[[2019/01/02, 2497.8805, 2465.291, 9.76E10]]
	[[2019/01/03, 2461.7829, 2464.3628, 1.07E11], …]
	…

Date	Open	Close	Amount
2019/01/03	2461.7829	2464.3628	1.07E11
…	…	…	…

A4	Value
	4

【Exercise 13.10】Find the longest days of continuous rise of various indexes in 2019. (Using data from StockIndex.txt; the first trading day is regarded as rising in closing price)

Date	Code	Open	Close	Amount
2019/1/2	399001	7259.489	7149.2665	63756033352
2019/1/2	399006	1250.623	1228.7668	14891216036
2019/1/2	399007	3046.3877	2998.0988	51134636181
…	…	…	…	…

13.5 Exercises

【Exercise 13.10】Find the longest days of continuous rise of various indexes in 2019. (Using data from StockIndex.txt; the first trading day is regarded as rising in closing price)

Solution

First, group by index code, and then perform ordered grouping for each subset. Use the **@u** option of the **A.group()** function to group without sorting the result set. Use the **@i** option with the **A.group()** function to start a new group when condition changes (closing price does not rise).

	A	B
1	=file("StockIndex.txt").import@t().select(year(Date)==2019)	/Import index table from file and select records of 2019
2	=A1.group@u(Code)	/Group by Code without sorting the result set
3	=A2.(~.sort(Date).group@i(Close<=Close[-1]))	/Use the @i option of the group function for each subset to start a new group when the condition changes
4	=A3.new(~.Code:Code, ~.max(~.len()):Day)	/Calculate the maximum days of continuous rise of each index respectively

13.5 Exercises

【Exercise 13.10】Find the longest days of continuous rise of various indexes in 2019. (Using data from StockIndex.txt; the first trading day is regarded as rising in closing price)

A2	Member
	[[2019/01/02,399001,7259.489,7149.2665,6.38E10], …]
	[[2019/01/02,399006,1250.623,1228.7668,1.49E10], …]
	…

Date	Code	Open	Close	Amount
2019/01/02	399001	7259.489	7149.2665	6.38E10
2019/01/03	399001	7134.1461	7089.4406	6.99E10
…	…	…	…	…

A3	Member
	[[[2019/01/02,399001,7259.489,7149.2665,6.38E10], …],…]
	[[[2019/01/02,399006,1250.623,1228.7668,1.49E10], …],…]
	…

Member
[[2019/01/02,399001,7259.489,7149.2665,6.38E10]]
[[2019/01/03,399001,7134.1461,7089.4406,6.99E10], …]
…

Note that after two groupings, the value of A3 is the set of sets of sets.

Date	Code	Open	Close	Amount
2019/01/03	399001	7134.1461	7089.4406	6.99E10
2019/01/04	399001	7022.1229	7284.8403	9.38E10
2019/01/07	399001	7333.3171	7400.2046	9.95E10

A4	Code	Day
	399001	7
	399006	7
	…	…

Chapter 14
Inter table association

14.1 Primary key and index

- Find a record by primary key value
- Index table

14.2 Inter table association

- Inter table association
- Common associated relationships

14.3 Foreign key table

14.4 Same dimension table and main sub table

14.5 Cross join and non equivalent join

- Cross join
- Non equivalent join

14.6 Exercises

Chapter 14
Inter table association

14.1 Primary key and index

14.1.1 Find a record by primary key value

A **primary key** is one or more fields in a table sequence. Its value is used to **uniquely identify** a record in the table sequence.

ID is the primary key of the employee table. Each ID can identify a **unique** employee. Find the employee with ID 2:

ID	Name	Birthday	EntryDate	Department	Salary
1	Rebecca	1974/11/20	2005/03/11	6	7000
2	Ashley	1980/07/19	2008/03/16	2	11000
3	Rachel	1970/12/17	2010/12/01	7	9000

14.1.1 Find a record by primary key value

【Example 14.1】Find the sales amount of order with OrderID 28571610. (Using data from Sales.txt)

OrderID	Customer	SellerId	OrderDate	Amount
81182311	VINET	5	2013/07/04	2440.0
98807954	TOMSP	6	2013/07/05	1863.4
…	…	…	…	…
28571610	MAISD	3	2015/04/07	295.38
…	…	…	…	…

14.1.1 Find a record by primary key value

【Example 14.1】Find the sales amount of order with OrderID 28571610. (Using data from Sales.txt)

OrderID	Customer	SellerId	OrderDate	Amount
81182311	VINET	5	2013/07/04	2440.0
98807954	TOMSP	6	2013/07/05	1863.4
…	…	…	…	…
28571610	MAISD	3	2015/04/07	295.38
…	…	…	…	…

Solution

The **primary key value** of the table can be used to find the specified record.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=A1.keys(OrderID)	/Set the primary key of sales table as OrderID
3	=A2.find(28571610)	/Use A.find(k) function to find the record with primary key value 28571610
4	=A3.Amount	/Get the sales amount from the record

A4
295.38

Note

- The function **A.find(k)** is used to find the matching record by the primary key value k .

14.1.1 Find a record by primary key value

【Example 14.1】Find the sales amount of order with OrderID 28571610. (Using data from Sales.txt)

Index	OrderID	Customer	SellerId	OrderDate	Amount
1	81182311	VINET	5	2013/07/04	2440.0
2	98807954	TOMSP	6	2013/07/05	1863.4
…	…	…	…	…	…
757	28571610	MAISD	3	2015/04/07	295.38
…	…	…	…	…	…

Solution

You can also find the sequence number of the record where the **primary key value** of the table is located, and then perform other calculations.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=A1.keys(OrderID)	/Set the primary key of sales table as OrderID
3	=A2.pfind(28571610)	/Use the function A.pfind(k) to find the sequence number of the sales record with the primary key value of 28571610
4	=A2(A3).Amount	/Get the sales amount from the record

A3
757

A4
295.38

Note

- The function **A.pfind(k)** is used to find the sequence number of the matching record according to the primary key value k.

14.1.1 Find a record by primary key value

By default, to find a record by primary key value is carried out by traversal, and the records are compared one by one until the target record is found.

Compare in order whether the primary key value is equal to 28571610. In row 757, the primary key value is matched. A total of 757 comparisons are made.

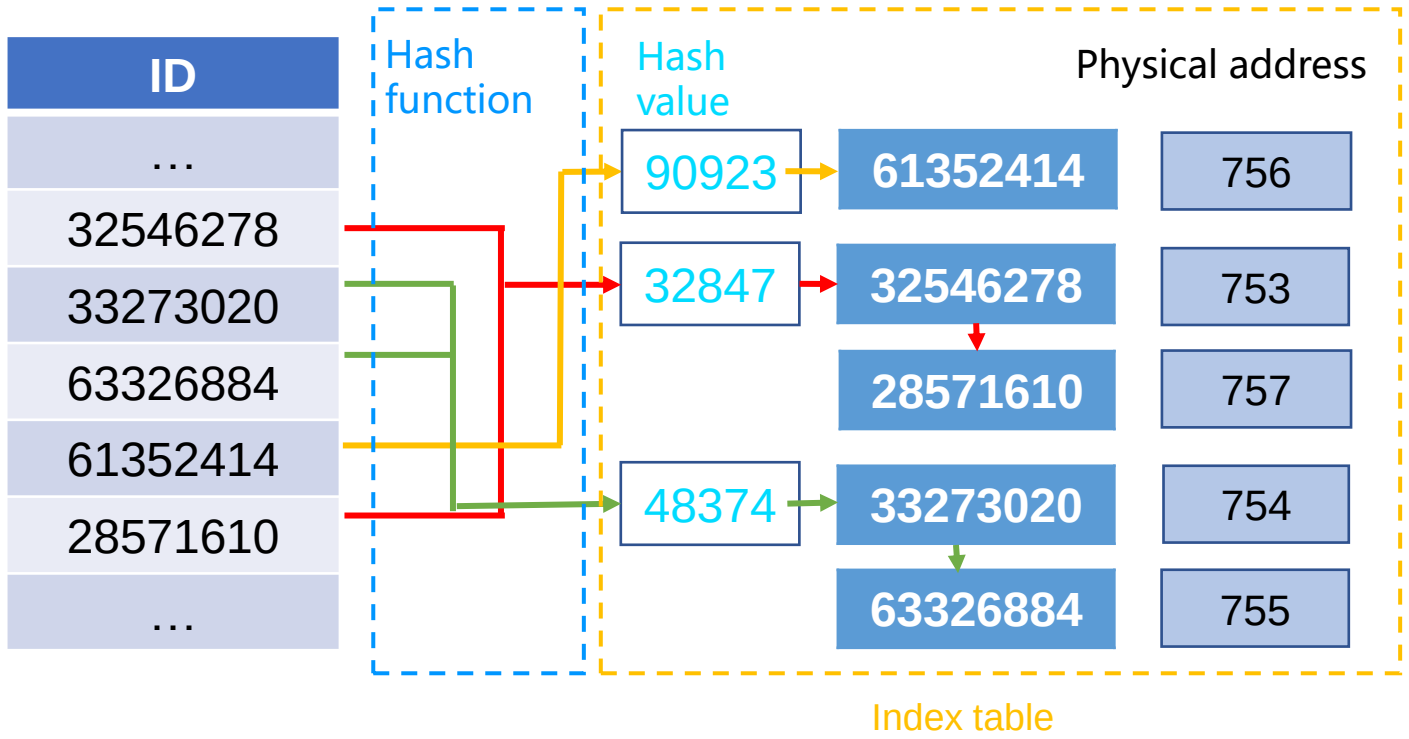
OrderID	Customer	SellerId	OrderDate	Amount
81182311	VINET	5	2013/07/04	2440.0
98807954	TOMSP	6	2013/07/05	1863.4
…	…	…	…	…
28571610	MAISD	3	2015/04/07	295.38
…	…	…	…	…

757 rows

The efficiency of traversal search is very poor. For the sequence table of N records, the average comparison is N/2 times.

14.1.2 Index table

To solve the problem of inefficient traversal search, we can use **index table**. The function of index is equivalent to the catalog of books, which can quickly find the required content according to the page number in the catalog.



14.1.2 Index table

Generate the **index table** by the **primary key**, and the efficiency will be obviously improved when searching by the **primary key**.

Search by traversal:

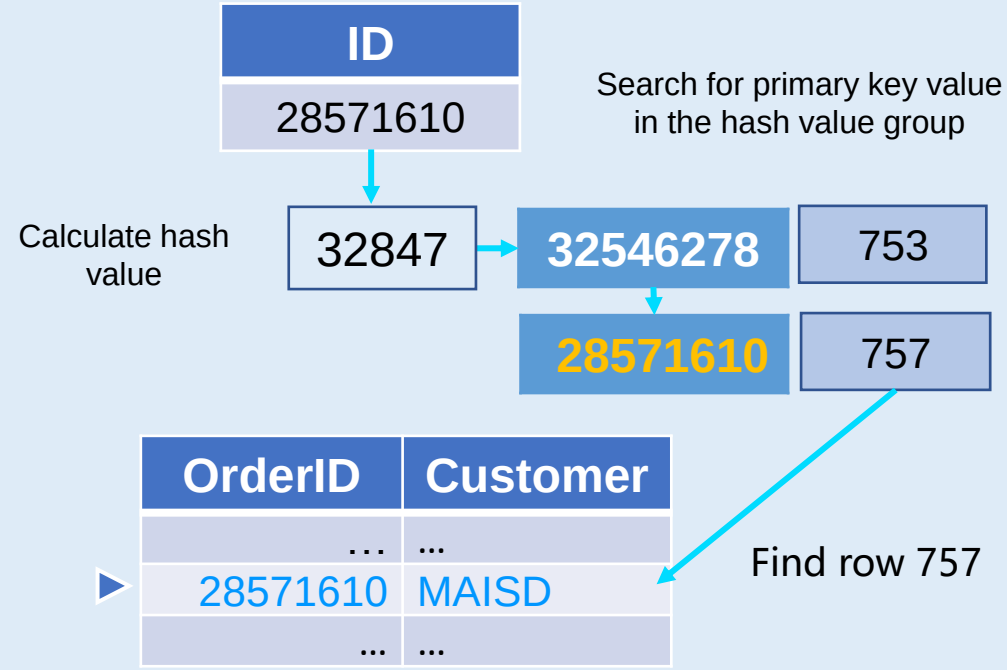
The table sequence of N records is searched **N/2** times on average. When the amount of data is large, the search efficiency is significantly reduced.

Traversal search 757 times

Index	OrderID	Customer
1	81182311	VINET
2	98807954	TOMSP
…	…	…
757	28571610	MAISD
…	…	…

Search by index:

The records are divided into k groups according to the hash value and are searched **N/2k** times on average. When k is larger, the search efficiency is higher. When the amount of data is large, it is obviously better than traversal search.



14.1.2 Index table

【Example 14.1】Find the sales amount of order with OrderID 28571610. How to improve search efficiency?

OrderID	Customer	SellerId	OrderDate	Amount
81182311	EASTC	1	2014/01/01	3063.0
98807954	HANAR	1	2014/01/01	3868.6
…	…	…	…	…
28571610	BLAUS	4	2015/07/17	3774.05
…	…	…	…	…

Solution

Creating a **primary key index** can improve efficiency and locate quickly when searching by the primary key.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=A1.keys@i(OrderID)	/Set the primary key of sales table as OrderID and create an index
3	=A2.find(28571610)	/Use A.find(k) function to find the record with primary key value 28571610
4	=A3.Amount	/Get the sales amount from the record

A4
3774.05

Note

- The **T.keys@i(k)** function is used to create a primary key index when setting the primary key. The index table only needs to be created once, and the processing of the primary key will automatically use the index.

Chapter 14
Inter table association

14.2 Inter table association

14.2.1 Inter-table association

The Department field in the employee table stores the ID of the Department to which the employee belongs, which is either null or comes from the value of the ID field of the Department table.

Employee table

ID	Name	Department
1	Rebecca	6

Department table

ID	Name
6	R&D

The data dependence relationship between employee table and department table is called inter table association. The associated relationship between tables can be divided into the following three types:

- 1:1
- 1:N
- M:N

14.2.1 Inter-table association

One to one association: One record in a table can only correspond to one record in another table; vice versa.

Student table

ID	Name	Birthday
1	Rebecca	2010/09/08

1 : 1

Student contact table

ID	Contacts	Address
1	Mrs. Moore	124 Guangming North Road

As shown in the figure above, each ID in the student table is unique, and each ID in the student contact table is also unique. In this way, a record in the student table can only correspond to one record in the student contact table, and vice versa. Therefore, the relationship between the student table and the student contact table is one-to-one.

14.2.1 Inter-table association

One to many association : One record of a table can correspond to any number of records of another table.

Order table

OrderID	Date
10040	2014/07/01

1 : N

Order detail table

OrderID	OrderNumber	Amount
10040	1	3576.5
10040	2	1874.0

In the above figure, the relationship between the order table and the order detail table is one to many. Each order may have any number of order details.

14.2.1 Inter-table association

Many to many association: A record in table A can correspond to any number of records in table B; at the same time, a record in table B can also correspond to any number of records in table A.

Matrix A

row	col	value
1	1	1
1	2	2
2	1	3
2	2	4

M : N

Matrix B

row	col	value
1	1	1
1	2	3
2	1	2
2	2	4

Take matrix multiplication as an example: C=AB, $C_{ij} = \sum_{k=0}^{n}(A_{ik} * B_{kj})$. In other words, C_{ij} is obtained by multiplying each column of the i^{th} row of matrix A and each row of the j-th column of matrix B in alignment and then summing them. Therefore, it is necessary to find out all the correspondence between the columns of matrix A and the rows of matrix B. As shown in the figure above, the lines of each color represent a group of correspondence between the columns of A and the rows of B.
See [example 14.14] for detailed implementation process.

14.2.2 Common associated relationships

Tables with **association relationship** can be joined by **associated field**. When joining by associated field, if the filter condition is one or more equality relations, it is called **equivalent join**.

Employee table

ID	Name	Department
1	Rebecca	6

Department table

ID	Name
6	R&D

For example, when joining the employee table and department table, take Employee.Department == Department.ID as the joining condition, and it is an equivalent join.

Equivalent join is the most commonly used join forms, and is often used for the following associated relationships:

- **Foreign key table**
- **Same dimension table**
- **Main and sub table**

14.2.2 Common associated relationships

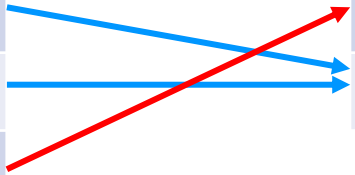
Some fields in table A are associated with the primary key of table B. The fields associated with the primary key of table B in table A are called **foreign keys** pointing to table B, and B is also called **foreign key table** of A. Table A has a many to one relationship with its foreign key table B.

Employee table

EMP_ID	EMP_NAME	DEPT_ID
1	Rebecca	2
2	Ashley	2
3	Rachel	1

Department table

DEPT_ID	DEPT_NAME
1	Admin
2	R&D



The Department field in the employee table is called a **foreign key** pointing to the Department table.

14.2.2 Common associated relationships

The fact table contains data that describes the business, as well as foreign keys that point to the dimension table. The dimension table contains details about the specified attribute in the fact table.

Daily attendance table

ID	Time	EMP_ID
1	09:05:32	2
2	09:15:17	1
3	18:01:55	2

Employee table

ID	EMP_NAME	DEPT_ID
1	Rebecca	2
2	Ashley	2
3	Rachel	1

Department table

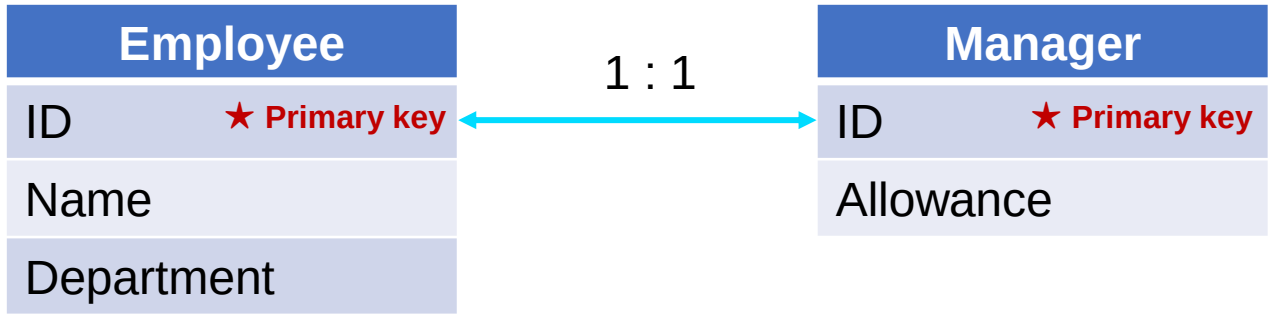
ID	DEPT_NAME
1	Admin
2	R&D

The basic information of employees is recorded in employee table, which is fact table; descriptive information of employee department is recorded in department table, which is dimension table.

The relationship between fact table and dimension table is relative. For the daily attendance table and employee table, the daily attendance table is the fact table, while the employee table becomes the dimension table.

14.2.2 Common associated relationships

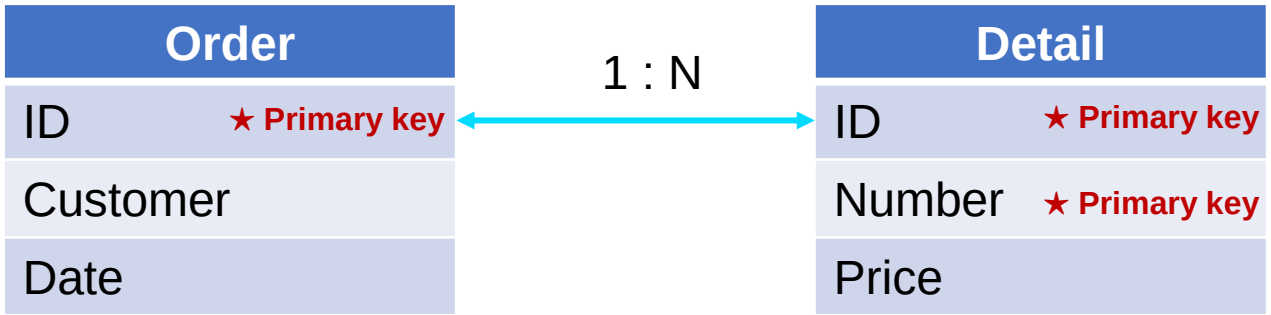
The primary key of table A is associated with the primary key of table B, both A and B are called the **same dimension tables**. The relationship between the same dimension tables is one to one, and the relationship between the same dimension tables is equivalent.



In the above figure, the employee table and the manager table are the **same dimension tables**. Managers are also employees, but they have more attributes than employees.

14.2.2 Common associated relationships

Part field of the primary keys of table B is associated with the primary key of table A. A is called the **main table**, and B is called the **sub table**. There is a one to many relationship between the main table and the sub table. The relationship between the main table and the sub table is not equal, and the reference of the main table from the sub table is similar to that of the foreign key table.



In the above figure, the order table and the order detail table are the main sub tables.

14.2.2 Common associated relationships

Foreign key table, same dimension table and main sub table are all related to **primary key** and correspond to many to one, one to one and one to many relationships respectively. When there is no primary key association, there will be many to many relationship. In practical application scenarios, many to many relationship is generally business error.

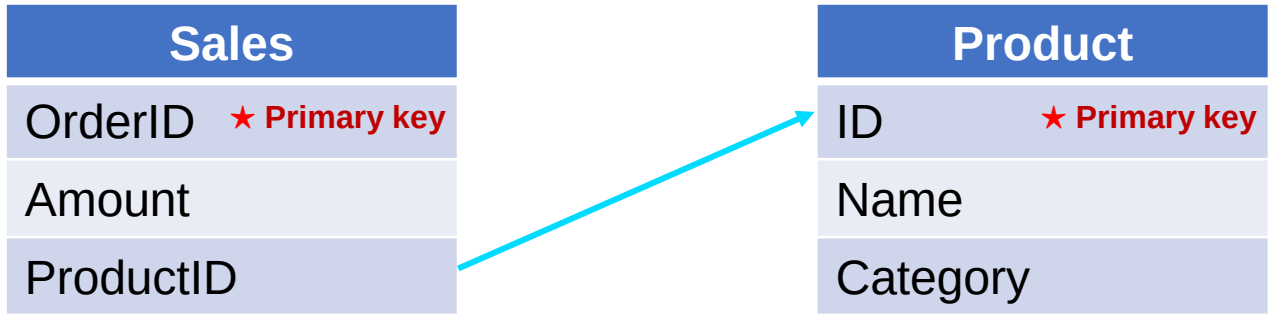
Note

- The **primary key** in this section refers to the logical primary key, that is, the field (fields) with unique value in the table. It is not necessarily the primary key created on a physical table.

Chapter 14
Inter table association

14.3 Foreign key

【Example 14.2】Query sales order details with sales amount over 1000. The data of sales table and product table is respectively from the files Sales.txt and Products.txt:



【Example 14.2】Query sales order details with sales amount over 1000. The data of sales table and product table is respectively from the files Sales.txt and Products.txt:

Solution

Join the foreign key table through the foreign key field and add the required foreign key table field.

Sales table

OrderID	Amount	ProductID
81182311	2440.0	59

Find the record with ID 59 in the product table

ID	Name	Category
59	Bright cheese	4

OrderID	Amount	ProductID	ProductName
81182311	2440.0	59	Bright cheese

Add foreign key table field: ProductName

【Example 14.2】Query sales order details with sales amount over 1000. The data of sales table and product table is respectively from the files Sales.txt and Products.txt:

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=file("Product.txt").import@t().keys@i(ID)	/Import product table from file
3	=A1.select(Amount>1000)	/Select records with sales greater than 1000
4	=A3.derive(A2.find(ProductID).Name:ProductName)	/According to the product ID, find the corresponding record in the product table, and add the ProductName field to the sales table.

A4	OrderID	Amount	ProductID	ProductName
	81182311	2440.0	59	Bright cheese
	98807954	1863.4	38	Green tea
	…	…	…	…

Note

► **A.derive(xi:Fi, …)** function is used to generate a new table sequence and copy the original fields first.

【Example 14.2】Query sales order details with sales amount over 1000. The data of sales table and product table is respectively from the files Sales.txt and Products.txt:

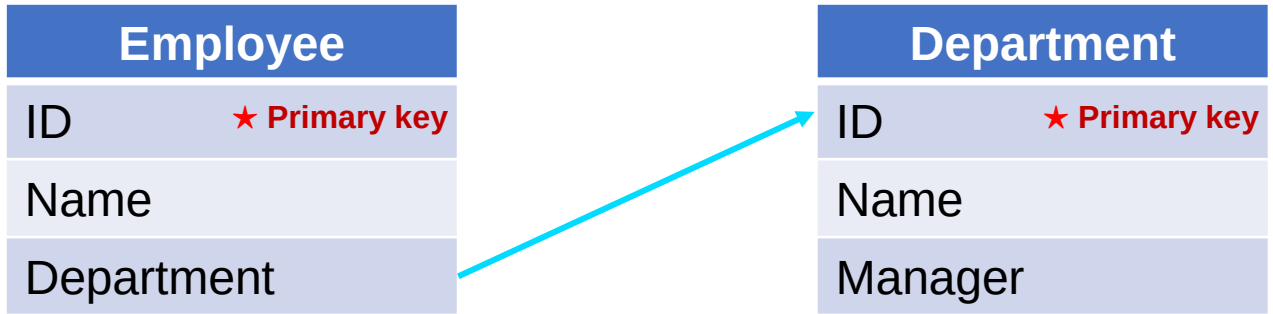
	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=file("Product.txt").import@t()	/Import product table from file
3	=A1.select(Amount>1000)	/Select records with sales greater than 1000
4	=A3.join(ProductID, A2:ID, Name:ProductName)	**/A.join() function can be used to simplify, add the foreign key field after join with the foreign key table**

A4	OrderID	Amount	ProductID	ProductName
	81182311	2440.0	59	Bright cheese
	98807954	1863.4	38	Green tea
	…	…	…	…

Note

- Function **A.join(C, T:K, x:F, …; …)** joins the field K of the table sequence T through the field C of the table sequence A, and add the x field of the table T (when x is ~ , the foreign key referenced record is added) to the F field of the table A.

【Example 14.3】Query employee name and employee's department name. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:



【Example 14.3】Query employee name and employee's department name. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:

Solution

The record in the foreign key table can be referenced directly through foreign key objectification. According to the value of the foreign key field, find the corresponding record in the foreign key table and set it in the foreign key field (the original value will be changed). In this way the two tables can be treated as a single table.

Employee table

ID	Name	Department
1	Rebecca	6

Find the record with ID 6 in the Department table

ID	Name	Manager
6	R&D	1

Foreign key objectification

ID	Name	Department
1	Rebecca	[6, R&D, 1]

【Example 14.3】Query employee name and employee's department name. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=file("Department.txt").import@t().keys@i(ID)	/Import department table from file, set primary key as ID and create index
3	=A1.run(Department=A2.find(Department))	/ Loop through the employee table, find the corresponding referenced record in the Department table and set it to the Department field
4	=A3.new(Name, Department.Name:DeptName)	/Create a table sequence and return employee name and department name

A3

ID	Name	Department
1	Rebecca	[6, R&D, 1]
…	…	…

ID	Name	Manager
6	R&D	1

A4

Name	DeptName
Rebecca	R&D
Ashley	Finance
…	…

Note

- After the foreign key objectification, the foreign key field value is converted to the corresponding record in the foreign key table. When the corresponding record is modified, the two tables will be affected synchronously.

【Example 14.3】Query employee name and employee's department name. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=file("Department.txt").import@t().keys@i(ID)	/Import department table from file, set primary key as ID and create index
3	=A1.switch(Department, A2:ID)	**/switch function can be used to simplify and implement foreign key objectification**
4	=A3.new(Name, Department.Name:DeptName)	/Create a table sequence and return employee name and department name

A3

ID	Name	Department
1	Rebecca	[6, R&D, 1]
…	…	…

ID	Name	Manager
6	R&D	1

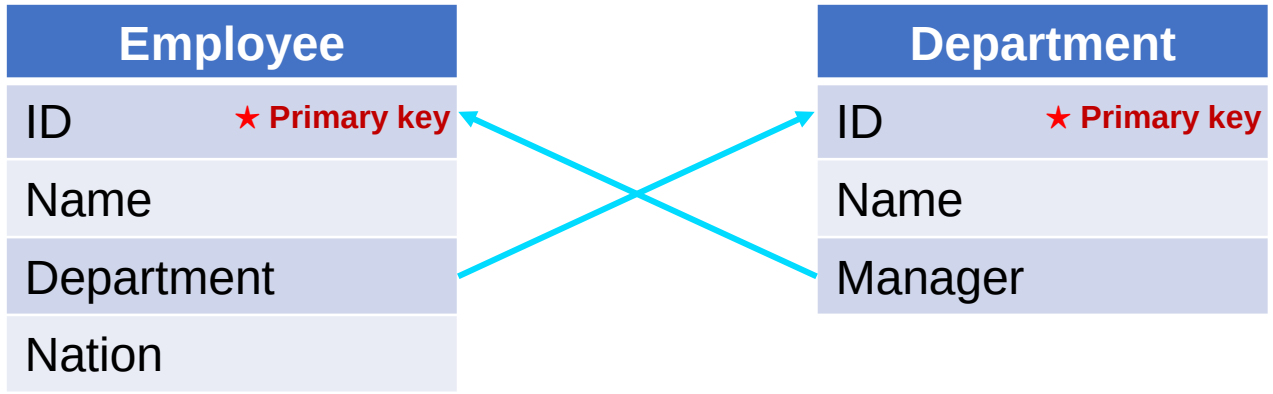
A4

Name	DeptName
Rebecca	R&D
Ashley	Finance
…	…

Note

- The function **A.switch(F, T:x)** is used to join the F field of sequence A with the x field of sequence T (if x is omitted, the primary key of T is used to join), and convert it to the corresponding record in T.

【Example14.4】Find which American employees have a Chinese manager. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:



【Example14.4】Find which American employees have a Chinese manager. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:

Solution

For multi-layer foreign key table join, the foreign key can be **objectified** layer by layer, and finally treated as a single table.

Department table

ID	Name	Manager
4	Marketing	6

Foreign key objectification

ID	Name	Manager
4	Marketing	[6, Matthew, 4]

Employee table

ID	Name	Department
8	Megan	4

Foreign key objectification

ID	Name	Department
8	Megan	[4, Marketing, […]]

【Example14.4】Find which American employees have a Chinese manager. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:

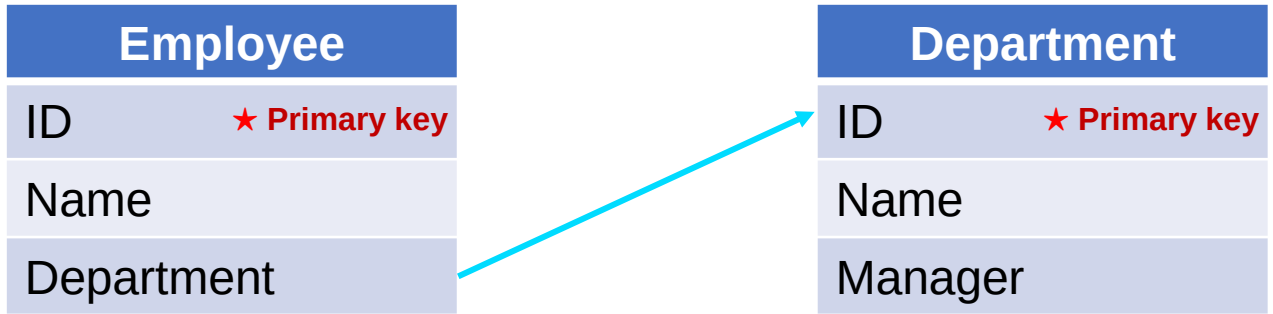
	A	B
1	=file("Employee.txt").import@t().keys(ID)	/Import employee table from file
2	=file("Department.txt").import@t().keys(ID)	/Import department table from file
3	=A2.switch(Manager, A1)	/Convert the manager field of the Department table to the corresponding employee record in the employee table
4	=A1.switch(Department, A2)	/Convert the Department field of the employee table to the corresponding record in the Department table.
5	=A4.select(Nation=="American" && Department.Manager.Nation=="Chinese")	/Select American employees whose manage is Chinese

A5	ID	Name	Department
	8	Megan	4
	27	Alexis	4
	…	…	…

Note

- In this example, the employee table and department table are referenced circularly. The Department field value in the employee table is converted to the corresponding record in the Department table, and the manager field value in the Department table is converted to the corresponding record in the employee table.

【Example 14.5】Query the names and department names of all employees. The Department that is not in the Department table uses the original value. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:



【Example 14.5】Query the names and department names of all employees. The Department that is not in the Department table uses the original value. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:

Solution

When the foreign key points to a null record, the foreign key objectification will cause the field value to be lost.
Solution 1: do not use foreign key objectification, add the required foreign key table field.
Solution 2: still use foreign key objectification, but the foreign key object is generated in the new field.

Foreign key points to the corresponding record

ID	Name	Department
1	Rebecca	6

Add new foreign key table field

ID	Name	Department	DEPT_NAME
1	Rebecca	6	R&D

ID	Name	Department
1	Rebecca	6

Add new foreign key object

ID	Name	Department	DEPT
1	Rebecca	6	[6, R&D, 1]

Foreign key points to null record

ID	Name	Department
93	Andrew	100

Add new foreign key table field

ID	Name	Department	DEPT_NAME
93	Andrew	100	

ID	Name	Department
93	Andrew	100

Add new foreign key object

ID	Name	Department	DEPT
93	Andrew	100	

【Example 14.5】Query the names and department names of all employees. The Department that is not in the Department table uses the original value. The data of employee table and department table is respectively from the files Employee.txt and Department.txt:

Solution 1:add the required foreign key table field

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=file("Department.txt").import@t()	/Import department table from file
3	=A1.join(Department, A2:ID, Name:DEPT_NAME)	/Using the join function, join the employee table and the Department table by the Department field, and add the Department name field.
4	=A3.new(Name, if(DEPT_NAME, DEPT_NAME, Department):DEPT_NAME)	/Create a table sequence, and use the original value if the Department name is null

Solution 2: the foreign key object is generated in the new field

	A
3	=A1.join(Department, A2:ID, ~:DEPT)
4	=A3.new(Name, if(DEPT, DEPT.Name, Department):DEPT_NAME)

A4	Name	DEPT_NAME
	Rebecca	R&D
	Andrew	100
	…	…

Note

- if(a, b, c) returns b if a is not null and not false, otherwise c is returned.

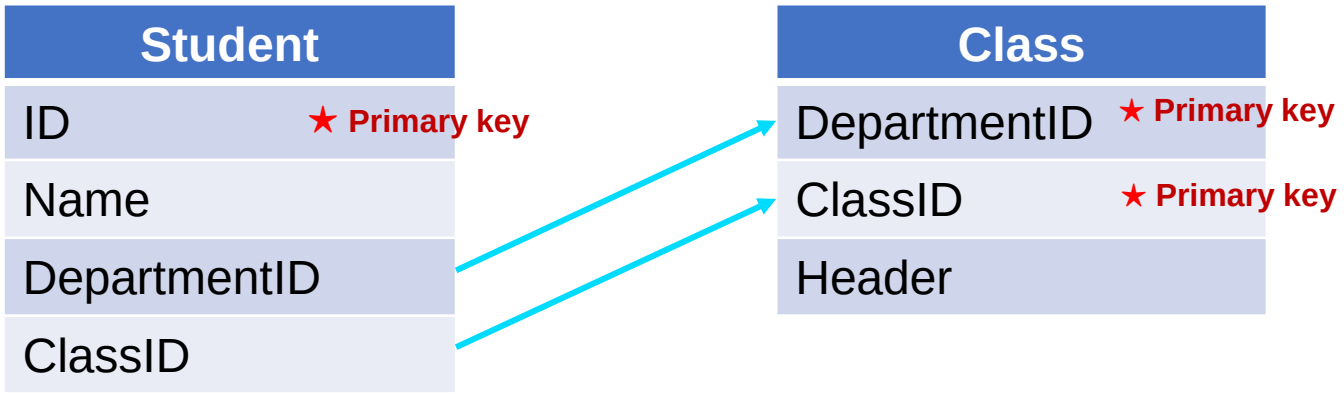
As mentioned earlier, a primary key consists of one or more fields in a table. When do you need to combine multiple fields as a primary key?

Class	
DepartmentID	★ Primary key
ClassID	★ Primary key
Header	

DepartmentID	ClassID	Header
1	1	Sarah
1	2	Andrew
2	1	Madison
2	2	Emily

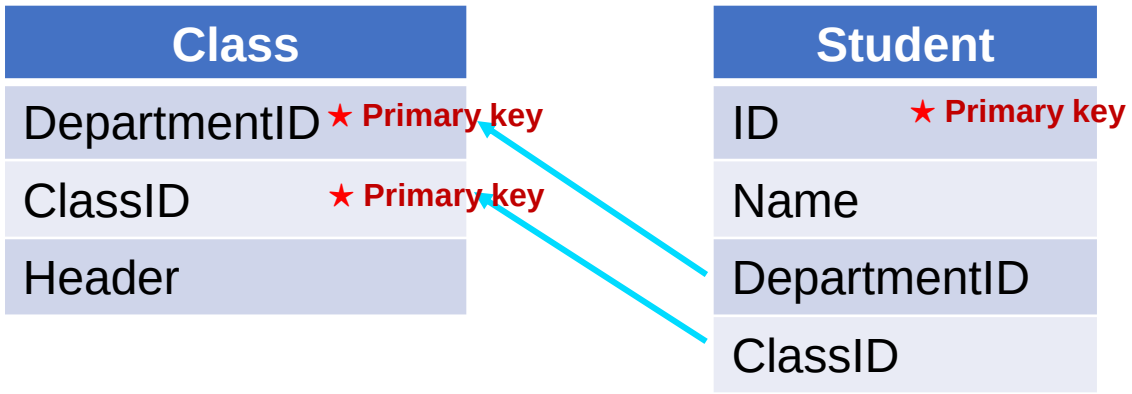
Take the class table as an example. There are classes 1 and 2 in the mathematics department and classes 1 and 2 in the computer department. Department ID or class ID can't be used as a unique identification alone. Only when department ID and class ID are combined can we form a primary key with unique identification.

When the foreign key table has a multi field primary key, the data table needs to use the **multi field foreign key** combined together to join the primary key of the foreign key table.



In the above figure, the student's department ID and class ID constitute a multi field foreign key pointing to the class table.

【Example 14.6】Check the head teacher of each student. Class table and student table data is respectively from the file Class.txt and Student.txt:



【Example 14.6】Check the head teacher of each student. Class table and student table data is respectively from the file Class.txt and Student.txt:

Solution

When using multi field foreign key to join, you can't objectify foreign key in the original field. You can add new field to save the field value of foreign key table during join.

Student table

ID	Name	DepartmentID	ClassID
1	Rebecca	1	1
2	Ashley	1	2

Class table

DepartmentID	ClassID	Header
1	1	Sarah
1	2	Andrew

Add foreign key mapped field

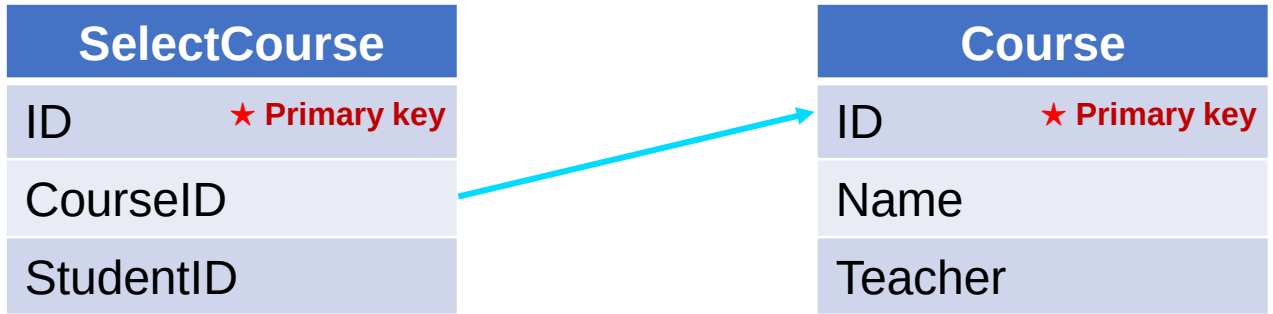
ID	Name	DepartmentID	ClassID	Header
1	Rebecca	1	1	Sarah
2	Ashley	1	2	Andrew

【Example 14.6】Check the head teacher of each student. Class table and student table data is respectively from the file Class.txt and Student.txt:

	A	B
1	=file("Student.txt").import@t()	/Import student table from file
2	=file("Class.txt").import@t()	/Import class table from file
3	=A1.join(DepartmentID:ClassID, A2:DepartmentID:ClassID, Header)	/Use the DepartmentID and ClassID fields to join the student table and the class table. Add the header field of the class table to the student table during join. .
4	=A3.new(Name, Header)	/Create a table sequence, return student name and header

A4	Name	Header
	Rebecca	Sarah
	Ashley	Sarah
	…	…

【Example 14.7】Find how many students who have taken "Matlab" course. The data of *Course* table and *SelectCouese* table is from Course.txt and SelectCourse.txt respectively:



【Example 14.7】Find how many students who have taken "Matlab" course. The data of *Course* table and *SelectCouese* table is from Course.txt and SelectCourse.txt respectively:

Solution

Select the record of MATLAB course in the course table first, and then perform a join filter.

Course table

ID	Name	Teacher
~~1~~	~~Math~~	~~3~~
2	Matlab	18

Filter

ID	Name	Teacher
2	Matlab	18

SelectCourse table

ID	CourseID	StudentID
1	2	12
~~2~~	~~1~~	~~37~~

Join filter

ID	CourseID	StudentID
1	2	12

【Example 14.7】Find how many students who have taken "Matlab" course. The data of *Course* table and *SelectCouese* table is from Course.txt and SelectCourse.txt respectively:

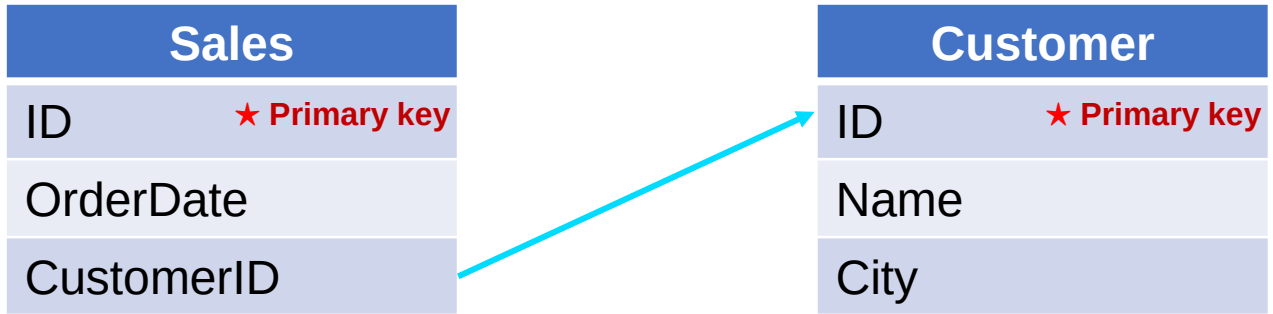
	A	B
1	=file("Course.txt").import@t()	/Import course table from file
2	=file("SelectCourse.txt").import@t()	/Import SelectCourse table from file
3	=A1.select(Name:"Matlab")	/Select record with course name as Matlab
4	=A2.switch@i(ID, A3:ID).count()	/Use the @i option with switch function to perform a join filter. The function count is used to count.

A4
5

Note

- Function **A.switch@i() / A.join@i()** is used to join and filter, delete unmatched records during join. A4 can be replaced as: =A2.join@i(ID, A3:ID).count()

【Example 14.8】Query the sales of new customers in 2014 (i.e. the records of the sales table with customer IDs that are not in the customer table). The data of sales table and customer table is respectively from Sales.txt and Customer.txt:



【Example 14.8】Query the sales of new customers in 2014 (i.e. the records of the sales table with customer IDs that are not in the customer table). The data of sales table and customer table is respectively from Sales.txt and Customer.txt:

Solution

Keep only unmatched records during join.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=file("Customer.txt").import@t()	/Import customer table from file
3	=A1.select(year(OrderDate)==2014)	/Select the 2014 records from the sales table
4	=A3.switch@d(Customer, A2:ID)	/Using the @d option of the switch function, keep only unmatched records during join.

A4

ID	OrderDate	Customer
10439	2014/02/07	1001
10504	2014/04/11	1002
…	…	…

Note

- Function **A.switch@d(F, T:x)** is used to keep only unmatched records during join, and the foreign key field will not be set as null in this case.

The table of China's administrative regions is as follows. ID is the number of administrative regions, parent_ ID is the number of the superior administrative region. For example, the superior administrative region of Tangshan is Hebei Province, and two fields in the same table are associated:

ID	NAME	PARENT_ID
13	Hebei	1
…	…	…
1302	Tangshan	13

Self join is to treat a mirror image of its own table as another table to join. It is often used when two different fields in a table are associated with each other.

【Example 14.9】Query the name of the superior organization of each organization. Organization table data is from Organization.txt.

ID	ORG_NAME	PARENT_ID
1	Head Office	0
2	Beijing Branch Office	1
3	Shanghai Branch Office	1
4	Chengdu Branch Office	1
5	Beijing R&D Center	2
6	Beijing Marketing Department	2
…	…	…

【Example 14.9】Query the name of the superior organization of each organization. Organization table data is from Organization.txt.

Solution

The organization table is joined with itself through foreign key, and the field of superior organization is converted to corresponding organization record through foreign key objectification.

ID	ORG_NAME	PARENT_ID
1	Head Office	0
2	Beijing Branch Office	1
…	…	…
6	Beijing Marketing Department	2

Foreign key objectification

ID	ORG_NAME	PARENT_ID
1	Head Office	(null)
2	Beijing Branch Office	[1, Head Office,]
…	…	…
6	Beijing Marketing Department	[2, Beijing Branch Office,]

【Example 14.9】Query the name of the superior organization of each organization. Organization table data is from Organization.txt.

	A	B
1	=file("Organization.txt").import@t()	/Import organization table from file
2	=A1.switch(PARENT_ID, A1:ID)	/Through foreign key objectification, the ID field of superior organization is converted to corresponding organization record
3	=A2.new(ID, ORG_NAME, PARENT_ID.ORG_NAME:PARENT_NAME)	/Create a table sequence to list the organization ID, organization name and superior organization name

A3

ID	ORG_NAME	PARENT_NAME
1	Head Office	(null)
2	Beijing Branch Office	Head Office
3	Shanghai Branch Office	Head Office
…	…	…

The most common solution for foreign key table is **foreign key objectification**.

The cases when foreign key objectification can't be used:

- Foreign key has no corresponding record.
- Multi field foreign key.

Solution:

- Add a new field to save the field values of foreign key table during join.
- Add a new field to save the corresponding foreign key table record during join.

Chapter 14
Inter table association

14.4 Same dimension table and main sub table

Next, we introduce three types of join: **left join** (the direction of right join is different, which is not described separately), **inner join** and **full join**. The differences among the three join types are as follows:

- **Left join**
- **Inner join**
- **Full join**

Left join is also called left outer join. It joins two tables based on left table. The result will list all the query information in the left table, while only parts of the right table that satisfy the condition in the left table will be listed.

Order table

ID	Customer	Date
10814	VICTE	2014/01/05
10815	SAVEA	2014/01/05

Order detail table

ID	ProductID	Price
10814	48	100
10814	48	300
10848	9	291

Left join

Order	Detail
[10814, VICTE, 2014/01/05]	[10814, 48, 100]
[10814, VICTE, 2014/01/05]	[10814, 48, 300]
[10815, SAVEA, 2014/01/05]	(null)

Inner join, also called join, will delete all rows from the result table that have no matching rows in other joined tables.

Order table

ID	Customer	Date
10814	VICTE	2014/01/05
10815	SAVEA	2014/01/05

Order detail table

ID	ProductID	Price
10814	48	100
10814	48	300
10848	9	291

Inner join

Order	Detail
[10814, VICTE, 2014/01/05]	[10814, 48, 100]
[10814, VICTE, 2014/01/05]	[10814, 48, 300]

A **full join** returns all records in each table. When a record has a match in another table, the result set contains the matched record. Otherwise, it is set to null.

Order table

ID	Customer	Date
10814	VICTE	2014/01/05
10815	SAVEA	2014/01/05

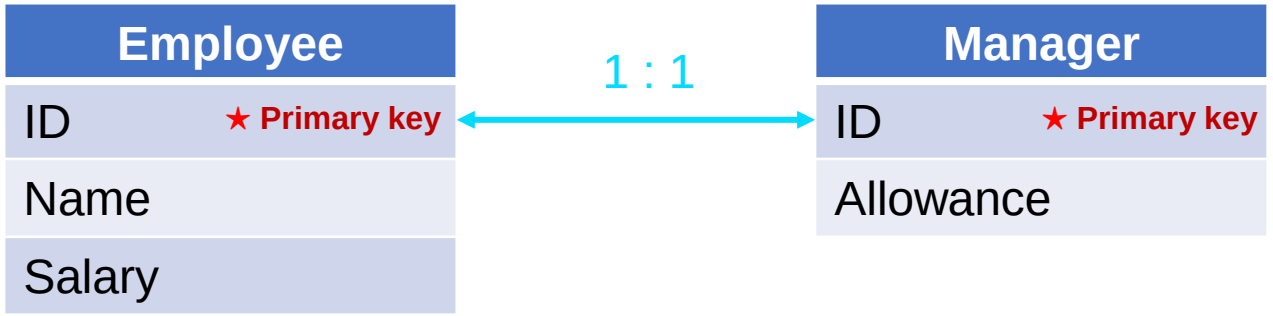
Order detail table

ID	ProductID	Price
10814	48	100
10814	48	300
10848	9	291

Full join

Order	Detail
[10814, VICTE, 2014/01/05]	[10814, 48, 100]
[10814, VICTE, 2014/01/05]	[10814, 48, 300]
[10815, SAVEA, 2014/01/05]	(null)
(null)	[10848, 9, 291]

【Example 14.10】Find the total income (plus allowance) of all employees (including managers). The data of employee table and manager table is respectively from Employee.txt and Manager.txt:



【Example 14.10】Find the total income (plus allowance) of all employees (including managers). The data of employee table and manager table is respectively from Employee.txt and Manager.txt:

Solution

The employee table and manager table are the same dimension tables, which can be joined directly through the **primary key**. Because all employees are to be queried, the employee table is used as the left table to perform left join.

Employee table

ID	Name	Salary
1	Rebecca	7000

Manager table

ID	Allowance
1	8000

Left join by the primary key ID

Employee	Manage
[1, Rebecca, 7000]	[1, 8000]

【Example 14.10】Find the total income (plus allowance) of all employees (including managers). The data of employee table and manager table is respectively from Employee.txt and Manager.txt:

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=file("Manager.txt").import@t().keys(ID)	/Import manager table from file
3	=create(Employee, Manager)	/Create a table sequence with employees and manager as fields
4	>A1.(A3.insert(0, A1.~, A2.find(A1.~.ID)))	/Loop through the employee table, find the corresponding record in the manager table according to the employee ID, and add the employee record and the corresponding manager record to the table sequence of A3
5	=A3.new(Employee.ID: ID, Employee.Name:Name, Employee.Salary+Manager.Allowance:Income)	/Create a table sequence, and calculate the total income(salary plus allowance) of employees

A5	ID	Name	Income
	1	Rebecca	15000
	2	Ashley	22000
	…	…	…

【Example 14.10】Find the total income (plus allowance) of all employees (including managers). The data of employee table and manager table is respectively from Employee.txt and Manager.txt:

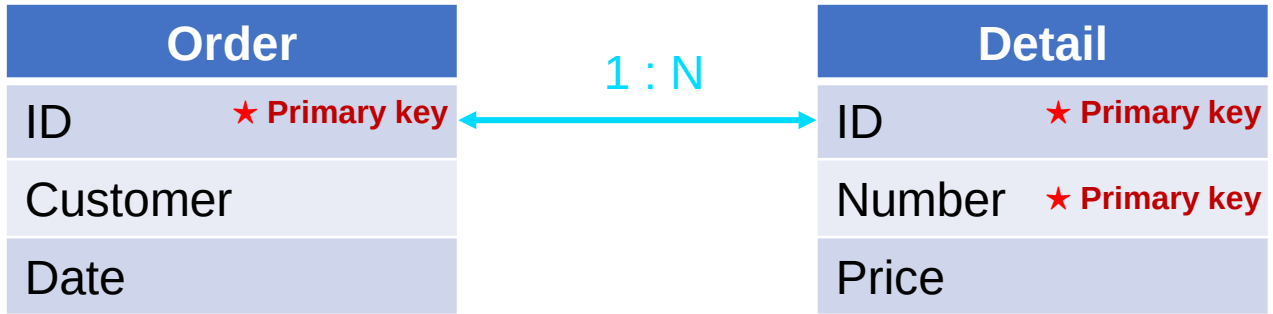
Solution

The join function can be used to simplify the join process. Function **join(Ai:Fi; ...)** is used to join table sequences by the specified field. The result set is composed of two tuples of set members. join@1() indicates that left join is performed for the first table sequence.

	A	B
1	=file("Employee.txt").import@t()	/Import employee table from file
2	=file("Manager.txt").import@t()	/Import manager table from file
3	=join@1(A1:Employee, ID; A2:Manager, ID)	/use join@1 function to join the employee table and the manager table by the primary key ID, and perform a left join to the employee table
4	=A3.new(Employee.ID: ID, Employee.Name:Name, Employee.Salary+Manager.Allowance:Income)	/Create a table sequence, and calculate the total income(salary plus allowance) of employees

A4	ID	Name	Income
	30	Joseph	18000
	400	Joshua	16000
	…	…	…

【Example 14.11】Calculate the total amount of each order. The data of order table and order detail table is respectively from Order.txt and Detail.txt.



【Example 14.11】Calculate the total amount of each order. The data of order table and order detail table is respectively from Order.txt and Detail.txt.

Solution

When a single main sub table is used, the sub table can be joined through partial primary key with the primary key of the main table.

Order table

ID	Customer	Date
10248	VINET	2012-07-04

Order detail table

ID	Number	Price
10248	1	238
10248	2	475

Join by primary key ID of main table

Order	Detail
[10248, VINET, 2012-07-04]	[10248, 1, 238]
[10248, VINET, 2012-07-04]	[10248, 2, 475]

Group aggregation

ID	Customer	Amount
10248	VINET	713

14.4 Same dimension table and main sub table

【Example 14.11】Calculate the total amount of each order. The data of order table and order detail table is respectively from Order.txt and Detail.txt.

	A	B
1	=file("Order.txt").import@t()	/Import order table from file
2	=file("Detail.txt").import@t()	/Import order detail table from file
3	=join(A1:Order, ID; A2:Detail, ID)	/Use join function to join order table and order detail table through order ID
4	=A3.groups(Order.ID:ID, Order.Customer:Customer; sum(Detail.Price):Amount)	/Calculate the total amount of each order

A4	ID	Customer	Amount
	10814	VICTE	816.0
	10848	CONSH	3201.0
	…	…	…

Note

- If the order table and order detail table are ordered for ID, we can use the function **join@m()** to perform an order-based merging to improve the speed and performance of join.

【Example 14.12】Compare the sales amount of various products in 2014 and 2015. Sales table data is from sales2014.txt and sales2015.txt:

OrderID	Customer	Product	Date	Amount
10400	VINET	Cheese	2014/01/01	2440.0
10401	TOMSP	Coffee	2014/01/01	1863.4
10402	HANAR	Milk	2014/01/02	1813.0
…	…	…	…	…

【Example 14.12】Compare the sales amount of various products in 2014 and 2015. Sales table data is from sales2014.txt and sales2015.txt:

Solution

The product should be listed regardless of the year in which it is sold, so **full join** should be used in this example. You can first calculate the sales amount of each product in each year. The sales amount of products in different years is the same dimension table relationship, which can be directly joined.

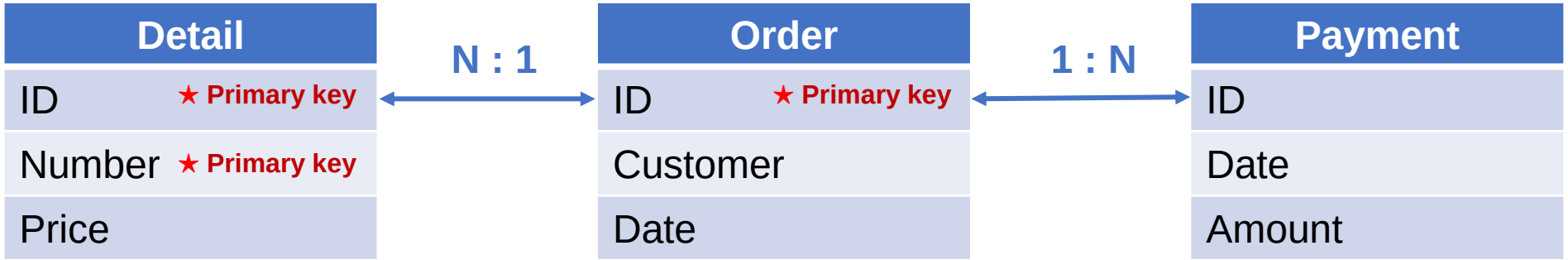
	A	B
1	=file("Sales2014.txt").import@t()	/Import 2014 sales table from file
2	=file("Sales2015.txt").import@t()	/Import 2015 sales table from file
3	=A1.groups(Product; sum(Amount):Amount)	/Group and calculate the sales amount of each product in 2014
4	=A2.groups(Product; sum(Amount):Amount)	/Group and calculate the sales amount of each product in 2015
5	=join@f(A3:A2014, Product; A4:A2015, Product)	/Perform a full join to the sales of 2014 and 2015
6	=A5.new(ifn(A2014.Product, A2015.Product):Product, A2014.Amount:A2014, A2015.Amount:A2015)	/Create a table sequence with product, sales in 2014 and sales in 2015 as fields.

A6	Product	C2014	C2015
	Biscuits	8931.47	266.0
	Shrimp eggs	4345.05	6726.05
	…	…	…

Note

- Function **join@f(Ai:Fi; …)** is used to perform a full join.

【Example 14.13】The order table also has a sub table to record the payment. Now we want to know which orders have not been fully paid, that is, the orders with the accumulated payment less than the total order amount. Order payment table data is from Payment.txt.



14.4 Same dimension table and main sub table

【Example 14.13】The order table also has a sub table to record the payment. Now we want to know which orders have not been fully paid, that is, the orders with the accumulated payment less than the total order amount. Order payment table data is from Payment.txt.

Solution

It is not right to simply join the three tables. There will be a many to many relationship between the order detail table and the payment table. After sub tables are grouped by ID, they have unique de facto primary keys, and multiple tables can be directly joined with primary keys.

Order detail table

ID	Number	Price
10248	1	238
10248	2	475

Group sub table by ID

Member
[[10248,1,238],[10248,2,475]]

Payment table

ID	Date	Amount
10248	2012-07-26	238
10248	2012-08-15	475

Group sub table by ID

Member
[[10248,2012-07-26,238],[10248,2012-08-15,475]]

Order table

ID	Customer	Date
10248	VINET	2012-07-04

Join multi tables through ID

Order	Detail	Payment
[10248,VINET,2012-07-04]	[[10248,1,238],[10248,2,475]]	[[10248,2012-07-26,238], …]

14.4 Same dimension table and main sub table

【Example 14.13】The order table also has a sub table to record the payment. Now we want to know which orders have not been fully paid, that is, the orders with the accumulated payment less than the total order amount. Order payment table data is from Payment.txt.

	A	B
1	=file("Order.txt").import@t()	/Import order table from file
2	=file("Detail.txt").import@t()	/Import order detail table from file
3	=file("Payment.txt").import@t()	/Import order payment table from file
4	=A2.groups(ID; sum(Price):a)	/Group the order detail table by ID and calculate the sales amount
5	=A3.groups(ID; sum(Amount):p)	/Group the order payment table by ID and calculate the payment amount
6	=join(A1:Order, ID; A4:Detail, ID; A5:Payment, ID)	/Use join function to join multi tables by ID
7	=A6.new(Order.ID:ID, Detail.a:Amount, Payment.p:Payment)	/Create a table sequence to calculate the total amount and payment amount of each order
8	=A7.select(Payment<Amount)	/Select the records whose payment amount is less than the total amount, that is, the records without full payment

A8

ID	Amount	Payment
10848	3201.0	2400.75
…	…	…

Note

- In the case of multiple main sub tables, the sub tables are first grouped according to the main table primary key. And then the primary key is the actual primary key. Then multiple tables can be joined through the primary key of the main table.

Chapter 14
Inter table association

14.5 Cross join and non equivalent join

14.5.1 Cross join

The student table is joined with the subject table, and each student record is combined with all the subject records to form a new record. The length of the result set is the product of the length of the student table and the length of the subject table.

Student table

STU_ID	STU_NAME
1	Rudy
2	Simon

×

Subject table

SUB_ID	SUB_NAME
1	Math
2	PE

=

Student	Subject
[1, Rudy]	[1, Math]
[1, Rudy]	[2, PE]
[2, Simon]	[1, Math]
[2, Simon]	[2, PE]

The above join mode is **cross join**. In the cross join result set of tables X and Y, the first field is a member of X, and the second field is one of all possible ordered pairs of Y.

14.5.1 Cross join

【Example 14.14】Find the product of two matrices. The data of two matrix tables is respectively from MatrixA.txt and MatrixB.txt.

The data structure of the matrix table is as follows:

row	col	value
1	1	1
1	2	2
1	3	3
…	…	…

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

The mathematical formula of this example is as follows:

$$C = AB = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 1\times1+2\times2+3\times3 & 1\times4+2\times5+3\times6 \\ 4\times1+5\times2+6\times3 & 4\times4+5\times5+6\times6 \end{pmatrix} = \begin{pmatrix} 14 & 32 \\ 32 & 77 \end{pmatrix}$$

14.5.1 Cross join

【Example 14.14】Find the product of two matrices. The data of two matrix tables is respectively from MatrixA.txt and MatrixB.txt.

Solution

Step 1, cross multiply the two matrix tables, filter the cross product at the same time, and select the records whose column number of A is equal to the row number of B.
Step 2, get the sum of the product of values with the same row and column number.

Step 1:

row	col	value
1	1	1
1	2	2
1	3	3
2	1	4
2	2	5
2	3	6

×

row	col	value
1	1	1
1	2	4
2	1	2
2	2	5
3	1	3
3	2	6

=

A	B
[1, 1, 1]	[1, 1, 1]
[1, 1, 1]	[1, 2, 4]
~~[1, 1, 1]~~	~~[2, 1, 2]~~
~~[1, 1, 1]~~	~~[2, 2, 5]~~
~~[1, 1, 1]~~	~~[3, 1, 3]~~
…	…

Step 2:

row	col	sum(A.value*B.value)
1	1	14
1	2	32
2	1	32
2	2	77

14.5.1 Cross join

【Example 14.14】Find the product of two matrices. The data of two matrix tables is respectively from MatrixA.txt and MatrixB.txt.

	A	B
1	=file("MatrixA.txt").import@t()	/Import matrix table A from file
2	=file("MatrixB.txt").import@t()	/Import matrix table B from file
3	=xjoin(A1:A; A2:B, A.col==B.row)	/Use xjoin function to perform a cross join, and filter according to condition during the join
4	=A3.groups(A.row:row, B.col:col; sum(A.value * B.value):value)	/Group and calculate the sum of product of values with the same row and column number

A4	row	col	value
	1	1	14
	1	2	32
	2	1	32
	2	2	77

$$\begin{bmatrix} 14 & 32 \\ 32 & 77 \end{bmatrix}$$

Note

- Function **xjoin(A_i:F_i, x_i; ...)** is used to cross join multiple tables A_i. If condition x_i is available, filter is performed at the same time of join.

14.5.2 Non equivalent join

When the join condition is not equivalence comparison, it is called **non equivalent join**. Non equivalent join is rare in practical application.

For example, to query an employee's income level:

Employee table

EID	NAME	SALARY
1	Simon	12000
2	Rudy	7000

Income level table

BAND	START	END
Low income	0	8000
Middle income	8000	15000
High income	15000	30000

The join between the employee table and the income level table is non equivalent join: when the employee's income is between the upper and lower limits of a certain income level, it is the employee's income level.

14.5.2 Non equivalent join

【Example 14.15】Find the age group of the community residents. The community residents table and age group table are read from Community.txt and AgeGroup.txt respectively.

Community residents table

ID	Name	Age
1	David	28
2	Daniel	15
3	Andrew	65
4	Rudy	(null)

Age group table

Group	Start	End
Children	0	15
Youth	16	40
Middle	41	60
Old	61	100

14.5.2 Non equivalent join

【Example 14.15】Find the age group of the community residents. The community residents table and age group table are read from Community.txt and AgeGroup.txt respectively.

Solution

Cross multiply the two tables, and find the age group that the age belongs to by comparing the age with the starting and ending ages of age group.

ID	Name	Age
1	David	28
2	Daniel	15
3	Andrew	65
…	…	…

×

Group	Start	End
Children	0	15
Youth	16	40
Middle	41	60
Old	61	100

=

Person	Age
~~[1, David, 28]~~	~~[Children, 0, 15]~~
[1, David, 28]	[Youth, 16, 40]
~~[1, David, 28]~~	~~[Middle, 41, 60]~~
~~[1, David, 28]~~	~~[Old, 61, 100]~~
…	…

14.5.2 Non equivalent join

【Example 14.15】Find the age group of the community residents. The community residents table and age group table are read from Community.txt and AgeGroup.txt respectively.

	A	B
1	=file("Community.txt").import@t()	/Import community residents table from file
2	=file("AgeGroup.txt").import@t()	/Import age group table from file
3	=xjoin@1(A1:Person; A2:Age, A2.Start<=Person.Age && A2.End>=Person.Age)	/Cross join the two tables, and select the records that age is in the corresponding age group
4	=A3.new(Person.ID:ID, Person.Name:Name, Person.Age:Age, Age.Group:Group)	/Create a new table sequence

A4	ID	Name	Age	Group
	1	David	28	Youth
	2	Daniel	15	Children
	3	Andrew	65	Old
	4	Rudy	(null)	(null)

Note

- Function **xjoin@1()** uses left join when cross join is performed, keep the record in the community residents table when the matching age group can't be found, and set the age field to null.

Chapter 14
Inter table association

14.6 Exercises

14.6 Exercises

【Exercise 14.1】Find the sale amount of the order record which OrderID is 55080325. Sales table data is from Sales.txt.

OrderID	Customer	OrderDate	Amount
81182311	VINET	2013/07/04	2440.0
98807954	TOMSP	2013/07/05	1863.4
…	…	…	…
55080325	WHITC	2015/05/01	928.75
…	…	…	…

14.6 Exercises

【Exercise 14.1】Find the sale amount of the order record which OrderID is 55080325. Sales table data is from Sales.txt.

ID	Customer	OrderDate	Amount
81182311	VINET	2013/07/04	2440.0
98807954	TOMSP	2013/07/05	1863.4
…	…	…	…
55080325	WHITC	2015/05/01	928.75
…	…	…	…

Solution

The primary key value of the table can be used to find the specified record.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=A1.keys(OrderID)	/Set the primary key of sales table as OrderID
3	=A2.find(55080325)	/Use the find function to find the record with the primary key value of 55080325
4	=A3.Amount	/get name from record

A4
Value
928.75

14.6 Exercises

【Exercise 14.2】Find each student's name and their math score. The data of the score table and student table is respectively from Score.txt and Student.txt.



14.6 Exercises

【Exercise 14.2】Find each student's name and their math score. The data of the score table and student table is respectively from Score.txt and Student.txt.

Solution

The foreign key table can be objectified by the foreign key, and the student field of the score table can be converted to the corresponding record in the student table.

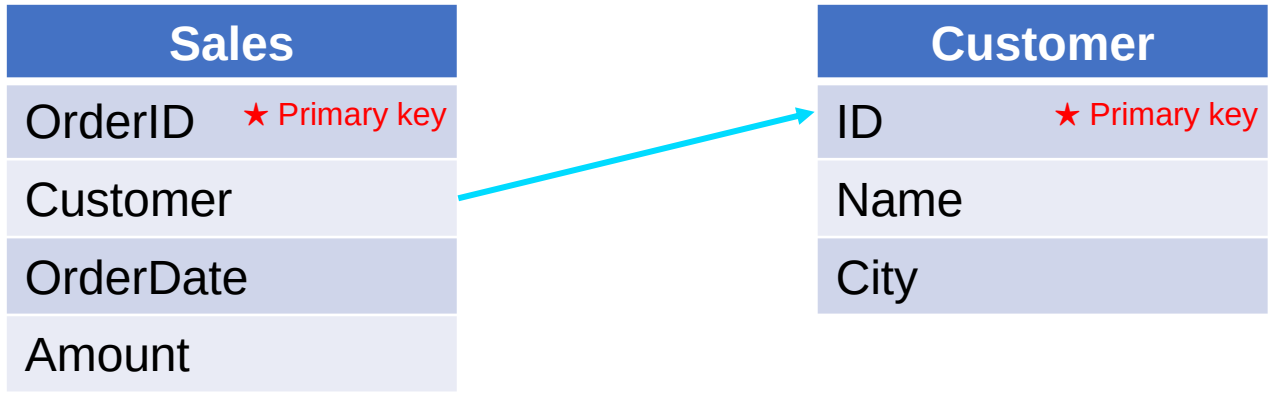
	A	B
1	=file("Score.txt").import@t()	/Import score table from file
2	=file("Student.txt").import@t()	/Import student table from file
3	=A1.select(SUBJECT:"Math")	/Select records with SUBJECT as Math from score table
4	=A3.switch(STUDENTID, A2:ID)	/Use switch function to convert STUDENTID field value to corresponding record in student table
5	=A4.new(STUDENTID.Name, SCORE)	/Create a table sequence to generate each student's name and math score

A5

ID	Name	SCORE
1	Rebecca	77
…	…	…

14.6 Exercises

【Exercise 14.3】Find the customer name, date and sales amount of each sales record and export to CustomerSales.txt. Some customers may not be in the customer table. The data of sales table and customer table is respectively from Sales.txt and Customer.txt.



14.6 Exercises

【Exercise 14.3】Find the customer name, date and sales amount of each sales record and export to CustomerSales.txt. Some customers may not be in the customer table. The data of sales table and customer table is respectively from Sales.txt and Customer.txt.

Solution

If the foreign key points to a null record, you can add the required foreign key table fields.

	A	B
1	=file("Sales.txt").import@t()	/Import sales table from file
2	=file("Customer.txt").import@t()	/Import customer table from file
3	=A1.join(Customer, A2:ID, Name:CustomerName)	/Use join function to join two tables, add the CustomerBame field of customer table to the sales table
4	=A3.new(CustomerName, OrderDate, Amount)	/Create a table sequence with customer name, date, and amount as fields
5	>file("CustomerSales.txt").export@t(A4)	/Export the A4 table sequence to file CustomerSales.txt

CustomerSales.txt is as follows:

```
CustomerName    OrderDate    Amount
Shantai    2013/07/04    2440.0
Shiyi    2013/07/05    1863.4
...
```

14.6 Exercises

【Exercise 14.4】Find the name and home address of each student. The data of student table and student contact table is respectively from Student.txt and StudentContacts.txt.



14.6 Exercises

【Exercise 14.4】Find the name and home address of each student. The data of student table and student contact table is respectively from Student.txt and StudentContacts.txt.

Solution

The relationship of the same dimension table is equal, and two tables can be joined directly.

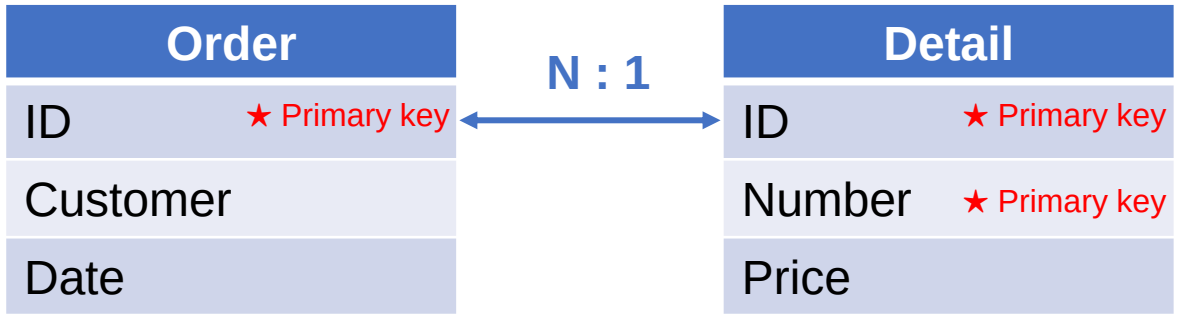
	A	B
1	=file("Student.txt").import@t()	/Import student table from file
2	=file("StudentContacts.txt").import@t()	/Import student contact table from file
3	=join(A1:Student, ID; A2:Contacts, ID)	/Use the join function to join the student table and the student contact table by the primary key student ID
4	=A3.new(Student.Name:Name, Contacts.Address:Address)	/Create a table sequence to generate the name and address of students

A4

ID	Name	Address
1	Rebecca	124 Guangming North Road
…	…	…

14.6 Exercises

【Exercise 14.5】Calculate the monthly sales in 2014. The data of order table and order detail table is respectively from Order.txt and Detail.txt.



14.6 Exercises

【Exercise 14.5】Calculate the monthly sales in 2014. The data of order table and order detail table is respectively from Order.txt and Detail.txt.

Solution

A single main and sub table can be joined directly by the common field of the primary key.

ID	Customer	Date
10814	VICTE	2014-01-05

ID	Number	Price
10814	1	102.0
10814	2	102.0
…	…	…

Join by common field order ID

Order	Detail
[10814, VICTE, 2014-01-05]	[10814, 1, 102.0]
[10814, VICTE, 2014-01-05]	[10814, 2, 102.0]
…	…

14.6 Exercises

【Exercise 14.5】Calculate the monthly sales in 2014. The data of order table and order detail table is respectively from Order.txt and Detail.txt.

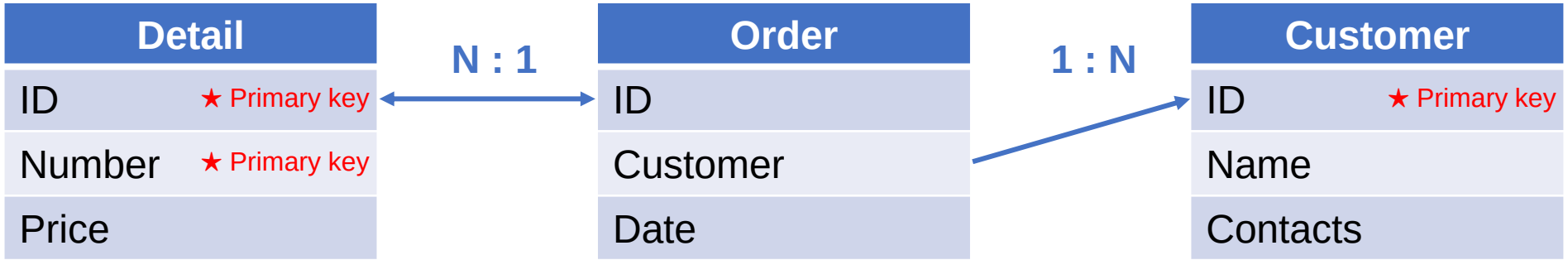
	A	B
1	=file("Order.txt").import@t()	/Import order table from file
2	=A1.select(year(Date)==2014)	/Select the 2014 records from the order table
3	=file("Detail.txt").import@t()	/Import order detail table from file
4	=join(A2:Order,ID; A3:Detail,ID)	/Use the join function to join order table and order detail table by common field order ID
5	=A4.groups(month(Order.Date):Month; sum(Detail.Price):Amount)	/Group and calculate monthly sales amount

A5

Month	Amount
1	4017.0
2	31841.0
…	…

14.6 Exercises

【Exercise 14.6】Find the names of customers with total sales over 10000. The data of each table comes from the txt file with the same name. The relationship between tables is as follows:



14.6 Exercises

【Exercise 14.6】Find the names of customers with total sales over 10000. The data of each table comes from the txt file with the same name.

Solution

In this exercise, there is a foreign key table relationship, as well as a single main sub table relationship. You can objectify the foreign key first, and then join the main sub table.

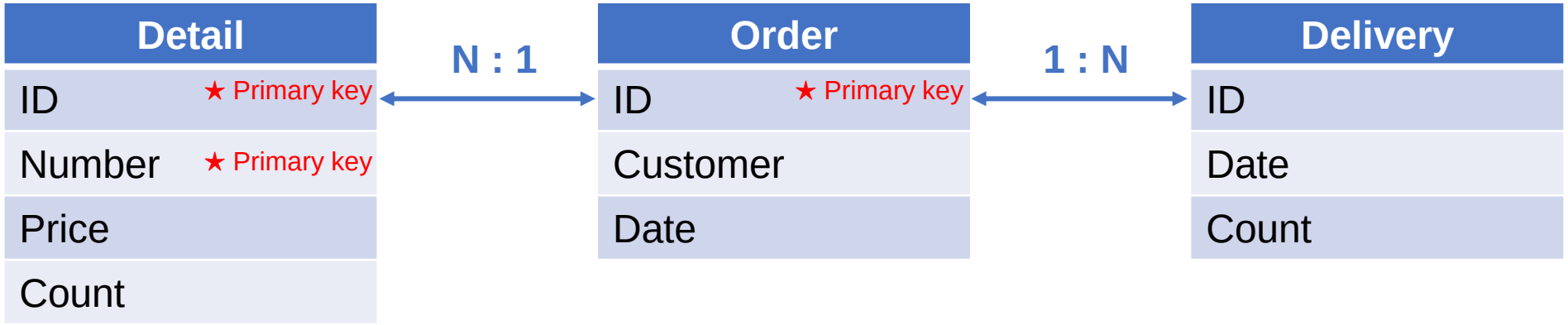
	A	B
1	=file("Order.txt").import@t()	/Import order table from file
2	=file("Detail.txt").import@t()	/Import order detail table from file
3	=file("Customer.txt").import@t()	/Import customer table from file
4	=A1.switch(Customer, A3:ID)	/Use the switch function to convert the customer ID of the order table to the corresponding record
5	=join(A4:Order, ID; A2:Detail, ID)	/Use join function to join order table and order detail table by order ID
6	=A5.groups(Order.Customer.Name:Name; sum(Detail.Price):Amount)	/Group and calculate the sum of each order
7	=A6.select(Amount>10000)	/Select records with total order amount over 10000

A7

Name	Amount
ALFKI	39661.0
…	…

14.6 Exercises

【Exercise 14.7】Find the orders that are not all delivered in 2014. The data of each table comes from the txt file with the same name. The relationship between tables is as follows:



14.6 Exercises

【Exercise 14.7】Find the orders that are not all delivered in 2014. The data of each table comes from the txt file with the same name. The relationship between tables is as follows:

Solution

In this exercise, there are multiple main and sub table relationships. The sub tables are grouped by ID first, and the ID becomes the actually primary key. Then the relationship among multiple tables is the same dimension table, which can be directly joined through ID.

Order detail table

ID	Number	Count
10814	1	8
10814	2	24

Group sub table by ID

Member
[[10814, 1, 8], [10248, 2, 24]]

Delivery table

ID	Date	Count
10814	2014/01/08	8
10814	2014/01/13	24

Group sub table by ID

Member
[[10248, 2014/01/08, 8], [10248, 2012/08/13, 24]]

Order table

ID	Customer	Date
10814	VICTE	2014/01/05

Join multi tables through ID

Order	Detail	Delivery
[10248, VICTE, 2014/01/05]	[[10814, 1, 8], [10248, 2, 24]]	[[10248, 2014/01/08, 8], [10248, 2012/08/13, 24]]

14.6 Exercises

【Exercise 14.7】Find the orders that are not all delivered in 2014. The data of each table comes from the txt file with the same name. The relationship between tables is as follows:

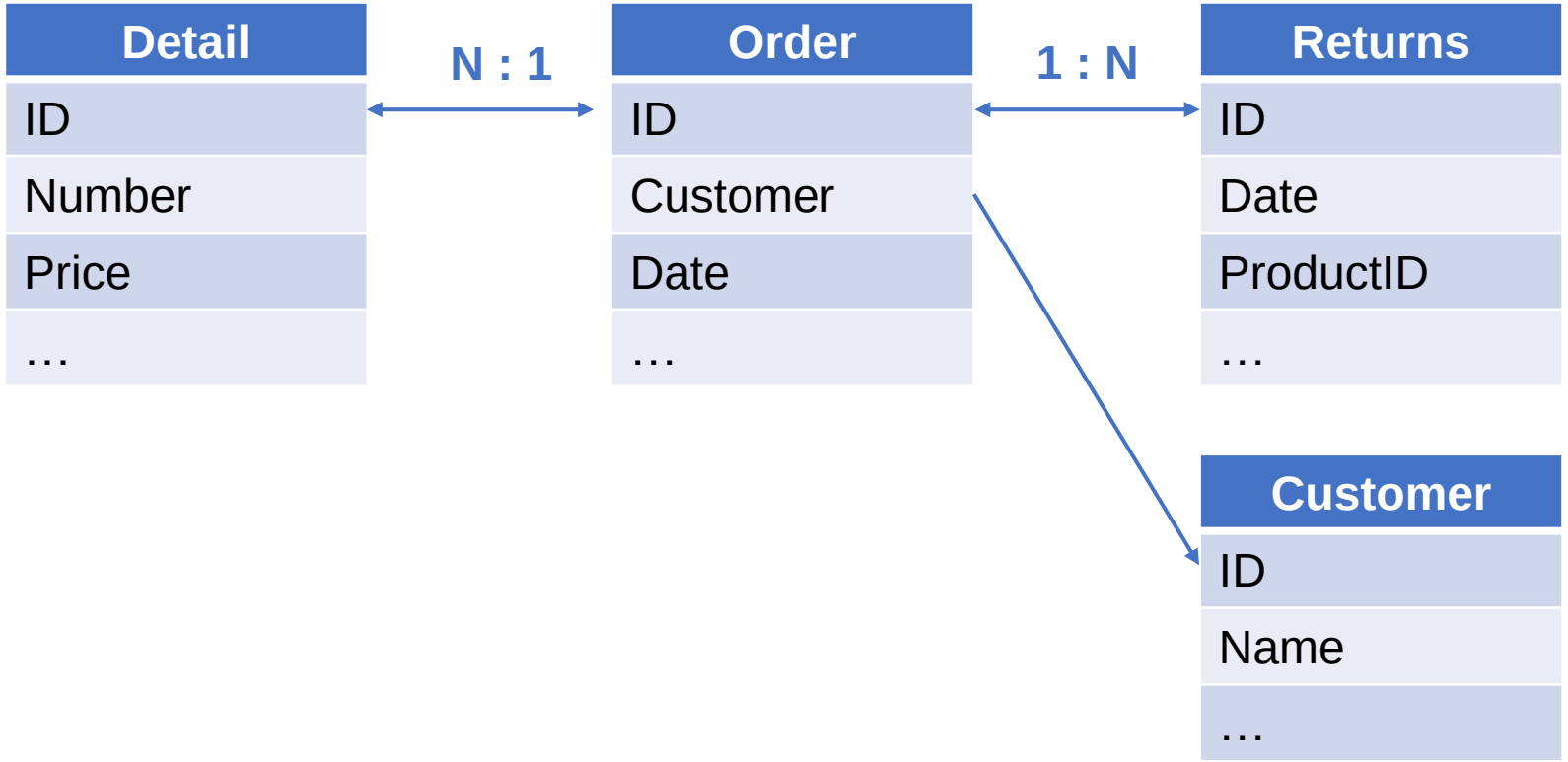
	A	B
1	=file("Order.txt").import@t()	/Import order table from file
2	=file("Detail.txt").import@t()	/Import order detail table from file
3	=file("Delivery.txt").import@t()	/Import delivery table from file
4	=A2.group(ID)	/Group order detail table by ID
5	=A3.group(ID)	/Group delivery table by ID
6	=join(A1:Order,ID; A4:Detail,ID; A5:Delivery,ID)	/Use join function to join multi tables through ID
7	=A6.new(Order.ID:ID, Detail.sum(Count):Count, Delivery.sum(Count):DeliveryCount)	/Create a table sequence, calculate the quantity of goods and delivered quantity of each order
8	=A7.select(DeliveryCount<Count)	/Select the records that the delivered quantity is less than the order quantity, that is, the records that the delivery has not been completed.

A8

ID	Count	DeliveryCount
10979	150	90
…	…	…

14.6 Exercises

【Exercise 14.8】Find the number of refund orders of each customer in 2015. The data of each table comes from the txt file with the same name. The relationship between tables is as follows:



14.6 Exercises

【Exercise 14.8】Find the number of refund orders of each customer in 2015. The data of each table comes from the txt file with the same name.

Solution

This exercise is a mixed example of foreign key table, same dimension table and main sub table. Use the methods from the previous sections: first objectify the foreign key, then group the sub tables according to the primary key of the main table. Finally, the relationships among multiple tables are all same dimension tables, and then join them through the primary key.

Order table

ID	Customer	Date
11087	ANTON	2015/02/03

Foreign key objectification →

ID	Customer	Date
11087	[ANTON, …]	2015/02/03

Detail table

ID	Number	Price
11087	1	124.0
11087	2	31.0

Group sub table by ID →

Member
[[11087, 1, 124.0], [11087, 2, 31.0]]

Returns table

ID	Date	ProductID
11087	2015/02/13	77

Group sub table by ID →

Member
[[11087, 2015/02/13, 77]]

Join multi tables through ID

Detail, Order, Returns →

Detail	Order	Returns
[[11087, 1, 124.0], [11087, 2, 31.0]]	[11087, [ANTON, …], 2015/02/03]	[[11087, 2015/02/13, 77]]

14.6 Exercises

【Exercise 14.8】Find the number of refund orders of each customer in 2015. The data of each table comes from the txt file with the same name.

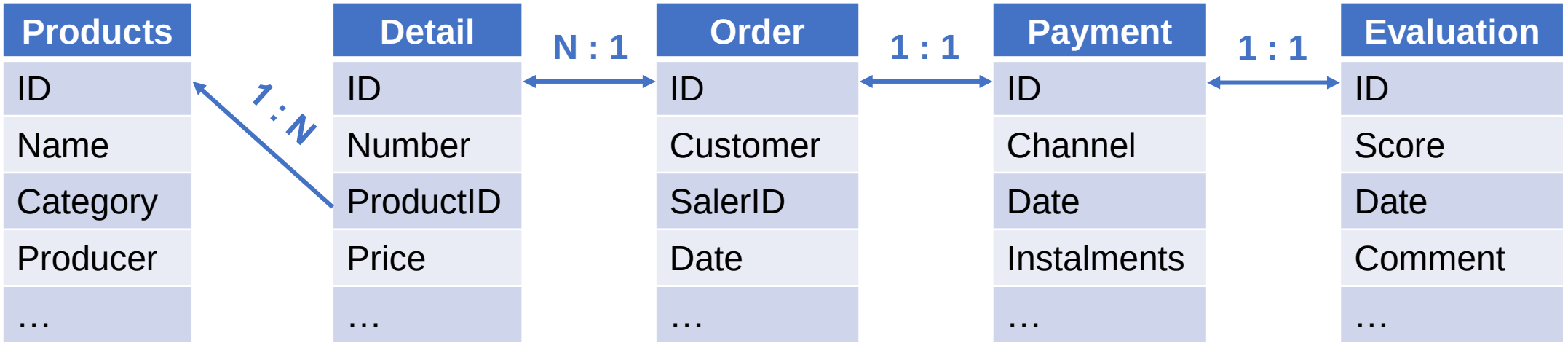
	A	B
1	=file("Order.txt").import@t()	/Import order table from file
2	=file("Detail.txt").import@t()	/Import order detail table from file
3	=file("Returns.txt").import@t()	/Import returns table from file
4	=file("Customer.txt").import@t()	/Import customer table from file
5	=A1.select(year(Date)==2015).switch(Customer, A4:ID)	/Select the records in 2015 from the order table and objectify the foreign key customer
6	=A2.group(ID)	/Group order detail table by ID
7	=A3.group(ID)	/Group returns table by ID
8	=join(A5:Order, ID;A6:Detail, ID;A7:Returns, ID)	/Use join function to join multi tables through ID
9	=A8.groups(Order.Customer.Name; count(~):Count)	/Use groups function to group and count the number of refund orders of each customer

A9

Name	Count
Guoding Co., Ltd	1
…	…

14.6 Exercises

【Exercise 14.9】Find the information of the orders that satisfy the following conditions: the product name contains "water", the order was placed in January 2014, the total order amount was more than 200 yuan, instalment was not used, and the evaluation was 5 stars. The data of each table comes from the txt file with the same name. The relationship between tables is as follows:



14.6 Exercises

【Exercise 14.9】Find the information of the orders that satisfy the following conditions: the product name contains "water", the order was placed in January 2014, the total order amount was more than 200 yuan, instalment was not used, and the evaluation was 5 stars. The data of each table comes from the txt file with the same name.

Solution

This exercise is a mixed example of foreign key table, same dimension table and main sub table. Use the methods from the previous sections: first objectify the foreign key, then group the sub tables according to the primary key of the main table. Finally, the relationships among multiple tables are all same dimension tables, and then join them through the primary key.

ID	Number	ProductID	Price
10248	1	1	238

Foreign key objectification

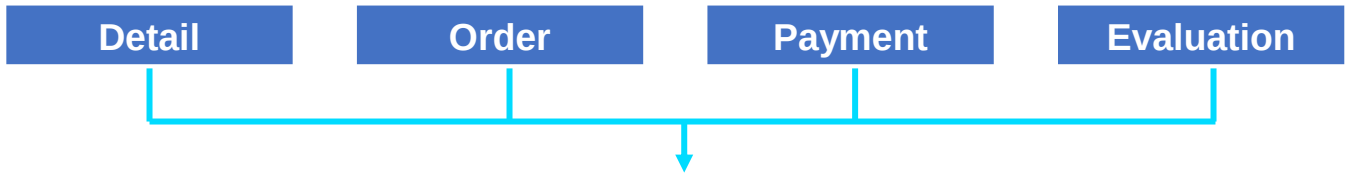
ID	Number	ProductID	Price
10248	1	[1, cake, 3, 12]	238

ID	Number	ProductID	Price
10248	1	[1, cake, 3, 12]	238
10248	2	[2, apple, 3, 5]	475

Group sub table by ID

Member
[[10248, 1, [1, cake, 3, 12], 238], [10248, 2, [2, …], 475]]

Join multi tables through ID



Detail	Order	Payment	Evaluation
[[10248, 1, [1, cake, 3, 12], 238], […]]	[10248, VINET, 2012-07-04]	[10248, 3, 2012-07-04, 0]	[10248, 5, 2012-07-16, "Good"]

14.6 Exercises

【Exercise 14.9】Find the information of the orders that satisfy the following conditions: the product name contains "water", the order was placed in January 2014, the total order amount was more than 200 yuan, instalment was not used, and the evaluation was 5 stars. The data of each table comes from the txt file with the same name.

	A	B	C
1	=file("Products.txt").import@t()	=A1.select(like(Name, "*water*"))	/Import the product table from file and select the records whose product name contains water
2	=file("Detail.txt").import@t()	=A2.switch@i(ProductID, A1:ID)	/Import order detail table from file. The foreign key of the product ID field is objectified in the order detail table, converted to the corresponding records in the product table, and the unmatched records are deleted.
3	=B2.group(ID)	=A3.select(sum(Price)>200)	/Group the order detail table by order ID, and select records with total order amount exceeding 200
4	=file("Order.txt").import@t()	=A4.select(year(Date)==2014 && month(Date)==1)	/Import order table from file, and select records of Jan 2014
5	=file("Payment.txt").import@t()	=A5.select(Instalments==0)	/Import payment table from file, and select records with no instalment
6	=file("Evaluation.txt").import@t()	=A6.select(Score==5)	/Import evaluation table from file, and select records with score as 5
7	=join(B3:Detail,ID; B4:Order,ID; B5:Payment, ID; B6:Evaluation,ID)		/Use join function to join multi tables through ID
8	=A7.new(Order.ID:ID, Order.Customer:Customer, Order.Date:Date, Detail.sum(Price):Amount)		/Create a table sequence, and generate order information

A8

ID	Customer	Date	Amount
10848	CONSH	2014/01/23	3201.0

Chapter 15
Graphics drawing

15.1 Canvas

- Canvas
- Element

15.2 Physical coordinates

- Absolute coordinates
- Relative coordinates

15.3 Logical coordinates

- Numeric axis
- Enumeration axis
- Date axis

15.4 More logical mappings

- Polar coordinates
- Combined coordinate system
- Legend

15.5 Finished graph

- Column graph
- Dual-axis graph

15.6 Exercises

- Trigonometric function
- Multiple combinations
- K-line chart

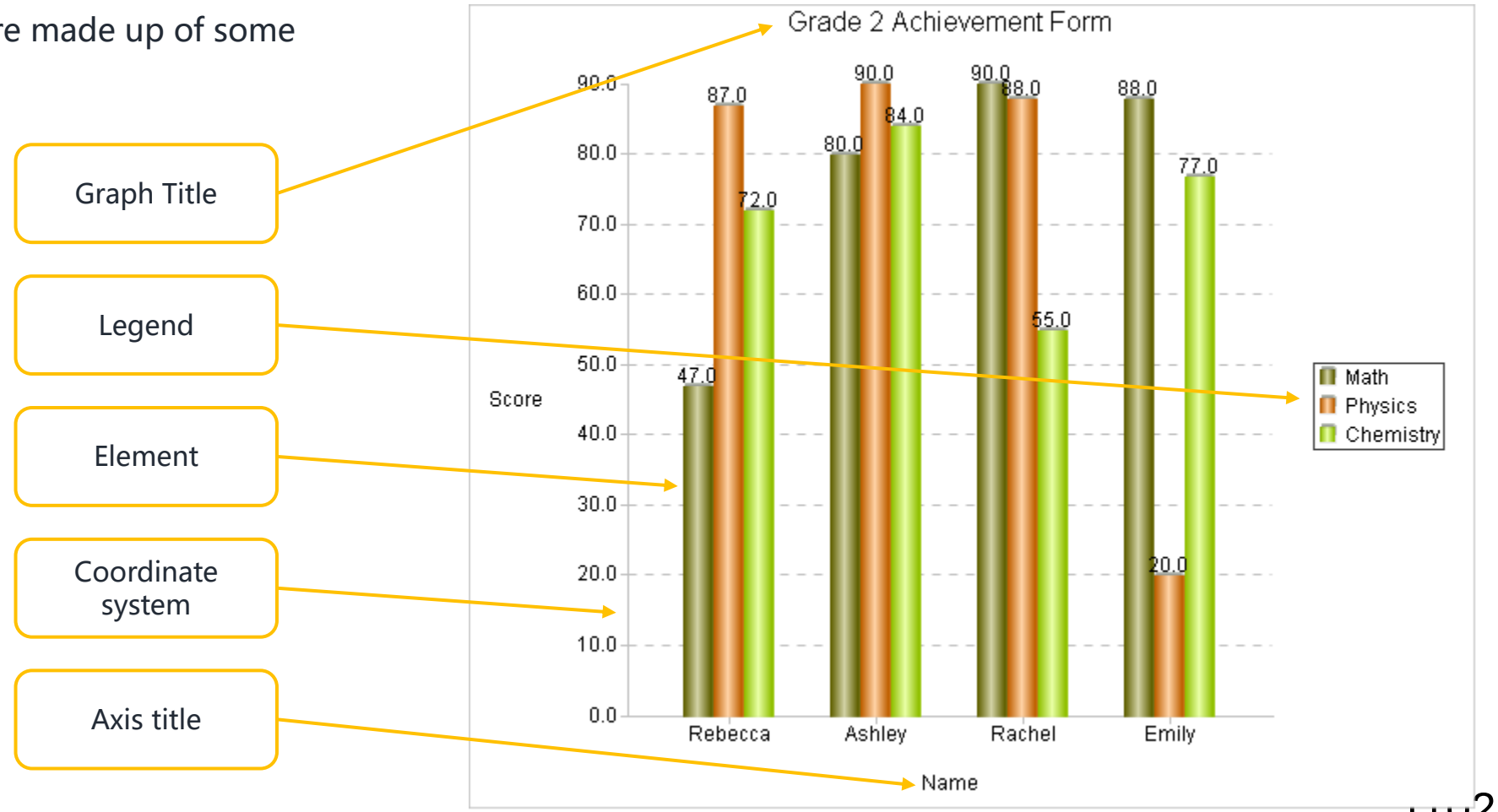
Chapter 15
Graphics drawing

15.1 Canvas

15.1.1 Canvas

Drawing requires a blank area, which is usually a rectangle on a computer, and the drawing is made up of points in this area. This area is called **canvas**.

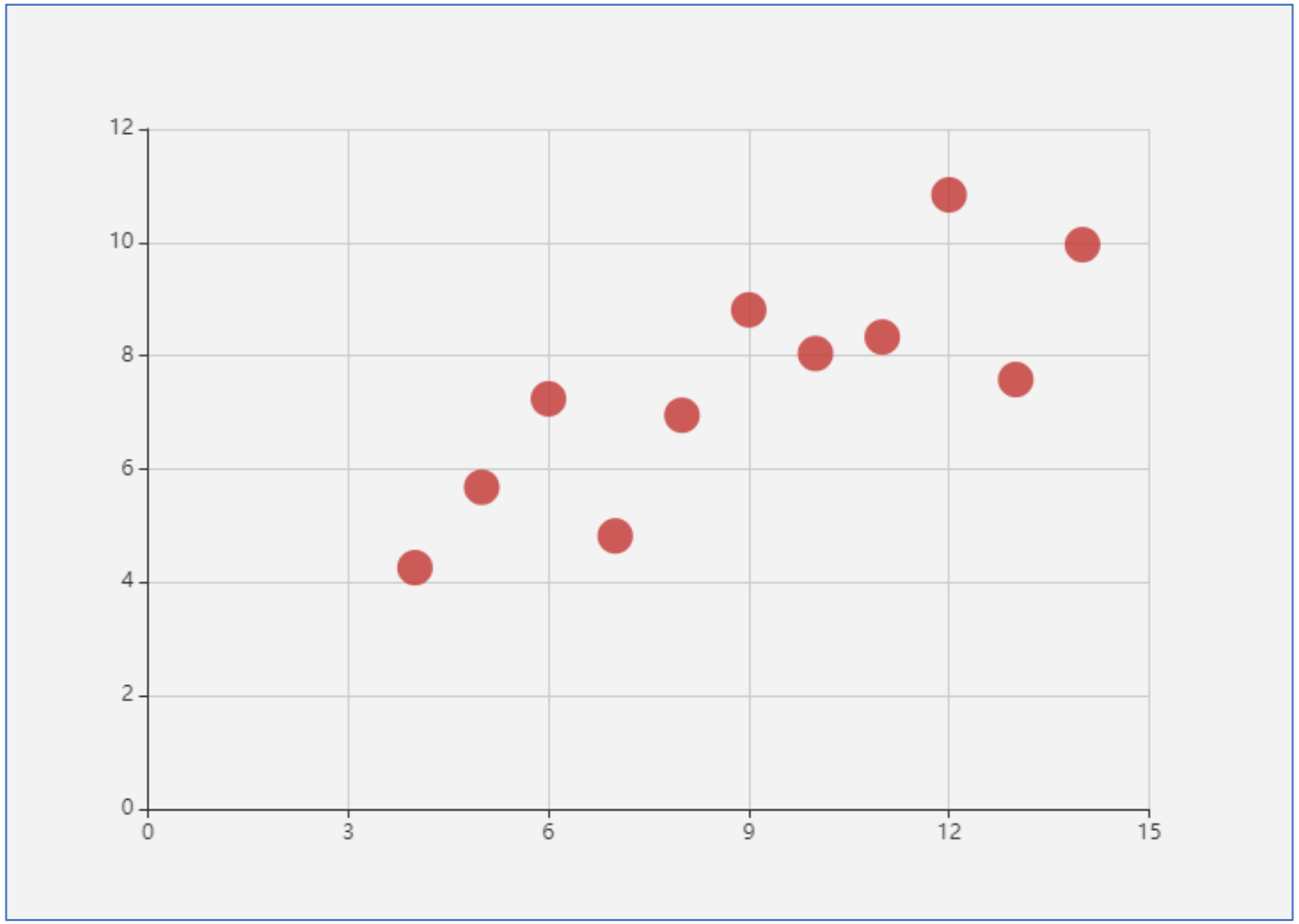
The graphics in the canvas are made up of some components:



15.1.2 Element

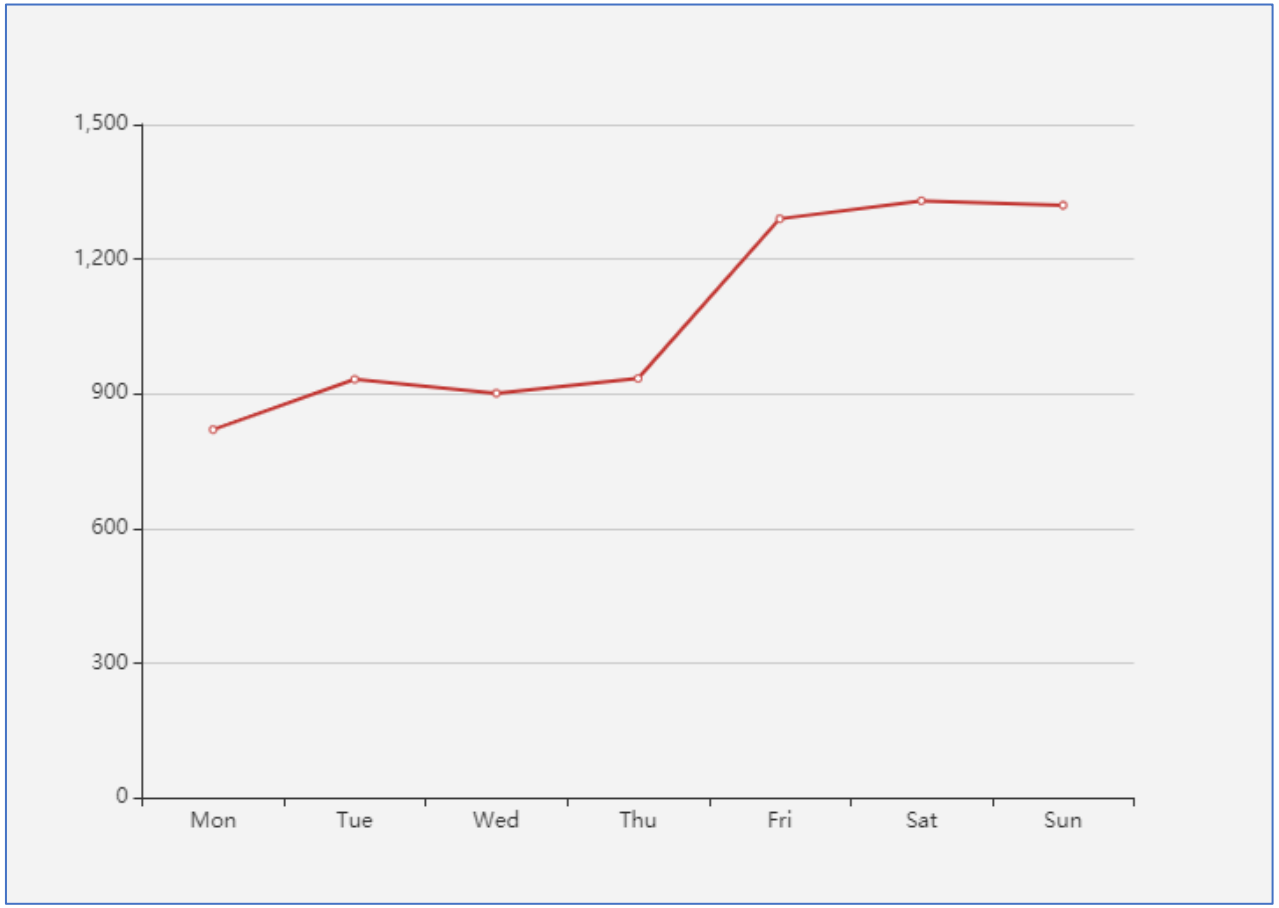
In these components, the basic graphics that represent business data in various forms are called **elements**.

For example, the **dot** in the right figure:



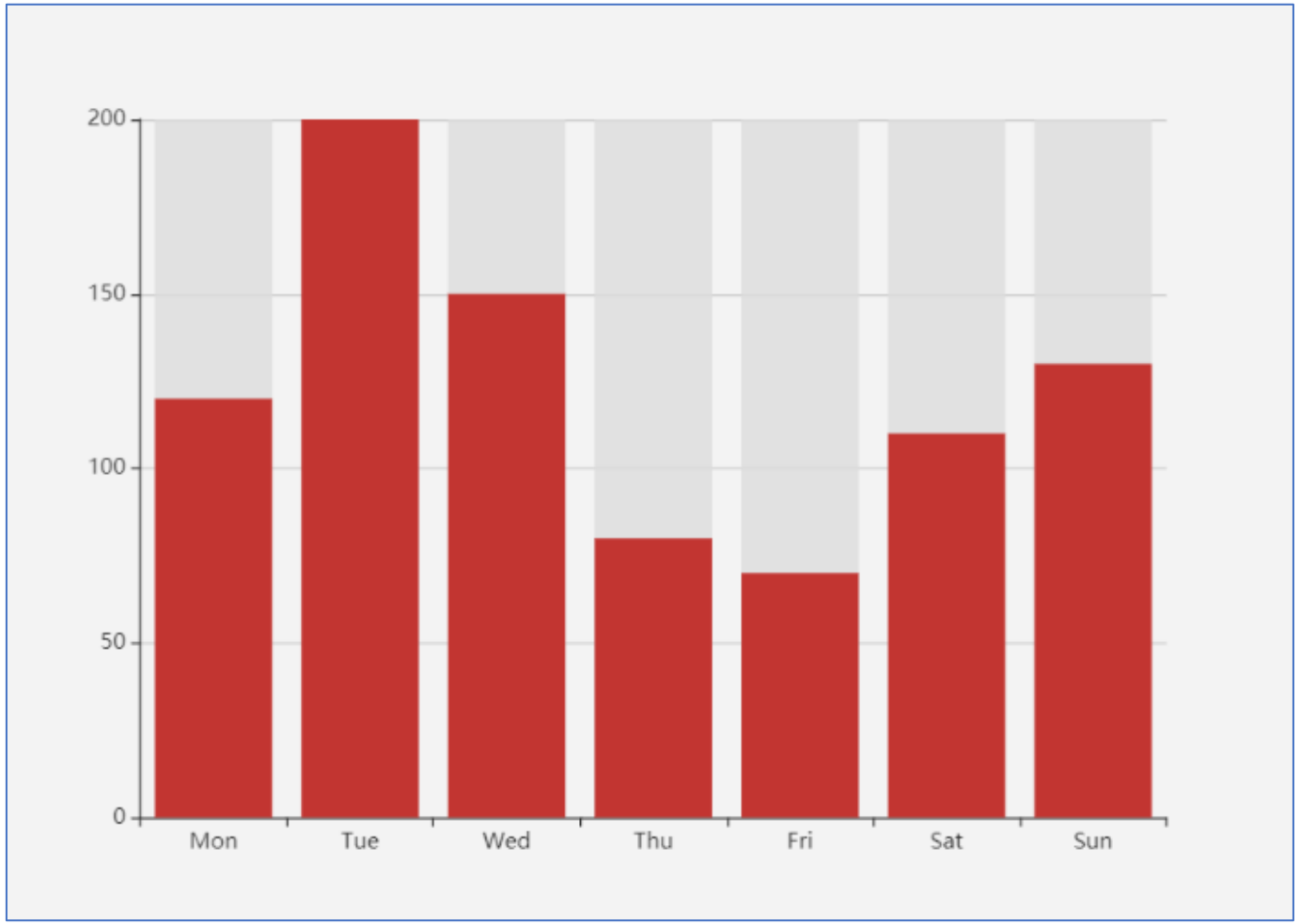
15.1.2 Element

For example, the **line** in the right figure:



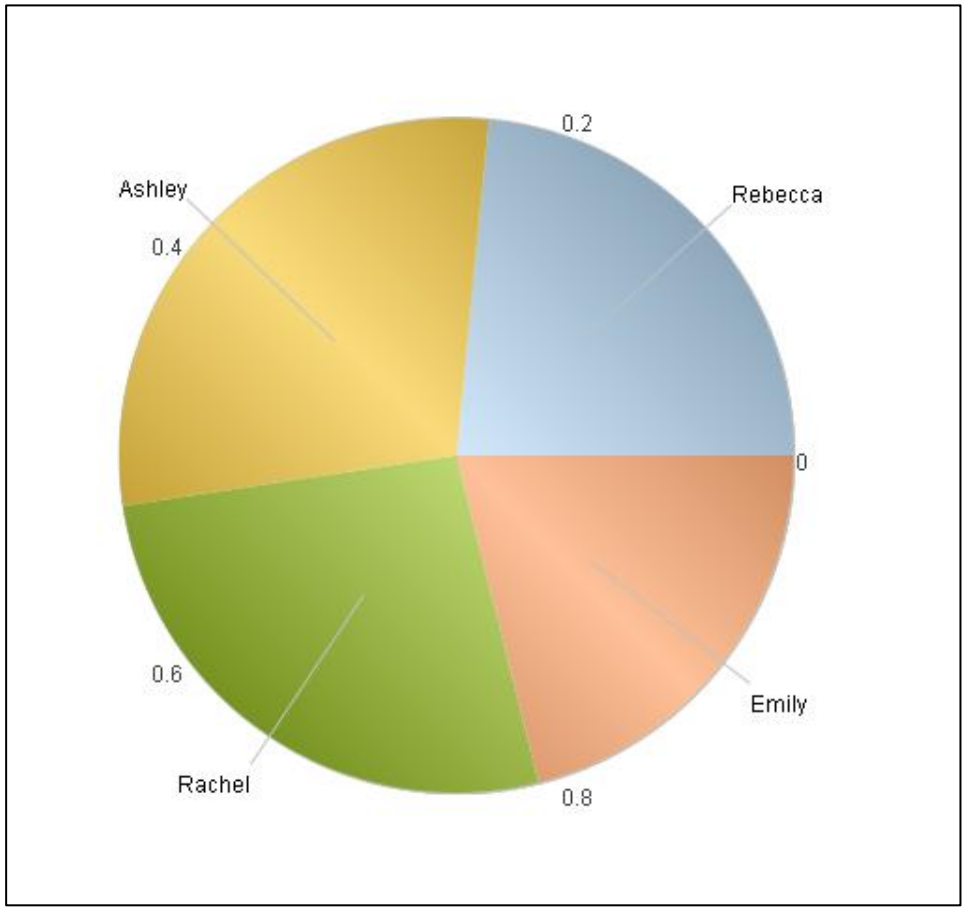
15.1.2 Element

For example, the **column** in the right figure:



15.1.2 Element

For example, the **pie chart** on the right:



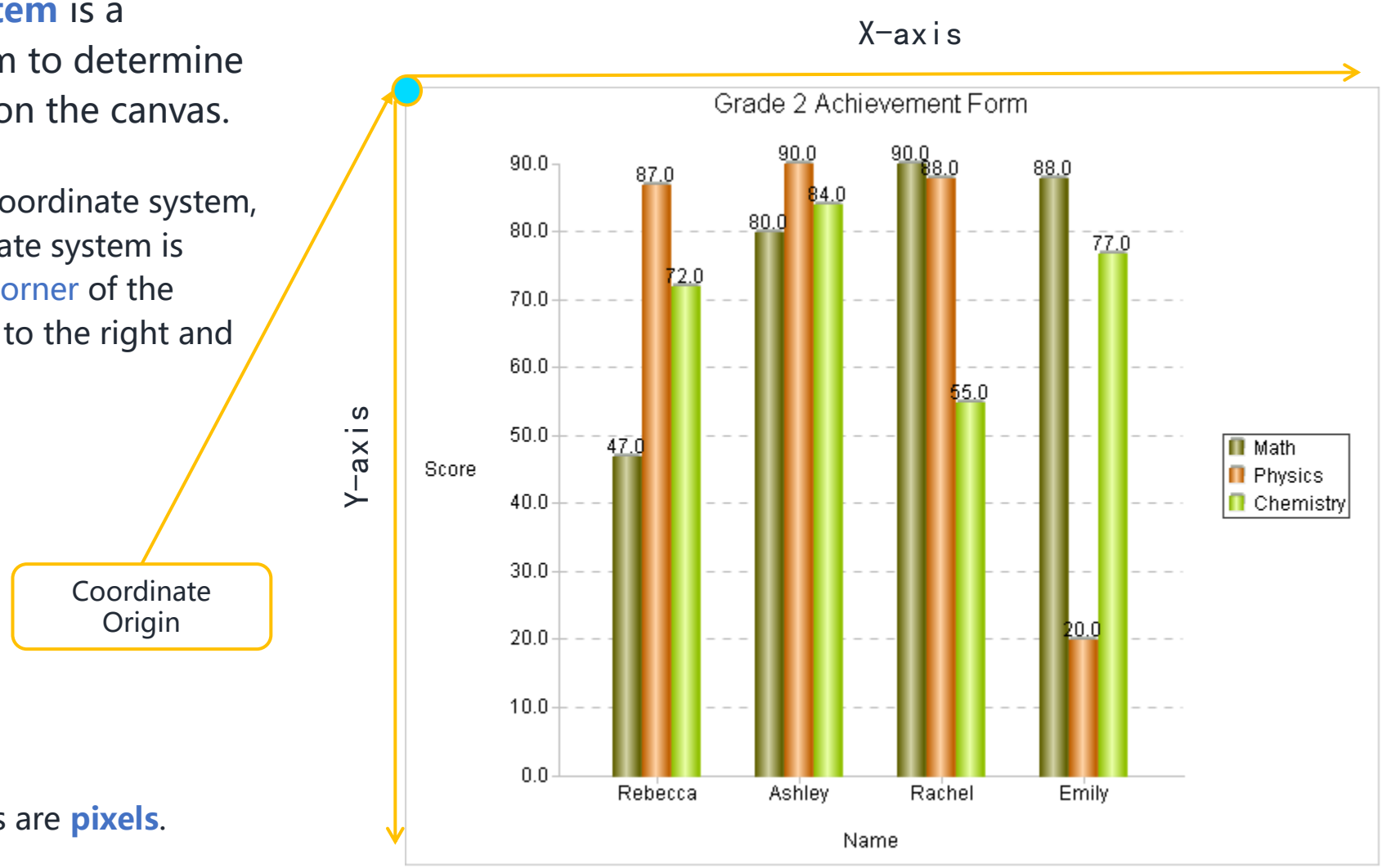
Chapter 15
Graphics drawing

15.2 Physical coordinates

15.2 Physical coordinates

The **physical coordinate system** is a rectangular coordinate system to determine the location of the elements on the canvas.

Different from the conventional coordinate system, the origin of the physical coordinate system is usually located in the upper left corner of the canvas, with the x-axis increasing to the right and the y-axis increasing downward.



The units of physical coordinates are **pixels**.

15.2 Physical coordinates

Understand the coordinates through a piece of code.

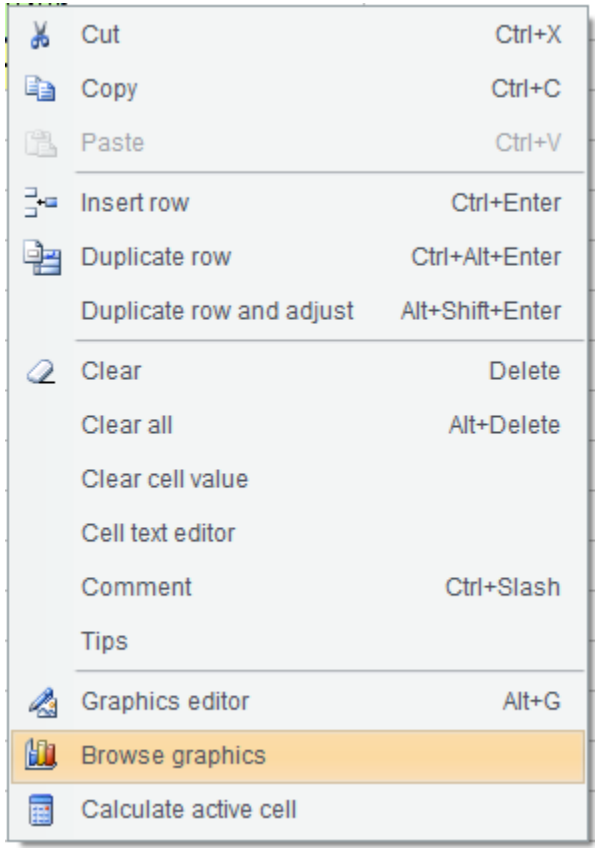
【Example 15.1】Build a 600 pixel wide and 480 pixel high canvas, and then draw the text *Hello, chart!* at coordinates (100, 50).

	A	B
1	=canvas()	/1. Create canvas
2	=A1.plot("Text","text":"Hello,chart!","x":100,"y":50)	/2. Layout text elements into the canvas and set coordinates and values
3	=A1.draw(600,480)	/3. The canvas performs drawing

Find the basic process of drawing from the example code:

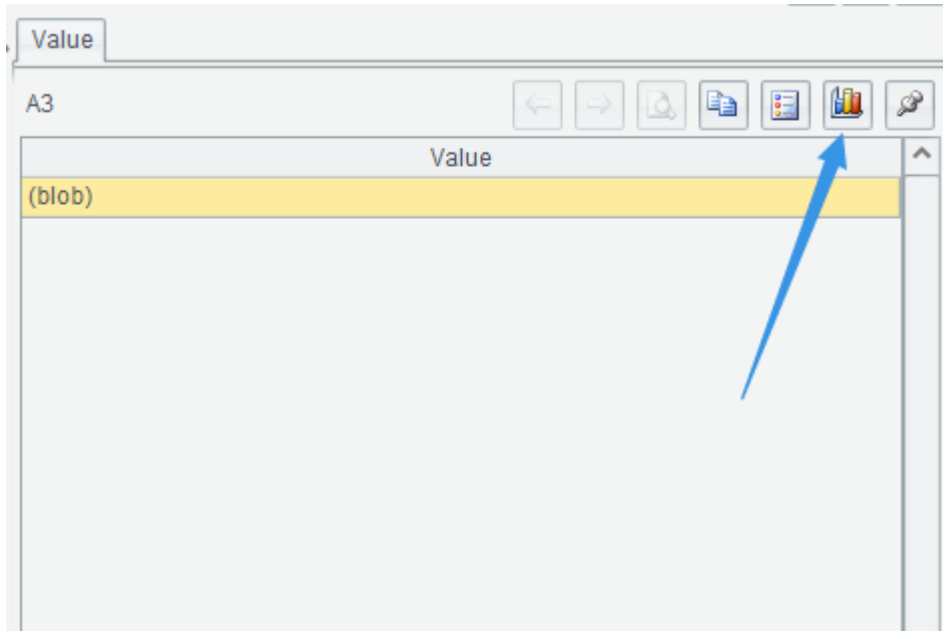
1. Create a canvas with **canvas()**.
2. Plot text into the canvas and set the corresponding parameters.
3. Use **draw(w, h)** to draw the figure.

After executing the code, select the A3 cell, and then click the shortcut menu item *Browse graphics*:

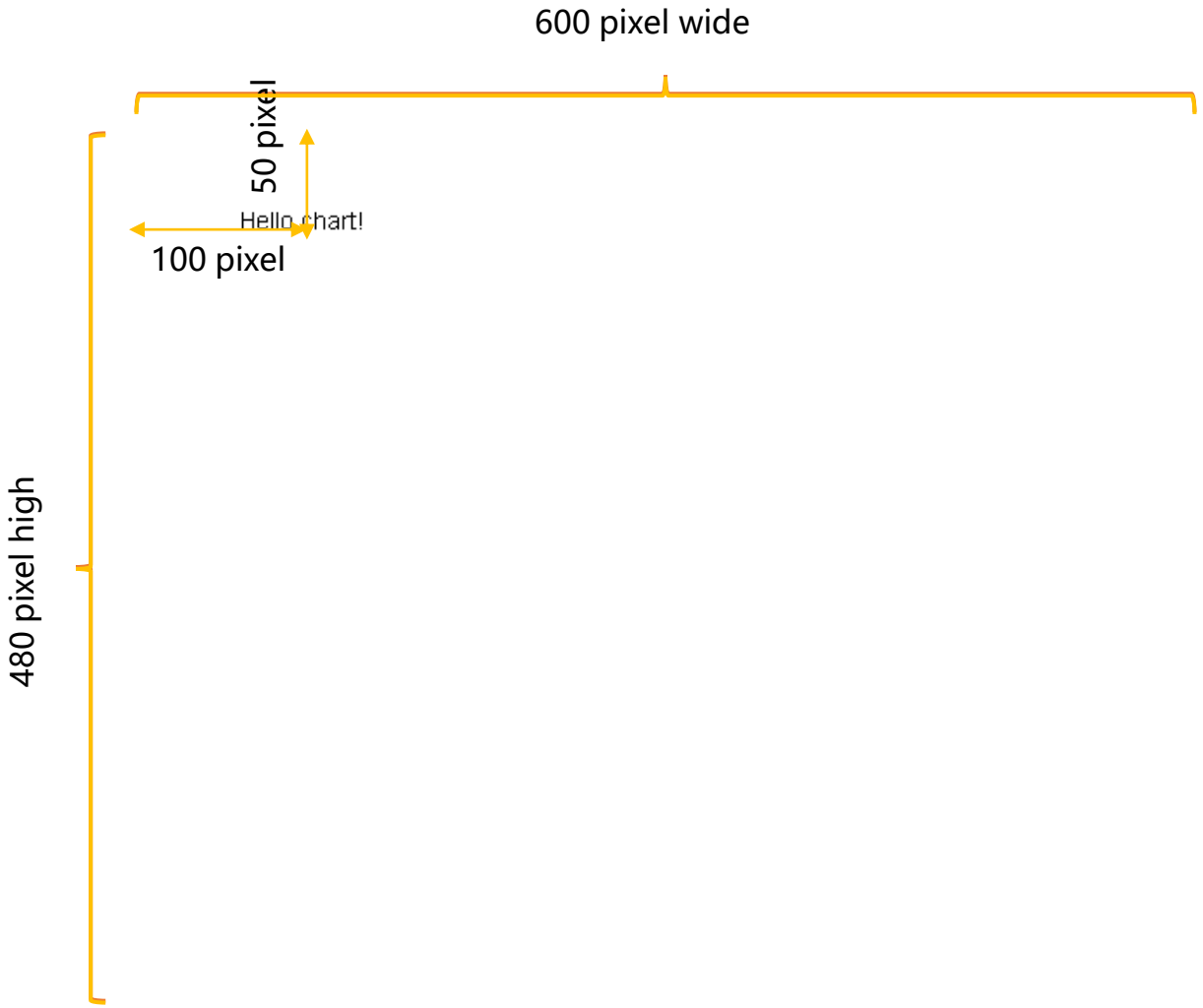


15.2 Physical coordinates

You can also select the A3 cell and click the icon in the value panel on the right :



To view the effect picture of the text example in A3:



15.2 Physical coordinates

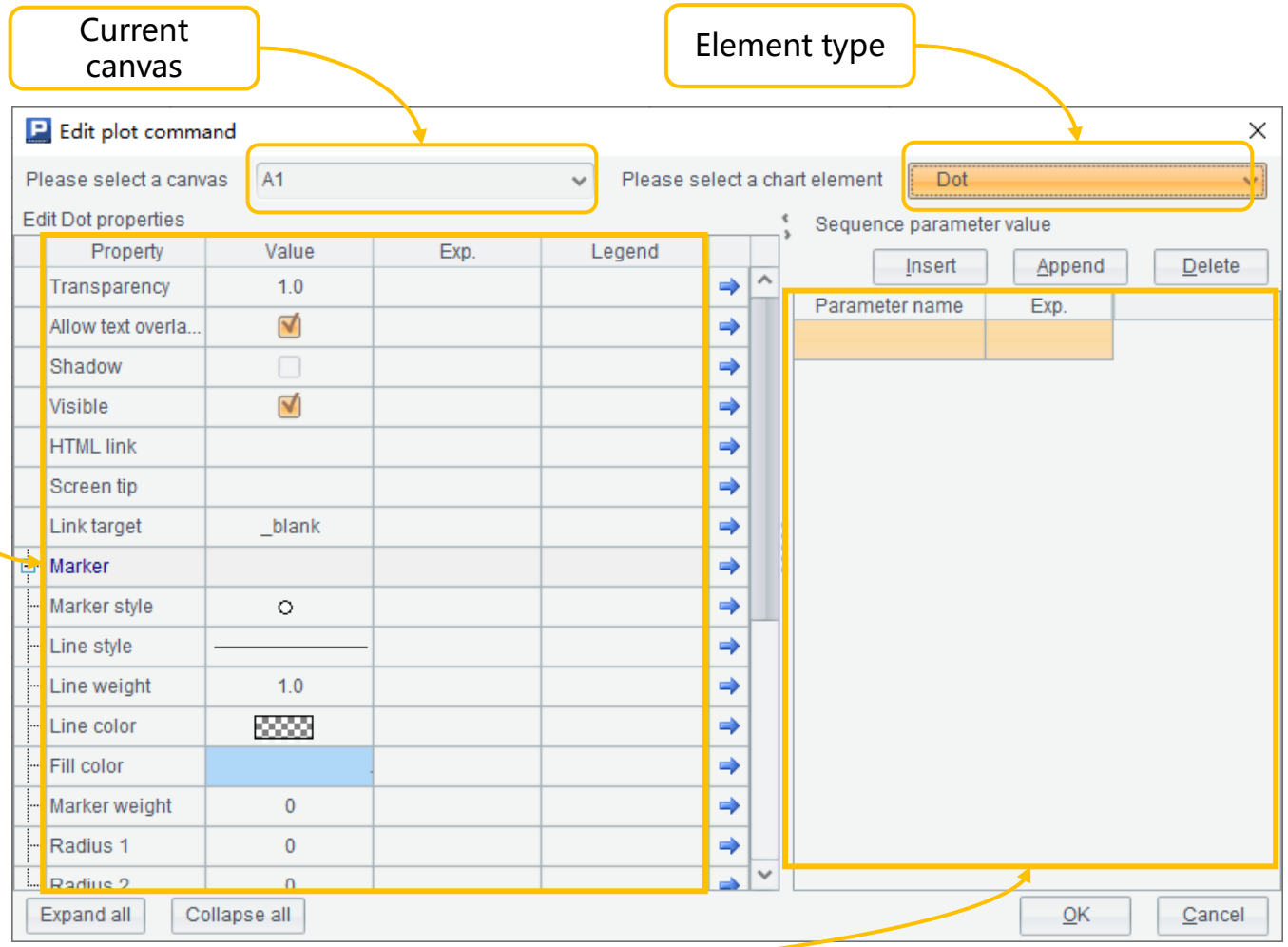
There are many attributes of elements, and it is inconvenient to edit them directly, so the **interface** is usually used to assist.

Right click the cell to pop up the menu and use the ***graphics editor*** window:

Property editing area:
When the value is a constant, just edit the **Value** column.
Edit the **Exp** column when the value needs to be obtained by calculation.

When the content of Value is graph, color or shape, the current expression description value will be displayed in Exp column after editing.

Sequence editing area :
When the property value is a sequence, you can click the **blue arrow** to the right of the property, and then maintain the members of the sequence in this area.



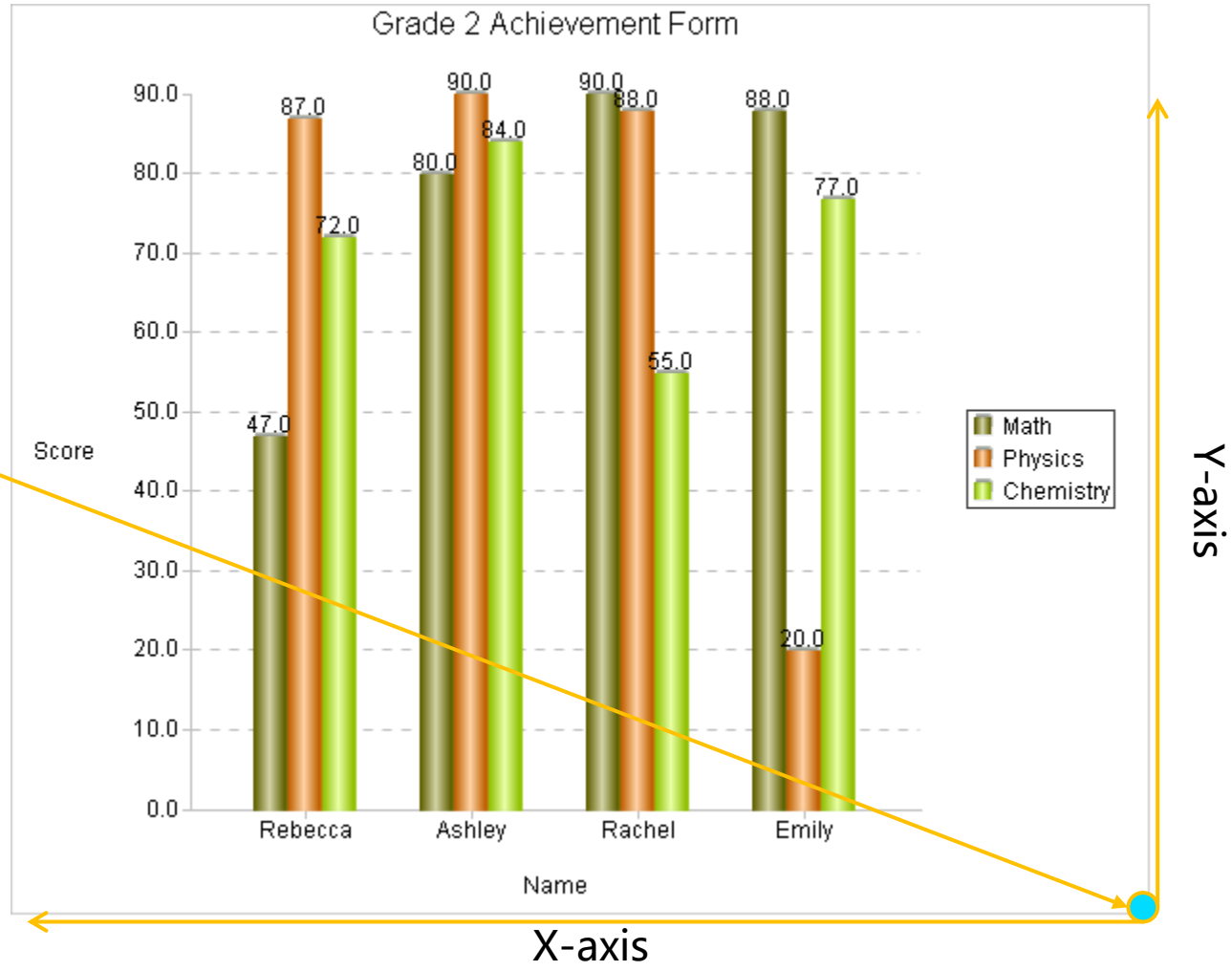
15.2 Physical coordinates

When the physical coordinates are negative, it means that the origin is located at the lower right corner of the canvas. In this case, the absolute value of X axis increases to the left, and the absolute value of Y axis increases upward.

Note:

- The coordinate value determines which origin to use by its value of positive or negative.
- When two coordinate values are positive and negative respectively, their respective origins are used.

Origin when coordinate value is negative



15.2.1 Absolute coordinates

In the physical coordinate system, we directly use **pixel** as the unit of position and size, which is called **absolute coordinate**.

【Example 15.2】Using the absolute coordinates (100, 100), draw the text *Hello, chart!* as a QR code with width and height of 100, and compare the effect under different size canvas.

As shown in the red box in the figure, set the property values of the **text element**.

Edit Text properties

Property	Value	Exp.
Text	Hello,chart!	
Visible	☑	
Multiline style	☐	
Coordinates		
X	100	
Y	100	
Width	100	
Height	100	
Text		
Alignment		
Barcode		
Barcode type	QRCode	
Character set	UTF-8	
Show text	☐	

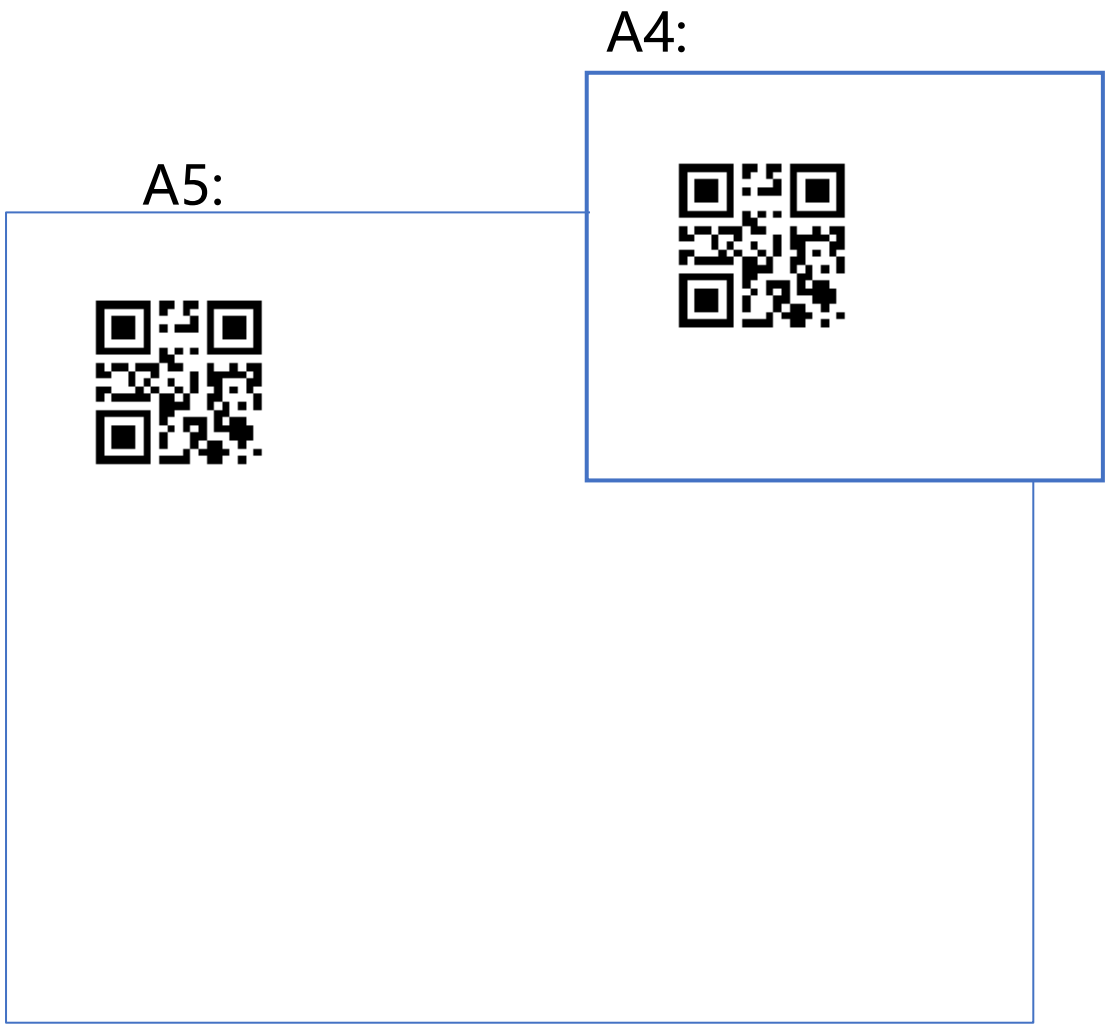
15.2.1 Absolute coordinates

Sample script:

	A	B
1	=canvas()	/1. Create canvas
2	=A1.plot("BackGround")	/2. Set the default white background
3	=A1.plot("Text","text":"Hello,chart!","x":100,"y":100,"width":100,"height":100,"barType":256)	/3. Layout text elements into the canvas and set coordinates and values
4	=A1.draw@p(300,240)	/4. Drawing a small figure
5	=A1.draw@p(600, 480)	/5. Drawing a big figure

15.2.1 Absolute coordinates

Execute the code and view the graphics in cell A4 and A5. You can see that the position and size of the QR code remain unchanged.



Features:

- Absolute coordinates do not change with the size of the canvas.
- The absolute value of absolute coordinates is always greater than 1.

15.2.2 Relative coordinates

When absolute coordinates are used, graphics cannot change proportionally with the size of the canvas. In order to solve this problem, we introduce relative coordinates.

The coordinates expressed by the ratio to the width and the height of the canvas are called **relative coordinates**.

【Example 15.3】Using relative coordinates (0.5, 0.5), draw the text *Hello, chart!* as a QR code with width and height of 0.2, and compare the effect under different canvas.

As shown in the red box in the figure, set the property values of the **text element**:

Edit Text properties

Property	Value	Exp.
Text	Hello,chart!	
Visible	☑	
Multiline style	☐	
Coordinates		
X	0.5	
Y	0.5	
Width	0.2	
Height	0.2	
Text		
Alignment		
Barcode		
Barcode type	QRCode	
Character set	UTF-8	
Show text	☐	

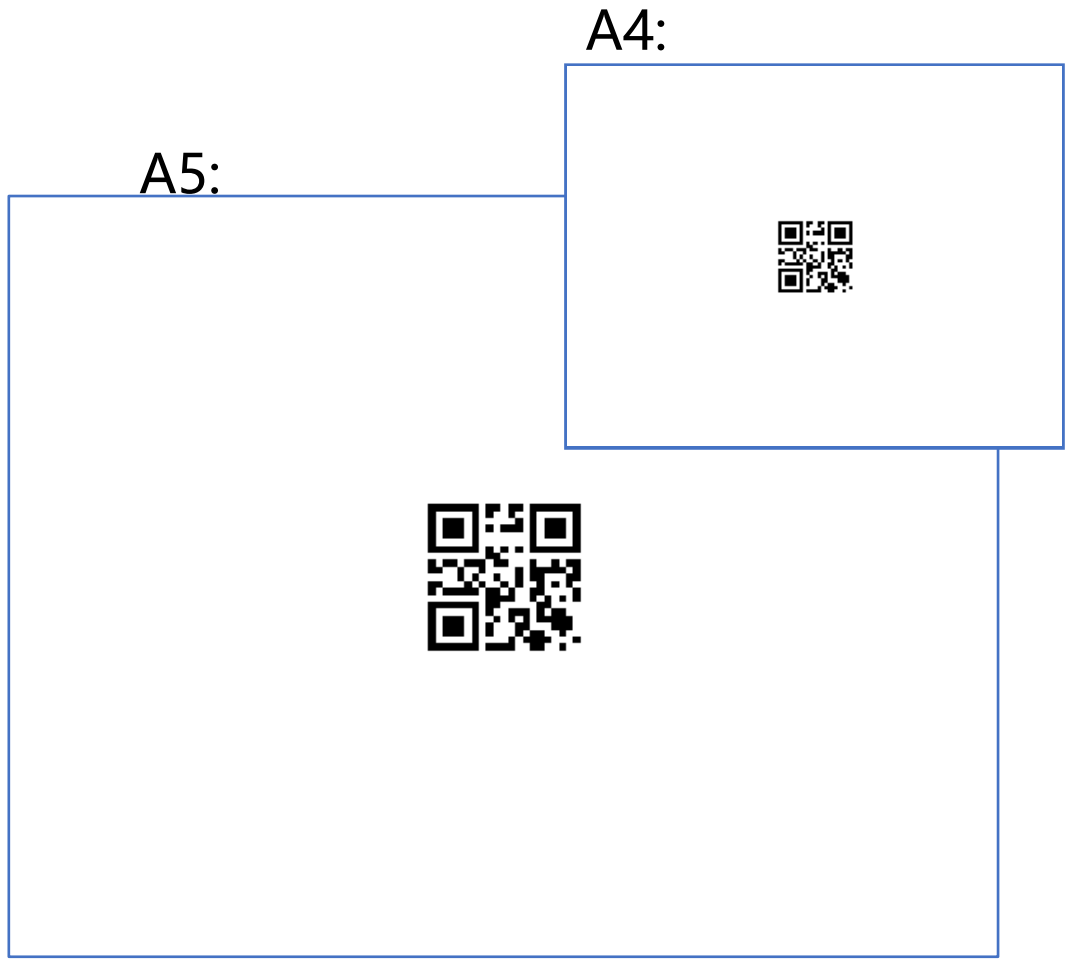
15.2.2 Relative coordinates

Sample script:

	A	B
1	=canvas()	/1. Create canvas
2	=A1.plot("BackGround")	/2. Set the default white background
3	=A1.plot("Text","text":"Hello,chart!","width":0.2,"height":0.2,"barType":256)	/3. Plot text elements and set coordinates and values
4	=A1.draw@p(300,240)	/4. Drawing a small figure
5	=A1.draw@p(600, 480)	/5. Drawing a big figure

15.2.2 Relative coordinates

Execute the code and view the graphics in cell A4 and A5. You can see that the position and size of the QR code change proportionally with the size of the canvas.



Features:

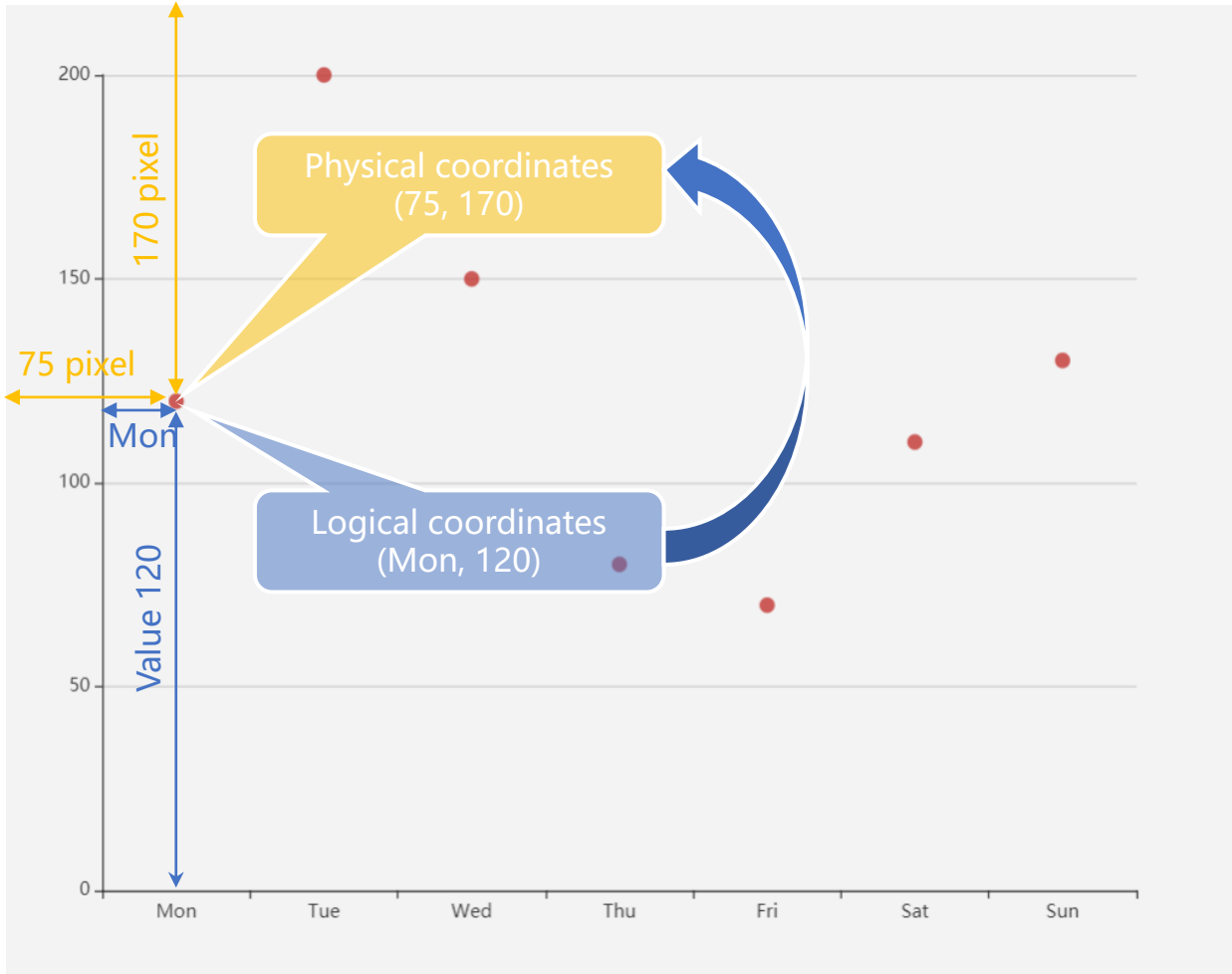
- The relative coordinates change proportionally with the size of the canvas.
- The absolute value of the relative coordinates is always less than or equal to 1.

Chapter 15
Graphics drawing

15.3 Logical coordinates

15.3 Logical coordinates

View the following statistical chart:



The first point represents the business meaning that the sales on Monday is 120. In other words, we want to draw a point on the abscissa of Monday and the ordinate of 120.

However, the computer does not recognize such coordinate values (Monday, 120), and it must be converted into physical coordinates. As a result, a point is drawn at the pixel value of (75, 170).

Obviously, it would be much easier to have a coordinate system that allows us to draw directly at (Monday, 120).

This requires the introduction of the concept of logical coordinate system and logical axis.

15.3 Logical coordinates

The coordinate system that can convert business data (logical coordinates) into physical coordinates in statistical chart is called **logical coordinate system**.

The axis that forms the logical coordinate system is called the **logical axis**.

The corresponding coordinates on the logical axis are **logical coordinates**.

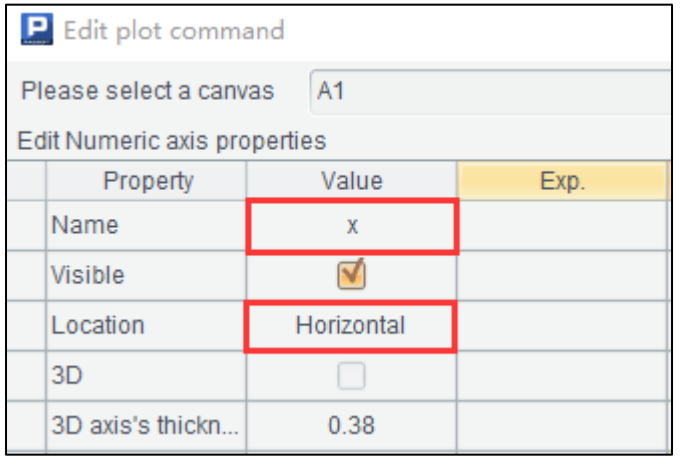
After using the logical coordinate system, we can directly use the logical coordinate value to draw, and the logical axis is responsible for converting the logical coordinates to the physical coordinates.

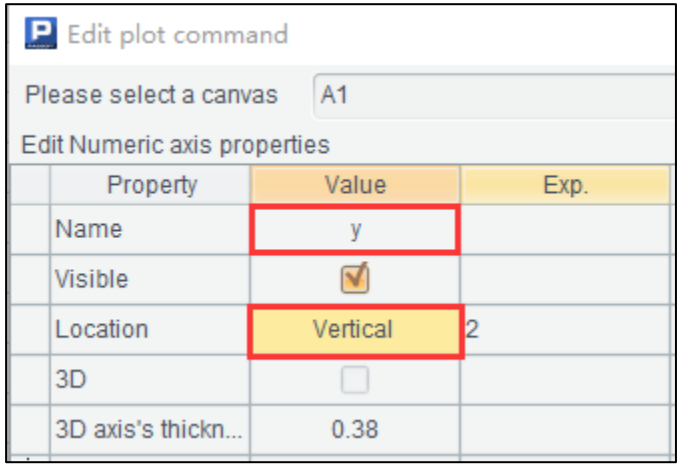
15.3 Logical coordinates

The logical coordinate system is still a rectangular coordinate system, which requires a horizontal logical axis and a vertical logical axis to determine the position.

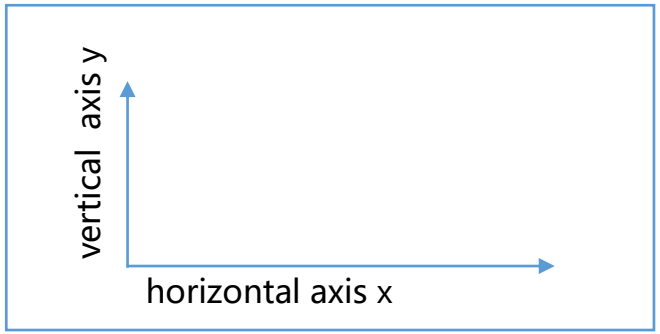
The position of logical axis is determined by physical coordinates.

The default physical coordinates of the logical axis have been set to the appropriate orientation in the canvas using relative coordinates. When using the default coordinates, simply select the position of the axis.





As shown in the right figure, after setting the names and positions of the two logical axes, the logical coordinate system as shown in the figure below can be formed:



15.3.1 Numeric axis

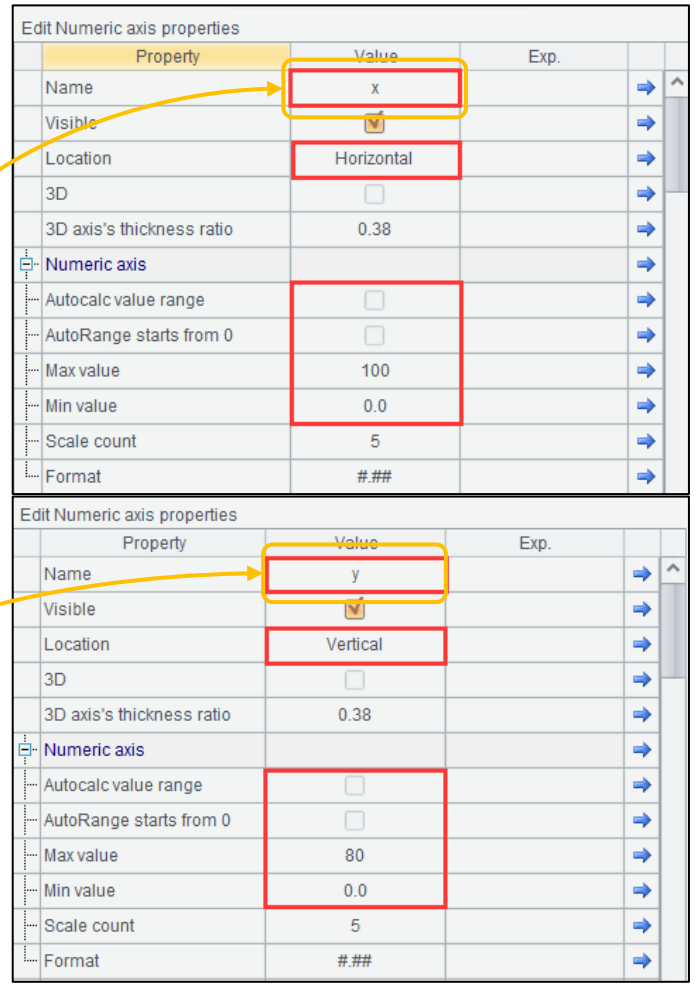
The logical axis with a numeric coordinate value is called the **numeric axis**, which is the simplest logical axis. Its values are continuous.

【Example 15.4】Construct a rectangular coordinate system, specify the value range of the horizontal axis is 0 to 100, and the vertical axis is 0 to 80, and then draw a dot at the numerical coordinates (60, 64).

Select the numeric axis and use the default physical coordinates, and the properties of the numeric axis are shown in the right figure:

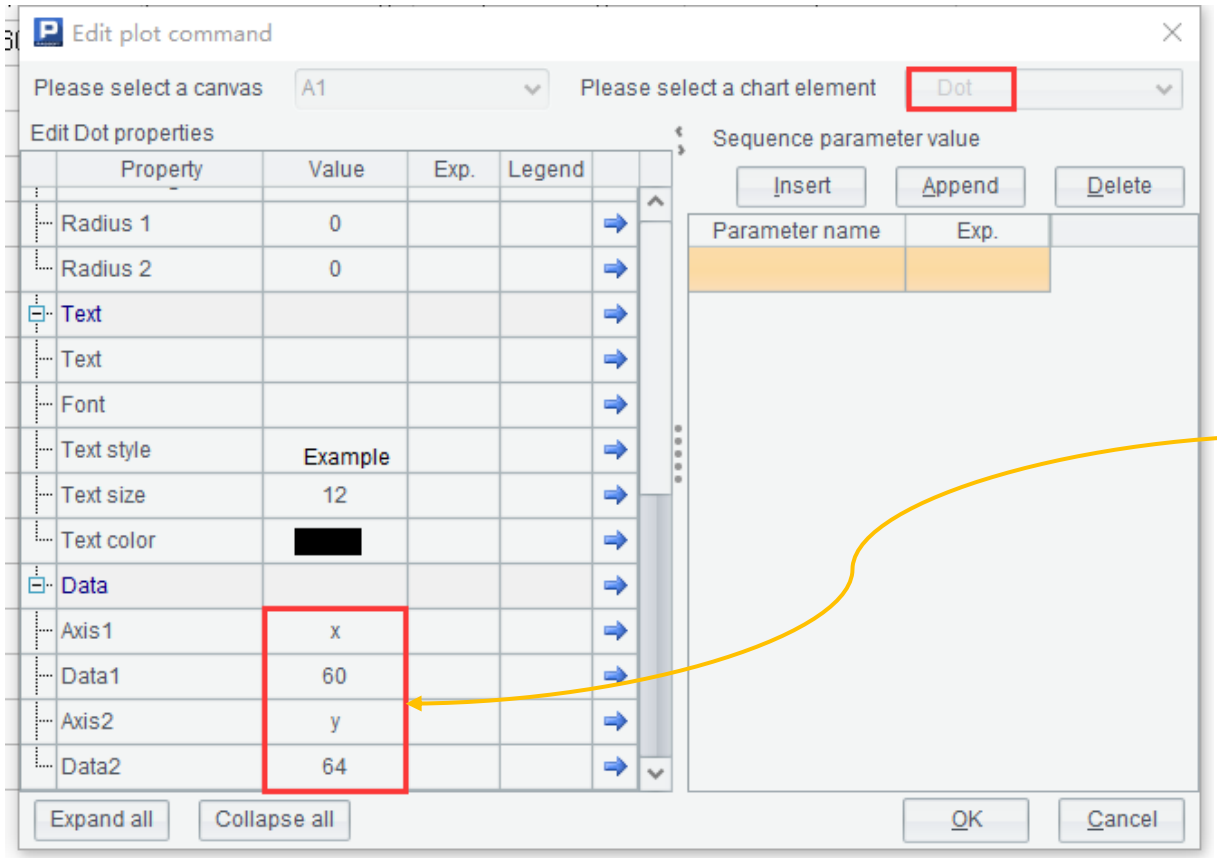
The horizontal axis name is set to x

The vertical axis name is set to y



15.3.1 Numeric axis

Add **dot element** and set properties as shown in the figure. Since the data is a constant value, it is directly filled in the value column:



- The coordinate system of an element can be determined by specifying the axis X and Y through axis1 and axis2 under the data property.
- The business data on each axis corresponds to data1 and data2 respectively.

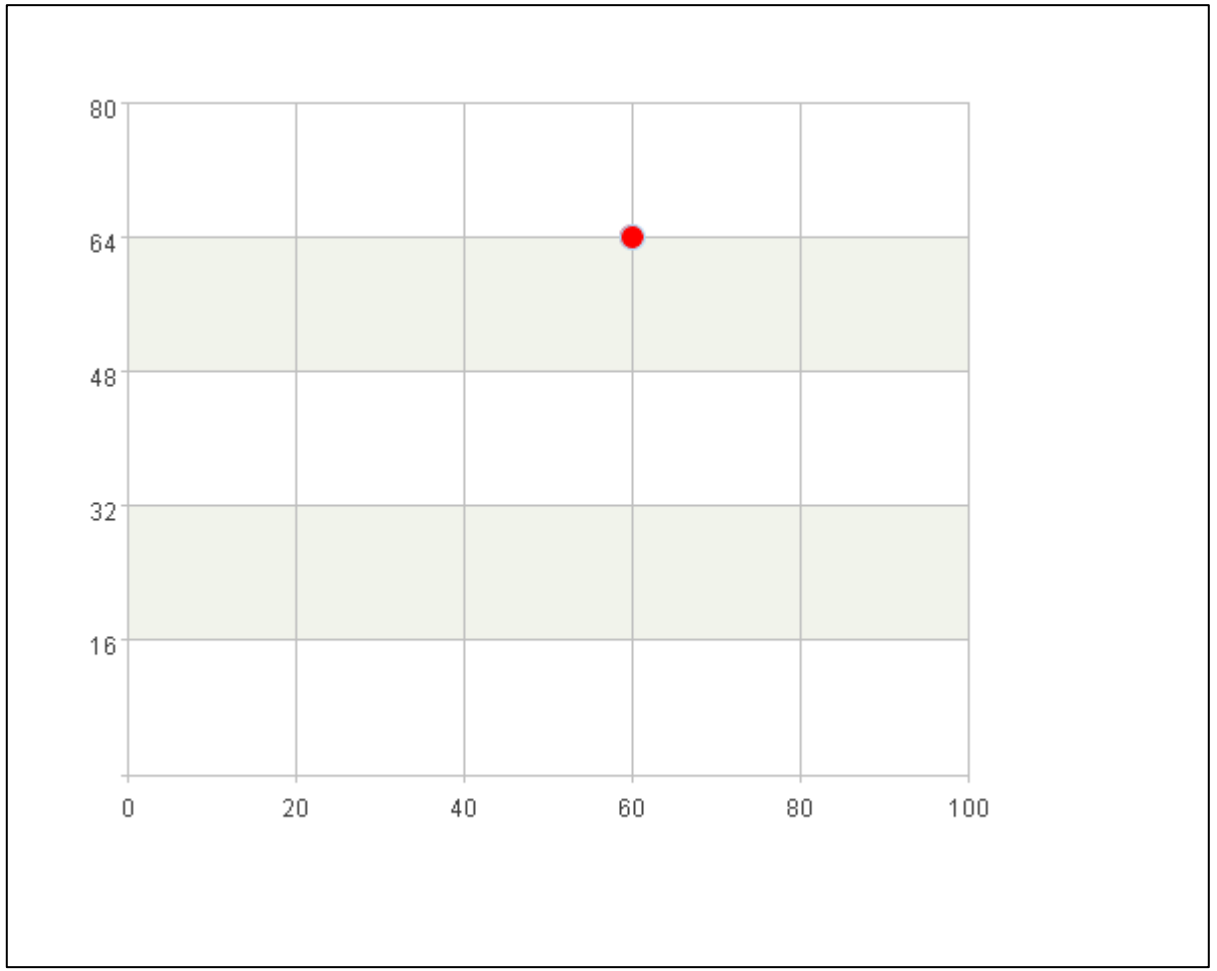
15.3.1 Numeric axis

Sample script:

	A	B
1	=canvas()	/Create canvas
2	=A1.plot("NumericAxis","name":"x","autoCalcValueRange":false,"autoRangeFromZero":false,"maxValue":100,"xStart":0.1,"labelOverlapping":true)	/Plot horizontal axis X
3	=A1.plot("NumericAxis","name":"y","location":2,"autoCalcValueRange":false,"autoRangeFromZero":false,"maxValue":80)	/Plot vertical axis Y
4	=A1.plot("Dot","markerColor":["ChartColor",0,false,-65536,-16777216,0],"markerWeight":6,"axis1":"x","data1":"60","axis2":"y","data2":"64")	/Plot dot element and set values and coordinate system
5	=A1.plot("BackGround")	/Default white background
6	=A1.draw@p(600,480)	/Draw the figure

15.3.1 Numeric axis

After execution, check the **dot graph** in A6:



15.3.1 Numeric axis

【Example 15.5】Construct a rectangular coordinate system, and draw 3 groups * 20 points randomly. The vertical axis interval for each point is [0, 1000]. The horizontal axis interval for the first group of points is [0, 10], for the second group is [0, 100], and for the third group is [0, 1000].

Similar to the previous section, we only need to add 20 random dots to each interval circularly. The example script is as follows:

	A	B
1	=canvas()	/Create canvas
2	=A1.plot("NumericAxis","name":"x","autoCalcValueRange":false,"maxValue":1000,"scaleNum":3)	/Plot horizontal axis X
3	=A1.plot("NumericAxis","name":"y","location":2)	/Plot vertical axis Y
4	for 20	=A1.plot("Dot","axis1":"x","data1":rand(10),"axis2":"y","data2":rand(1000))
5		=A1.plot("Dot","axis1":"x","data1":rand(100),"axis2":"y","data2":rand(1000))
6		=A1.plot("Dot","axis1":"x","data1":rand(1000),"axis2":"y","data2":rand(1000))
7	=A1.plot("BackGround")	
8	=A1.draw@p(600,480)	

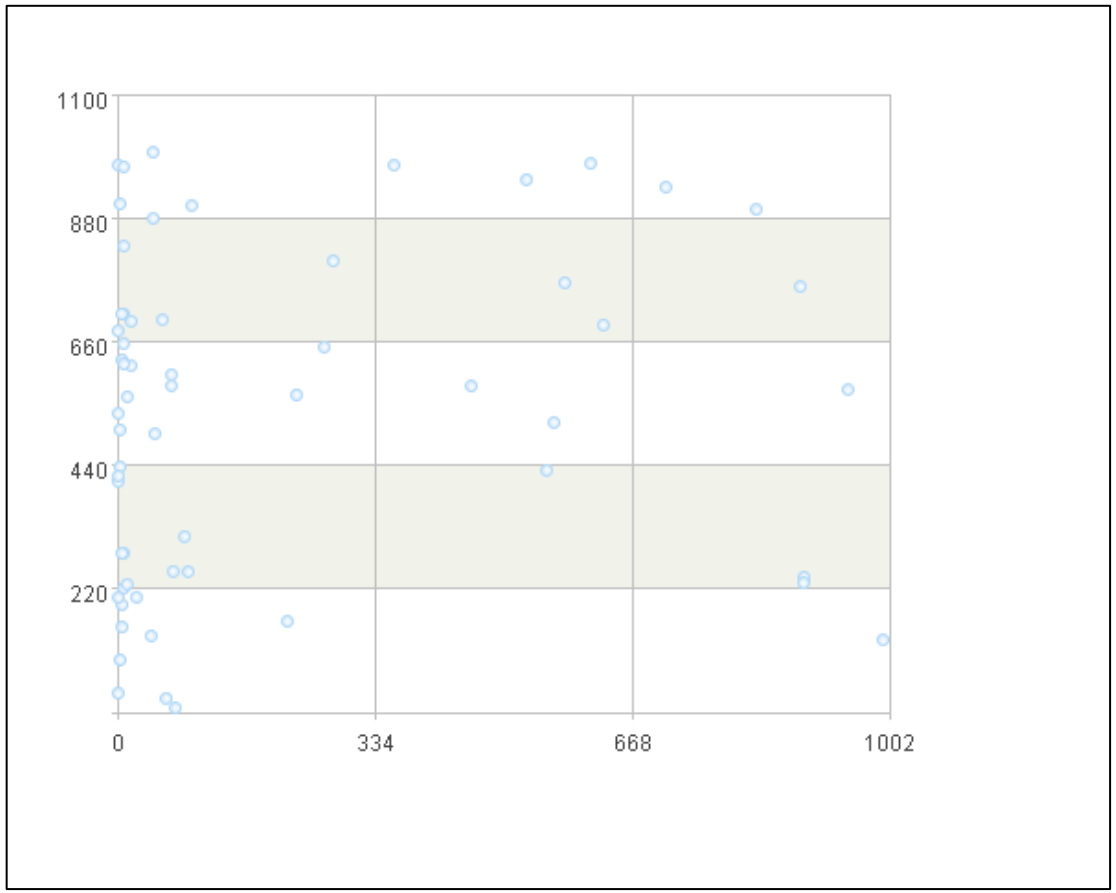
The property settings of **dot element** are shown in the figure. Now the values are expressions and need to be calculated, so enter them in Exp column:

Edit Dot properties

Property	Value	Exp.
Transparency	1.0	
Allow text overla...	☑	
Shadow	☐	
Visible	☑	
HTML link		
Screen tip		
Link target	_blank	
⊞ Marker		
⊞ Text		
⊟ Data		
Axis1	x	
Data1		rand(1000)
Axis2	y	
Data2		rand(1000)

15.3.1 Numeric axis

After execution, check the **random dot graph** in A8:



Note:

- The coordinate range of the numeric axis is automatically calculated according to the range of given value by default.
- The coordinate range of the numeric axis can also be specified manually.
- The value of the numeric axis also supports numbers of string type.

15.3.1 Numeric axis

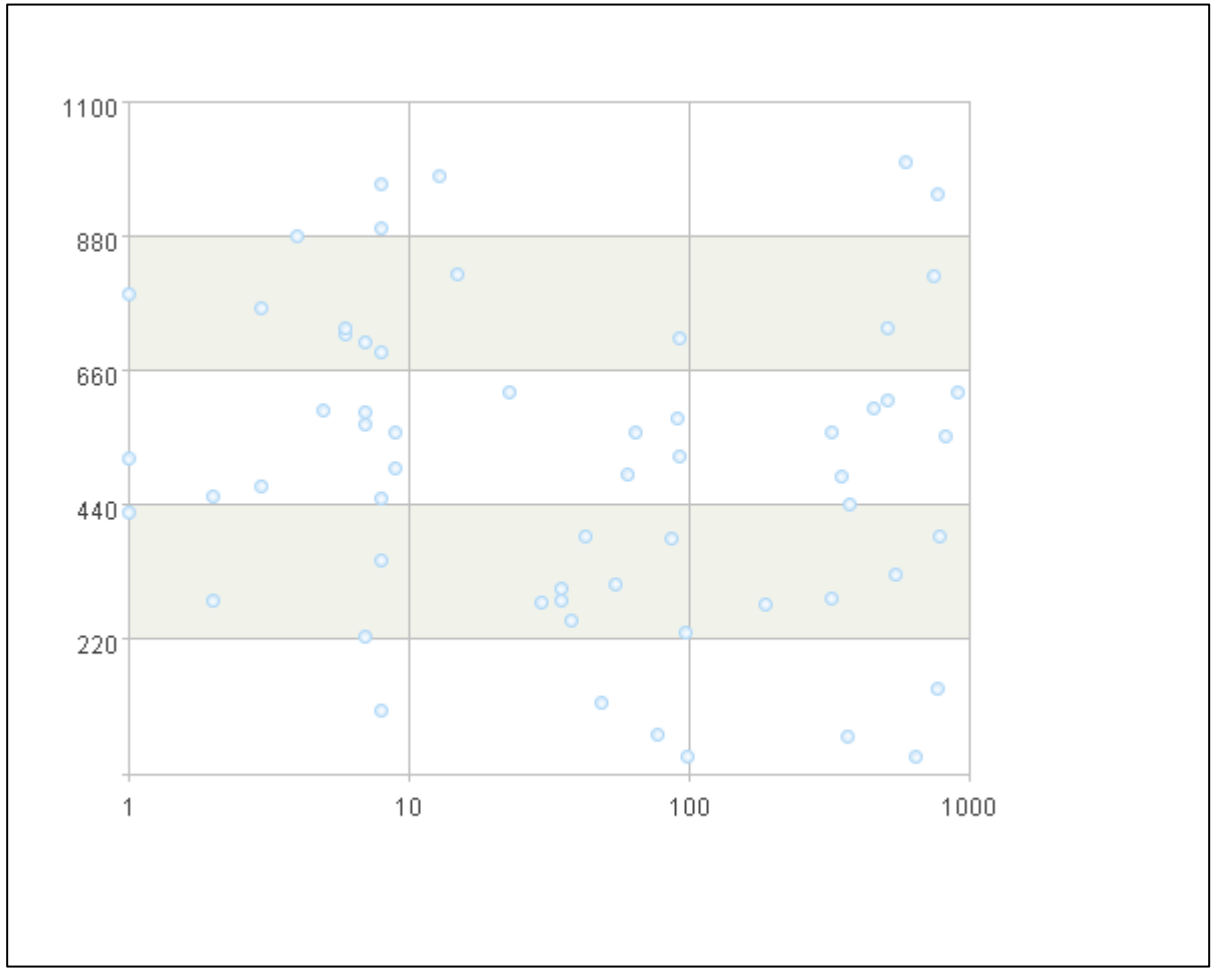
The numerical axis can also have nonlinear transformation.

For example, the dots in the figure above are not evenly distributed in the X direction. In this case, set the transformation type of the coordinate axis, and set the **Transform** property of the x-axis in the above example to **logarithm**.

Edit Numeric axis properties

Property	Value
Name	x
Visible	☑
Location	Horizontal
3D	☐
3D axis's thickn...	0.38
⊞ Numeric axis	
⊟ Transform	
Transform	Logarithm
Scale	1.0
Logarithmic ba...	10.0
Power exponent	2.718281828459045

After executing it again, check the evenly distributed random dot graph in A8 after logarithmic transformation:



15.3.1 Numeric axis

Generally, there are many data used for drawing. It is complex to draw with loop statement. We can also use sequence parameters to make the element draw multiple dots at one time.

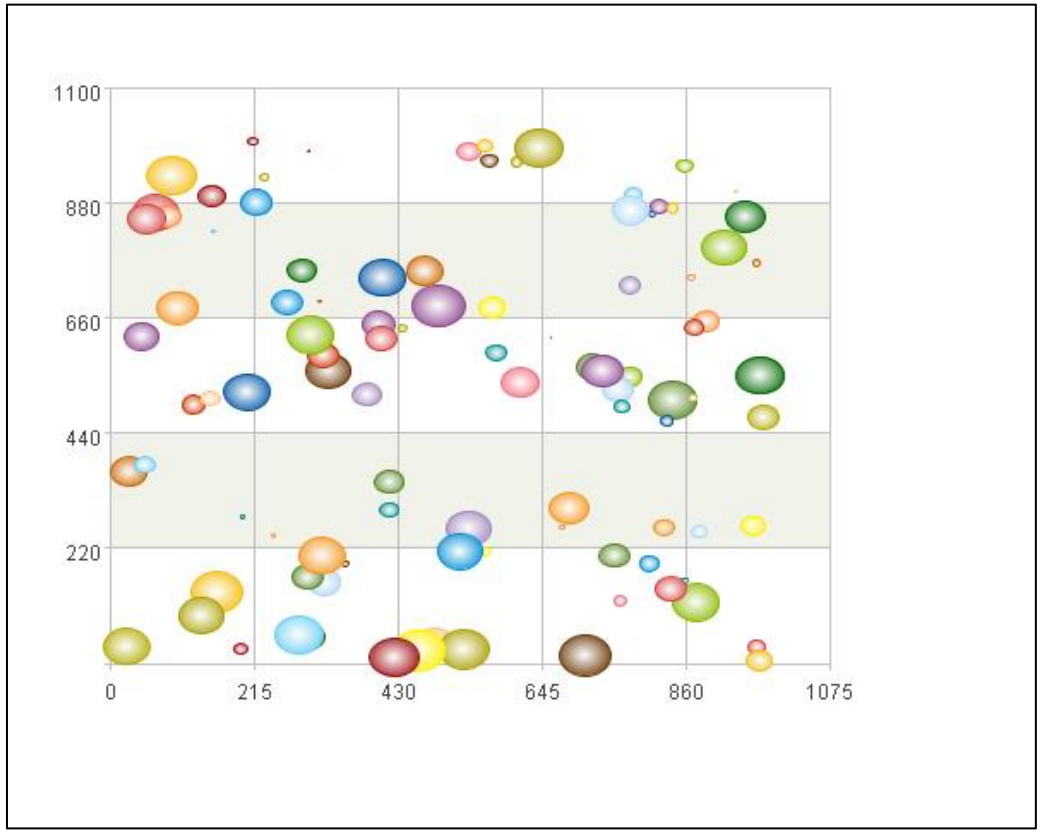
【Example 15.6】Construct a rectangular coordinate system, and draw100 dots with different radii randomly to show the bubble chart.

The example script is as follows:

	A	B
1	>xx=100.(rand(1000))	/Generate 100 x-coordinates with random range within 1000
2	>yy=100.(rand(1000))	/Y coordinates
3	>radius=100.(rand(40))	/Radius of dot
4	=canvas()	/Create canvas
5	=A4.plot("NumericAxis","name":"x")	/Plot numeric axis x
6	=A4.plot("NumericAxis","name":"y","location":2)	/Plot numeric axis y
7	=A4.plot("Dot","radius1":radius,"radius2":radius,"axis1":"x","data1":xx,"axis2":"y","data2":yy)	/Plot dot elements and specify data sequence directly
8	=A4.draw(600,480)	/Draw the figure

15.3.1 Numeric axis

View the **bubble chart** in A8 after execution:



Note:

- The radius of a dot has two values, radius1 and radius2, which respectively represent the radius values on the corresponding axis.
- The unit of radius is not pixel, but also the value of business meaning.

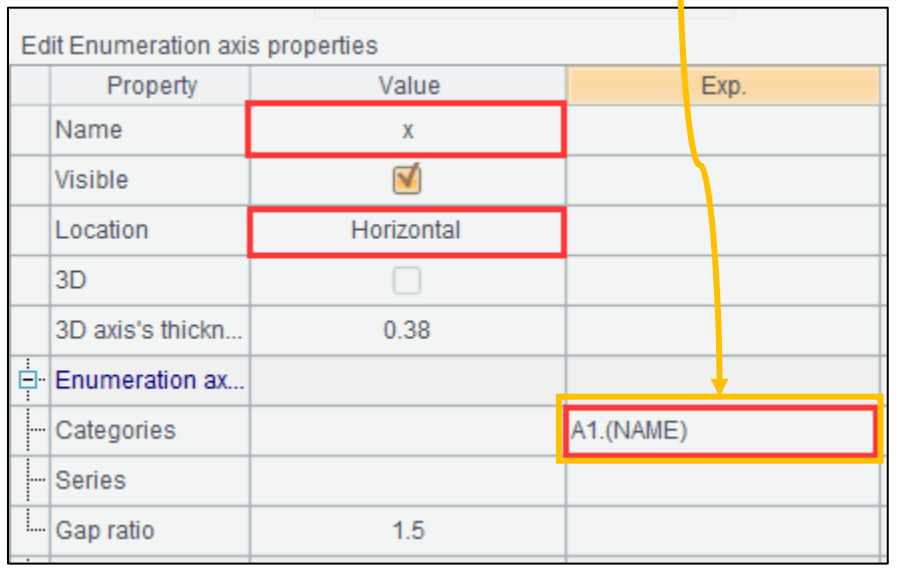
15.3.2 Enumeration axis

The logic axis with the coordinate values that can only be of members of a sequence is called **enumeration axis**, and the values of enumeration axis are discontinuous.
The coordinate values of the enumeration axis are often referred to as **categories**, such as category Rebecca.

【Example 15.7】Draw the data of a score table (summary.txt) as a line chart. The data is as follows:

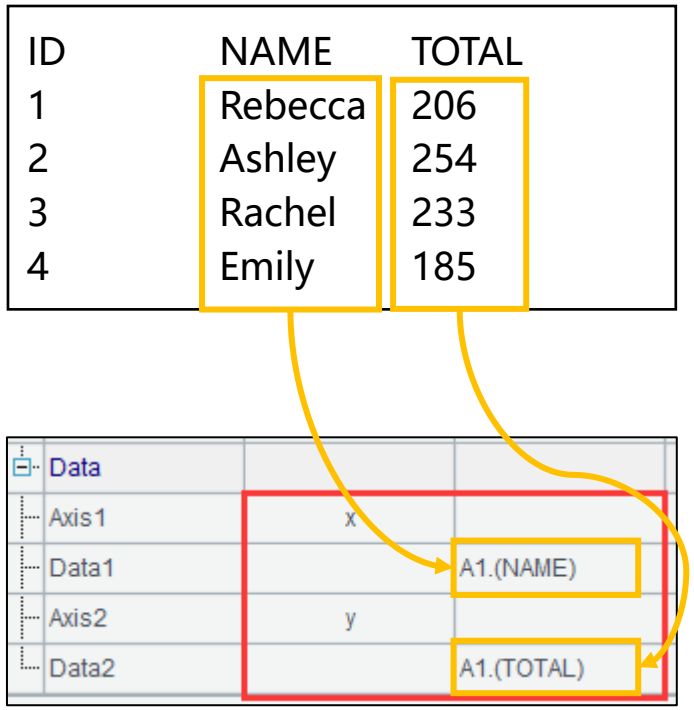
ID	NAME	TOTAL
1	Rebecca	206
2	Ashley	254
3	Rachel	233
4	Emily	185

Set the name, location and category data of **enumeration axis**, as well as the data content of categories, as shown in the figure on the right:



15.3.2 Enumeration axis

Then edit the **line element**, and set the relevant coordinate system and data expression. The data schematic diagram of the corresponding coordinate axis is as follows:

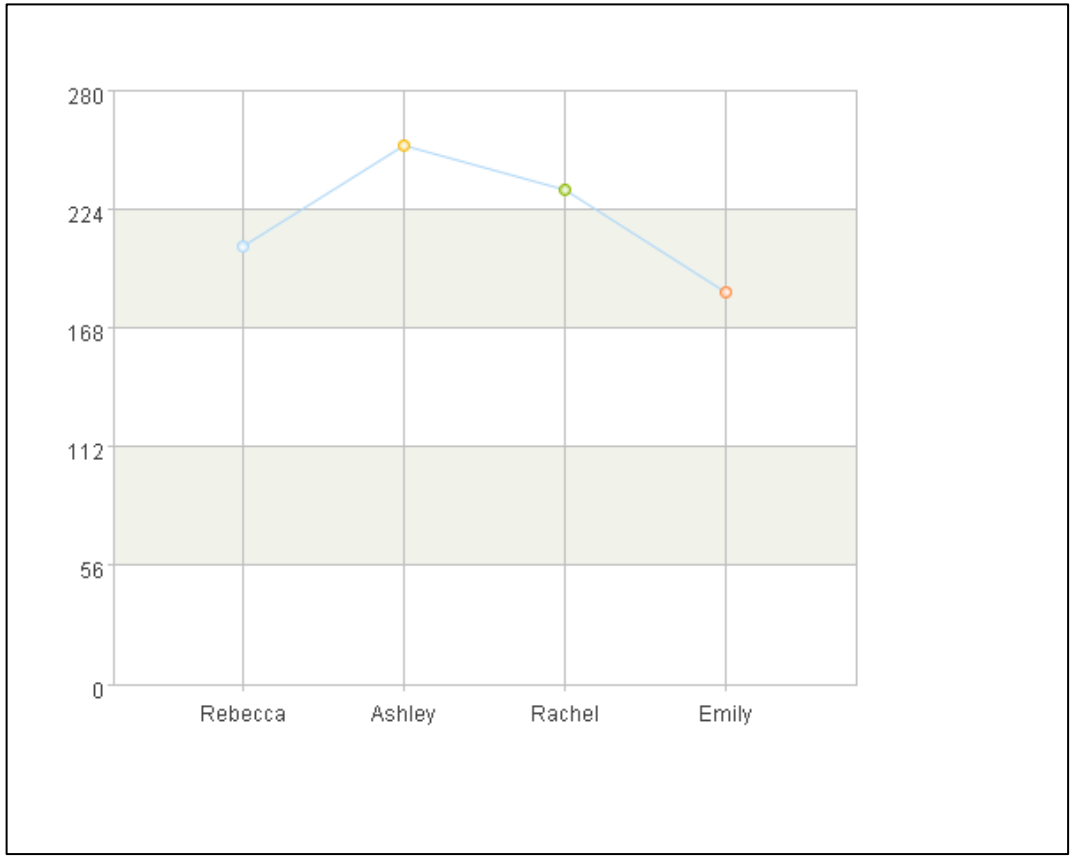


The sample script is as follows:

	A	B
1	=file("summary.txt").import@t()	/1.Import data
2	=canvas()	/2. Create canvas
3	=A2.plot("EnumAxis","name":"x","categories":A1.(NAME))	/3. Enumeration horizontal axis X
4	=A2.plot("NumericAxis","name":"y","location":2)	/4. Numeric vertical axis Y
5	=A2.plot("Line","axis1":"x","data1":A1.(NAME),"axis2":"y","data2":A1.(TOTAL))	/5. Plot line elements in the canvas, set data and associate coordinate system
6	=A2.draw(600,480)	/6. Draw the figure

15.3.2 Enumeration axis

After execution, view the **line chart** in A6:



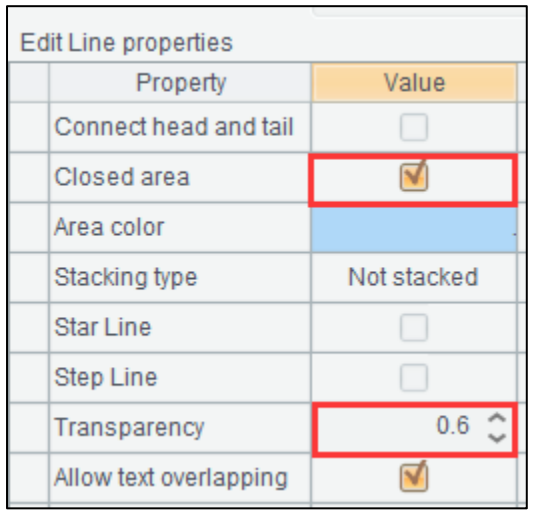
Note:

- If the categories and series content of enumeration axis are not specified, it will be calculated automatically according to the data of corresponding element.
- If you manually specify the category or series content of the enumeration axis, the range must contain the content of the actual element.

15.3.2 Enumeration axis

【Example 15.8】Check the closed area of the line chart, so that the line chart is displayed as an area chart.

Just add two properties of A5 cell in the script of 【example 15.7】, the added property settings are as follows:

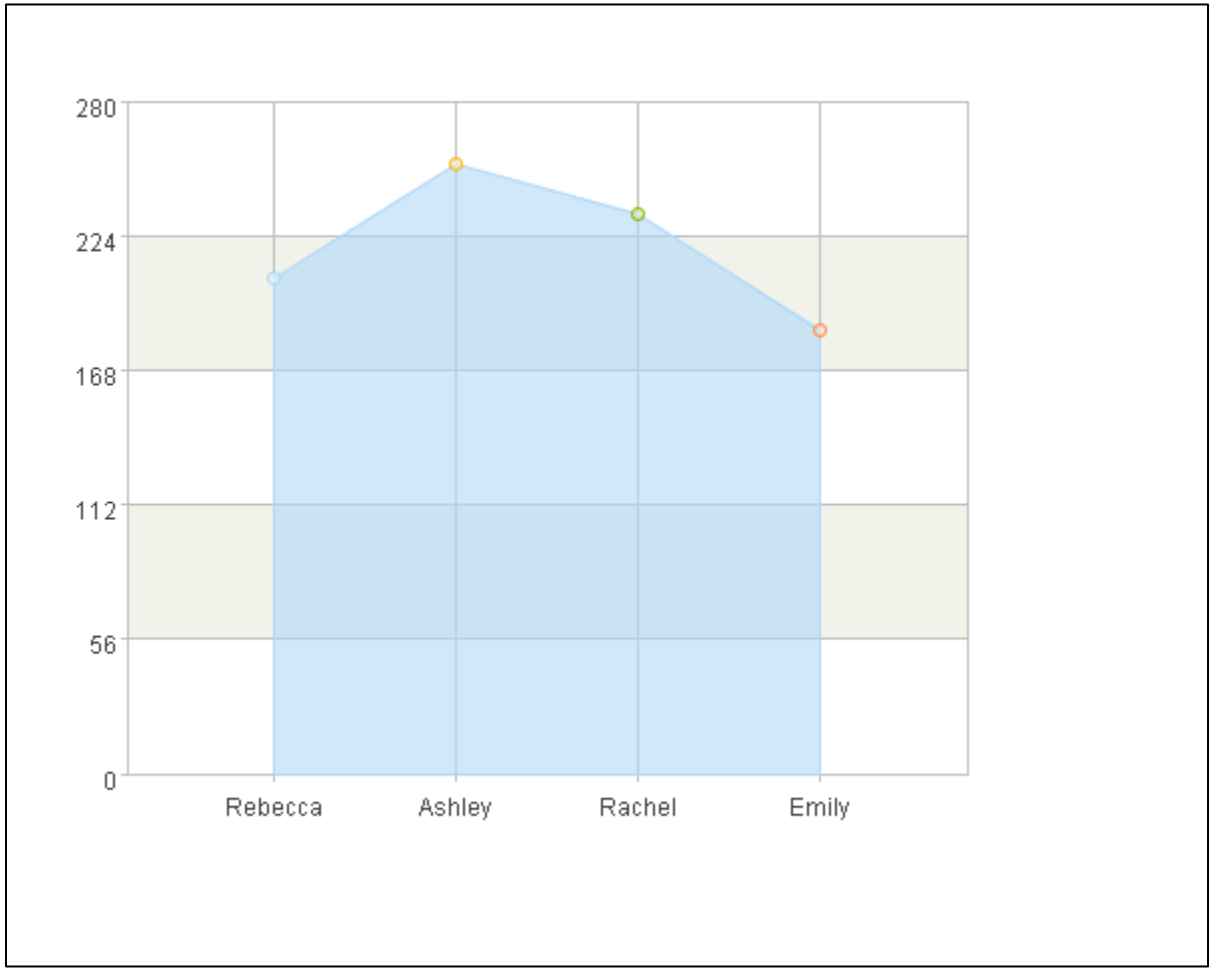


The sample script is as follows:

	A	B
1	=file("summary.txt").import@t()	/1.Import data
2	=canvas()	/2. Create canvas
3	=A2.plot("EnumAxis","name":"x")	/3. Enumeration horizontal axis X
4	=A2.plot("NumericAxis","name":"y","location":2)	/4. Numeric vertical axis Y
5	=A2.plot("Line","closedArea":true,"transparent":0.6,"axis1":"x","data1":A1.(NAME),"axis2":"y","data2":A1.(TOTAL))	/5. Plot line elements in the canvas, set data and associate coordinate system
6	=A2.draw(600,480)	/6. Draw the figure

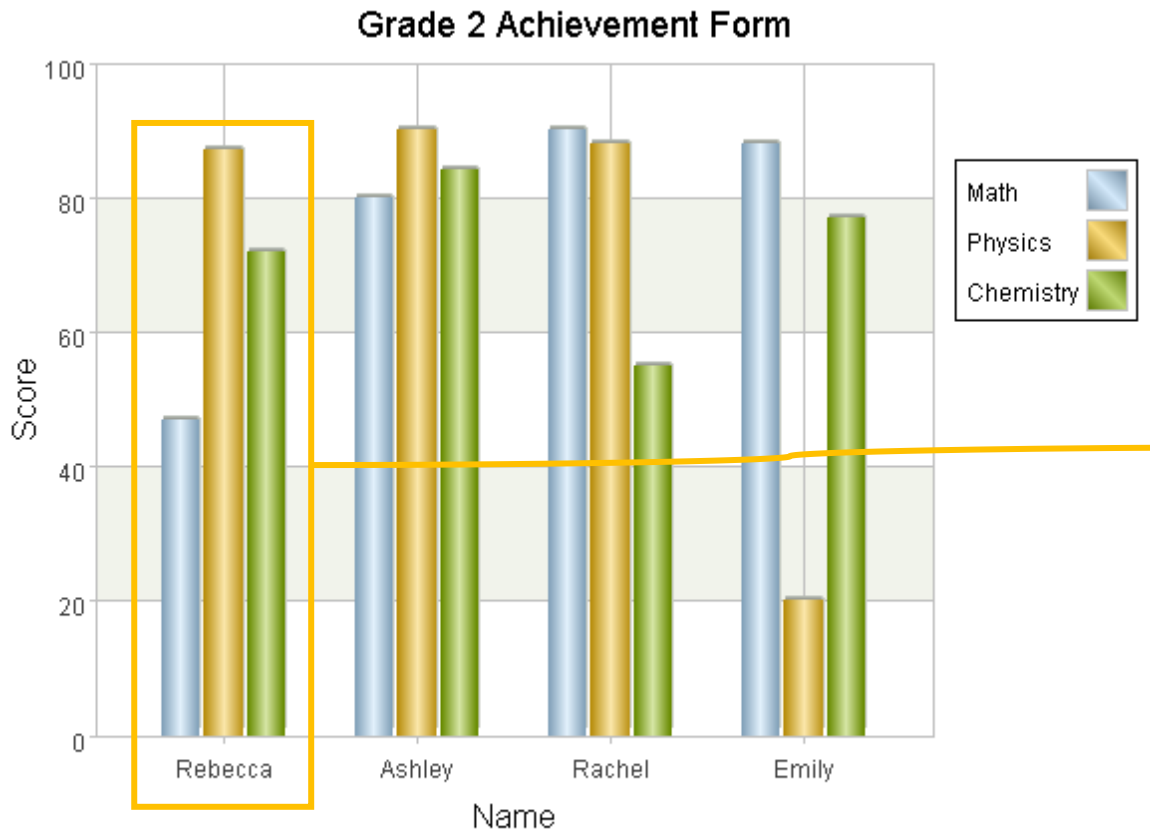
15.3.2 Enumeration axis

After execution, view the **area chart** in A6:



15.3.2 Enumeration axis

Observe the following statistical chart:



Under the category coordinate Rebecca, there are three columns corresponding to math, physics and chemistry.

In other words, the data of enumeration axis can have two layers. In this case, the first layer is called **category**, and the second layer is called **series**.

15.3.2 Enumeration axis

【Example 15.9】Draw the scores of each subject in score table (scores.txt) into a multi-series column chart with column elements, and realize the effect of complete statistical chart in the previous page.

The score table data is as follows:

ID	NAME	Math	Physics	Chemistry
1	Rebecca	47	87	72
2	Ashley	80	90	84
3	Rachel	90	88	55
4	Emily	88	20	77

Category value

Series value

As shown in the figure below, set the properties of **enumeration axis**, where the data of category and series can also be omitted. If omitted, it will be automatically obtained from the associated element:



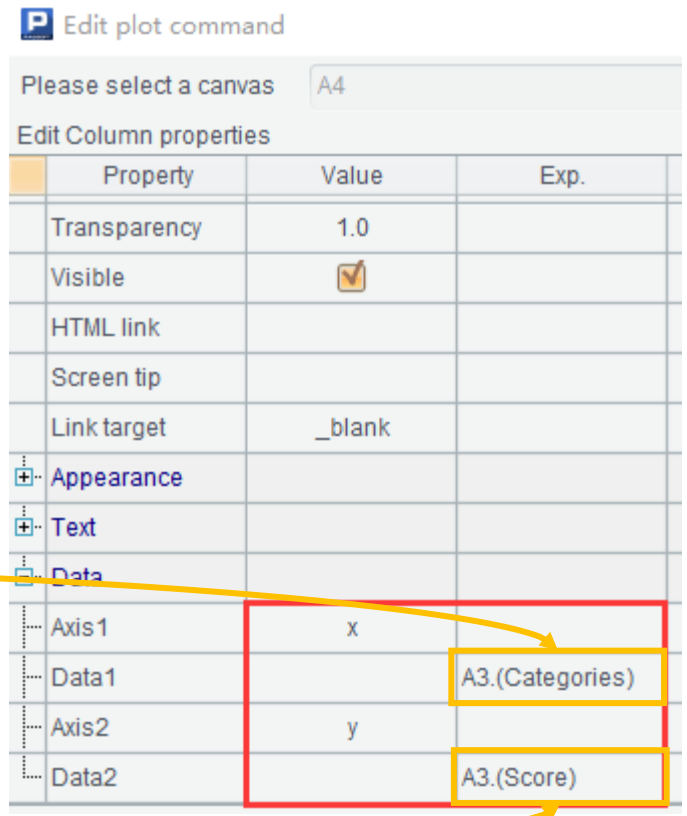
15.3.2 Enumeration axis

When the data of the element corresponding to the enumeration axis contains two layers, it is necessary to separate the category and series with a comma. Such as "Rebecca,Math".

Before drawing, convert the data into the following structure:

NAME	Subject	Score	Categories
Rebecca	Math	47	Rebecca,Math
Rebecca	Physics	87	Rebecca,Physics
Rebecca	Chemistry	72	Rebecca,Chemistry
Ashley	Math	80	Ashley,Math
Ashley	Physics	90	Ashley,Physics
Ashley	Chemistry	84	Ashley,Chemistry
Rachel	Math	90	Rachel,Math
Rachel	Physics	88	Rachel,Physics
Rachel	Chemistry	55	Rachel,Chemistry
Emily	Math	88	Emily,Math
Emily	Physics	20	Emily,Physics
Emily	Chemistry	77	Emily,Chemistry

Then edit the data properties of the **column element** and the corresponding data sequence diagram as shown in the following figure:



15.3.2 Enumeration axis

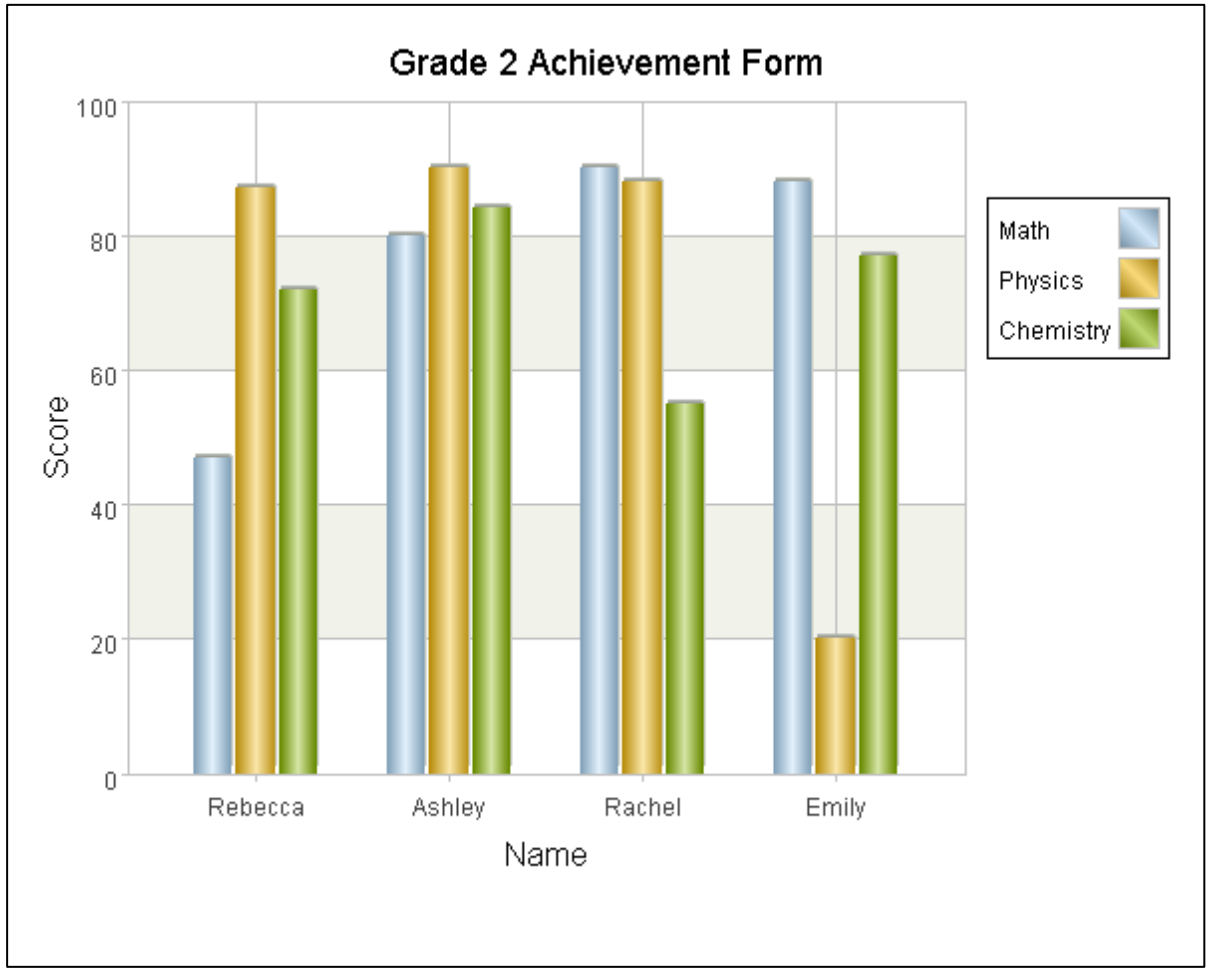
The sample script is as follows:

	A	B
1	=file("scores.txt").import@t()	/Import score table
2	=A1.pivot@r(ID,NAME;Subject,Score)	/Transpose the score tale
3	=A2.derive(NAME+","+Subject:Categories)	/Add Categories field
4	=canvas()	/Create canvas
5	=A4.plot("EnumAxis","name":"x","categories":A1.(NAME),"series":["Math","Physics","Chemistry"],"title":"Name")	/Enumeration type horizontal axis, named X axis, set axis title
6	=A4.plot("NumericAxis","name":"y","location":2,"title":"Score","titleAngle":-90)	/Numeric vertical axis, named Y axis, set the axis title and rotation angle
7	=A4.plot("Column","axis1":"x","data1":A3.(Categories),"axis2":"y","data2":A3.(Score))	/Plot column element in canvas, set data and associate coordinate system
8	=A4.plot("Legend")	/Plot default legend
9	=A4.plot("Text","text":"Grade 2 Achievement Form","y":0.05,"textSize":18)	/Plot statistical chart title with text
10	=A4.plot("BackGround")	/Set default white background
11	=A4.draw@p(600,480)	/Draw the figure

The default plotting of legend is used here. Please refer to the legend in [15.4.3] for detailed usage.

15.3.2 Enumeration axis

After execution, check the **multi series column chart** in A11:



15.3.3 Date axis

The logical axis whose coordinate value is date type is called **date axis**. Its value is similar to the numeric axis and is continuous.

【Example 15.10】Draw the project schedule with Gantt chart (project.csv).

The data is in CSV format, as follows:

```
Task Description,Start,Duration,End
Brainstorm,7/16/2013,5,7/21/2013
Outline,7/21/2013,10,7/31/2013
First Draft,7/25/2013,7,8/1/2013
First Draft Edit,8/1/2013,3,8/4/2013
Final Draft,8/4/2013,7,8/11/2013
```

- The horizontal axis of the Gantt chart uses the date axis.
- Use the **column element** to describe the plan of the project, and use the **data3** property to indicate the beginning of the column.
- When the data imported from text contains date, pay attention to set the **date format** in the option to be consistent with the data, otherwise the date type data will not be recognized.

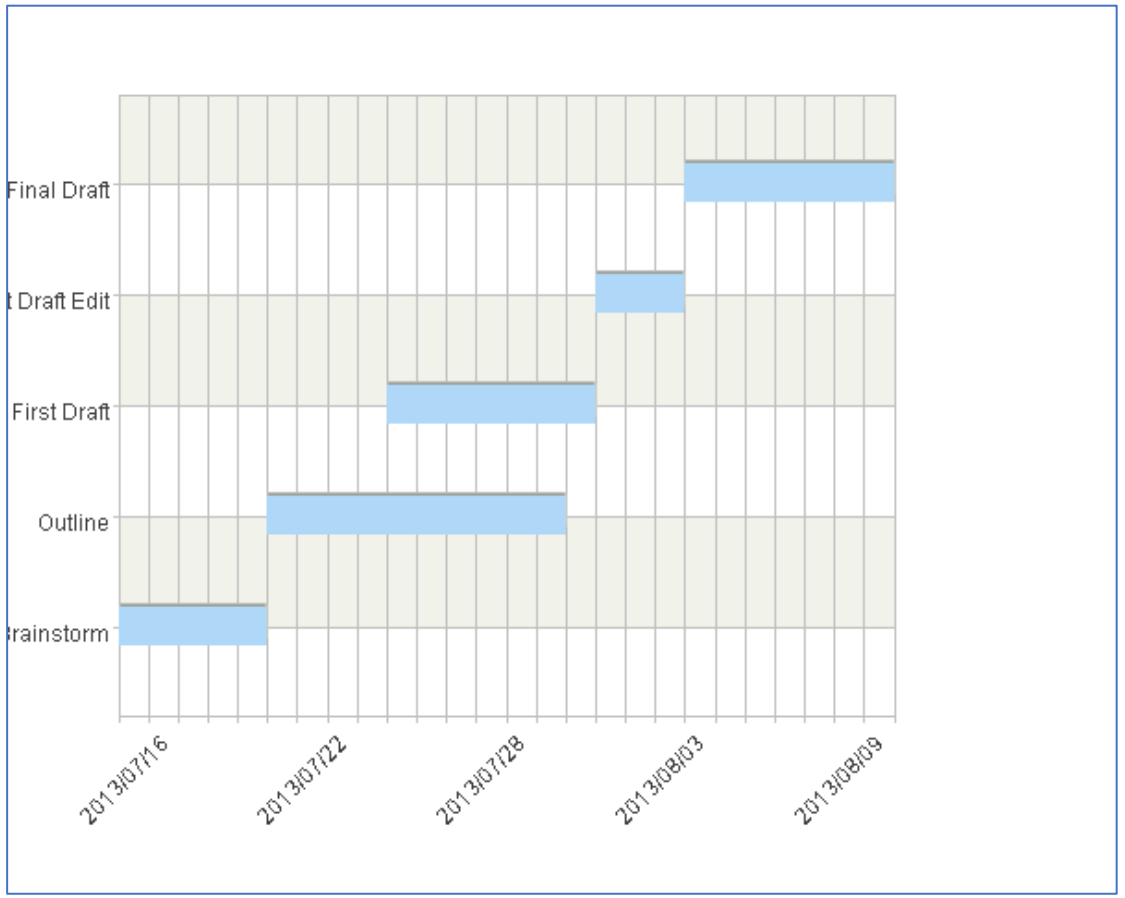
15.3.3 Date axis

The sample script is as follows:

	A	B
1	=file("project.csv").import@t(;,",")	/1. Import CSV data
2	=canvas()	/2. Create canvas
3	=A2.plot("DateAxis","name":"x","labelStep":2,"labelAngle":-45)	/3. Date type horizontal axis X
4	=A2.plot("EnumAxis","name":"y","location":2)	/4. Enumeration type vertical axis Y
5	=A2.plot("Column","data3":A1.(End),"fillColor":["ChartColor",0,false,-5252872,-5252872,0],"axis1":"y","data1":A1.(#1),"axis2":"x","data2":A1.(#2))	/5. Plot column element in canvas, set data and associate coordinate system
6	=A2.plot("Background")	/6. Set default white background
7	=A2.draw@p(600,480)	/7. Draw the figure

15.3.3 Date axis

After execution, view the **Gantt chart** in A7:



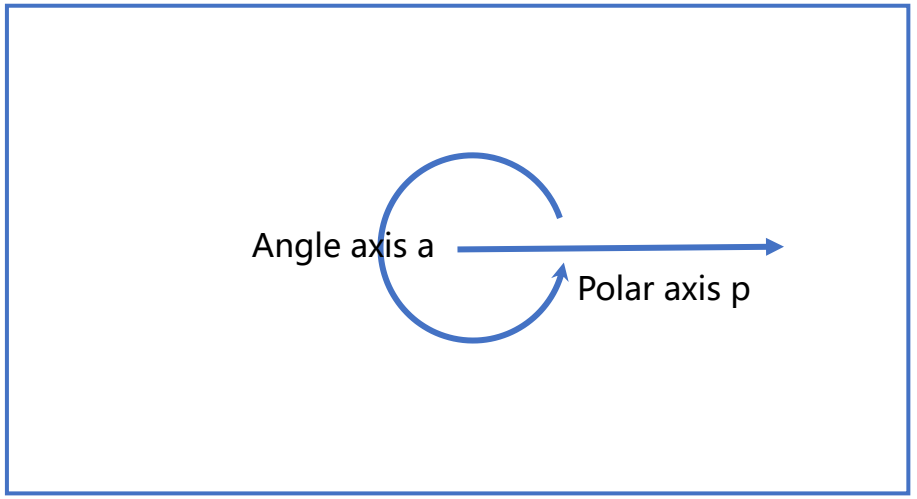
Chapter 15
Graphics drawing

15.4 More logical mappings

15.4.1 Polar coordinates

The **polar coordinate system** also consists of two axes. According to the location of the axis, the polar coordinate system is composed of **polar axis** and **angle axis**.

As shown in the right figure, the polar coordinate system can be formed by setting the locations of the axes as polar axis and angle axis respectively:



Edit Enumeration axis properties

Property	Value	Exp.
Name	p	
Visible	☑	
Location	Polar	
3D	☐	
3D axis's thickn...	0.38	

Edit Numeric axis properties

Property	Value	Exp.
Name	a	
Visible	☑	
Location	Angle	
3D	☐	
3D axis's thickn...	0.38	

15.4.1 Polar coordinates

Radar chart is actually the deformation of line chart in polar coordinate system. When the enumeration axis type of the line chart is set as angle axis, the numerical axis type is set as polar axis. Then set the **end to end** property of the line element, and the displayed graph is the radar chart.

【Example 15.11】Construct a polar coordinate system draw a radar map with data in the score table (summary.txt).

The property settings are as follows:

Edit Enumeration axis properties

Property	Value	Exp.
Name	a	
Visible	☑	
Location	Angle	
3D	☐	

Edit Numeric axis properties

Property	Value	Exp.
Name	p	
Visible	☑	
Location	Polar	
3D	☐	

Line element properties:

Edit Line properties

Property	Value	Exp.
End to end	☑	
Closed area	☑	
Area color		
Stacking type	Not stacked	
Star Line	☐	
Step Line	☐	
Transparency	0.6	
Allow text overla...	☑	
Shadow	☐	
Visible	☑	

Edit Line properties

Property	Value	Exp.
Data		
Axis1	a	
Data1		A1.(NAME)
Axis2	p	
Data2		A1.(TOTAL)

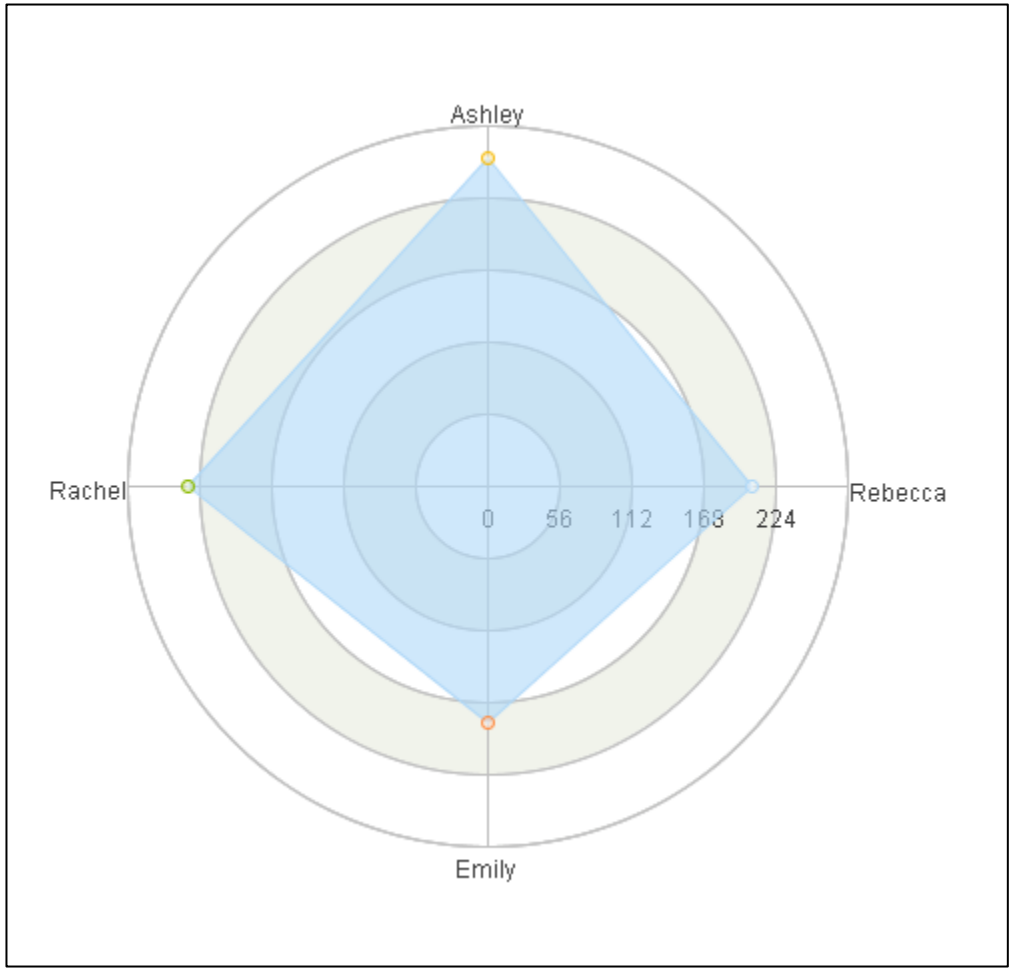
15.4.1 Polar coordinates

The script after setting the properties is as follows:

	A	B
1	=file("summary.txt").import@t()	/Import score table
2	=canvas()	/Create canvas
3	=A2.plot("EnumAxis","name":"a","location":4)	/Enumeration type angle axis, named a axis
4	=A2.plot("NumericAxis","name":"p","location":3)	/Numeric type polar axis, named p axis
5	=A2.plot("Line","endToHead":true,"closedArea":true,"transparent":0.6,"axis1":"a","data1":A1.(NAME),"axis2":"p","data2":A1.(TOTAL))	/Plot line element in the canvas, set data and associate coordinate system
6	=A2.draw(600,480)	/Draw the figure

15.4.1 Polar coordinates

After execution, check the **radar chart** in A6:



15.4.1 Polar coordinates

【Example 15.12】Construct a polar coordinate system draw a pie chart with sector using data in the score table (summary.txt).

The drawing process is also simple, as long as construct the polar coordinate system and then the plot the sector elements.

The data properties of **sector element** are as follows:

Edit Sector properties

Property	Value	Exp.
Text color		
Data		
Axis1	p	
Data1		A1.(NAME)
Axis2	a	
Data2		A1.(TOTAL)

Note:

The polar axis of a **pie chart** can only be an enumeration axis.

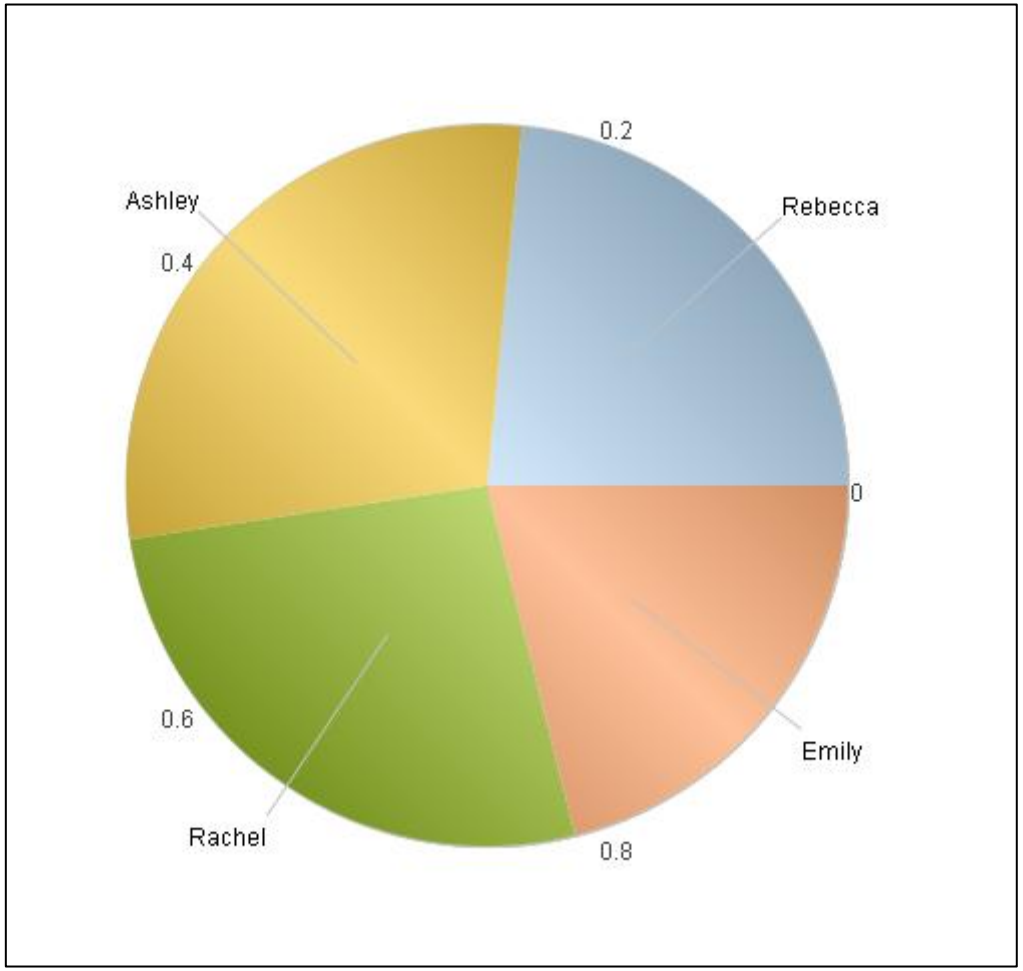
15.4.1 Polar coordinates

The sample script is as follows:

	A	B
1	=file("summary.txt").import@t()	/Import score table
2	=canvas()	/Create canvas
3	=A2.plot("EnumAxis","name":"p","location":3)	/Enumeration type polar axis, named p axis
4	=A2.plot("NumericAxis","name":"a","location":4)	/Numeric type angle axis, named a axis
5	=A2.plot("Sector","axis1":"p","data1":A1.(NAME),"axis2":"a","data2":A1.(TOTAL))	/Plot sector element in the canvas, set data and associate coordinate system
6	=A2.draw(600,480)	/Draw the figure

15.4.1 Polar coordinates

After execution, view the **pie chart** in A6

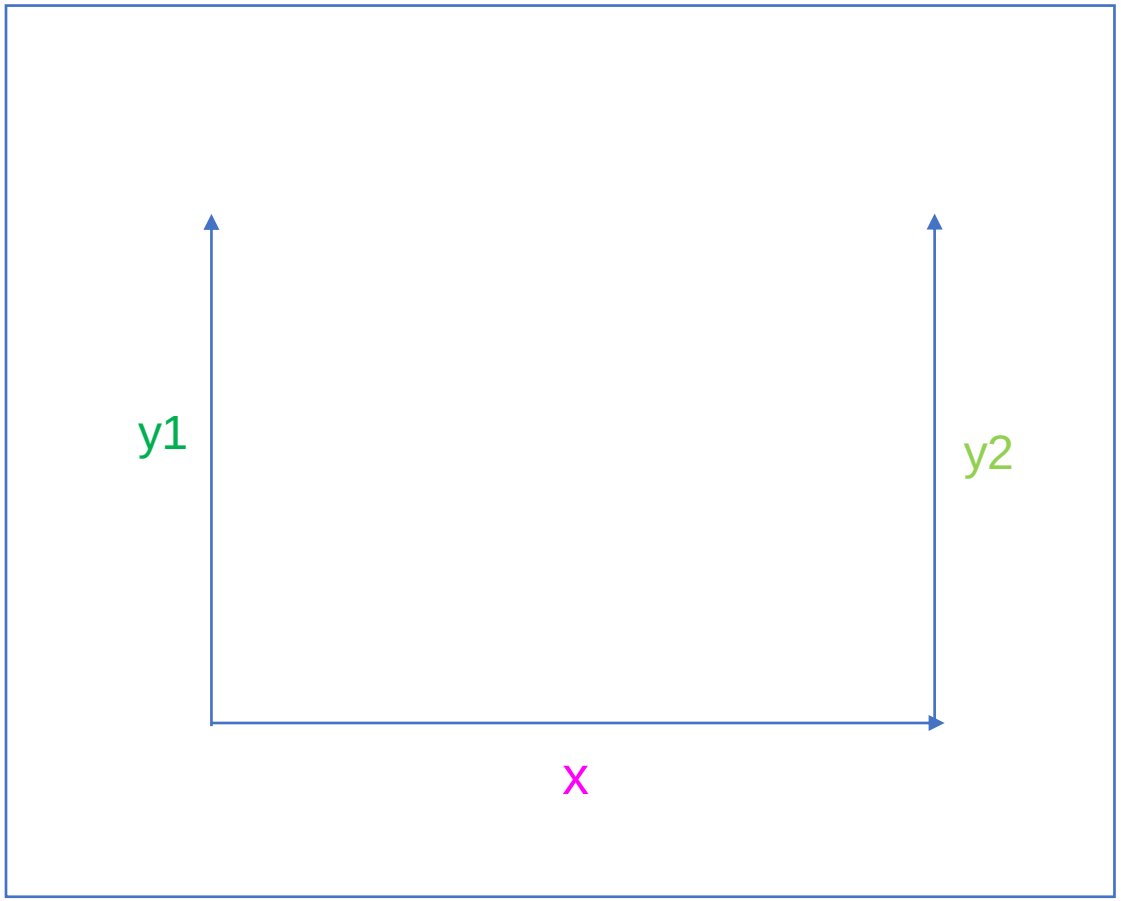


15.4.2 Combined coordinate system

All the previous examples only use the coordinate system of the combination of two axes. In order to draw composite elements, a **combined coordinate system** containing common axis can be constructed.

The combined coordinate system can draw the **dual-axis graph** conveniently.

For example, after plotting the coordinate axes in the figure on the right, the dual-axis graph containing different elements can be drawn under the coordinate system xy1 and the coordinate system xy2 respectively.



15.4.2 Combined coordinate system

When setting the coordinates of the axis, the corresponding physical coordinate properties of the axis should be set according to the location of the axis.

The corresponding relationship between location and coordinate properties is shown in the figure:

Horizontal axis

Vertical axis

Polar axis

Angle axis



15.4.2 Combined coordinate system

【Example 15.13】Draw the score table into a column chart, and then draw the summary table into a line chart with the right axis to generate a dual-axis column line chart (summary.txt and scores.txt).

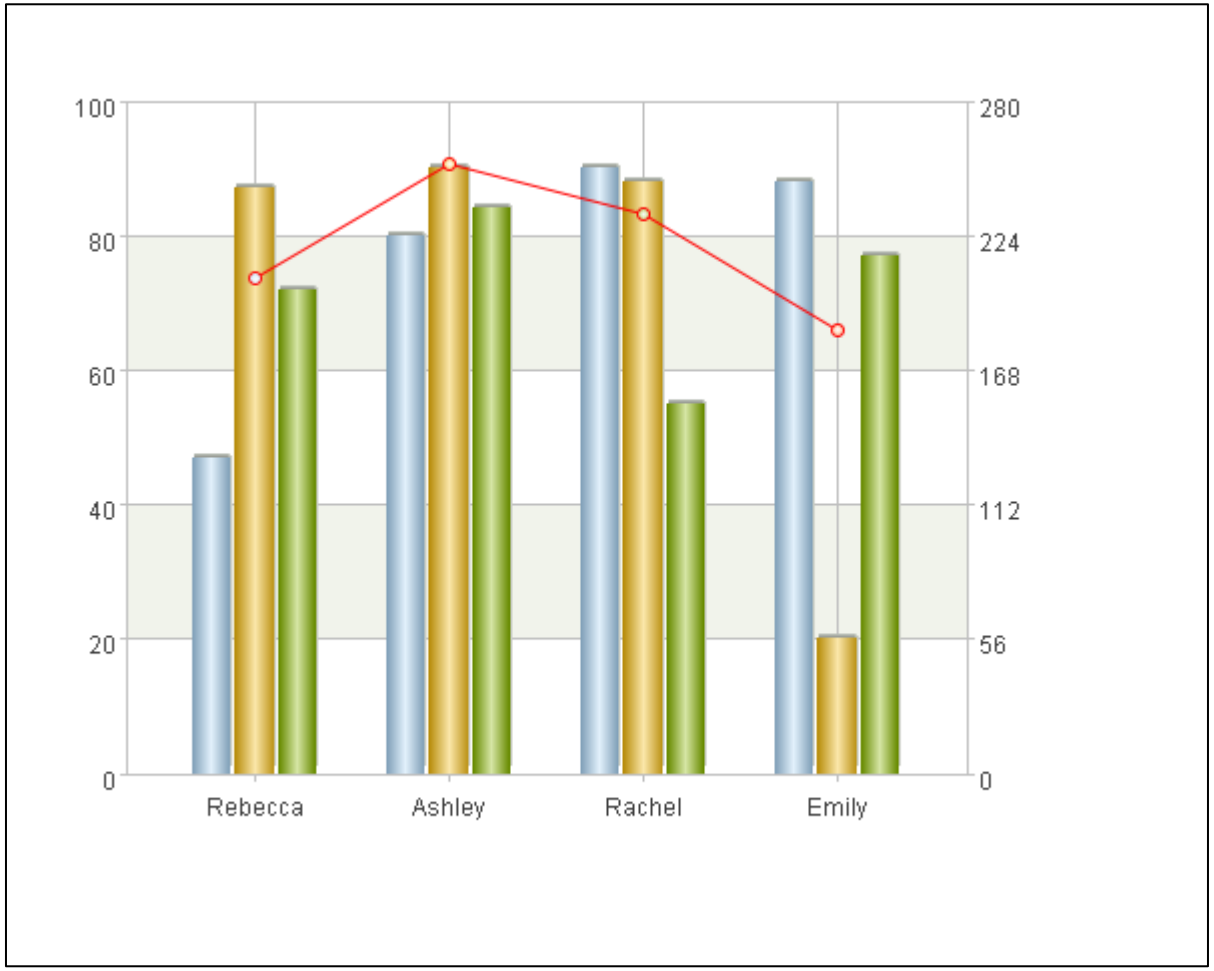
According to the schematic diagram of the axes introduced earlier, set up three coordinate axes, and then draw the graphs with different elements using their respective coordinate system.

The settings of properties are no longer described. The sample script is shown in the right table:

	A	B
1	=file("summary.txt").import@t()	/Import summary table
2	=file("scores.txt").import@t()	/Import score table
3	=A2.pivot@r(ID,NAME;Subject,Score)	/Transpose the score table
4	=A3.(NAME+","+Subject)	/Concatenate the categories and series
5	=canvas()	/Create canvas
6	=A5.plot("EnumAxis","name":"x")	/Plot enumeration axis x
7	=A5.plot("NumericAxis","name":"y1","location":2)	/Plot numeric vertical axis y1
8	=A5.plot("Column","axis1":"x","data1":A4,"axis2":"y1","data2":A3.(Score))	/Plot column graph under coordinate system x-y1
9	=A5.plot("NumericAxis","name":"y2","location":2,"yPosition":0.8,"scalePosition":0)	/Plot the right vertical axis y2
10	=A5.plot("Line","lineColor":-65536,"axis1":"x","data1":A1.(NAME),"axis2":"y2","data2":A1.(TOTAL))	/Plot line chart under coordinate system x-y2
11	=A5.draw(600,480)	/Draw dual-axis graph

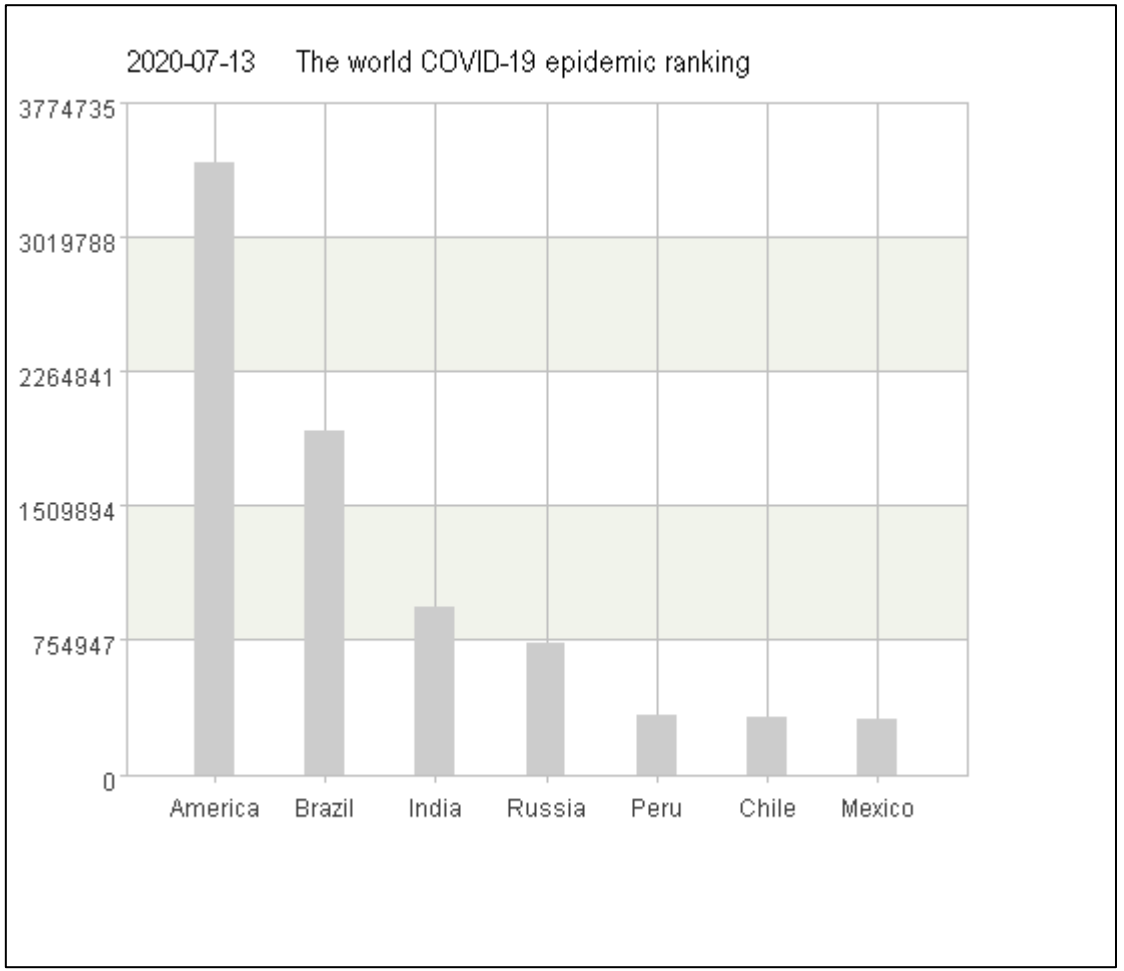
15.4.2 Combined coordinate system

After execution, check the **dual-axis column line graph** in A11



15.4.3 Legend

Looking at the world epidemic map on the right, each column is determined by a pair of logical coordinates (country, cumulative number of cases). However, if you want to represent one more logical quantity, such as the severity based on the number of cases, because there are only two coordinates in the coordinate system, color or shape should be used to describe it.



To describe business information with graphic attributes such as color or shape, it is necessary to establish the corresponding relationship between business information and graphic attributes, which is called **legend**.

15.4.3 Legend

【Example 15.14】Draw the top seven epidemic cases in the world on July 13, 2020 into a column chart, and mark the severity with different colors according to the interval of accumulated cases (covid.txt).

The specific data extracted from the Internet is as follows:

Country	Accumulation	Level
America	3431574	>1 Million
Brazil	1926824	>1 Million
India	936181	500 Thousand~1 Million
Russia	738787	500 Thousand~1 Million
Peru	333867	<500 Thousand
Chile	319493	<500 Thousand
Mexico	311486	<500 Thousand

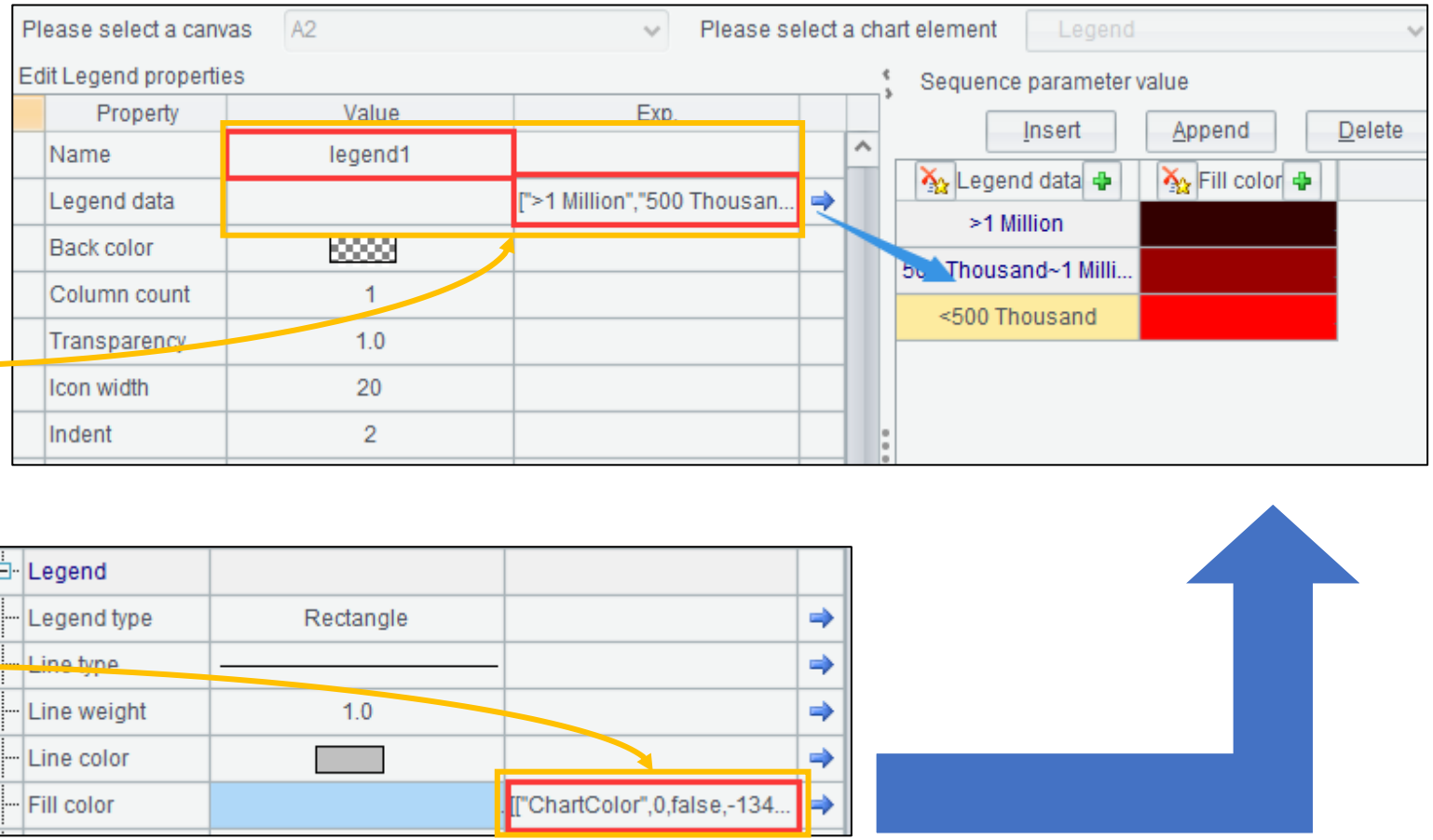
Based on the current data, the enumeration axis is needed to represent the country.
The numeric axis is used to represent the accumulation.
Then use the **legend** to draw different colors to represent the level.

15.4.3 Legend

Edit the legend, and set the legend name, legend data and corresponding color settings as shown in the figure:

Set the legend name legend1 and legend data

Use the sequence editing module to set the colors corresponding to the intervals of cumulative number of cases



15.4.3 Legend

Edit the **column element**, directly set the fill color of column element to logical coordinate level, and specify legend1. Legend1 will convert the logical value to the actual drawing color according to the legend settings.

For other related properties and the corresponding data, please refer to the schematic diagram:

Country	Accumulation	Level
America	3431574	>1 Million
Brazil	1926824	>1 Million
India	936181	500 Thousand~1 Milli..
Russia	738787	500 Thousand~1 Milli..
Peru	333867	<500 Thousand
Chile	319493	<500 Thousand
Mexico	311486	<500 Thousand

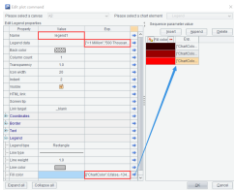
Edit Column properties

Property	Value	Exp.	Legend
Border weight	0		
Border color			
Fill color		A1.(Level)	legend1
Text			
Horizontal align...	Center		
Vertical alignm...	Top		
Text			
Font			
Text style	Example		
Text size	12		
Text color			
Data			
Axis1	x		
Data1		A1.(Country)	
Axis2	y		
Data2		A1.(Accumulation)	

Conversion via legend1

Logical color value

<500 Thousand



Fill color value

["ChartColor", 0, false, -65536, -5252872, 0]

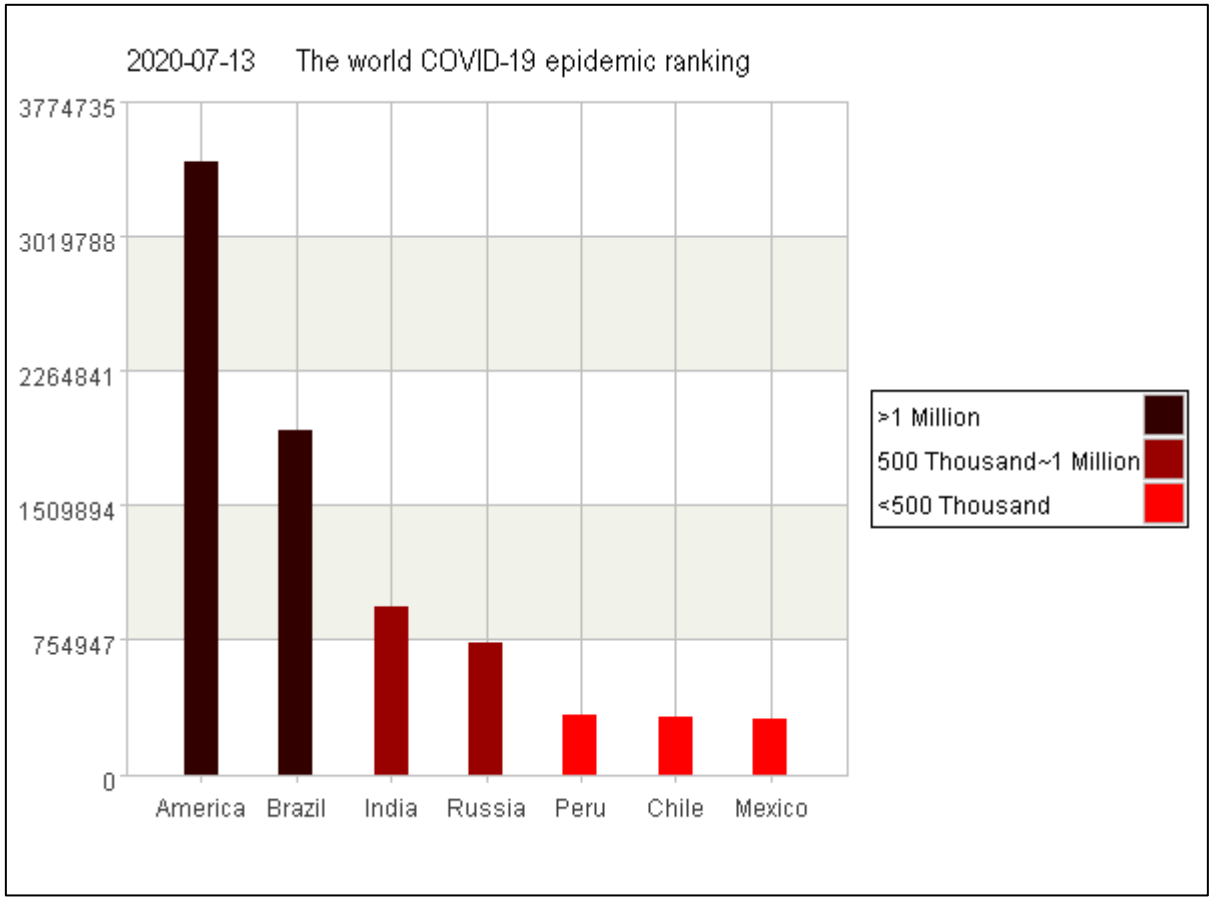
15.4.3 Legend

The sample script is as follows:

	A	B
1	=file("covid.txt").import@t()	/1.Import data
2	=canvas()	/2. Create canvas
3	=A2.plot("Legend","name":"legend1","legendText":[">1 Million","500 Thousand~1 Million","<500 Thousand"],"edgeIndent":2,"x":0.72,"y":0.4,"legendFillColor":[["ChartColor",0,false,-13434880,-5252872,0],["ChartColor",0,false,-6750208,-5252872,0],["ChartColor",0,false,-65536,-5252872,0]])	/3. Set the legend name, value and other related properties
4	=A2.plot("EnumAxis","name":"x","xEnd":0.7)	/4. Enumeration horizontal axis X
5	=A2.plot("NumericAxis","name":"y","location":2)	/5. Numeric vertical axis Y
6	=A2.plot("Column","shadow":false,"fillColor":A1.(Level):"legend1","axis1":"x","data1":A1.(Country),"axis2":"y","data2":A1.(Accumulation))	/6. Plot column element
7	=A2.plot("Text","text":"2020-07-13 The world COVID-19 epidemic ranking","x":0.1,"y":0.03,"textFont":"Arial Narrow","textSize":16,"hAlign":0,"vAlign":8)	/7. Set figure title
8	=A2.plot("BackGround")	/Default white background
9	=A2.draw@p(600,480)	/Draw the figure

15.4.3 Legend

After execution, check the **world epidemic column chart** in A9:



15.4.3 Legend

There is no problem when using the color legend to view on the computer or print it out in color. But if printed in black and white, the color becomes gray, which is not easy to distinguish. In this case, you can use the dot graph to set the legend through the shape of the dots.

【Example 15.15】 Based on the score table (scores.txt), draw a black and white dot graph according to different marker shapes.

Edit Legend properties

Property	Value	Exp.
Name	legend	
Legend data		["Math","Physics","Chemistry"]

Edit the **legend**, set the legend type to dot, and set the related transparent fill and black border.

The marker shape is a sequence of shapes corresponding to the legend name.

Text		
Font		
Text style	Example	
Text size	12	
Text color		
Legend		
Legend type	Dot	
Line type		
Line weight	1.0	
Line color		
Fill color		
Marker shape		[1,2,3]

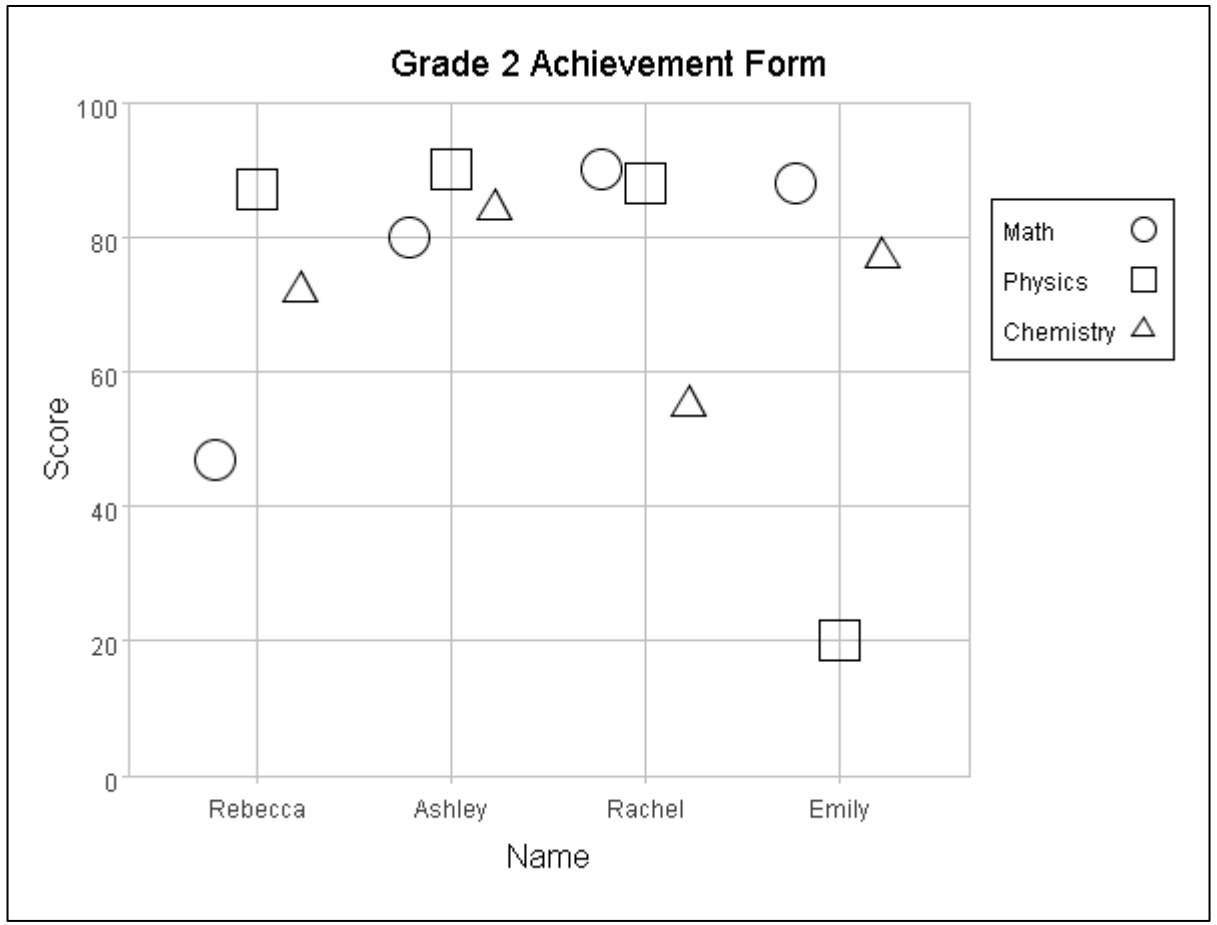
15.4.3 Legend

The sample script is as follows:

	A	B
1	=file("scores.txt").import@t()	/Import score table
2	=A1.pivot@r(ID,NAME;Subject,Score)	/Transpose the score table
3	=A2.(NAME+","+Subject)	/Concatenate the categories and series
4	=canvas()	/Create canvas
5	=A4.plot("EnumAxis","name":"x","title":"Name","regionColors":["ChartColor",0,false,16777215,-5252872,0])	/Enumeration horizontal axis, named X axis, set axis title
6	=A4.plot("NumericAxis","name":"y","location":2,"title":"Score","titleAngle":-90,"regionColors":["ChartColor",0,false,16777215,-5252872,0])	/Numeric vertical axis, named Y axis, set axis title and rotation angle
7	=A4.plot("Legend","name":"legend","legendText":["Math","Physics","Chemistry"],"legendType":2,"legendLineColor":-16777216,"legendFillColor":["ChartColor",0,false,16777215,-5252872,0],"legendMarkerShape":[1,2,3])	/Plot the legend in the canvas, set the legend data and fill color
8	=A4.plot("Dot","markerStyle":A2.(Subject):"legend","lineColor":-16777216,"markerColor":["ChartColor",0,false,16777215,-5252872,0],"markerWeight":10,"axis1":"x","data1":A3,"axis2":"y","data2":A2.(Score))	/Plot dot element, set logical dot shape, and associate legend
9	=A4.plot("Text","text":"Grade 2 Achievement Form","y":0.05,"textSize":18)	/Plot statistical chart title with text
10	=A4.plot("BackGround")	/Default white background
11	=A4.draw@p(600,480)	/Draw the figure

15.4.3 Legend

After execution, the **black and white dot graph** in A11 is obtained.



15.4.3 Legend

If the legend is drawn directly using the categories or series of the enumeration axis in the current coordinate system, and if the element and legend adopt the default color sequence of the palette, the legend is automatically simplified to the simplest case, and the legend data can be automatically obtained from the enumeration axis, and the legend only needs to edit one legend object.

【Example 15.16】Draw the summary table (dsummary.txt) into a column chart with legend.

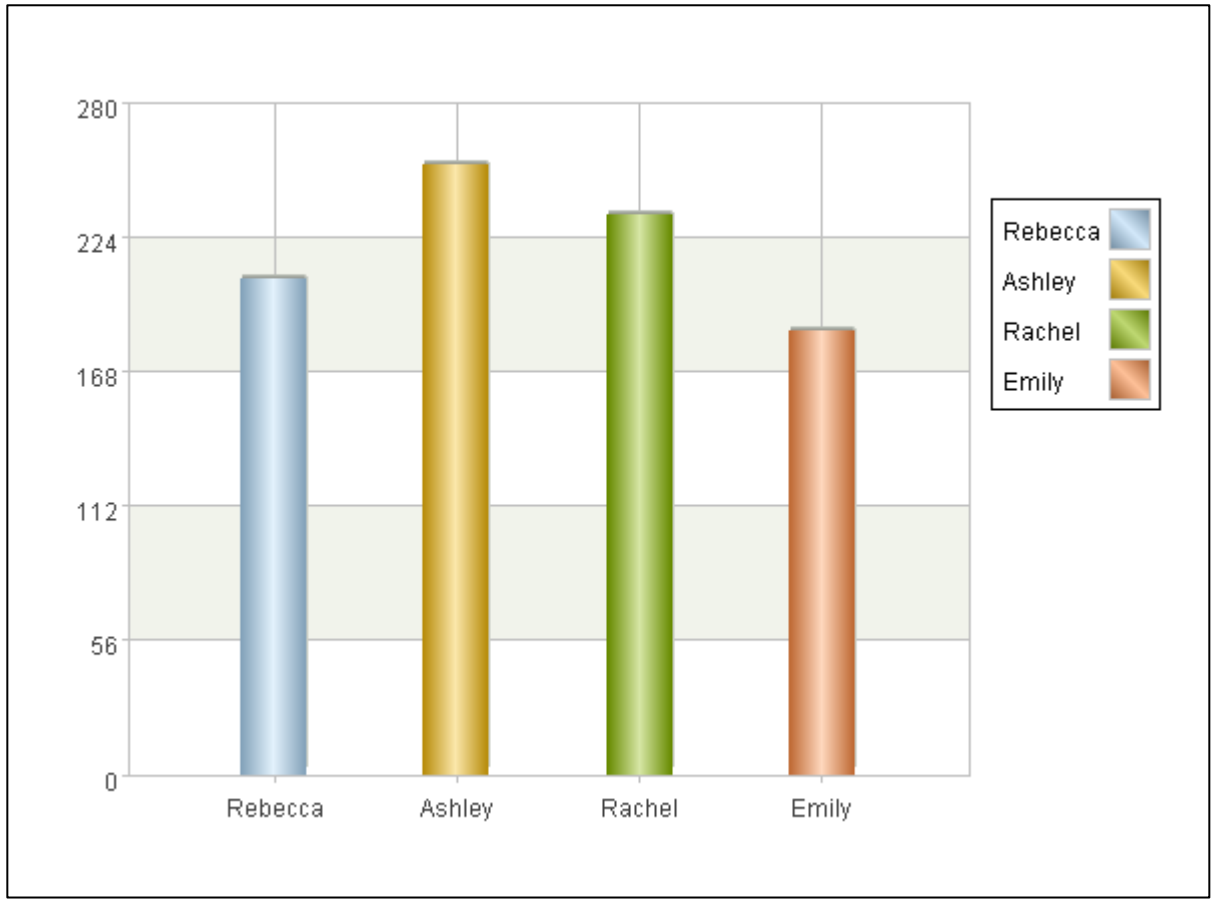
Because all default properties are used, editing is very simple. The example script edited is as follows:

	A	B
1	=file("summary.txt").import@t()	/1. Import data
2	=canvas()	/2. Create canvas
3	=A2.plot("EnumAxis","name":"x")	/3. Enumeration horizontal axis X
4	=A2.plot("NumericAxis","name":"y","location":2)	/4. Numeric vertical axis Y
5	=A2.plot("Column","axis1":"x","data1":A1.(NAME),"axis2":"y","data2":A1.(TOTAL))	/5. Plot column elements in the canvas, set the data and associate coordinate system, and use the default palette
6	=A2.plot("Legend")	/6. Plot the legend, using the default palette as well
7	=A2.draw(600,480)	/7. Draw the figure

You only need to edit one legend object. Because it is automatic matching, the name can be omitted.

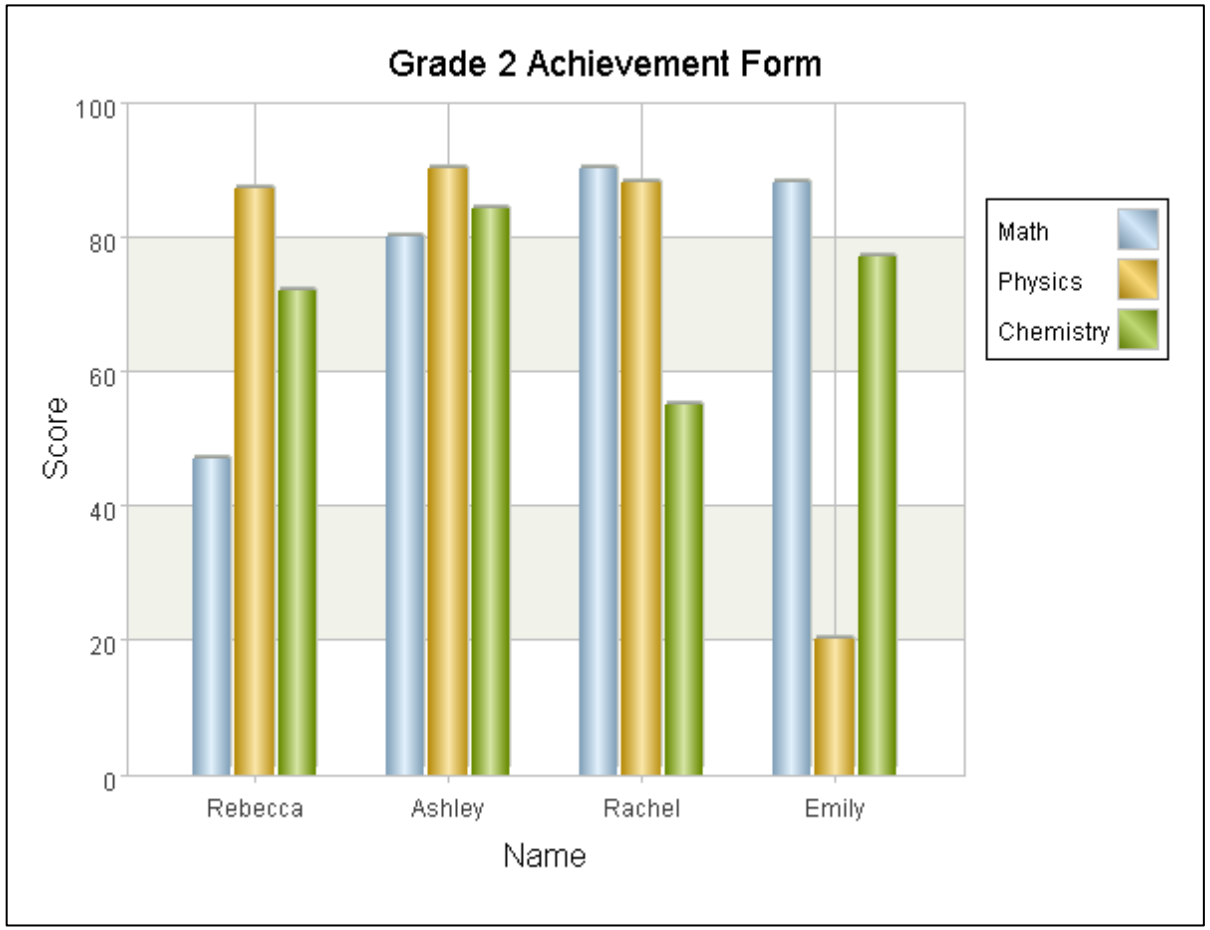
15.4.3 Legend

After execution, check the **column chart with legend** in A7:



15.4.3 Legend

Looking back at 【Example 15.9】, when the enumeration axis contains series, the legend will be drawn automatically according to the series:



Chapter 15
Graphics drawing

15.5 Finished graph

15.5 Finished graph

When drawing a statistical chart with elements, it is always necessary to set various graphic elements and coordinate system, so the drawing process is complicated. But sometimes it is only necessary to draw a single graph, and it is not necessary to draw a flexible and complex statistical chart.

The graphic elements required by several common statistical charts are integrated into a **graph**. The packaged single graph is called **finished graph**, and the finished graph is represented by graph in the following.

The drawing of a graph only needs to specify **categories** and **values**. The requirements of these two kinds of data are consistent with the data in the element, and they are also two sequences.

15.5.1 Column Graph

Column graph corresponds to column elements, and it can draw business data as column graph.

【Example 15.17】Draw the summary table (summary.txt) as column graph.

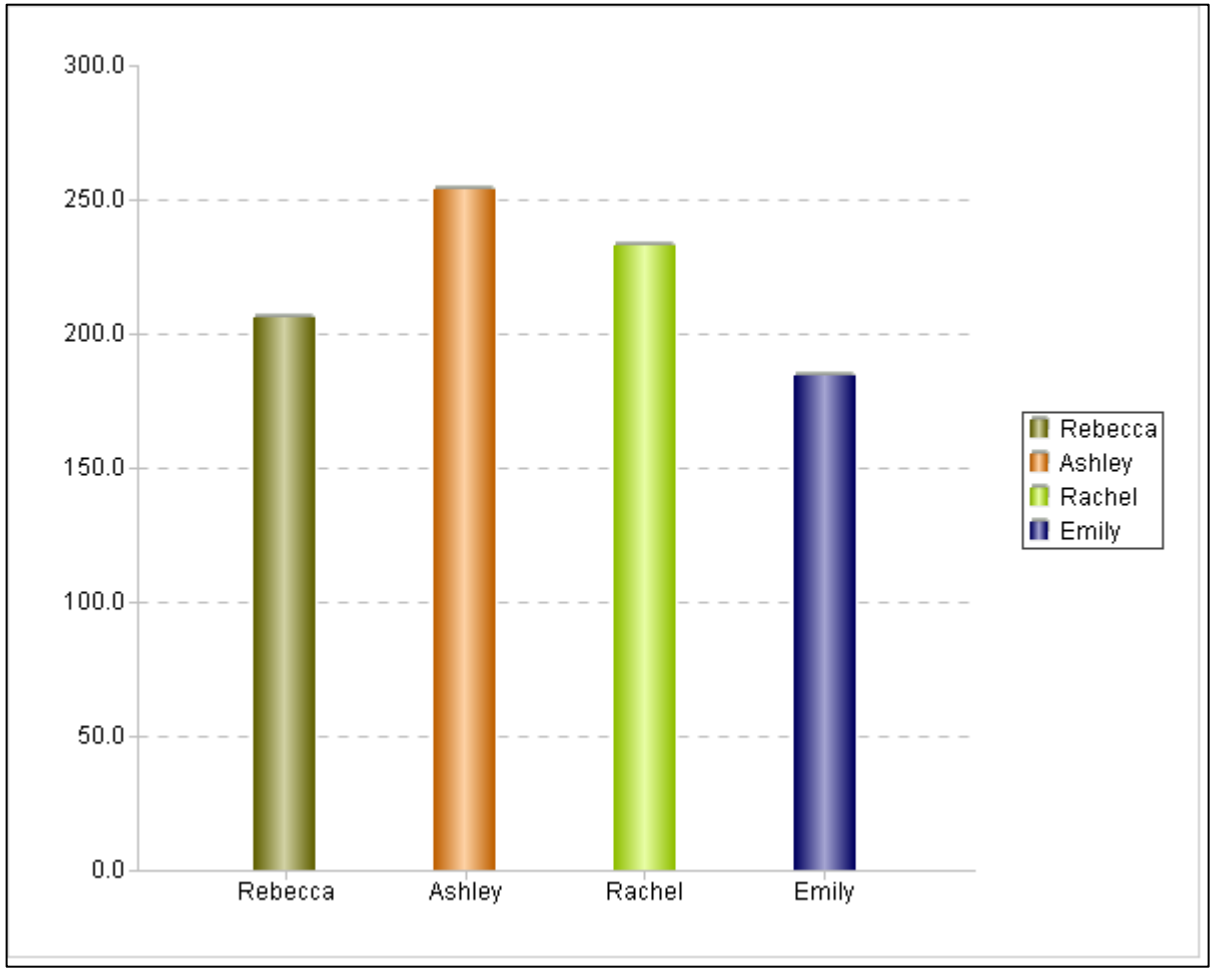
- Compared with the drawing of elements, the drawing of graph is more simple. Directly plotting the corresponding graph and specifying the business data is OK.

The example script after setting the properties is as follows:

	A	B
1	=file("summary.txt").import@t()	/1.Import data
2	=canvas()	/2. Create canvas
3	=A2.plot("GraphColumn","categories":A1.(NAME),"values":A1.(TOTAL))	/3.Plot Column Graph in the canvas and set data
4	=A2.draw(600,480)	/4. Draw the figure

15.5.1 Column Graph

After execution, check the **column graph** in A4:



15.5.2 Dual-axis Graph

Dual-axis graph is also widely used. When the numerical range of series in the same graph is greatly different, it is more intuitive to use dual-axis graph.

【Example 15.18】 Calculate the total score of each student in the score table(scores.txt), and then use the dual-axis graph to draw a graph with the score as a column and the total score as a line.

Dual-axis graph can only draw business data containing multiple series. Since there are two axes, it is necessary to specify specific series to the corresponding axis.

As shown in the right figure, the total score of the calculated columns is assigned to the right axis series, the remaining series are automatically classified as the left axis series.

Edit Dual-axis graph properties

Property	Value	Exp.
Categories		A4
Values		A3.(Score)
Type	2AxisColLine	
Left axis series		
Right axis series	Total	
Line properties		
Plot data points	☐	

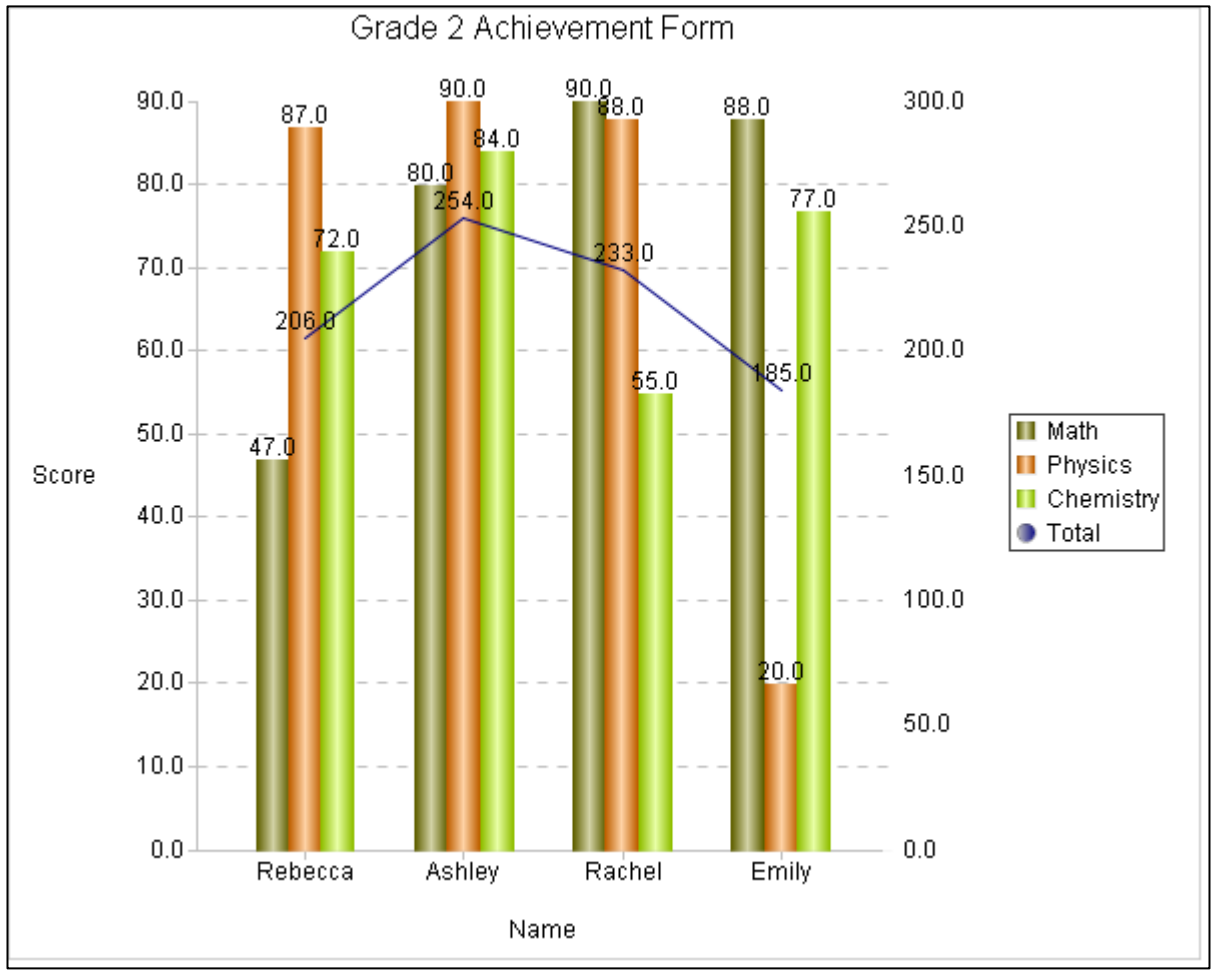
15.5.2 Dual-axis Graph

The sample script is as follows:

	A	B
1	=file("scores.txt").import@t()	/Import score table
2	=A1.derive(Math+Physics+Chemistry:Total)	/Add calculated column total
3	=A2.pivot@r(ID,NAME;Subject,Score)	/Transpose the score table
4	=A3.(NAME+","+Subject)	/Spell the categories and series
5	=canvas()	/Create canvas
6	=A5.plot("Graph2Axis","categories":A4,"values":A3.(Score),"rightSeries":"Total","xTitle":"Name","yTitle":"Score","graphTitle":"Grade 2 Achievement Form","displayData":2)	/Plot Dual-axis graph and set data and title
7	=A5.draw(600,480)	/Draw the figure

15.5.2 Dual-axis Graph

After execution, check the **dual-axis graph** in A7:



Chapter 15
Graphics drawing

15.6 Exercises

15.6.1 Exercises: Trigonometric function

【Example 15.1】Draw cosine function graph in rectangular coordinate system.

Solution:

To draw a cosine function graph, we only need to define a coordinate system where the horizontal and vertical axes are numeric axes. After calculating the cosine value in a certain radian, it can be displayed with a line graph.

Since the cosine value is between - 1 and 1, the "**Auto range starts from 0**" option of the default property is cancelled on the vertical axis.

Set the vertical axis properties as follows:

Edit Numeric axis properties

Property	Value	Exp.
Numeric axis		
Autocalc value range	☑	
AutoRange starts from 0	☐	
Max value	10.0	

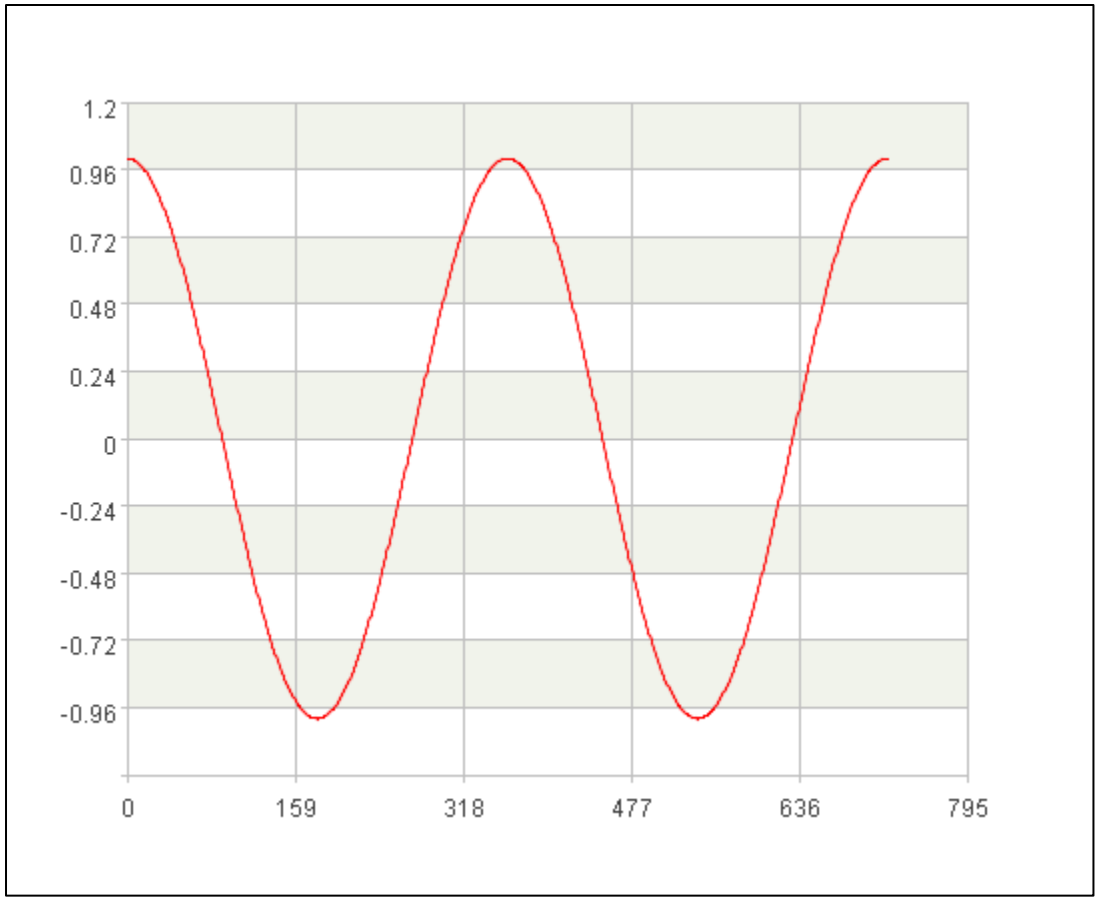
15.6.1 Exercises: Trigonometric function

The sample script is as follows:

	A	B	C
1	>rads=[],angs=[]	/Define the sequence to store data	
2	for 720	=cos(A2*pi()/180)	/Calculate the cosine of the angle in loop
3		>angs=angs|A2	/Construct angle sequence
4		>rads=rads|B2	/Construct cosine value sequence
5	=canvas()	/Create canvas	
6	=A5.plot("NumericAxis","name":"x")	/Plot numeric horizontal axis X	
7	=A5.plot("NumericAxis","name":"y","location":2,"autoRangeFromZero":false)	/Plot numeric vertical axis Y, cancel "auto range from zero"	
8	=A5.plot("Line","markerStyle":0,"lineColor":-65536,"axis1":"x","data1":angs,"axis2":"y","data2":rads)	/Plot Line element, set coordinate system and data value	
9	=A5.plot("BackGround")	/Plot default white background	
10	=A5.draw@p(600,480)	/Draw the figure	

15.6.1 Exercises: Trigonometric function

After execution, view the **cosine line chart** in A10:



15.6.2 Exercises:Multiple combinations

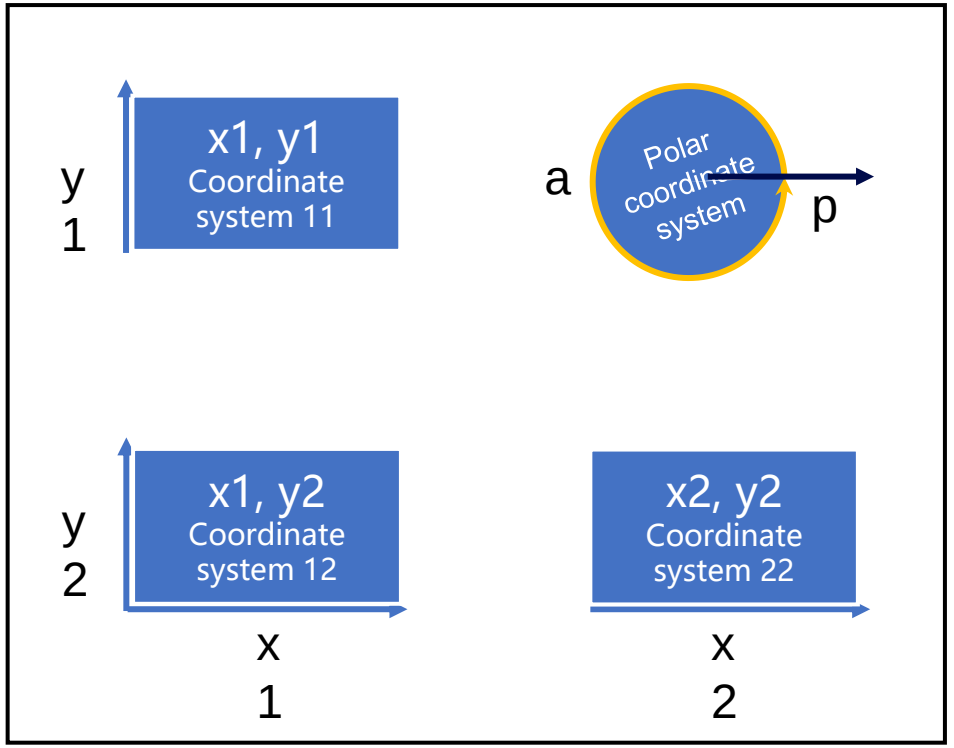
【Example 15.2】Based on the score table(scores.txt), draw a column chart and a stacked column chart in the same chart to facilitate the view of single subject score and total score ranking. Then draw the summary table(summary.txt) in the chart as pie chart and line chart.

Solution:

Through flexible use of the combination of multiple coordinate systems to draw the relevant elements, you can draw complex multi graphs.

Because two elements are drawn with the same score table, the common coordinate system in the combined graphs can be used.

The layout of the coordinate system is shown in the figure on the right:



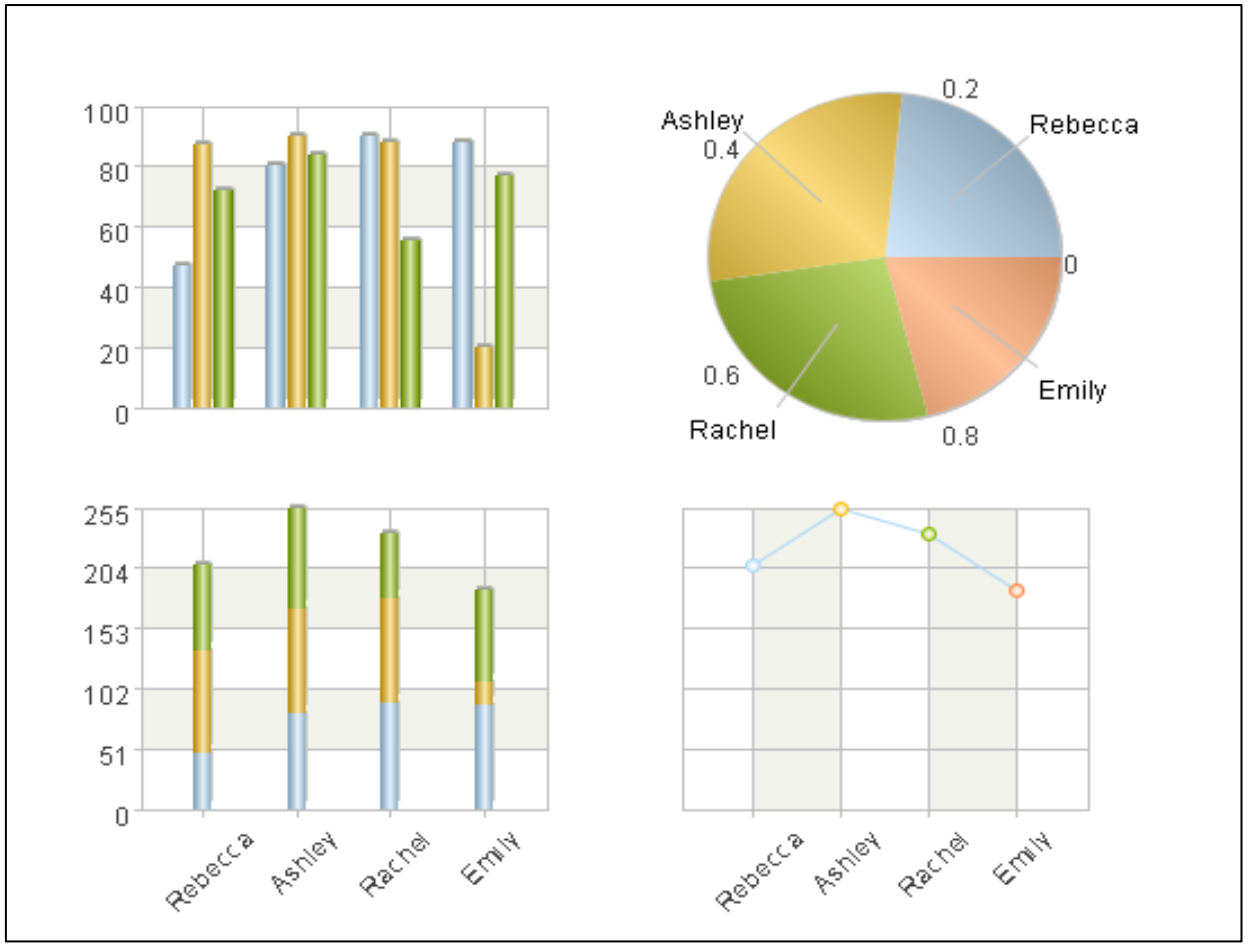
15.6.2 Exercises:Multiple combinations

The example script after editing the corresponding element properties is as follows:

	A	B	C	D
1	=file("summary.txt").import@t()	/Import summary table		
2	=file("scores.txt").import@t()	/Import score table		
3	=A2.pivot@r(ID,NAME;Subject,Score)	/Transpose the score table		
4	=A3.(NAME+","+Subject)	/Spell the categories and series		
5	=canvas()	/Create canvas		
6	=A5.plot("EnumAxis","name":"x1","xEnd":0.4,"labelAngle":-45,"labelOverlapping":true)	/Plot enumeration horizontal axis x1	=A5.plot("EnumAxis","name":"p","location":3,"xEnd":0.4,"polarX":0.65,"polarY":0.25,"polarLength":0.13,"labelAngle":-45,"labelOverlapping":true)	/Plot polar axis p
7	=A5.plot("NumericAxis","name":"y1","location":2,"yStart":0.4,"yPosition":0.1)	/Plot numeric vertical axis y1	=A5.plot("NumericAxis","name":"a","location":4)	/Plot angle axis a
8	=A5.plot("Column","axis1":"x1","data1":A4,"axis2":"y1","data2":A3.(Score))	/Plot a column chart in coordinate system 11	=A5.plot("Sector","axis1":"p","data1":A1.(NAME),"axis2":"a","data2":A1.(TOTAL))	/Plot a pie chart in polar coordinates
9	=A5.plot("NumericAxis","name":"y2","location":2,"yStart":0.8,"yEnd":0.5)	/Plot the vertical axis y2 and the common axis x1 to construct the coordinate system 12	=A5.plot("EnumAxis","name":"x2","xStart":0.5,"labelAngle":-45,"labelOverlapping":true)	/Plot horizontal axis x2
10	=A5.plot("Column","stackType":2,"axis1":"x1","data1":A4,"axis2":"y2","data2":A3.(Score))	/Plot a stacked column chart in coordinate system 12	=A5.plot("Line","axis1":"x2","data1":A1.(NAME),"axis2":"y2","data2":A1.(TOTAL))	/Plot a line chart in coordinate system 22
11	=A5.draw(600,480)	/Draw the figure		

15.6.2 Exercises:Multiple combinations

After execution, view the **combined multiple graphs** in A11:



15.6.3 Exercises:K-line chart

【Exercise 15.3】Draw a K-line chart of the stock market within a month.

Part of the data extracted from the Internet is as follows:

Date	Close	Vo...	Open	High	Low
07/13/2020	10390.84	0	10729.92	10824.78	10368.04
07/10/2020	10617.44	0	10545.91	10622.35	10447.01
07/09/2020	10547.75	0	10563.72	10578.1	10379.91
07/08/2020	10492.5	0	10409.35	10494.63	10350.96
07/07/2020	10343.89	0	10412.46	10518.98	10337.98
07/06/2020	10433.65	0	10360.38	10462.05	10354.98
07/02/2020	10207.63	0	10268.67	10310.36	10194.06
07/01/2020	10154.63	0	10063.67	10197.19	10048.04
06/30/2020	10058.77	0	9875.29	10085.59	9863.67
06/29/2020	9874.15	0	9771.72	9877.34	9663.61
06/26/2020	9757.22	0	9995.12	10000.67	9749.07
06/25/2020	10017.0	0	9899.36	10023.28	9810.47
06/24/2020	9909.17	0	10092.92	10137.5	9842.22
06/23/2020	10131.37	0	10130.83	10221.85	10112.44
06/22/2020	10056.47	0	9945.49	10059.61	9916.6
06/19/2020	9946.12	0	10042.13	10053.91	9872.94
06/18/2020	9943.05	0	9892.48	9959.2	9885.66

Solution:

The horizontal axis of the K-line chart is the date, so the date axis is used as the horizontal axis, and then there are four data every day. The opening and closing prices are in a group, represented by a rectangle; the lowest and the highest prices are in a group, and are represented by a line (i.e., rectangle with very narrow width). Column element has data3 property, which can be used to indicate the starting position of column to meet the requirements.

Construct a rectangular coordinate system, and plot two column elements with different widths and using data3 property respectively. The color of the columns are set using a custom legend.

15.6.3 Exercises:K-line chart

	A	B
1	=file("NASDAQ.csv").import@tc()	/Import data
2	=A1.select(month(Date)==6)	/Select data of June
3	=A2.derive(Open>Close:Status)	/Calculate the day's rise or fall
4	=canvas()	/Create canvas
5	=A4.plot("Legend","name":"legend","legendText":[true,false],"visible":false,"legendFillColor":[["ChartColor",0,false,-3407872,-5252872,0],["ChartColor",0,false,-16737997,-5252872,0]])	/Define a hidden legend for column color mapping
6	=A4.plot("DateAxis","name":"d","format":"MM/dd/yyyy","labelStep":3,"labelAngle":-45,"allowRegions":false)	/Date type horizontal axis d
7	=A4.plot("NumericAxis","name":"y","location":2,"autoRangeFromZero":false)	/Numeric vertical axis y
8	=A4.plot("Column","shadow":false,"data3":A3.(Open),"columnWidth":8,"fillColor":A3.(Status):"legend","axis1":"d","data1":A3.(Date),"axis2":"y","data2":A3.(Close))	/Column element, a rectangle with a width of 8 pixels(open and close)
9	=A4.plot("Column","shadow":false,"data3":A3.(Low),"columnWidth":2,"fillColor":A3.(Status):"legend","axis1":"d","data1":A3.(Date),"axis2":"y","data2":A3.(High))	/Column element, a rectangle with a width of 2 pixels(high and low)
10	=A4.plot("Text","text":"NASDAQ Composite","y":0.05,"textFont":"Arial Black")	=A4.plot("BackGround")
11	=A5.draw@p(600,480)	/Draw the figure

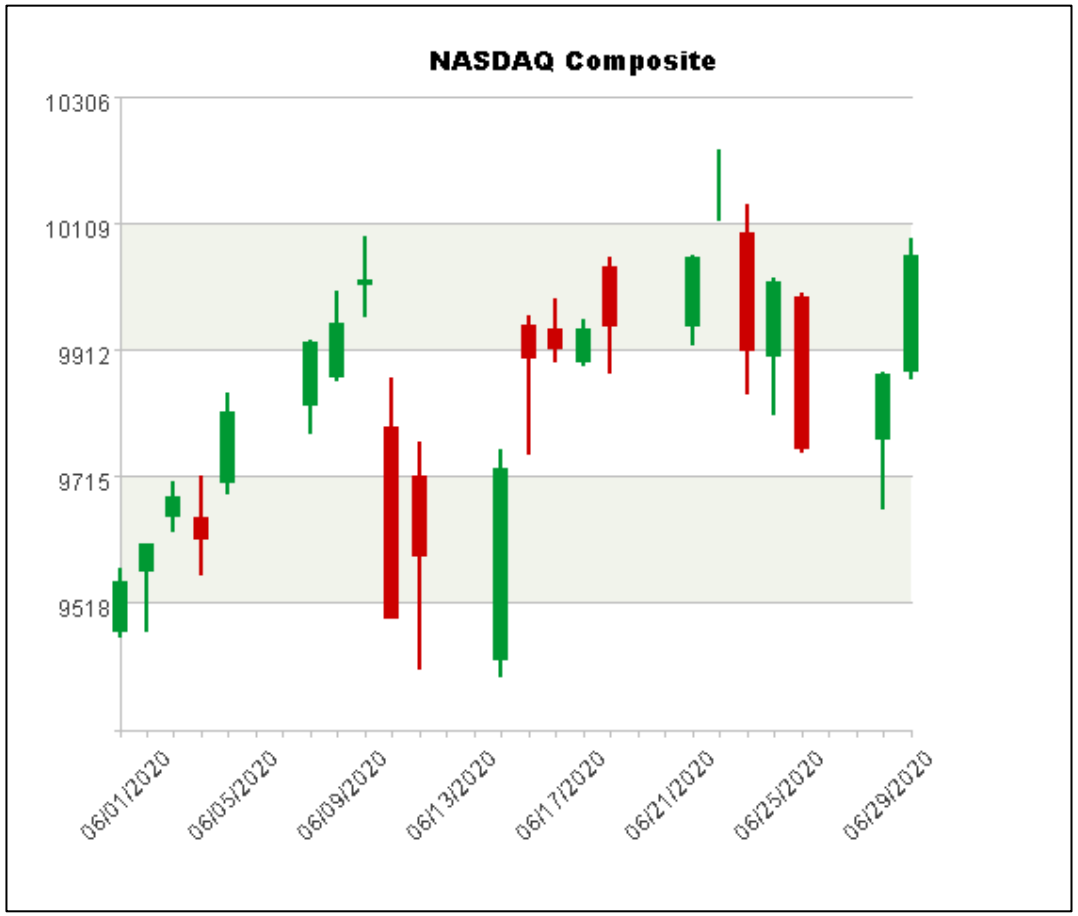
Edit Column properties

Property	Value	Exp.	Legend
Data3		A3.(Open)	
Stacking type	Not stacked		
Allow text overla...	☑		
Transparency	1.0		
Visible	☑		
HTML link			
Screen tip			
Link target	_blank		
Appearance			
Column width	8		
Column shape	Column		
Border style	———		
Border weight	0		
Border color			
Fill color		A3.(Status)	legend
Text			
Data			
Axis1	d		
Data1		A3.(Date)	
Axis2	y		
Data2		A3.(Close)	

The meaning of setting properties in the above figure is to draw a column at data1 of date axis and data2 to data3 of numeric axis, and use legend mapping for filling color.

15.6.3 Exercises:K-line chart

After execution, check the **column chart of NASDAQ Composite Index** in A11:



Chapter 16
Big data and cursor

16.1 Big data and cursor

- Big data
- Concept of cursor

16.2 Basic cursor operations

- Use a cursor to read data
- Loop the cursor

16.3 Cursor functions

- Aggregate operations of a cursor
- Use cursor functions
- Deferred calculation vs. immediate calculation

16.4 Ordered cursor

16.5 Big cursor

- Big sorting
- Big grouping
- Combine
- Merge
- Join
- Multi cursors

16.6 Exercises

Chapter 16
Big data and cursor

16.1 Big data and cursor

16.1.1 Big data

All the operations we mentioned before are carried out in memory, but sometimes the amount of data is too large to be fully loaded into the memory, so the methods mentioned before can not be used directly.

We call such data **big data**, which is generally stored on the hard disk in the form of file or database. This chapter discusses the calculation of big data stored in the form of **file**.

16.1.1 Big data

Big data is in our daily life:

An e-commerce order table

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
3	123314	F	193846.25	1993/10/14	5-LOW	Clerk#000000955	0
4	136777	O	32151.78	1995/10/11	5-LOW	Clerk#000000124	0
5	44485	F	144659.2	1994/7/30	5-LOW	Clerk#000000925	0
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED	Clerk#000000058	0
7	39136	O	252004.18	1996/1/10	2-HIGH	Clerk#000000470	0
...	...	...	...	...	...	...	...

The order records contain the order number, customer number, order status, order total price, order date, order priority and other related information.

With the expansion of users and business, the amount of data has doubled every year.

Call record table

SERIALNUMBER	OUTID	INID	CALLTIME	DURATION	CHARGE
1	13781722635	18635273625	2006-08-11 08:53:22	97	0.00
2	18572635487	18173625183	2006-08-11 09:01:03	108	0.15
3	18572635182	13526384957	2006-08-11 09:02:11	34	0.00
4	13782769374	18502938475	2006-08-11 09:24:21	87	0.00
5	13672394817	18591820385	2006-08-11 09:33:12	132	0.15
6	18638274625	18108273645	2006-08-11 09:35:37	876	0.00
7	18127364918	13582937465	2006-08-11 09:43:18	56	0.15
...	...	…	…	…	…

The call records contain the serial number, the caller ID, the receiver ID, the dialing time, the call duration, and the charge.

There may be hundreds of millions of mobile phone users, and we can imagine the scale of the actual call recording information.

16.1.2 Concept of cursor

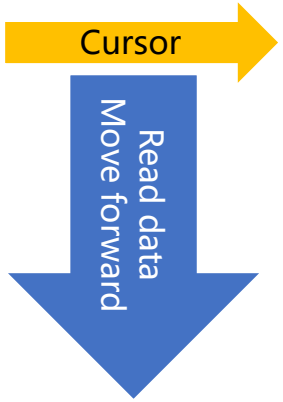
When processing big data calculation, we can not read all the data into memory and can only read in batches. We call this mechanism of reading data into the memory as **cursor**.

Cursor makes it easy for us to use the method of processing small data in loop to complete the calculation of big data.

16.1.2 Concept of cursor

A cursor is similar to a pointer in that it gets data by moving the pointer's position.

A cursor is generally **unidirectional**, that is, it can only read forward or skip data, and cannot go back.



O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
3	123314	F	193846.25	1993/10/14	5-LOW	Clerk#000000955	0
4	136777	O	32151.78	1995/10/11	5-LOW	Clerk#000000124	0
5	44485	F	144659.2	1994/7/30	5-LOW	Clerk#000000925	0
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED	Clerk#000000058	0
7	39136	O	252004.18	1996/1/10	2-HIGH	Clerk#000000470	0
...	...	...	...	...	...	...	...

Chapter 16
Big data and cursor

16.2 Basic cursor operations

16.2.1 Use a cursor to read data

【Example 16.1】Use the cursor to read the first five order records of the order table file. The order table file is orders.txt :

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW
2	78002	O	46929.18	1996/12/1	1-URGENT
3	123314	F	193846.25	1993/10/14	5-LOW
4	136777	O	32151.78	1995/10/11	5-LOW
5	44485	F	144659.2	1994/7/30	5-LOW
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED
7	39136	O	252004.18	1996/1/10	2-HIGH

Note: The purpose of the example is to understand the principle and function usage, and the amount of data used is small. The following examples also have the similar situation, and we will not explain again.

16.2.1 Use a cursor to read data

【Example 16.1】Use the cursor to read the first five order records of the order table file. The order table file is orders.txt.

Solution

Use **f.cursor()** to create a cursor for the file. If the first line of the file is field names, use the **@t** option. Use **cs.fetch()** to read data.

	A	B
1	=file("orders.txt")	/Open the file orders.txt
2	=A1.cursor@t()	/Create a cursor for the file. The usage here is similar to import. You can also handle files with a CSV suffix
3	=A2.fetch(5)	/Read the first five records from the cursor
4	=A2.close()	/Not all data in the cursor has been read. The cursor needs to be closed

A3 returned result:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW
2	78002	O	46929.18	1996/12/1	1-URGENT
3	123314	F	193846.25	1993/10/14	5-LOW
4	136777	O	32151.78	1995/10/11	5-LOW
5	44485	F	144659.2	1994/7/30	5-LOW

16.2.1 Use a cursor to read data

【Example 16.2】Use the cursor to skip the first two records and read the remaining order records of the order table file. The order table file is orders.csv :

```
O_ORDERKEY, O_CUSTKEY, O_ORDERSTATUS, O_TOTALPRICE, O_ORDERDATE, O_ORDERPRIORITY
1, 36901, O, 173665.47, 1996/1/2, 5-LOW
2, 78002, O, 46929.18, 1996/12/1, 1-URGENT
3, 123314, F, 193846.25, 1993/10/14, 5-LOW
4, 136777, O, 32151.78, 1995/10/11, 5-LOW
5, 44485, F, 144659.2, 1994/7/30, 5-LOW
6, 55624, F, 58749.59, 1992/2/21, 4-NOT SPECIFIED
7, 39136, O, 252004.18, 1996/1/10, 2-HIGH
```

16.2.1 Use a cursor to read data

【Example 16.2】Use the cursor to skip the first two records and read the remaining order records of the order table file. The order table file is orders.csv.

Solution

Use **cs.skip()** to skip records. Use **cs.fetch()** to read data.

	A	B
1	=file("orders.csv")	/Open file orders.csv
2	=A1.cursor@tc(;)	/Create a cursor for the csv file
3	=A2.skip(2)	/Skip two records from the current cursor
4	=A2.fetch()	/Read remaining records from cursor

A4 returned result:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3	123314	F	193846.25	1993/10/14	5-LOW
4	136777	O	32151.78	1995/10/11	5-LOW
5	44485	F	144659.2	1994/7/30	5-LOW
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED
7	39136	O	252004.18	1996/1/10	2-HIGH

Note: There are 7 records in the example. cs.fetch() read all remaining records directly. In practice, we need to consider the amount of data to avoid memory overflow.

cs.skip(n;x) detailed description: skip n records or until x changes, return the actual number of records skipped. Return the number of remaining records and close the cursor when all parameters are omitted.

16.2.1 Use a cursor to read data

【Example 16.3】Use the cursor to filter the order record with customer number 36901 in the order table file orders.txt :

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW
2	78002	O	46929.18	1996/12/1	1-URGENT
3	123314	F	193846.25	1993/10/14	5-LOW
4	136777	O	32151.78	1995/10/11	5-LOW
5	44485	F	144659.2	1994/7/30	5-LOW
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED
7	39136	O	252004.18	1996/1/10	2-HIGH

16.2.1 Use a cursor to read data

【Example 16.3】Use the cursor to filter the order record with customer number 36901 in the order table file orders.txt.

Solution

Use **cs.select()** to attach filter operation to cursor.

	A	B
1	=file("orders.txt")	/Open file orders.txt
2	=A1.cursor@t()	/Create a cursor for the file
3	=A2.select(O_CUSTKEY==36901)	/Add filter operation to cursor
4	=A2.fetch()	/Get the result of the cursor operation
5	=A2.fetch(1)	/Get the first record of cursor

A4 returned result:

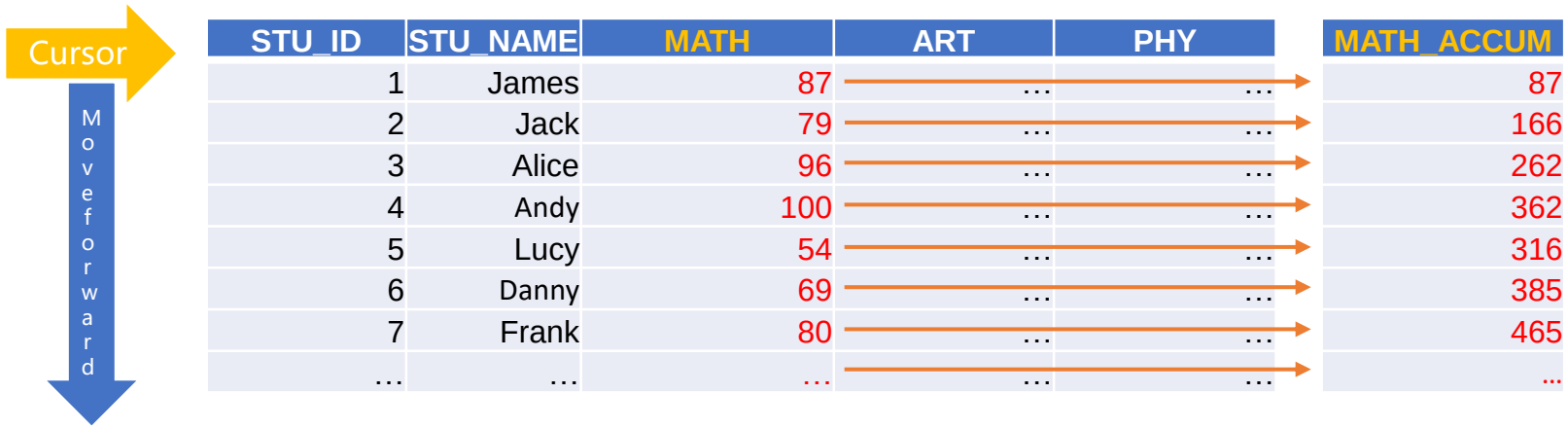
O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW

Note: Because the cursor is unidirectional, after executing A4, the cursor is closed, and the returned result of A5 is **null**.

16.2.2 Loop the cursor

To **loop the cursor** refers to reading one or more records from the cursor according to certain rules in the loop process.

For example: Get the total math score of a class



The cursor starts from the beginning, moves forward, reads the record (that meets the condition) each time, and performs the calculation.

16.2.2 Loop the cursor

【Example 16.4】Calculate the sum of order amount in order table. The order table file is orders.txt :

O_ORDERKEY		O_CUSTKEY	O_ORDERSTATUS		O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW		
2	78002	O	46929.18	1996/12/1	1-URGENT		
3	123314	F	193846.25	1993/10/14	5-LOW		
4	136777	O	32151.78	1995/10/11	5-LOW		
5	44485	F	144659.2	1994/7/30	5-LOW		
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED		
7	39136	O	252004.18	1996/1/10	2-HIGH		

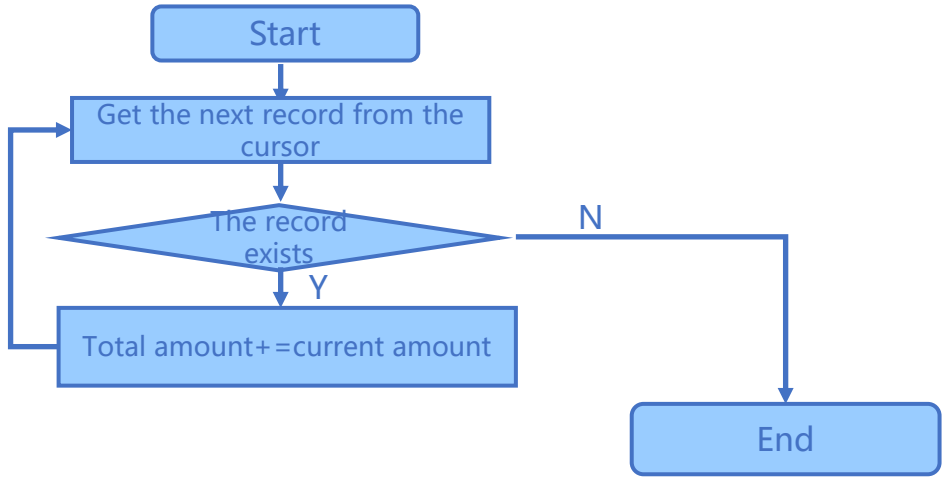
16.2.2 Loop the cursor

【Example 16.4】Calculate the sum of order amount in order table. The order table file is orders.txt.

Solution

Use the **for cs** statement to read the data in the cursor in loop and do calculations as required.

	A	B	C
1	=file("orders.txt").cursor@t(O_TOTALPRICE:decimal)	0	/A1 opens the file and creates a cursor. Here, only the total price field is used, and its type is set to decimal. The initial value of B1 is 0
2	for A1	>B1+=A2.O_TOTALPRICE	/A2 loops the A1 cursor, gets one record each time. B2 accumulates A2's amount to B1



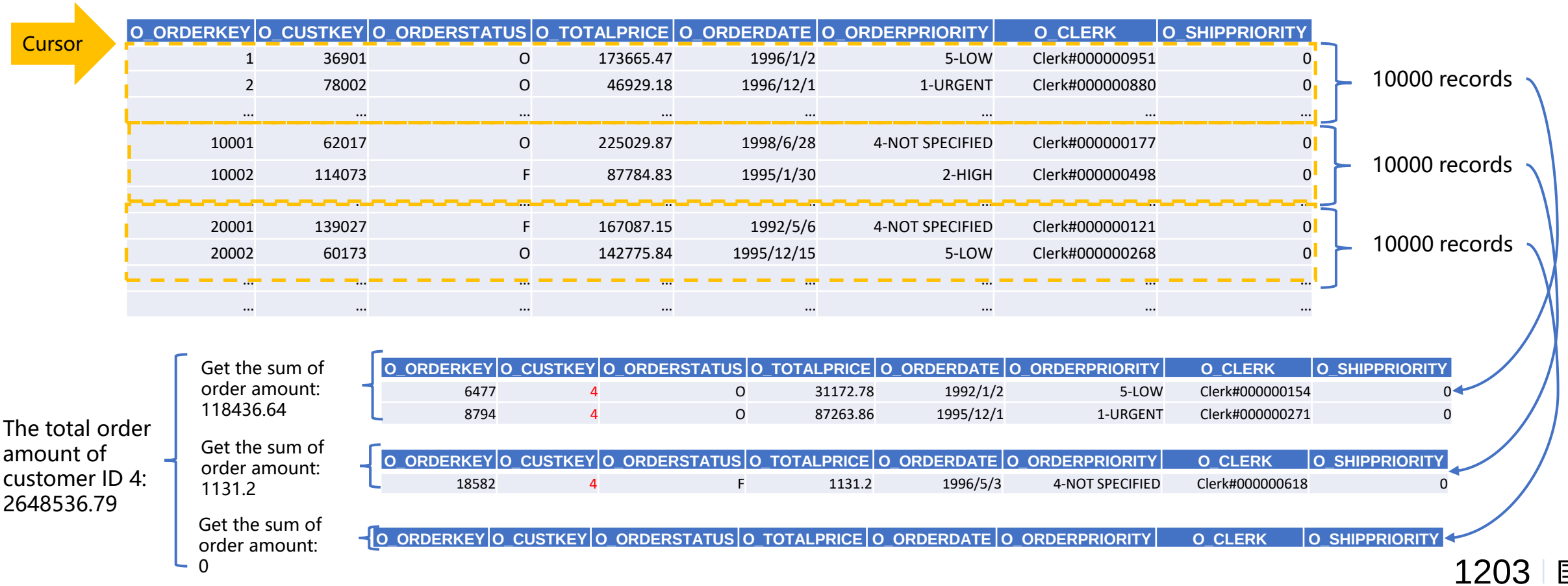
B1
Value
902005.65

Note that in the calculation of big data, the accuracy of double precision data is often not enough. In this case, the **decimal** type can be used.

16.2.2 Loop the cursor

We can also read specified number of data and do calculations by looping the cursor.

For example: for the data of an order table, read 10000 records each time, and calculate the total order amount of customer ID 4.



16.2.2 Loop the cursor

【Example 16.5】Calculate the sum of O_TOTALPRICE with O_CUSTKEY as 4 in the order table. The order table file is orders.txt:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	4	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
…	…	…	…	…	…	…	…
10001	62017	O	225029.87	1998/6/28	4-NOT SPECIFIED	Clerk#000000177	0
10002	114073	F	87784.83	1995/1/30	2-HIGH	Clerk#000000498	0
…	…	…	…	…	…	…	…
20001	4	F	167087.15	1992/5/6	4-NOT SPECIFIED	Clerk#000000121	0
20002	60173	O	142775.84	1995/12/15	5-LOW	Clerk#000000268	0
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…

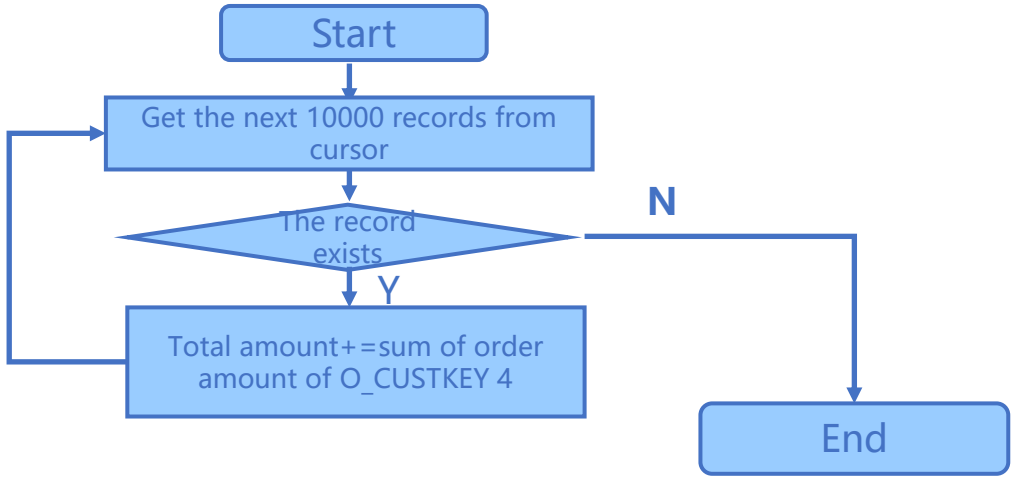
16.2.2 Loop the cursor

【Example 16.5】Calculate the sum of O_TOTALPRICE with O_CUSTKEY as 4 in the order table. The order table file is orders.txt.

Solution

Use the **for cs, n** statement to read the data in the cursor circularly, and calculate the n pieces of data read out each time as required.

	A	B	C
1	=file("orders.txt").cursor@t(O_CUSTKEY,O_TOTALPRICE:decimal)	0	/A1 opens the file and creates a cursor, and the initial value of B1 is 0
2	for A1,10000	=A2.select(O_CUSTKEY==4).sum(O_TOTALPRICE)	/A2 loops through the cursor and gets 10000 records each time, B2 filters and calculates the data
3		>B1=B1+B2	/B3 accumulates the calculated result of B2 to B1

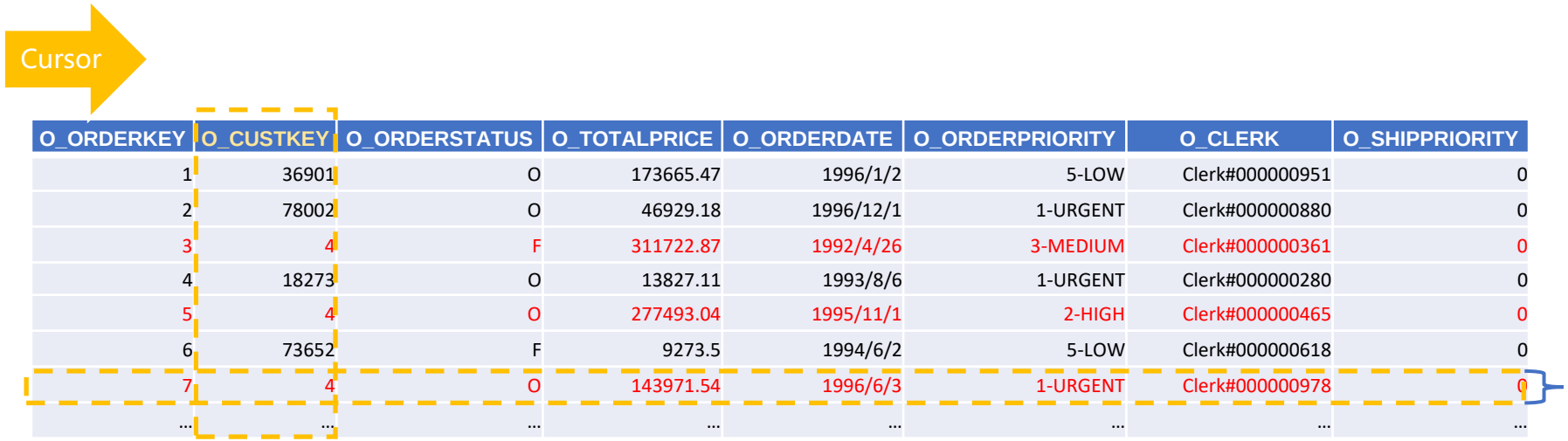


B1
Value
2648536.79

16.2.2 Loop the cursor

Sometimes looping the cursor does not traverse all data.

For example: find the first three records with O_CUSTKEY 4.



O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
3	4	F	311722.87	1992/4/26	3-MEDIUM	Clerk#000000361	0
4	18273	O	13827.11	1993/8/6	1-URGENT	Clerk#000000280	0
5	4	O	277493.04	1995/11/1	2-HIGH	Clerk#000000465	0
6	73652	F	9273.5	1994/6/2	5-LOW	Clerk#000000618	0
7	4	O	143971.54	1996/6/3	1-URGENT	Clerk#000000978	0
...	...	...	...	...	...	...	...

Stop the loop if the conditions are met, and the data after record 7 will no longer be read.

16.2.2 Loop the cursor

【Example 16.6】Find the first three records with O_CUSTKEY 4 in the order table. The order table file is orders.txt:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
…	…	…	…	…	…	…	…
164711	4	F	311722.87	1992/4/26	3-MEDIUM	Clerk#000000361	0
…	…	…	…	…	…	…	…
385825	4	O	277493.04	1995/11/1	2-HIGH	Clerk#000000465	0
…	…	…	…	…	…	…	…
1192231	4	O	143971.54	1996/6/3	1-URGENT	Clerk#000000978	0
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…

16.2.2 Loop the cursor

【Example 16.6】Find the first three records with O_CUSTKEY 4 in the order table. The order table file is orders.txt.

Solution

Calculate the data read out by the loop according to the requirements, and **terminate** the loop when certain conditions are met.

	A	B	C	D
1	=file("orders.txt").cursor@t()	[]		/A1 opens the file and creates a cursor, B1 sets the initial value as empty set
2	for A1,10000	=A2.select(O_CUSTKEY==4)	>B1=B1|B2	/A2 loops through the cursor and gets 10000 records each time, B2 gets records with O_CUSTKEY 4 , C2 appends result to B1
3		if B1.len()>=3	>B1=B1(to(3))	/B3 judges whether the number of records is greater than or equal to 3. If the condition is met, C3 gets the first three records because B1 may have more than 3 records
4			break	/If the B3 condition is met, terminate the loop
5	>A1.close()			/Because the data may not be read completely, close the cursor actively

B1

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
164711	4	F	311722.87	1992/4/26	3-MEDIUM	Clerk#000000361	0
385825	4	O	277493.04	1995/11/1	2-HIGH	Clerk#000000465	0
1192231	4	O	143971.54	1996/6/3	1-URGENT	Clerk#000000978	0

In the example, the cursor is looped only 30 times, which is far less than the number of loops needed for traversing all data.

Chapter 16
Big data and cursor

16.3 Cursor functions

16.3.1 Aggregate operations of a cursor

The previous section talked about handling aggregate operations of the cursor by looping the cursor, but it is obviously too cumbersome.

Similar to table sequence, a cursor also has encapsulated direct aggregate operations.

16.3.1 Aggregate operations of a cursor

【Example 16.7】Calculate the order quantity, the maximum order amount (max(O_TOTALPRICE)), and the sum of order amount (sum(O_TOTALPRICE)) in the order table(orders.txt).

Part of the data in orders.txt is as follows:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
3	123314	F	193846.25	1993/10/14	5-LOW	Clerk#000000955	0
4	136777	O	32151.78	1995/10/11	5-LOW	Clerk#000000124	0
5	44485	F	144659.2	1994/7/30	5-LOW	Clerk#000000925	0
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED	Clerk#000000058	0
7	39136	O	252004.18	1996/1/10	2-HIGH	Clerk#000000470	0
32	130057	O	208660.75	1995/7/16	2-HIGH	Clerk#000000616	0
33	66958	F	163243.98	1993/10/27	3-MEDIUM	Clerk#000000409	0

16.3.1 Aggregate operations of a cursor

【Example 16.7】Calculate the order quantity, the maximum order amount (max(O_TOTALPRICE)), and the sum of order amount (sum(O_TOTALPRICE)) in the order table(orders.txt).

Solution

Use cs.total() function to complete all kinds of aggregation of the complete set.

	A	B
1	=file("orders.txt").cursor@t()	/Open the file and create a cursor
2	=A1.total(count(~),max(O_TOTALPRICE),sum(O_TOTALPRICE))	/Calculate the total number of orders, the maximum order amount and the total amount

A2

Index	Member
1	1500000
2	555285.16
3	2.268293064474466E11

The results of three rows are respectively: the total number of orders, the maximum order amount and the total amount

It is different from the total set aggregate operation(A.sum/A.count, etc.) of table sequence. Because the cursor can only traverse once, to calculate multiple values at the same time, it is encapsulated in the same function(cs.total).

16.3.1 Aggregate operations of a cursor

【Example 16.8】Find the top three highest and lowest order amount (O_TOTALPRICE)in each year (year(O_ORDERDATE)) from order table(orders.txt).

Part of the data in orders.txt is as follows:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
3	123314	F	193846.25	1993/10/14	5-LOW	Clerk#000000955	0
4	136777	O	32151.78	1995/10/11	5-LOW	Clerk#000000124	0
5	44485	F	144659.2	1994/7/30	5-LOW	Clerk#000000925	0
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED	Clerk#000000058	0
7	39136	O	252004.18	1996/1/10	2-HIGH	Clerk#000000470	0
32	130057	O	208660.75	1995/7/16	2-HIGH	Clerk#000000616	0
33	66958	F	163243.98	1993/10/27	3-MEDIUM	Clerk#000000409	0

16.3.1 Aggregate operations of a cursor

【Example 16.8】Find the top three highest and lowest order amount (O_TOTALPRICE)in each year (year(O_ORDERDATE)) from order table(orders.txt).

Solution

Use cs.groups() function to group, and use aggregate function top.

	A	B
1	=file("orders.txt").cursor@t()	/Open the file and create a cursor
2	=A1.groups(year(O_ORDERDATE):Year;top(3;O_TOTALPRICE):Min, top(-3;O_TOTALPRICE):Max)	/find the three order records with the lowest and highest order amount by year. To find the three highest order amounts, use a negative sign in the top function

A2:

Index	Year	Min	Max
1	1992	[[1600323,53551,F, ...],[82...	[[1750466,21433,F, ...],[358...
2	1993	[[5267200,107698,F, ...],[4...	[[4515876,100685,F, ...],[38...
3	1994	[[5363650,1351,F, ...],[404...	[[4722021,128120,F, ...],[3...
4	1995	[[5062848,148522,F, ...],[1...	[[3967937,109186,O, ...],[57...
5	1996	[[1788706,120961,O, ...],[...	[[2199712,66790,O, ...],[215...
6	1997	[[3790757,85468,O, ...],[2...	[[3043270,144617,O, ...],[45...
7	1998	[[2159139,3115,O, ...],[51...	[[1395745,100159,O, ...],[33...

Index	O_ORDER...	O_CUSTKEY	O_ORDER...	O_TOTAL...	O_ORDER...	O_O
1	1600323	53551	F	866.9	1992-04...	1-U
2	823814	64043	F	870.88	1992-01...	1-U
3	3309383	82949	F	908.18	1992-07...	1-U

Index	O_ORDER...	O_CUSTKEY	O_ORDER...	O_TOTAL...	O_ORDER...	O_O
1	1750466	21433	F	555285.16	1992-11...	4-N
2	3586919	24049	F	522644.48	1992-11...	1-U
3	2185667	51796	F	511359.88	1992-10...	1-U

Cursor aggregate operations can also be used on the cursor returned by cursor functions.

16.3.1 Aggregate operations of a cursor

【Example 16.9】Calculate the order quantity, the maximum order amount (max(O_TOTALPRICE)), and the sum of order amount (sum(O_TOTALPRICE))in the order table(orders.txt).

Part of the data in orders.txt is as follows:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
3	123314	F	193846.25	1993/10/14	5-LOW	Clerk#000000955	0
4	136777	O	32151.78	1995/10/11	5-LOW	Clerk#000000124	0
5	44485	F	144659.2	1994/7/30	5-LOW	Clerk#000000925	0
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED	Clerk#000000058	0
7	39136	O	252004.18	1996/1/10	2-HIGH	Clerk#000000470	0
32	130057	O	208660.75	1995/7/16	2-HIGH	Clerk#000000616	0
33	66958	F	163243.98	1993/10/27	3-MEDIUM	Clerk#000000409	0

16.3.1 Aggregate operations of a cursor

【Example 16.9】Calculate the order quantity, the maximum order amount (max(O_TOTALPRICE)), and the sum of order amount (sum(O_TOTALPRICE))in the order table(orders.txt).

Solution

Use cs.groups(;...) function to complete all kinds of aggregation of the complete set.

	A	B
1	=file("orders.txt").cursor@t()	/Open the file and create a cursor
2	=A1.groups(;count(~):count, max(O_TOTALPRICE):max, sum(O_TOTALPRICE):total)	/Calculate the total number of orders, the maximum order amount and the total amount

A2

Index	count	max	total
1	1500000	555285.16	2.268293064474466E11

The aggregate functions in cs.groups(;...) support sum/count/max/min/top/avg.

16.3.2 Use cursor functions

Similarly, there are corresponding **cursor functions** for **filtering**, **aligned generation**, **join** and other operations related to table sequence.

However, unlike the table sequence, these functions on the cursor cannot get the results immediately. Instead, these operations are attached to the cursor and then returns the cursor. These operations are actually performed when the cursor data is **fetch** later.

16.3.2 Use cursor functions

Generating new data with cursor function(**cs.new()**):

For example: For the employee table, combine the NAME field with the SURNAME field to generate a new field FULLNAME, and remove the original fields NAME and SURNAME.

Cursor

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D

Cursor

EID	FULLNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley Smith	F	Texas	1975/5/13	2004/7/30	R&D

【Example 16.10】Combine the NAME field and the SURNAME field in the employee table as field FullName. The employee table file is emp.txt:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D
6	Matthew	Johnson	M	California	1984/7/7	2005/7/7	Sales
7	Alexis	Smith	F	Illinois	1972/8/16	2002/8/16	Sales
8	Megan	Wilson	F	California	1979/4/19	1984/4/19	Marketing
9	Victoria	Davis	F	Texas	1983/12/7	2009/12/7	HR
10	Ryan	Johnson	M	Pennsylvania	1976/3/12	2006/3/12	R&D
...	...	...	...	...	...	...	...

16.3.2 Use cursor functions

【Example 16.10】Combine the NAME field and the SURNAME field in the employee table as field FullName. The employee table file is emp.txt:

Solution

Use cursor function **cs.new()** to generate a new data structure on the original cursor.

	A	B
1	=file("emp.txt").cursor@t(EID,NAME,SURNAME,GENDER,STATE,BIRTHDAY)	/Open the file and create a cursor
2	=A1.new(EID,NAME+" "+SURNAME:FullName,GENDER,STATE,BIRTHDAY)	/Combine the name and surname into full name, and remove the original fields of name and surname
3	=A2.fetch()	/Fetch data

A3

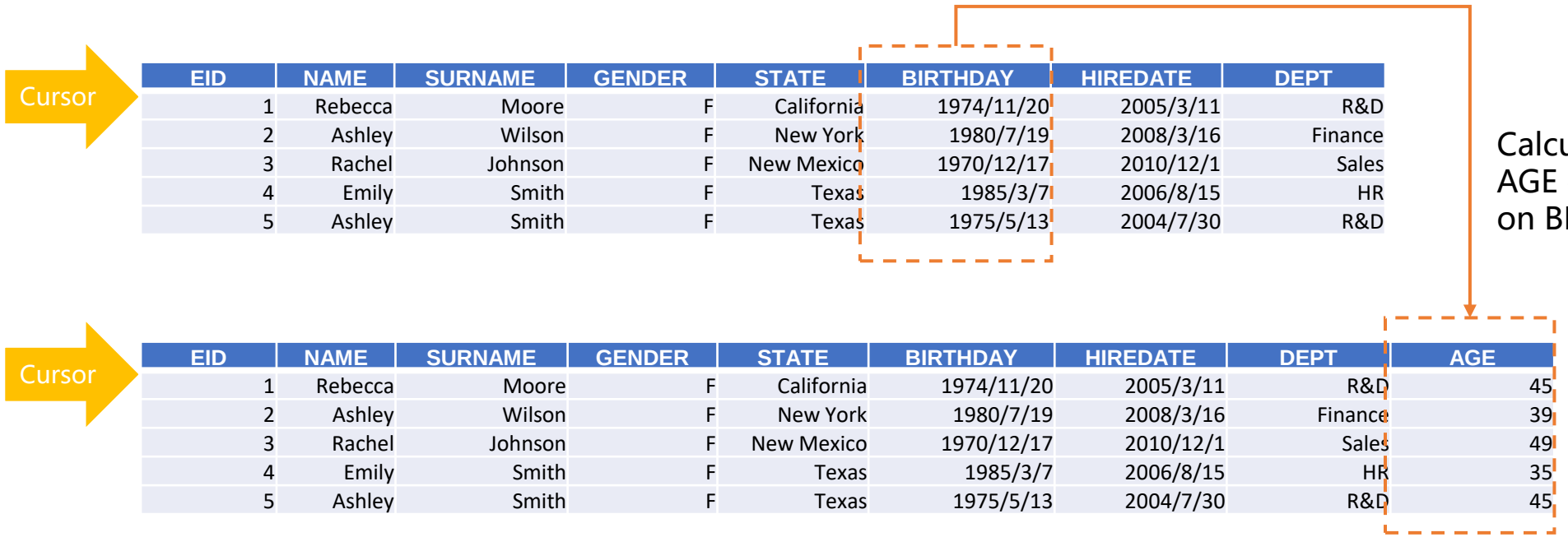
Index	EID	FullName	GENDER	STATE	BIRTHDAY
1	1	Rebecca Moore	F	California	1974-11-20
2	2	Ashley Wilson	F	New York	1980-07-19
3	3	Rachel Johnson	F	New Mexico	1970-12-17
4	4	Emily Smith	F	Texas	1985-03-07
5	5	Ashley Smith	F	Texas	1975-05-13
…	…	…	…	…	…

The amount of data used in the example is small, and fetch gives all the results. In practice, we need to consider whether there will be **memory overflow**.

The first five records of the results are shown.

Adding fields with cursor function(**cs.derive()**):

For example: For the employee table, calculate and add the AGE column according to the employee's birthday.



EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	AGE
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D	45
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance	39
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales	49
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR	35
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D	45

【Example 16.11】For the employee table, calculate and add the Age column according to the employee's birthday. The employee table file is emp.txt :

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D
6	Matthew	Johnson	M	California	1984/7/7	2005/7/7	Sales
7	Alexis	Smith	F	Illinois	1972/8/16	2002/8/16	Sales
8	Megan	Wilson	F	California	1979/4/19	1984/4/19	Marketing
9	Victoria	Davis	F	Texas	1983/12/7	2009/12/7	HR
10	Ryan	Johnson	M	Pennsylvania	1976/3/12	2006/3/12	R&D
...	...	...	...	...	...	...	...

16.3.2 Use cursor functions

Add calculated column

【Example 16.11】For the employee table, calculate and add the Age column according to the employee's birthday. The employee table file is emp.txt.

Solution

Use cursor function **cs.derive()** to calculate and add a column on the original cursor.

	A	B
1	=file("emp.txt").cursor@t(EID,NAME,SURNAME,GENDER,STATE,BIRTHDAY)	/Create a cursor for the file and read the specified fields
2	=A1.derive(age(BIRTHDAY):Age)	/Add calculate column Age
3	=A2.fetch()	/Fetch data

A3

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	Age
1	1	Rebecca	Moore	F	California	1974-11-20	45
2	2	Ashley	Wilson	F	New York	1980-07-19	39
3	3	Rachel	Johnson	F	New Mexico	1970-12-17	49
4	4	Emily	Smith	F	Texas	1985-03-07	35
5	5	Ashley	Smith	F	Texas	1975-05-13	45
…	…	…		…	…	…	…

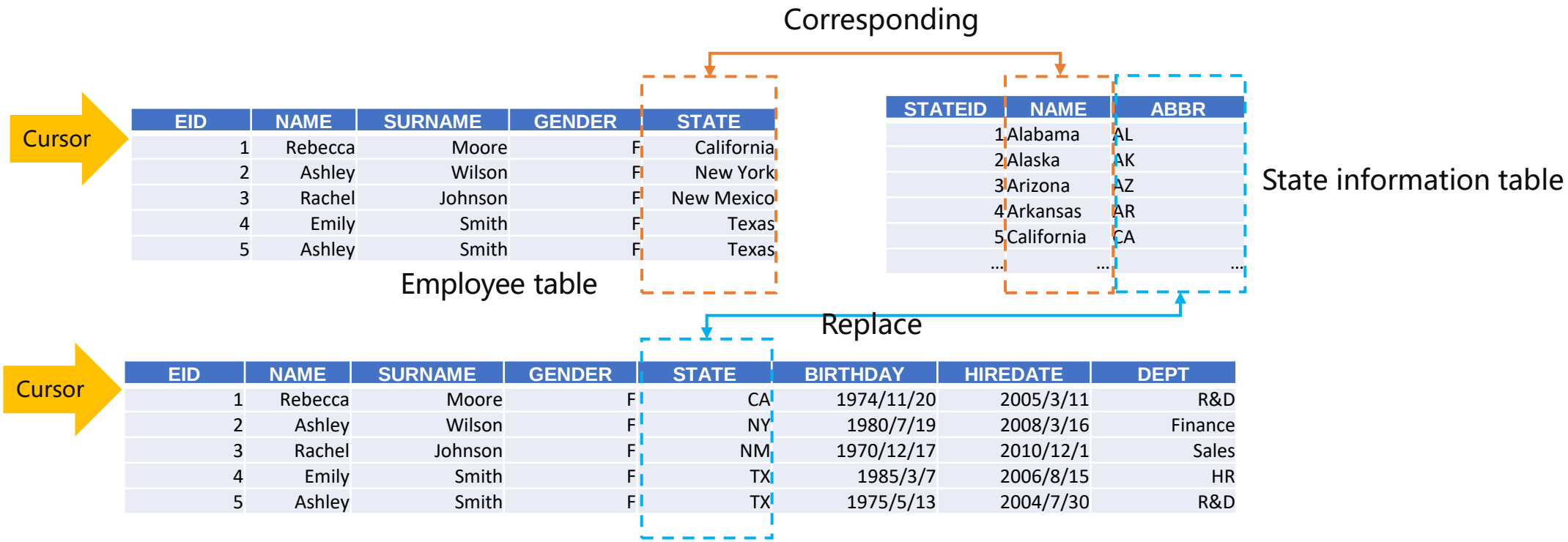
The amount of data used in the example is small, and fetch gives all the results. In practice, we need to consider whether there will be **memory overflow**.

The first five records of the results are shown.

16.3.2 Use cursor functions

Modifying fields with cursor function(**cs.run()**):

For example: Replace STATE in employee table by abbreviation (ABBR) according to state information table.



【Example 16.12】Replace state in employee table by abbreviation(ABBR) according to state information table. The employee table file is emp.txt, and the state information table file is states.txt:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D
6	Matthew	Johnson	M	California	1984/7/7	2005/7/7	Sales
7	Alexis	Smith	F	Illinois	1972/8/16	2002/8/16	Sales
8	Megan	Wilson	F	California	1979/4/19	1984/4/19	Marketing
9	Victoria	Davis	F	Texas	1983/12/7	2009/12/7	HR
10	Ryan	Johnson	M	Pennsylvania	1976/3/12	2006/3/12	R&D
...	...	...	...	...	...	...	...

Employee table (emp.txt)

STATEID	NAME	ABBR
1	Alabama	AL
2	Alaska	AK
3	Arizona	AZ
4	Arkansas	AR
5	California	CA
6	Colorado	CO
7	Connecticut	CT
8	Delaware	DE
9	Florida	FL
10	Georgia	GA
...	...	...

State information table (states.txt)

16.3.2 Use cursor functions

【Example 16.12】Replace state in employee table by abbreviation(ABBR) according to state information table. The employee table file is emp.txt, and the state information table file is states.txt.

Solution

Use cursor function **cs.run()** to modify the field values on the original cursor.

	A	B
1	=file("emp.txt").cursor@t(EID,NAME,SURNAME,GENDER,STATE,BIRTHDAY)	/Create a cursor for the file and read the specified fields
2	=file("states.txt").import@t()	/Read state information
3	=A1.run(STATE=A2.select@1(NAME==STATE).ABBR)	/Replace the state field with the abbreviation
4	=A1.fetch()	/Fetch the data after modifying the field value of the original cursor

A4

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY
1	1	Rebecca	Moore	F	CA	1974-11-20
2	2	Ashley	Wilson	F	NY	1980-07-19
3	3	Rachel	Johnson	F	NM	1970-12-17
4	4	Emily	Smith	F	TX	1985-03-07
5	5	Ashley	Smith	F	TX	1975-05-13
…	…	…		…	…	…

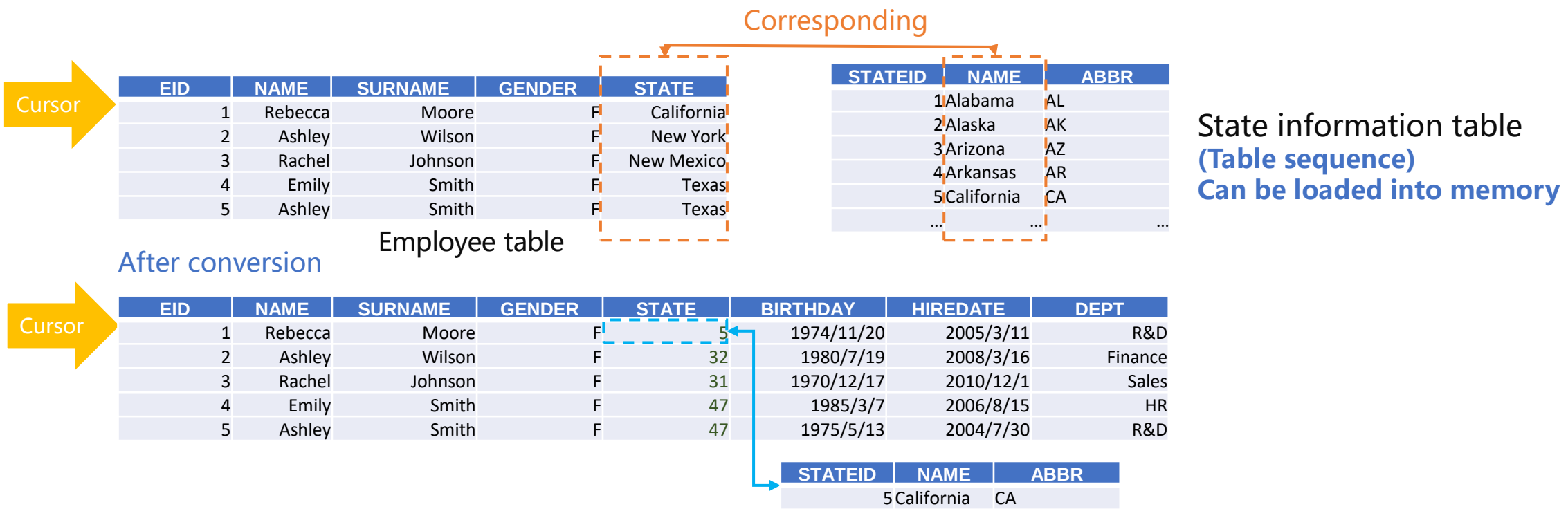
The amount of data used in the example is small, and fetch gives all the results. In practice, we need to consider whether there will be **memory overflow**.

The first five records of the results are shown.

16.3.2 Use cursor functions

Foreign key objectification is a common method to deal with foreign key table. Chapter 14 introduces the functions used when the amount of data is small. When the amount of data is large, similar cursor functions(**cs.switch ()**) are used to objectify the foreign key. Note that the foreign key table must be a table sequence that can be loaded into memory.

For example: Convert the state in the employee table to the corresponding state record;



【Example 16.13】Convert the state in the employee table to the corresponding state record. The employee table file is emp.txt , state information table file is states.txt :

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D
6	Matthew	Johnson	M	California	1984/7/7	2005/7/7	Sales
7	Alexis	Smith	F	Illinois	1972/8/16	2002/8/16	Sales
8	Megan	Wilson	F	California	1979/4/19	1984/4/19	Marketing
9	Victoria	Davis	F	Texas	1983/12/7	2009/12/7	HR
10	Ryan	Johnson	M	Pennsylvania	1976/3/12	2006/3/12	R&D
…	…	…	…	…	…	…	…

Employee table (emp.txt)

STATEID	NAME	ABBR
1	Alabama	AL
2	Alaska	AK
3	Arizona	AZ
4	Arkansas	AR
5	California	CA
6	Colorado	CO
7	Connecticut	CT
8	Delaware	DE
9	Florida	FL
10	Georgia	GA
…	…	…

State information table (states.txt)

16.3.2 Use cursor functions

【Example 16.13】Convert the state in the employee table to the corresponding state record. The employee table file is emp.txt , state information table file is states.txt :

Solution

Use cursor function **cs.switch()** to convert the field values to the records on the original cursor.

	A	B
1	=file("emp.txt").cursor@t(EID,NAME,SURNAME,GENDER,STATE,BIRTHDAY)	/Create a cursor for the file and read the specified fields
2	=file("states.txt").import@t()	/Read state information to a table sequence
3	=A1.switch(STATE,A2:NAME)	/Replace the state field with the corresponding record
4	=A1.fetch()	/Fetch the data after modifying the field value of the original cursor

A4

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY
1	1	Rebecca	Moore	F	[5,Califor…	1974-11-20
2	2	Ashley	Wilson	F	[32,New …	1980-07-19
3	3	Rachel	Johnson	F	[31,New …	1970-12-17
4	4	Emily	Smith	F	[43,Texa…	1985-03-07
5	5	Ashley	Smith	F	[43,Texa…	1975-05-13
…	…	…		…	…	…

STATEID	NAME	ABBR
5	California	CA

【Example 16.14】Add a new field to the employee table to save the corresponding record of the state field. The employee table file is emp.txt , state information table file is states.txt:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D
6	Matthew	Johnson	M	California	1984/7/7	2005/7/7	Sales
7	Alexis	Smith	F	Illinois	1972/8/16	2002/8/16	Sales
8	Megan	Wilson	F	California	1979/4/19	1984/4/19	Marketing
9	Victoria	Davis	F	Texas	1983/12/7	2009/12/7	HR
10	Ryan	Johnson	M	Pennsylvania	1976/3/12	2006/3/12	R&D
…	…	…	…	…	…	…	…

Employee table (emp.txt)

STATEID	NAME	ABBR
1	Alabama	AL
2	Alaska	AK
3	Arizona	AZ
4	Arkansas	AR
5	California	CA
6	Colorado	CO
7	Connecticut	CT
8	Delaware	DE
9	Florida	FL
…	…	…

State information table (states.txt)

16.3.2 Use cursor functions

【Example 16.14】Add a new field to the employee table to save the corresponding record of the state field. The employee table file is emp.txt , state information table file is states.txt (There is missing data in state information table) :

Solution

Use cursor function cs.join to add a field to the original cursor to save the foreign key table record.

	A	B
1	=file("emp.txt").cursor@t(EID, NAME, SURNAME, GENDER, STATE, BIRTHDAY)	/Create a cursor for the file and read the specified fields
2	=file("states.txt").import@t()	/Read state information as a table sequence
3	=A1.join(STATE, A2:NAME, ~:STATER)	/Add a new field to save the foreign key table record
4	=A1.fetch()	/Fetch the data after adding a field to the original cursor

A4

Index	EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT	SALARY	STATER
1	1	Rebecca	Moore	F	California	1974-11-20	2005-03-11	R&D	7000	[5,California,CA]
2	2	Ashley	Wilson	F	New York	1980-07-19	2008-03-16	Finance	11000	[32,New York,NY]
3	3	Rachel	Johnson	F	New Mexico	1970-12-17	2010-12-01	Sales	9000	[31,New Mexico,NM]
4	4	Emily	Smith	F	Texas	1985-03-07	2004-07-30	HR	7000	[43,Texas,TX]
5	5	Ashley	Smith	F	Texas	1975-05-13	2005-07-07	R&D	16000	[43,Texas,TX]
…	…	…	…	…	…	…	…	…	…	…

16.3.2 Use cursor functions

Calculate immediately —— Cursor attached operations are performed immediately, such as: cs.groups/cs.total/cs.iterate/cs.id etc.

For example, calculate the total number of orders in each year in order table (orders.txt).

	A	B
1	=file("orders.txt").cursor@t()	/Create a cursor for the file
2	=A1.groups(year(O_ORDERDATE):Year;count(~):OrderCount)	/Calculate the total orders by year

A2

Index	Year	OrderCount
1	1992	227089
2	1993	226645
3	1994	227597
4	1995	228637
5	1996	228626
6	1997	227783
7	1998	133623
…	…	…

The immediate calculation of cursor refers to that after operations are attached to cursor, the cursor data is immediately traversed to do calculation and return the result.

16.3.2 Use cursor functions

Deferred calculation —— Cursor attached operations are not calculated immediately, but wait until the cursor is traversed. Such as: cs.select/cs.new/cs.derive/cs.run etc.

For example, find all records in the order table with o_ Custkey as 4.

	A	B
1	=file("orders.txt").cursor@t()	/Open file and create a cursor
2	=A1.select(O_CUSTKEY==4)	/Select records with o_ Custkey as 4 from cursor
3	=A2.fetch()	/Fetch data from cursor A2

A1

Value
com.raqsoft.dm.cursor.FileCursor@1f48da9f

A1 opens the file, creates a cursor, and returns the cursor

A2

Value
com.raqsoft.dm.cursor.FileCursor@1f48da9f

A2 attaches the select operation on the cursor of A1, does not calculate immediately, and still returns the cursor

A3

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK
164711	4	F	311722.87	1992/4/26	3-MEDIUM	Clerk#000000361
385825	4	O	277493.04	1995/11/1	2-HIGH	Clerk#000000465
1192231	4	O	143971.54	1996/6/3	1-URGENT	Clerk#000000978
1226497	4	F	88317.19	1993/10/4	1-URGENT	Clerk#000000154
...	...	...	...	...	...	...

When A3 traverses the cursor, the attached select operation will be calculated and return the result

The advantage of deferred calculation is that it is similar to memory operation in writing and easy to understand, but it does not really generate intermediate result sets (in-memory computing will) to take up space (or cache).
It's similar to write immediate calculation, but multi-step will produce intermediate result sets, which take up a lot of memory.

Chapter 16
Big data and cursor

16.4 Ordered cursor

16.4 Ordered cursor

When the data in the cursor is **ordered**, you can also use the **ordered operation** methods similar to the table sequence mentioned before.

【Example 16.15】The order table with ordered order date is grouped by date, and get the first three groups of records. The order table file is orders.txt:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
3271	33058	F	125535.15	1992/1/1	1-URGENT	Clerk#000000421	0
5607	91021	F	37356.47	1992/1/1	4-NOT SPECIFIED	Clerk#000000137	0
20742	136801	F	121343.42	1992/1/1	4-NOT SPECIFIED	Clerk#000000913	0
23010	91141	F	97372.7	1992/1/1	2-HIGH	Clerk#000000831	0
…	…	…	…	…	…	…	…
3987111	364	F	78703.45	1992/1/2	2-HIGH	Clerk#000000129	0
1141924	700	F	173598.79	1992/1/2	2-HIGH	Clerk#000000603	0
631968	817	F	238394.75	1992/1/2	3-MEDIUM	Clerk#000000939	0
2102785	856	F	173343.08	1992/1/2	5-LOW	Clerk#000000678	0
...	...	...	...	...	...	...	...

16.4 Ordered cursor

【Example 16.15】The order table with ordered order date is grouped by date, and get the first three groups of records. The order table file is orders.txt.

Solution

Use **cs.group()** to do adjacent equivalent grouping of ordered cursor.

	A	B
1	=file("orders.txt").cursor@t()	/Open file and create a cursor
2	=A1.group(O_ORDERDATE)	/Group by date
3	=A1.fetch@x(3)	/Get the result of the first three groups

A3

Index	Member
1	[[3271,33058,F, ...],[5607,91021,F, ...],[20...
2	[[4505602,89161,F, ...],[4512870,18220,F...
3	[[6657,46195,F, ...],[22693,21748,F, ...],[2...

Index	O_ORDER...	O_CUSTKEY	O_ORDER...	O_TOTAL...	O_ORDERDA...
1	3271	33058	F	125535.15	1992-01-01
2	5607	91021	F	37356.47	1992-01-01
3	20742	136801	F	121343.42	1992-01-01
4	23010	91141	F	97372.7	1992-01-01
5	27015	10954	F	41753.82	1992-01-01

16.4 Ordered cursor

【Example 16.16】The order table with ordered order date is used to count the number of orders per day. The order table file is orders.txt:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
3271	33058	F	125535.15	1992/1/1	1-URGENT	Clerk#000000421	0
5607	91021	F	37356.47	1992/1/1	4-NOT SPECIFIED	Clerk#000000137	0
20742	136801	F	121343.42	1992/1/1	4-NOT SPECIFIED	Clerk#000000913	0
23010	91141	F	97372.7	1992/1/1	2-HIGH	Clerk#000000831	0
…	…	…	…	…	…	…	…
3987111	364	F	78703.45	1992/1/2	2-HIGH	Clerk#000000129	0
1141924	700	F	173598.79	1992/1/2	2-HIGH	Clerk#000000603	0
631968	817	F	238394.75	1992/1/2	3-MEDIUM	Clerk#000000939	0
2102785	856	F	173343.08	1992/1/2	5-LOW	Clerk#000000678	0
...	...	...	...	...	...	...	...

16.4 Ordered cursor

【Example 16.16】The order table with ordered order date is used to count the number of orders per day. The order table file is orders.txt.

Solution

Use cs.groups@o to do adjacent equivalent grouping aggregation for ordered cursor.

	A	B
1	=file("orders.txt").cursor@t()	/Open file and create a cursor
2	=A1.groups@o(O_ORDERDATE;count(~):count)	/Group by date and count

A3

Index	O_ORDERDATE	EID
1	1992-01-01	621
2	1992-01-02	612
3	1992-01-03	598
4	1992-01-04	670
5	1992-01-05	641
…	…	…

16.4 Ordered cursor

Sometimes it is necessary to complete some complex calculations for ordered cursor, and it is not possible to get the results directly by using cursor functions, and we need to loop the ordered cursor in this case.

Cursor

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
3271	33058	F	125535.15	1992/1/1	1-URGENT	Clerk#000000421	0
5607	91021	F	37356.47	1992/1/1	4-NOT SPECIFIED	Clerk#000000137	0
20742	136801	F	121343.42	1992/1/1	4-NOT SPECIFIED	Clerk#000000913	0
23010	91141	F	97372.7	1992/1/1	2-HIGH	Clerk#000000831	0
...	...	...	...	...	...	...	...
3987111	364	F	78703.45	1992/1/2	2-HIGH	Clerk#000000129	0
1141924	700	F	173598.79	1992/1/2	2-HIGH	Clerk#000000603	0
631968	817	F	238394.75	1992/1/2	3-MEDIUM	Clerk#000000939	0
2102785	856	F	173343.08	1992/1/2	5-LOW	Clerk#000000678	0
...	...	...	...	...	...	...	...

For example: Loop the order table with ordered order date until access condition (O_ORDERDATE) changes, count the number of orders per day.

Data of 1992-01-01

Data of 1992-01-02

16.4 Ordered cursor

【Example 16.17】Count the number of orders per day in the order table ordered by order date. The order table file is orders.txt:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
3271	33058	F	125535.15	1992/1/1	1-URGENT	Clerk#000000421	0
5607	91021	F	37356.47	1992/1/1	4-NOT SPECIFIED	Clerk#000000137	0
20742	136801	F	121343.42	1992/1/1	4-NOT SPECIFIED	Clerk#000000913	0
23010	91141	F	97372.7	1992/1/1	2-HIGH	Clerk#000000831	0
…	…	…	…	…	…	…	…
3987111	364	F	78703.45	1992/1/2	2-HIGH	Clerk#000000129	0
1141924	700	F	173598.79	1992/1/2	2-HIGH	Clerk#000000603	0
631968	817	F	238394.75	1992/1/2	3-MEDIUM	Clerk#000000939	0
2102785	856	F	173343.08	1992/1/2	5-LOW	Clerk#000000678	0
…	…	…	…	…	…	…	…

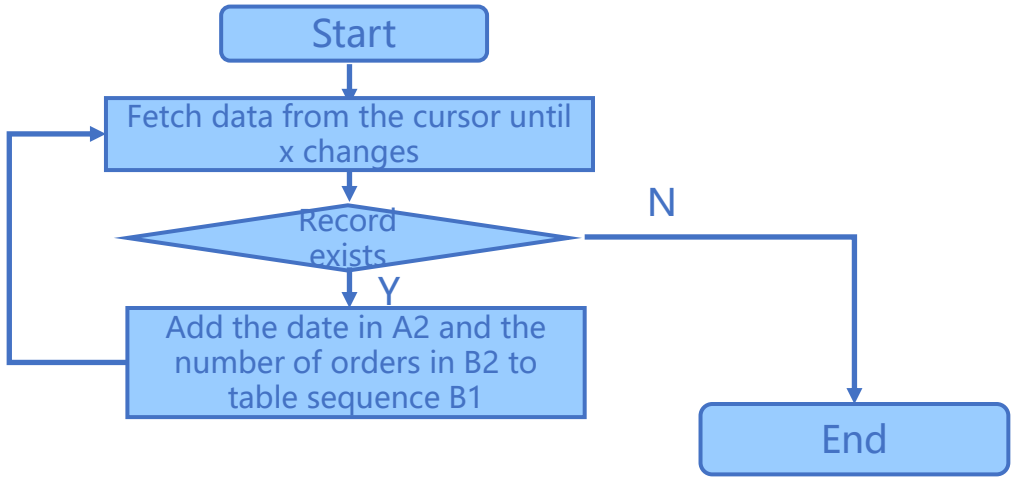
16.4 Ordered cursor

【Example 16.17】Count the number of orders per day in the order table ordered by order date. The order table file is orders.txt.

Solution

Use **for cs;x** statement to loop the data in the cursor, read out the data every time x changes, and calculate as required.

	A	B	C
1	=file("orders.txt").cursor@t()	=create(Date, Count)	/A1 opens the file and creates a cursor, B1 creates the empty table sequence
2	for A1;O_ORDERDATE	=A2.count()	/A2 gets the data in loop by date, B2 counts the order quantity of the day
3		>B1.insert(0, A2.O_ORDERDATE, B2)	/Add the date in A2 and the number of orders in B2 to the table sequence



B1

Date	Count
1922-01-01	621
1922-01-02	612
1922-01-03	598
...	...

【Example 16.18】Find the maximum number of record rows of a single event and the corresponding user tag in the log:

```
--flag--
usertag:0ohoi4
pybmbceiodlrafrelayzf
cxpiqlqoftsist
…
mbfcmgocnvixyueky
ogyzbllkwarvt
kjowchyuyxmvdde
--flag--
usertag:5sb9y
evhnjwtvxcy
dbepkgheyeelp
jmqpdrvtfodbzdszlvk
…
```

log.txt
The starting ID of each event is --flag--

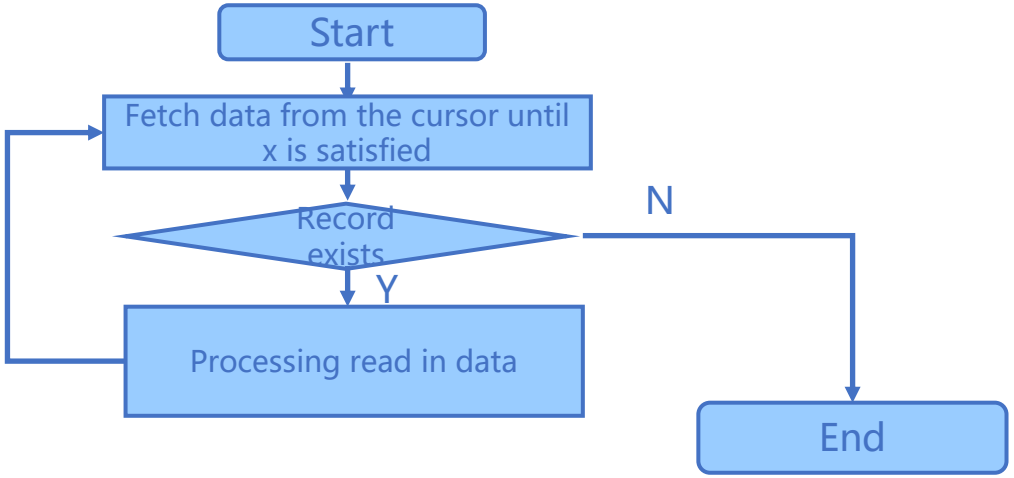
16.4 Ordered cursor

【Example 16.18】Find the maximum number of record rows of a single event and the corresponding user tag in the log.

Solution

Use the **for cs;x** statement to read the data in the cursor in loop until the expression x is true each time.

	A	B	C
1	=file("log.txt").cursor@i()		A1 opens the file and creates a cursor, @i returns a sequence
2	for A1;~=="--flag--"	=[A2(2), A2.len()-2]	/A1 reads the single event data, A2 gets the user tag and the number of event record lines. A2.len()-2 removes the first two lines
3		>if(B2(2)>B1(2), B1=B2)	/Judge the current result and replace the previous result if it satisfies the condition



B1

Index	Member
1	usertag:0ohoi4
2	999998

Chapter 16
Big data and cursor

16.5 Big cursor

16.5.1 Big sorting

The sorting of table sequence(see Chapter 11 - Calculation of table for details) introduces the sorting method (A.sort()) when the amount of data is small and can be put into memory.

If the amount of data to be sorted is huge and cannot be read into memory at one time, the ordinary sorting method can not be executed. In this case, **external storage sorting** is needed(**cs.sortx()**).

In the external storage sorting, part of the data is read for sorting each time, and the sorted data is written into temporary files. Finally, these temporary files are merged and sorted to obtain the final result.

16.5.1 Big sorting

【Example 16.19】Sort the call record table by caller number, and get the top 100 sorted records.

Part data of the records.txt is as follows:

Index	outid	duration
1	18500694558	11
2	18500665932	94
3	18600854809	176
4	18600677317	120
5	18100517144	321
6	18100857113	86
7	18100733553	81
8	18100796252	362
9	18500313020	354
10	18500113597	133
11	18500276539	204
12	18600581342	132
13	18600022612	173
14	18600424740	314
15	18100339412	302

16.5.1 Big sorting

【Example 16.19】Sort the call record table by caller number, and get the top 100 sorted records.

Solution

Use **cs.sortx()** function to sort cursor in external storage.

	A	B
1	=file("records.txt").cursor@t()	/Open file and create a cursor
2	=A1.sortx(outid)	/Use the external storage sorting function of cursor to sort by caller number
3	=A2.fetch(100)	/Get the first 100 records
4	>A1.close()	/Close the cursor

A3:

Index	outid	duration
1	13600000001	329
2	13600000001	72
3	13600000001	222
4	13600000001	261
5	13600000001	217
6	13600000001	193
7	13600000001	296
8	13600000001	123

16.5.2 Big grouping

Section 16.3.2 describes that cursor aggregate operations can use **cs.groups()** function. However, if the aggregated result is also big data, it is necessary to use external storage to complete the calculation.

For example, for the customers of small and medium-sized enterprises, the number of customers may be only tens of thousands, and the memory mode(**cs.groups()**) can be used for grouping and aggregation by customers. However, if it is a telecom company, the number of users is at least 10 million or even hundreds of millions. In this case, the result of grouping and aggregation usually can't fit into the memory, and external storage must be used for grouping(cs.groupx).

Large scale e-commerce's statistics of customers' orders and telecom companies' statistics of customers' daily Internet traffic will face such problems.

16.5.2 Big grouping

【Example 16.20】From the call record table, count the number of calls per user, and view the first five results.

Part data of the records.txt is as follows:

Index	outid	duration
1	18500694558	11
2	18500665932	94
3	18600854809	176
4	18600677317	120
5	18100517144	321
6	18100857113	86
7	18100733553	81
8	18100796252	362
9	18500313020	354
10	18500113597	133
11	18500276539	204
12	18600581342	132
13	18600022612	173
14	18600424740	314
15	18100339412	302

16.5.2 Big grouping

【Example 16.20】From the call record table, count the number of calls per user, and view the first five results.

Solution

Use cs.groupx() function to group cursor in external storage.

	A	B
1	=file("records.txt").cursor@t()	/Open file and create a cursor
2	=A1.groupx(outid;count(~):count)	/Use the external storage grouping function of cursor to group by caller number and count the number of calls
3	=A2.fetch(5)	/Get the first 5 results
4	>A1.close()	/Close the cursor

A3:

Index	outid	count
1	13600000001	27
2	13600000002	34
3	13600000003	27
4	13600000004	23
5	13600000005	19

The results of cs.groupx() and cs.sortx().group() are the same and ordered:

	A
1	=file("records.txt").cursor@t()
2	=A1.sortx(outid)
3	=A2.group(outid;count(~):count)
4	=A3.fetch(5)
5	>A1.close()

16.5.3 Combine

Sometimes the data to be processed is distributed in **multiple files**, which needs to be **combined** before query or calculation.

For example, when order data of different priorities is in different files, the order records of each priority are combined(**CS.conjx()**).

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	2243/9/13	1992/1/1	1-URGENT
176768	112850	2936/6/26	1992/1/1	1-URGENT
201414	11642	2367/2/8	1992/1/1	1-URGENT
228804	96325	2278/6/10	1992/1/1	1-URGENT
...	...	...	...	...

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
23010	91141	2166/8/4	1992/1/1	2-HIGH
27137	54109	2667/12/17	1992/1/1	2-HIGH
175975	7511	2682/3/11	1992/1/1	2-HIGH
249668	115781	2008/11/5	1992/1/1	2-HIGH
...	...	...	...	...

1-URGENT.txt 2-HIGH.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	2243/9/13	1992/1/1	1-URGENT
176768	112850	2936/6/26	1992/1/1	1-URGENT
...	...	...	...	...
23010	91141	2166/8/4	1992/1/1	2-HIGH
27137	54109	2667/12/17	1992/1/1	2-HIGH
...	...	...	...	...

16.5.3 Combine

【Example 16.21】Combine the cursors of the data files with different priorities. Get 100 records from the combined cursor each time. If the priorities of the first and last records in these 100 records are different, then jump out of the loop and observe the data characteristics of the last 100 records.

Part data of each priority file is as follows (There are more than 100 records in each file):

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	125535.00	1992/1/1	1-URGENT
176768	112850	378570.00	1992/1/1	1-URGENT
201414	11642	170608.00	1992/1/1	1-URGENT
228804	96325	138224.00	1992/1/1	1-URGENT
...	...	...	...	...

1-URGENT.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
23010	91141	97372.00	1992/1/1	2-HIGH
27137	54109	280493.00	1992/1/1	2-HIGH
175975	7511	285691.00	1992/1/1	2-HIGH
249668	115781	39757.00	1992/1/1	2-HIGH
...	...	...	...	...

2-HIGH.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
37543	124229	178835.23	1992/1/1	3-MEDIUM
88199	36158	268020.49	1992/1/1	3-MEDIUM
115585	143335	201582.36	1992/1/1	3-MEDIUM
126753	136505	207936.02	1992/1/1	3-MEDIUM
...	...	...	...	...

3-MEDIUM.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
5607	91021	37356.47	1992/1/1	4-NOT SPECIFIED
20742	136801	121343.42	1992/1/1	4-NOT SPECIFIED
45697	109417	345167.47	1992/1/1	4-NOT SPECIFIED
159266	99056	125324.77	1992/1/1	4-NOT SPECIFIED
...	...	...	...	...

4-NOT SPECIFIED.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
27015	10954	41753.82	1992/1/1	5-LOW
59718	24749	83608.04	1992/1/1	5-LOW
139655	25249	118709.27	1992/1/1	5-LOW
190656	23695	154110.00	1992/1/1	5-LOW
...	...	...	...	...

5-LOW.txt

16.5.3 Combine

【Example 16.21】Combine the cursors of the data files with different priorities. Get 100 records from the combined cursor each time. If the priorities of the first and last records in these 100 records are different, then jump out of the loop and observe the data characteristics of the last 100 records.

Solution

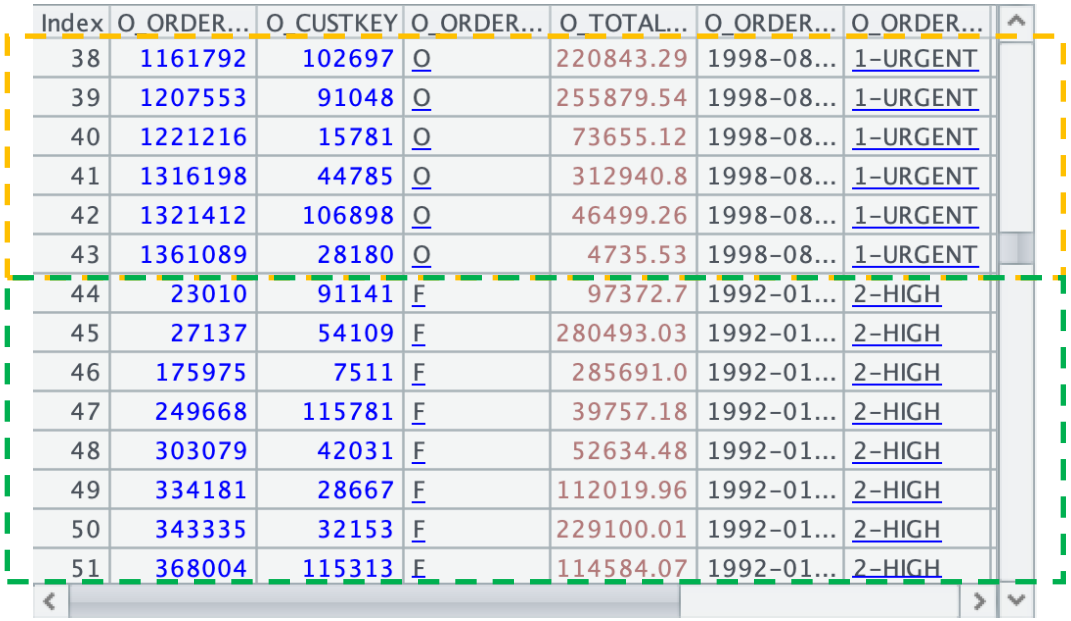
Use CS.conjx() function to combine cursors.

	A	B	C	D
1	=file("1-URGENT.txt")			/Open file
2	=file("2-HIGH.txt")			/Open file
3	=file("3-MEDIUM.txt")			/Open file
4	=file("4-NOT SPECIFIED.txt")			/Open file
5	=file("5-LOW.txt")			/Open file
6	=[A1:A5].(~.cursor@t())			/Create a cursor for each file in the file sequence
7	=A6.conjx()	100		/Combine the cursor sequence
8	for	=A7.fetch(B7)		/Loop the data in the cursor, get 100 pieces at a time
9		=B8(1).O_ORDERPRIORITY	=B8.m(-1).O_ORDERPRIORITY	/Observe the priority of the first and last records
10		if B9!=C9	break	/If the priority is inconsistent, jump out of the loop
11	>A7.close()			/Close the cursor

16.5.3 Combine

【Example 16.21】Combine the cursors of the data files with different priorities. Get 100 records from the combined cursor each time. If the priorities of the first and last records in these 100 records are different, then jump out of the loop and observe the data characteristics of the last 100 records.

After executing the code on the previous page, observe the result in B8 as follows:



Index	O_ORDER...	O_CUSTKEY	O_ORDER...	O_TOTAL...	O_ORDER...	O_ORDER...
38	1161792	102697	O	220843.29	1998-08...	1-URGENT
39	1207553	91048	O	255879.54	1998-08...	1-URGENT
40	1221216	15781	O	73655.12	1998-08...	1-URGENT
41	1316198	44785	O	312940.8	1998-08...	1-URGENT
42	1321412	106898	O	46499.26	1998-08...	1-URGENT
43	1361089	28180	O	4735.53	1998-08...	1-URGENT
44	23010	91141	F	97372.7	1992-01...	2-HIGH
45	27137	54109	F	280493.03	1992-01...	2-HIGH
46	175975	7511	F	285691.0	1992-01...	2-HIGH
47	249668	115781	F	39757.18	1992-01...	2-HIGH
48	303079	42031	F	52634.48	1992-01...	2-HIGH
49	334181	28667	F	112019.96	1992-01...	2-HIGH
50	343335	32153	F	229100.01	1992-01...	2-HIGH
51	368004	115313	F	114584.07	1992-01...	2-HIGH

It can be seen that the combined cursor will merge data according to the order of the original cursor sequence.

16.5.3 Merge

Sometimes the data to be processed is distributed in **multiple files**, and all of them is **ordered** by some field. The combining operation will lead to the disorder of the result, while the **merging** operation can keep the original order for further query or calculation.

For example, the order data with different priorities is in different files (ordered by date), and the data files with different priorities are merged orderly by date(**CS.megerx()**).

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	2243/9/13	1992/1/1	1-URGENT
176768	112850	2936/6/26	1992/1/1	1-URGENT
201414	11642	2367/2/8	1992/1/1	1-URGENT
228804	96325	2278/6/10	1992/1/1	1-URGENT
...	...	...	...	...

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
175975	7511	2682/3/11	1992/1/1	2-HIGH
...	...	...	...	...
23010	91141	2166/8/4	1992/1/2	2-HIGH
27137	54109	2667/12/17	1992/1/2	2-HIGH
...	...	...	...	...

1-URGENT.txt

2-HIGH.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	2243/9/13	1992/1/1	1-URGENT
176768	112850	2936/6/26	1992/1/1	1-URGENT
...	...	...	...	...
23010	91141	2166/8/4	1992/1/2	2-HIGH
27137	54109	2667/12/17	1992/1/2	2-HIGH
...	...	...	...	...

16.5.3 Merge

【Example 16.22】Merge the data files with different priorities orderly by date, and get the first 200 records to observe the data characteristics.

The partial data of each priority file is as follows(Each file is ordered by O_OrderDate and there are less than 200 pieces of data per day):

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	125535.00	1992/1/1	1-URGENT
176768	112850	378570.00	1992/1/1	1-URGENT
201414	11642	170608.00	1992/1/1	1-URGENT
228804	96325	138224.00	1992/1/1	1-URGENT
...	...	...	...	...

1-URGENT.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
23010	91141	97372.00	1992/1/1	2-HIGH
27137	54109	280493.00	1992/1/1	2-HIGH
175975	7511	285691.00	1992/1/1	2-HIGH
249668	115781	39757.00	1992/1/1	2-HIGH
...	...	...	...	...

2-HIGH.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
37543	124229	178835.23	1992/1/1	3-MEDIUM
88199	36158	268020.49	1992/1/1	3-MEDIUM
115585	143335	201582.36	1992/1/1	3-MEDIUM
126753	136505	207936.02	1992/1/1	3-MEDIUM
...	...	...	...	...

3-MEDIUM.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
5607	91021	37356.47	1992/1/1	4-NOT SPECIFIED
20742	136801	121343.42	1992/1/1	4-NOT SPECIFIED
45697	109417	345167.47	1992/1/1	4-NOT SPECIFIED
159266	99056	125324.77	1992/1/1	4-NOT SPECIFIED
...	...	...	...	...

4-NOT SPECIFIED.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
27015	10954	41753.82	1992/1/1	5-LOW
59718	24749	83608.04	1992/1/1	5-LOW
139655	25249	118709.27	1992/1/1	5-LOW
190656	23695	154110.00	1992/1/1	5-LOW
...	...	...	...	...

5-LOW.txt

16.5.3 Merge

【Example 16.22】Merge the data files with different priorities orderly by date, and get the first 200 records to observe the data characteristics.

Solution

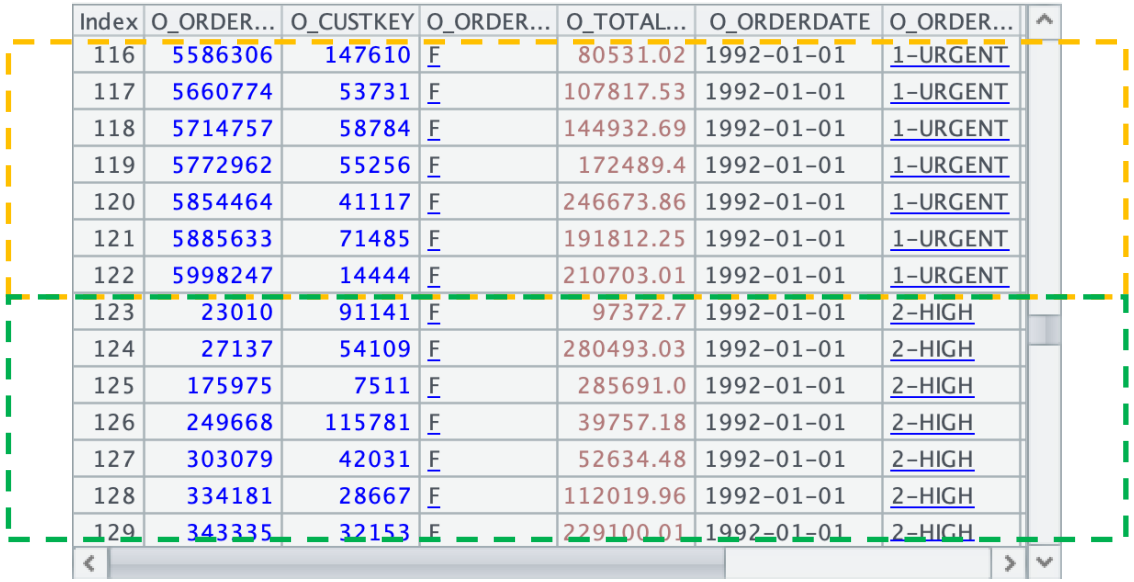
Use **CS.mergex(x)** function to perform an ordered merging. Ordered merging is a method to combine multiple ordered data into one ordered data.

	A	B
1	=file("1-URGENT.txt")	/Open file
2	=file("2-HIGH.txt")	/Open file
3	=file("3-MEDIUM.txt")	/Open file
4	=file("4-NOT SPECIFIED.txt")	/Open file
5	=file("5-LOW.txt")	/Open file
6	=[A1:A5].(~.cursor@t())	/Create a cursor for each file in the file sequence
7	=A6.mergex(O_ORDERDATE)	/Merge cursor sequence in order by date
8	=A7.fetch(200)	/Get the first 200 records
9	>A7.close()	/Close the cursor

16.5.3 Merge

【Example 16.22】Merge the data files with different priorities orderly by date, and get the first 200 records to observe the data characteristics.

After executing the code on the previous page, observe the result in A8 as follows:



Index	O_ORDER...	O_CUSTKEY	O_ORDER...	O_TOTAL...	O_ORDERDATE	O_ORDER...
116	5586306	147610	F	80531.02	1992-01-01	1-URGENT
117	5660774	53731	F	107817.53	1992-01-01	1-URGENT
118	5714757	58784	F	144932.69	1992-01-01	1-URGENT
119	5772962	55256	F	172489.4	1992-01-01	1-URGENT
120	5854464	41117	F	246673.86	1992-01-01	1-URGENT
121	5885633	71485	F	191812.25	1992-01-01	1-URGENT
122	5998247	14444	F	210703.01	1992-01-01	1-URGENT
123	23010	91141	F	97372.7	1992-01-01	2-HIGH
124	27137	54109	F	280493.03	1992-01-01	2-HIGH
125	175975	7511	F	285691.0	1992-01-01	2-HIGH
126	249668	115781	F	39757.18	1992-01-01	2-HIGH
127	303079	42031	F	52634.48	1992-01-01	2-HIGH
128	334181	28667	F	112019.96	1992-01-01	2-HIGH
129	343335	32153	F	229100.01	1992-01-01	2-HIGH

It can be seen that data merged by date is not like CS.conjx() to obtain data in the order of the original cursor sequence, but obtain data of different priority by the order of date.

When using ordered **merging** (**CS.mergex (x)**) , each cursor in the cursor sequence CS must be **ordered** to expression x.

16.5.5 Join

Sometimes it is necessary to synthesize the associated data in multiple cursors to generate a wide table. In this case, each cursor is required to be **ordered** to the associated field.

For example, order data with different priorities is stored in files ordered by date, and we need to find the number of order records of each priority per day. The data of different priorities needs to be **joined** by date (**joinx()**).

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	125535.00	1992/1/1	1-URGENT
176768	112850	378570.00	1992/1/1	1-URGENT
201414	11642	170608.00	1992/1/1	1-URGENT
228804	96325	138224.00	1992/1/1	1-URGENT
...	...	...	...	...

1-URGENT.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
23010	91141	97372.00	1992/1/1	2-HIGH
27137	54109	280493.00	1992/1/1	2-HIGH
175975	7511	285691.00	1992/1/1	2-HIGH
249668	115781	39757.00	1992/1/1	2-HIGH
...	...	...	...	...

2-HIGH.txt

O_ORDERPRIORITY	O_ORDERDATE	COUNT	O_ORDERPRIORITY	O_ORDERDATE	COUNT
1-URGENT	1992/1/1	283	2-HIGH	1992/1/1	315
1-URGENT	1992/1/2	749	2-HIGH	1992/1/2	246
...	...	...	...	...	...

16.5.5 Join

【Example 16.23】Count the order records of each priority per day.

Part data of each priority file is as follows:

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	125535.00	1992/1/1	1-URGENT
176768	112850	378570.00	1992/1/1	1-URGENT
201414	11642	170608.00	1992/1/1	1-URGENT
228804	96325	138224.00	1992/1/1	1-URGENT
...	...	...	...	...

1-URGENT.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
23010	91141	97372.00	1992/1/1	2-HIGH
27137	54109	280493.00	1992/1/1	2-HIGH
175975	7511	285691.00	1992/1/1	2-HIGH
249668	115781	39757.00	1992/1/1	2-HIGH
...	...	...	...	...

2-HIGH.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
37543	124229	178835.23	1992/1/1	3-MEDIUM
88199	36158	268020.49	1992/1/1	3-MEDIUM
115585	143335	201582.36	1992/1/1	3-MEDIUM
126753	136505	207936.02	1992/1/1	3-MEDIUM
...	...	...	...	...

3-MEDIUM.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
5607	91021	37356.47	1992/1/1	4-NOT SPECIFIED
20742	136801	121343.42	1992/1/1	4-NOT SPECIFIED
45697	109417	345167.47	1992/1/1	4-NOT SPECIFIED
159266	99056	125324.77	1992/1/1	4-NOT SPECIFIED
...	...	...	...	...

4-NOT SPECIFIED.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
27015	10954	41753.82	1992/1/1	5-LOW
59718	24749	83608.04	1992/1/1	5-LOW
139655	25249	118709.27	1992/1/1	5-LOW
190656	23695	154110.00	1992/1/1	5-LOW
...	...	...	...	...

5-LOW.txt

16.5.5 Join

【Example 16.23】Count the order records of each priority per day.

Solution

Use **joinx()** function to join multi same ordered cursors.

	A	B
1	=file("1-URGENT.txt").cursor@t()	/Open file and create a cursor
2	=file("2-HIGH.txt").cursor@t()	/Open file and create a cursor
3	=file("3-MEDIUM.txt").cursor@t()	/Open file and create a cursor
4	=file("4-NOT SPECIFIED.txt").cursor@t()	/Open file and create a cursor
5	=file("5-LOW.txt").cursor@t()	/Open file and create a cursor
6	=A1.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count number of orders
7	=A2.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count number of orders
8	=A3.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count number of orders
9	=A4.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count number of orders
10	=A5.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count number of orders
11	=joinx@1(A6:URGENT, O_ORDERDATE;A7:HIGH, O_ORDERDATE;A8:MEDIUM, O_ORDERDATE;A9:NOT_SPECIFIED, O_ORDERDATE;A10:LOW, O_ORDERDATE)	/Join the aggregate results of each priority by order date
12	=A11.fetch(10)	/Get the first 10 records
13	>A11.close()	/Close the cursor

16.5.5 Join

【Example 16.23】Count the order records of each priority per day.

After executing the code on the previous page, observe the result in A12 as follows:

Index	URGENT	HIGH	MEDIUM	NOT_SPECIFIED	LOW
1	[1992-01-01,...	[1992-01-01,...	[1992-01-01,...	[1992-01-01,...	[1992-01-01,...
2	[1992-01-02,...	[1992-01-02,...	[1992-01-02,...	[1992-01-02,...	[1992-01-02,...
3	[1992-01-03,...	[1992-01-03,...	[1992-01-03,...	[1992-01-03,...	[1992-01-03,...
4	[1992-01-04,...	[1992-01-04,...	[1992-01-04,...	[1992-01-04,...	[1992-01-04,...
5	[1992-01-05,...	[1992-01-05,...	[1992-01-05,...	[1992-01-05,...	[1992-01-05,...
6	[1992-01-06,...	[1992-01-06,...	[1992-01-06,...	[1992-01-06,...	[1992-01-06,...
7	[1992-01-07,...	[1992-01-07,...	[1992-01-07,...	[1992-01-07,...	[1992-01-07,...
8	[1992-01-08,...	[1992-01-08,...	[1992-01-08,...	[1992-01-08,...	[1992-01-08,...
9	[1992-01-09,...	[1992-01-09,...	[1992-01-09,...	[1992-01-09,...	[1992-01-09,...
10	[1992-01-10,...	[1992-01-10,...	[1992-01-10,...	[1992-01-10,...	[1992-01-10,...

O_ORDERDATE	O_ORDERPRIORITY	count
1992-01-01	4-NOT SPECIFIED	126

O_ORDERDATE	O_ORDERPRIORITY	count
1992-01-01	5-LOW	115

16.5.5 Join

【Example 16.23】Count the order records of each priority per day.

If the final result needs to be returned as a table sequence, such as the date and the number of orders under each priority, you can use the "new" function after A11.

	A	B
...	…	…
11	=joinx@1(A6:URGENT, O_ORDERDATE;A7:HIGH, O_ORDERDATE;A8:MEDIUM, O_ORDERDATE;A9:NOT_SPECIFIED, O_ORDERDATE;A10:LOW, O_ORDERDATE)	/Join the aggregate results of each priority by order date
12	=A11.new(URGENT.O_ORDERDATE:ORDERDATE, URGENT.count:URGENT, HIGH.count:HIGH, MEDIUM.count:MEDIUM, NOT_SPECIFIED.count:NOT_SPECIFIED, LOW.count:LOW)	/Attach "new" operation to cursor to change the structure to date and the number of orders under each priority
13	=A12.fetch(10)	/Get the first 10 records
14	>A12.close()	/Close the cursor

Index	ORDERDATE	URGENT	HIGH	MEDIUM	NOT_SPE...	LOW
1	1992-01-01	122	120	138	126	115
2	1992-01-02	113	126	125	119	129
3	1992-01-03	132	119	113	127	107
4	1992-01-04	103	144	133	149	141
5	1992-01-05	146	118	130	129	118
6	1992-01-06	116	128	127	132	123
7	1992-01-07	124	136	109	128	142
8	1992-01-08	120	124	124	121	117
9	1992-01-09	127	112	129	136	138
10	1992-01-10	138	135	114	121	118

16.5.6 Multi cursors

When processing big data files, the **multi-core CPU** can also be used to **segment and parallel calculate** the files.

For example: find the orders with transaction amount (O_TOTALPRICE) greater than 200000 in the order table (order table is ordered by O_OrderDate).

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE
1	9527	O	173665.47	1992/1/2
2	4	O	46929.18	1992/12/1
...	...	...	...	...
24903795	2732	F	823642.87	1993/6/25

Segment 1

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE
24903796	62017	O	225029.87	1993/6/28
24903797	14073	F	87784.83	1993/7/30
...	...	...	...	...
49971212	37621	O	62537.17	1994/9/30

Segment 2

4 cursors
Part content of each data segment

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE
49971213	82265	F	167087.15	1995/5/6
49971214	38273	O	142775.84	1995/12/15
...	...	...	...	...
73564722	12653	F	72638.21	

Segment 3

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE
73564723	36901	O	173665.47	1996/1/2
73564724	23126	O	46929.18	1996/12/1
73564725	22673	O	82652.39	1996/12/1
...	...	...	...	...

Segment 4

16.5.6 Multi cursors

【Example 16.24】Find the number of order records with transaction amount greater than 200000 in the order table (create 4-way cursors to complete calculation).

The order table file is orders.txt:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	4	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
...	...	...	...	...	...	...	...
10001	62017	O	225029.87	1998/6/28	4-NOT SPECIFIED	Clerk#000000177	0
10002	114073	F	87784.83	1995/1/30	2-HIGH	Clerk#000000498	0
...	...	...	...	...	...	...	...
20001	4	F	167087.15	1992/5/6	4-NOT SPECIFIED	Clerk#000000121	0
20002	60173	O	142775.84	1995/12/15	5-LOW	Clerk#000000268	0
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

16.5.6 Multi cursors

【Example 16.24】Find the number of order records with transaction amount greater than 200000 in the order table (create 4-way cursors to complete calculation).

Solution

Use the @m option of the cursor function to generate **multi cursors**.

	A	B
1	=file("orders.txt").cursor@tm(;4)	/Open file and create 4 cursors
2	=A1.select(O_TOTALPRICE>200000).total(count(~))	/Attach filtering function to multi cursors and count

A3
Value
450996

16.5.6 Multi cursors

Sometimes the data may come from multiple cursors. In this case, **multiple cursors** (**CS.mcursor()**) can also be used to do calculations in parallel.

For example, when order data of different priorities is in different files, find orders with all priorities with order date of March 1, 1992.

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	125535.00	1992/1/1	1-URGENT
176768	112850	378570.00	1992/1/1	1-URGENT
...	...	...	...	...
153271	33058	178835.23	1992/3/1	1-URGENT
196768	112850	268020.49	1992/3/1	1-URGENT
...	...	...	...	...

Thread 1:search URGENT.txt file

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
23010	91141	97372.00	1992/1/1	2-HIGH
27137	54109	280493.00	1992/1/1	2-HIGH
...	...	...	...	...
223010	91141	207936.02	1992/3/1	2-HIGH
437137	54109	178835.23	1992/3/1	2-HIGH
...	...	...	...	...

Thread 2:search HIGH.txt file

Combine the result:

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
153271	33058	178835.23	1992/3/1	1-URGENT
196768	112850	268020.49	1992/3/1	1-URGENT
...	...	...	...	...
223010	91141	207936.02	1992/3/1	2-HIGH
437137	54109	178835.23	1992/3/1	2-HIGH
...	...	...	...	...

16.5.6 Multi cursors

【Example 16.25】Find all priority orders with the order date of March 1, 1992 and the final result should be ordered by O_ORDERKEY.

The partial data of each priority file is as follows (each file is ordered by O_Orderkey):

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	125535.00	1992/1/1	1-URGENT
176768	112850	378570.00	1992/1/1	1-URGENT
201414	11642	170608.00	1992/1/1	1-URGENT
228804	96325	138224.00	1992/1/1	1-URGENT
...	...	...	...	...

1-URGENT.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
23010	91141	97372.00	1992/1/1	2-HIGH
27137	54109	280493.00	1992/1/1	2-HIGH
175975	7511	285691.00	1992/1/1	2-HIGH
249668	115781	39757.00	1992/1/1	2-HIGH
...	...	...	...	...

2-HIGH.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
37543	124229	178835.23	1992/1/1	3-MEDIUM
88199	36158	268020.49	1992/1/1	3-MEDIUM
115585	143335	201582.36	1992/1/1	3-MEDIUM
126753	136505	207936.02	1992/1/1	3-MEDIUM
...	...	...	...	...

3-MEDIUM.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
5607	91021	37356.47	1992/1/1	4-NOT SPECIFIED
20742	136801	121343.42	1992/1/1	4-NOT SPECIFIED
45697	109417	345167.47	1992/1/1	4-NOT SPECIFIED
159266	99056	125324.77	1992/1/1	4-NOT SPECIFIED
...	...	...	...	...

4-NOT SPECIFIED.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
27015	10954	41753.82	1992/1/1	5-LOW
59718	24749	83608.04	1992/1/1	5-LOW
139655	25249	118709.27	1992/1/1	5-LOW
190656	23695	154110.00	1992/1/1	5-LOW
...	...	...	...	...

5-LOW.txt

16.5.6 Multi cursors

【Example 16.25】Find all priority orders with the order date of March 1, 1992 and the final result should be ordered by O_ORDERKEY.

Solution

Use **CS.mcursor()** function to form multi cursors to multi-way cursors. When each cursor is ordered by a field, the result after **mergex** by the same field is also ordered.

	A	B
1	=file("1-URGENT.txt")	/Open file
2	=file("2-HIGH.txt")	/Open file
3	=file("3-MEDIUM.txt")	/Open file
4	=file("4-NOT SPECIFIED.txt")	/Open file
5	=file("5-LOW.txt")	/Open file
6	=[A1:A5].(~.cursor@t()).mcursor()	/Use files in A1~A5 to create a cursor sequence, and then generate multi-way cursors
7	=A6.select(O_ORDERDATE==date("1992-03-01")).mergex(O_ORDERKEY).fetch()	/Filter the data with the order date of 1992-03-01 and get the result after merging by the order number

A7:

Index	O_ORDER...	O_CUSTKEY	O_...	O_TOTAL...	O_ORDERD...	O_ORDER...	O_CLERK
1	16740	27046	F	101843.93	1992-03-01	5-LOW	Clerk#00...
2	45219	100177	F	246351.96	1992-03-01	5-LOW	Clerk#00...
3	56292	119560	F	304043.03	1992-03-01	5-LOW	Clerk#00...
4	59204	71213	F	250535.19	1992-03-01	3-MEDIUM	Clerk#00...
5	61091	36157	F	9558.5	1992-03-01	3-MEDIUM	Clerk#00...
6	78720	102899	F	191306.41	1992-03-01	2-HIGH	Clerk#00...
7	81057	129427	F	169503.95	1992-03-01	4-NOT SP...	Clerk#00...

Chapter 16
Big data and cursor

16.6 Exercises

16.6 Exercises

【Exercise 16.1】Read the first two records in the order table (orders.txt), skip three records, and then read four records.

The first ten records of the order table(orders.txt) are as follows:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0
3	123314	F	193846.25	1993/10/14	5-LOW	Clerk#000000955	0
4	136777	O	32151.78	1995/10/11	5-LOW	Clerk#000000124	0
5	44485	F	144659.2	1994/7/30	5-LOW	Clerk#000000925	0
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED	Clerk#000000058	0
7	39136	O	252004.18	1996/1/10	2-HIGH	Clerk#000000470	0
32	130057	O	208660.75	1995/7/16	2-HIGH	Clerk#000000616	0
33	66958	F	163243.98	1993/10/27	3-MEDIUM	Clerk#000000409	0
...	...	...	...	...	...	...	...

16.6 Exercises

【Exercise 16.1】Read the first two records in the order table (orders.txt), skip three records, and then read four records.

Solution

Use cs.fetch(n) function to read the records in the cursor, and use the cs.skip(n) function to skip the specified number of records.

	A	B
1	=file("orders.txt")	/Open file orders.txt
2	=A1.cursor@t()	/Create a cursor for the file
3	=A2.fetch(2)	/Read 2 records from the cursor
4	=A2.skip(3)	/The cursor skips three records
5	=A2.fetch(4)	/Continue to fetch 4 records from the cursor
6	>A2.close()	/Cursor data is not all fetched, close cursor

When reading only the required columns, A2 can be changed to:

	A	B
2	=A1.cursor@t(O_ORDERKEY, O_CUSTKEY, O_TOTALPRICE)	/Create a cursor, and read only specified fields

Execution result:

A3:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
1	36901	O	173665.47	1996/1/2	5-LOW	Clerk#000000951	0
2	78002	O	46929.18	1996/12/1	1-URGENT	Clerk#000000880	0

A5:

O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY
6	55624	F	58749.59	1992/2/21	4-NOT SPECIFIED	Clerk#000000058	0
7	39136	O	252004.18	1996/1/10	2-HIGH	Clerk#000000470	0
32	130057	O	208660.75	1995/7/16	2-HIGH	Clerk#000000616	0
33	66958	F	163243.98	1993/10/27	3-MEDIUM	Clerk#000000409	0

16.6 Exercises

【Exercise 16.2】Find the call records with OUTID of 13781722635 in the call record table(record.txt).

The partial data of call record table(record.txt) is as follows:

SERIALNUMBER	OUTID	INID	CALLTIME	DURATION	CHARGE
1	13781722635	18635273625	2006-08-11 08:53:22	97	0.00
2	18572635487	18173625183	2006-08-11 09:01:03	108	0.15
3	18572635182	13526384957	2006-08-11 09:02:11	34	0.00
4	13782769374	18502938475	2006-08-11 09:24:21	87	0.00
5	13672394817	18591820385	2006-08-11 09:33:12	132	0.15
6	18638274625	18108273645	2006-08-11 09:35:37	876	0.00
7	18127364918	13582937465	2006-08-11 09:43:18	56	0.15
…	…	…	…	…	…

16.6 Exercises

【Exercise 16.2】Find the call records with OUTID of 13781722635 in the call record table(record.txt).

Solution

Use cursor function cs.select() to find corresponding records.

	A	B
1	=file("record.txt").cursor@t()	/Open file and create a cursor
2	=A1.select(OUTID==13781722635)	/Select the call record with OUTID 13781722635 from the cursor
3	=A2.fetch()	/Fetch data from A2 cursor

A1

Value
com.raqsoft.dm.cursor.FileCursor@1f48da9f

A2

Value
com.raqsoft.dm.cursor.FileCursor@1f48da9f

A3

SERIALNUMBER	OUTID	INID	CALLTIME	DURATION	CHARGE
1	13781722635	18635273625	2006-08-11 08:53:22	97	0.00
273	13781722635	18173625183	2006-08-15 09:01:03	108	0.15
3715	13781722635	13526384957	2006-08-18 09:02:11	34	0.00
7625	13781722635	18502938475	2006-08-23 09:24:21	87	0.00
…	…	…	…	…	…

The result in A2 is the same as A1. When you use the select function in a cursor, the returned result is itself. When calculating A2, you do not actually get data from the cursor, but attach operations on the cursor. When fetch is executed in A3, the data is retrieved in A1 cursor according to the condition.

16.6 Exercises

【Exercise 16.3】Count the number of calls per day in the call record table. The call record table file is record.txt:

SERIALNUMBER	OUTID	INID	CALLTIME	DURATION	CHARGE
1	13781722635	18635273625	2006-08-11 08:53:22	97	0.00
2	18572635487	18173625183	2006-08-11 09:01:03	108	0.15
3	18572635182	13526384957	2006-08-11 09:02:11	34	0.00
4	13782769374	18502938475	2006-08-11 09:24:21	87	0.00
5	13672394817	18591820385	2006-08-11 09:33:12	132	0.15
6	18638274625	18108273645	2006-08-11 09:35:37	876	0.00
7	18127364918	13582937465	2006-08-11 09:43:18	56	0.15
…	…	…	…	…	…

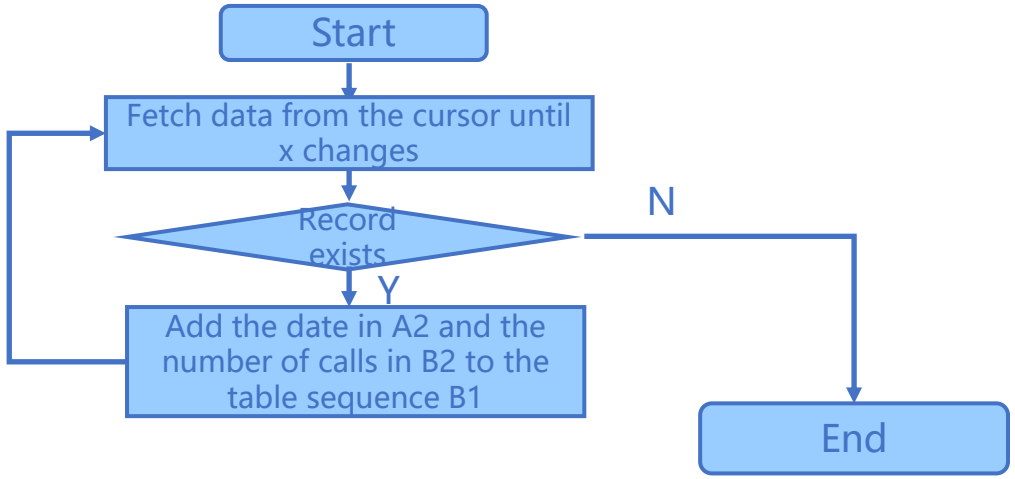
16.6 Exercises

【Exercise 16.3】Count the number of calls per day in the call record table. The call record table file is record.txt:

Solution

Use **for cs;x** statement to loop the data in the cursor, read out the data every time x changes, and calculate as required.

	A	B	C
1	=file("record.txt").cursor@t().run(CALLTIME=date(CALLTIME))	=create(Date, Count)	/A1 opens the file and creates a cursor, changes CALLTIME to date type, and B1 creates an empty table sequence
2	for A1;CALLTIME	=A2.count()	/A2 fetches the data in loop by date, and B2 counts the number of calls on that day
3		>B1.insert(0, A2.CALLTIME, B2)	/Add the date in A2 and the number of calls in B2 to the table sequence B1



B1

Date	Count
2006-08-11	621
2006-08-12	612
2006-08-13	598
...	...

16.6 Exercises

【Exercise 16.4】Calculate the number of calls, the average call duration and the total call charges per user in the call record table.

Call record table file is record.txt :

SERIALNUMBER	OUTID	INID	CALLTIME	DURATION	CHARGE
1	13781722635	18635273625	2006-08-11 08:53:22	97	0.00
2	18572635487	18173625183	2006-08-11 09:01:03	108	0.15
3	18572635182	13526384957	2006-08-11 09:02:11	34	0.00
4	13782769374	18502938475	2006-08-11 09:24:21	87	0.00
5	13672394817	18591820385	2006-08-11 09:33:12	132	0.15
6	18638274625	18108273645	2006-08-11 09:35:37	876	0.00
7	18127364918	13582937465	2006-08-11 09:43:18	56	0.15
…	…	…	…	…	…

16.6 Exercises

【Exercise 16.4】Calculate the number of calls, the average call duration and the total call charges per user in the call record table.

Solution

Use **cs.groups()** function to complete all kinds of aggregation operations.

	A	B
1	=file("record.txt").cursor@t()	/Open file and create a cursor
2	=A1.groups(OUTID;count(~):Count, avg(DURATION):AvgDuration, sum(CHARGE):Total)	/Calculate the number of calls, average call duration and total call charges by each user

A2:

OUTID	Count	AvgDuration	Total
13672394817	3	364	4.15
13782769374	7	672	9.55
13782769482	1	87	0.15
13782769674	9	1024	5.15
18127364983	6	781	3.45
18127364996	17	2038	9.95
18127365109	33	4095	13.45
…	…	…	…

16.6 Exercises

【Exercise 16.5】Employee table(emp.txt)and state information table(states.txt)

Requirements:

1. Merge the name and surname in the employee table into the full name;
2. Replace the state in the employee table with the abbreviation (abbr) according to the state information table;
3. According to the employee's birthday, add the calculated column: age.
4. Get the top ten records.

Part of the data in the two tables is as follows:

EID	NAME	SURNAME	GENDER	STATE	BIRTHDAY	HIREDATE	DEPT
1	Rebecca	Moore	F	California	1974/11/20	2005/3/11	R&D
2	Ashley	Wilson	F	New York	1980/7/19	2008/3/16	Finance
3	Rachel	Johnson	F	New Mexico	1970/12/17	2010/12/1	Sales
4	Emily	Smith	F	Texas	1985/3/7	2006/8/15	HR
5	Ashley	Smith	F	Texas	1975/5/13	2004/7/30	R&D
6	Matthew	Johnson	M	California	1984/7/7	2005/7/7	Sales
7	Alexis	Smith	F	Illinois	1972/8/16	2002/8/16	Sales
8	Megan	Wilson	F	California	1979/4/19	1984/4/19	Marketing
9	Victoria	Davis	F	Texas	1983/12/7	2009/12/7	HR
10	Ryan	Johnson	M	Pennsylvania	1976/3/12	2006/3/12	R&D
...	...	...	...	...	...	...	...

Employee table(emp.txt)

STATEID	NAME	ABBR
1	Alabama	AL
2	Alaska	AK
3	Arizona	AZ
4	Arkansas	AR
5	California	CA
6	Colorado	CO
7	Connecticut	CT
8	Delaware	DE
9	Florida	FL
10	Georgia	GA
...	...	...

State information table(states.txt)

16.6 Exercises

Solution

Use the cursor function **cs.new()** to finish the tasks.

	A	B
1	=file("emp.txt").cursor@t(EID, NAME, SURNAME, GENDER, STATE, BIRTHDAY)	/Select specified fields from the file and create a cursor
2	=file("states.txt").import@t()	/Read state information
3	=A1.new(EID, NAME+" "+SURNAME:FullName, GENDER, A2.select@1(NAME==STATE).ABBR:STATE, BIRTHDAY, age(BIRTHDAY):Age)	/Merge the name and surname into the full name, replace the state field with the abbreviation, and calculate the age by birthday
4	=A1.fetch(10)	/Fetch the first 10 records
5	>A1.close()	/Close the cursor

A4:

Index	EID	FullName	GENDER	STATE	BIRTHDAY	Age
1	1	Rebecca Moore	F	CA	1974-11-20	45
2	2	Ashley Wilson	F	NY	1980-07-19	39
3	3	Rachel Johnson	F	NM	1970-12-17	49
4	4	Emily Smith	F	TX	1985-03-07	35
5	5	Ashley Smith	F	TX	1975-05-13	45
6	6	Matthew Johnson	M	CA	1984-07-07	35
7	7	Alexis Smith	F	IL	1972-08-16	47
8	8	Megan Wilson	F	CA	1979-04-19	41
9	9	Victoria Davis	F	TX	1983-12-07	36
10	10	Ryan Johnson	M	PA	1976-03-12	44

16.6 Exercises

【Exercise 16.6】Count the number of order records of each priority level every day, and output the high priority statistics results that the number of high priority records is greater than the number of low priority records of the day to HIGH.txt.

The partial data of each priority file is as follows:

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
3271	33058	125535.00	1992/1/1	1-URGENT
176768	112850	378570.00	1992/1/1	1-URGENT
201414	11642	170608.00	1992/1/1	1-URGENT
228804	96325	138224.00	1992/1/1	1-URGENT
...	...	...	...	...

1-URGENT.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
23010	91141	97372.00	1992/1/1	2-HIGH
27137	54109	280493.00	1992/1/1	2-HIGH
175975	7511	285691.00	1992/1/1	2-HIGH
249668	115781	39757.00	1992/1/1	2-HIGH
...	...	...	...	...

2-HIGH.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
37543	124229	178835.23	1992/1/1	3-MEDIUM
88199	36158	268020.49	1992/1/1	3-MEDIUM
115585	143335	201582.36	1992/1/1	3-MEDIUM
126753	136505	207936.02	1992/1/1	3-MEDIUM
...	...	...	...	...

3-MEDIUM.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
5607	91021	37356.47	1992/1/1	4-NOT SPECIFIED
20742	136801	121343.42	1992/1/1	4-NOT SPECIFIED
45697	109417	345167.47	1992/1/1	4-NOT SPECIFIED
159266	99056	125324.77	1992/1/1	4-NOT SPECIFIED
...	...	...	...	...

4-NOT SPECIFIED.txt

O_ORDERKEY	O_CUSTKEY	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY
27015	10954	41753.82	1992/1/1	5-LOW
59718	24749	83608.04	1992/1/1	5-LOW
139655	25249	118709.27	1992/1/1	5-LOW
190656	23695	154110.00	1992/1/1	5-LOW
...	...	...	...	...

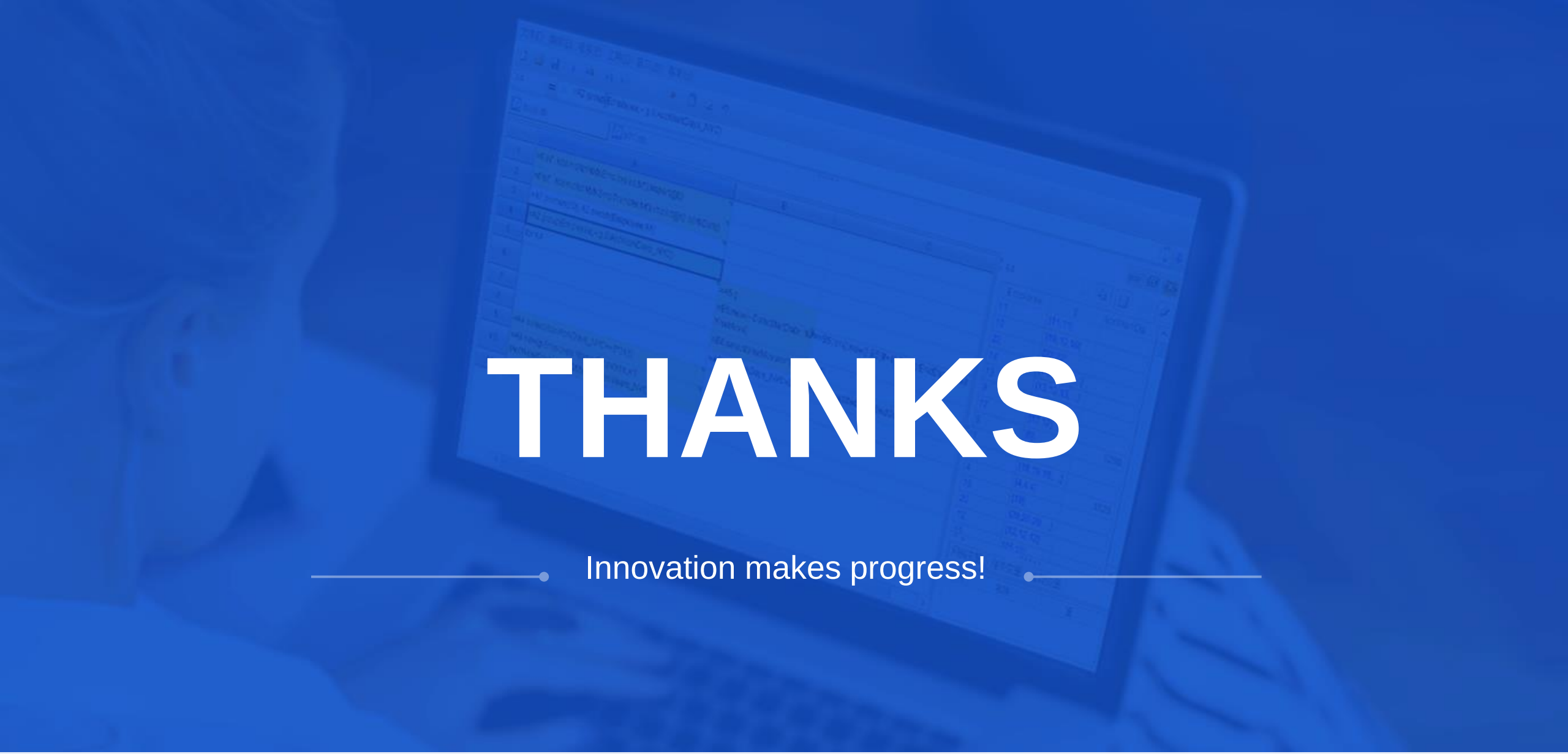
5-LOW.txt

16.6 Exercises

Solution

Use **joinx()** function to join multi ordered cursors; Use cursor function **cs.select()** to filter the result.

	A	B
1	=file("1-URGENT.txt").cursor@t()	/Open file and create a cursor
2	=file("2-HIGH.txt").cursor@t()	/Open file and create a cursor
3	=file("3-MEDIUM.txt").cursor@t()	/Open file and create a cursor
4	=file("4-NOT SPECIFIED.txt").cursor@t()	/Open file and create a cursor
5	=file("5-LOW.txt").cursor@t()	/Open file and create a cursor
6	=A1.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count the number of orders
7	=A2.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count the number of orders
8	=A3.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count the number of orders
9	=A4.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count the number of orders
10	=A5.groupx(O_ORDERDATE, O_ORDERPRIORITY;count(~):count)	/Group by order date, priority and count the number of orders
11	=joinx@1(A6:URGENT,O_ORDERDATE;A7:HIGH,O_ORDERDATE;A8:MEDIUM,O_ORDERDATE;A9:NOT_SPECIFIED,O_ORDERDATE;A10:LOW,O_ORDERDATE)	/Join the aggregate results of each priority by order date
12	=A11.select(HIGH.count>LOW.count)	/Filter results with more high priority records than low priority records
13	=file("HIGH.txt").export@t(A12.(HIGH))	/Output the result with high priority in the result set meeting the requirements
14	>A11.close()	/Close the cursor



THANKS

Innovation makes progress!

www.raqsoft.com