C/C++ Programming Language

CS205 Spring

Feng Zheng

2019.03.21

南方科技大学

SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Content

- Brief Review
- More About Loops
- Branching Statements
- Logical Expressions
- Summary

Brief Review

Content of Last Class

- Pointers
  - Address of array
  - new and delete operations
- Managing memory for data
  - Automatic memory
  - Dynamic memory
  - Static memory
- Loops
  - for

Review of The Address of an Array

- Address of an Array
  - short tell[10];
  - tell is type pointer-to-short
  - &tell is type pointer-to-array of 10 shorts
  - short (*pas)[10] = &tell;
  - (*pas) = tell is type pointer-to-short
  - pas = &tell is type pointer-to-array of 10 shorts
  - short* pas[10];
  - pas is an array of 10 pointers-to-short (short *)

➢ **&tell**

More About Loops

More for Increment/Decrement Operators

- Prefixing versus postfixing: ++x, x++, --x, x--
  - Prefix form is more efficient
- The increment/decrement operators and pointers
  - Adding an increment operator to a pointer increases its value by the number of bytes in the type it points to
  - The prefix increment, prefix decrement, and dereferencing operators have the same precedence (from right to left)
  - Postfix increment and decrement operators have the same precedence, which is higher than the prefix precedence(from left to right)
- See program example 7

And More for Loops

- Combination assignment operators
  - Example: combined addition and assignment operator

Operator	Effect (L=left operand, R=right operand)
+=	Assigns L + R to L
-=	Assigns L - R to L
*=	Assigns L * R to L
/=	Assigns L / R to L
%=	Assigns L % R to L

- Compound statements, or blocks: {}
  - Program example 8
- More syntax tricks—the comma operator
- ```
  int i, j;  // comma is a separator here, not an operator
  ++j, --i   // two expressions count as one for syntax purposes
  ```
  ```

Relational Expressions

- C++ provides six relational operators to compare numbers
  - Exclamation mark

Operator	Meaning
<	Is less than
<=	Is less than or equal to
==	Is equal to
>	Is greater than
>=	Is greater than or equal to
!=	Is not equal to

Comparisons in Test Expression

- Program example 9
  - A mistake you'll probably make
  - = or ==
- Program example 10
  - Comparing C-style strings
  - strcmp(str1,str2)
- Program example 11
  - Comparing string class strings
  - Using relational symbol (!=)

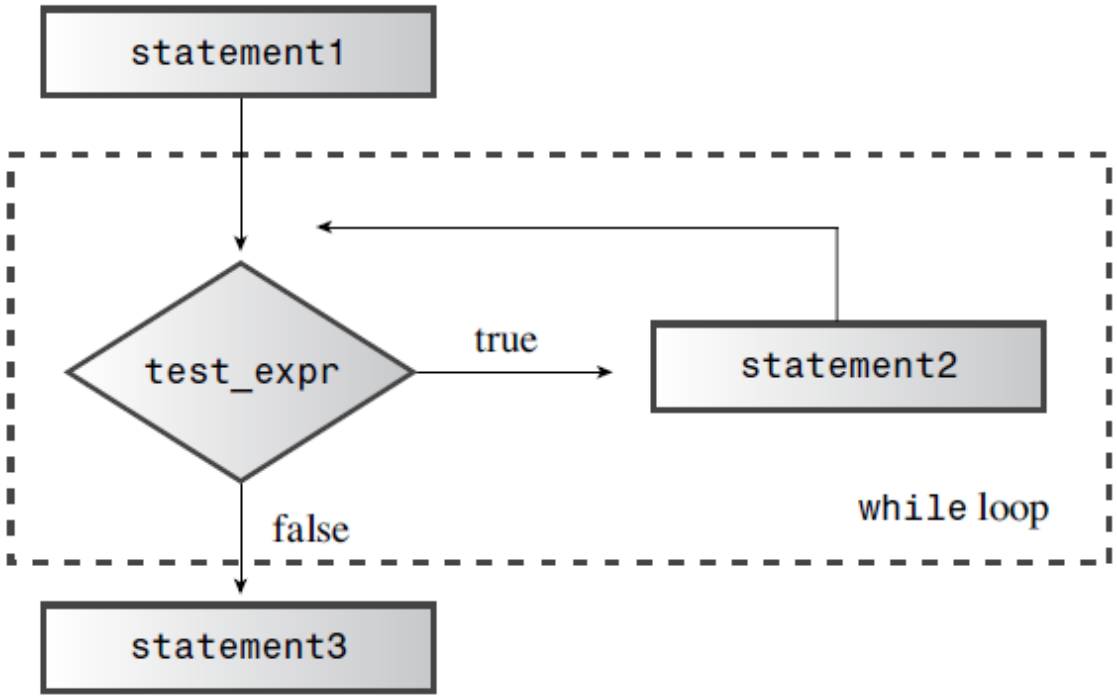
The while Loop

- while is entry-condition loop
- It has just a test condition and a body
  - Do something to affect the test-condition expression
- See Program example 12
  - Two types of condition expression

while (name[i] != '\0')

while (name[i])

```
statement1
while (test_expr)
      statement2
statement3
```



for Versus while

- In C++ the for and while loops are essentially equivalent

```
for (init-expression; test-expression; update-expression)
{
    statement(s)
}
```

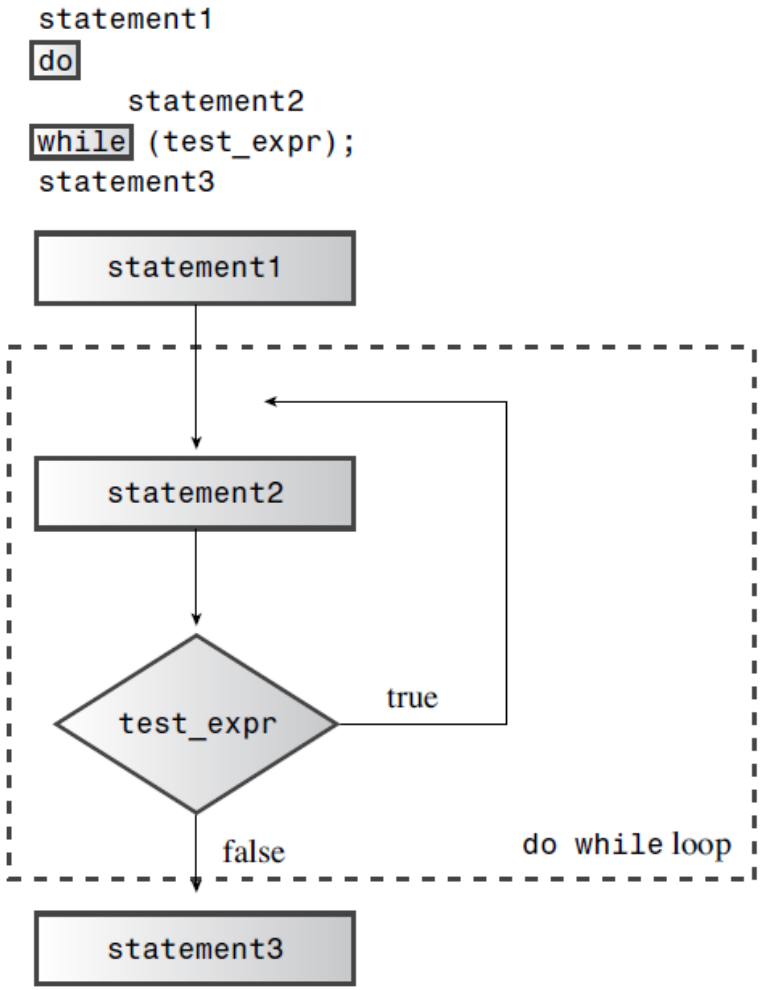
```
init-expression;
while (test-expression)
{
    statement(s)
    update-expression;
}
```

```
while (test-expression)
    body
```

```
for ( ;test-expression;)
    body
```

More Loops

- The do while Loop
  - It's an exit-condition loop
  - Such a loop always executes at least once
  - See Program example 13
- The range-based for loop (C++11)
  - See Program example 14
    - ✓ Colon symbol :
    - ✓ & symbol: reference variable
    - ✓ To modify the array contents



Example: Loops and Text Input

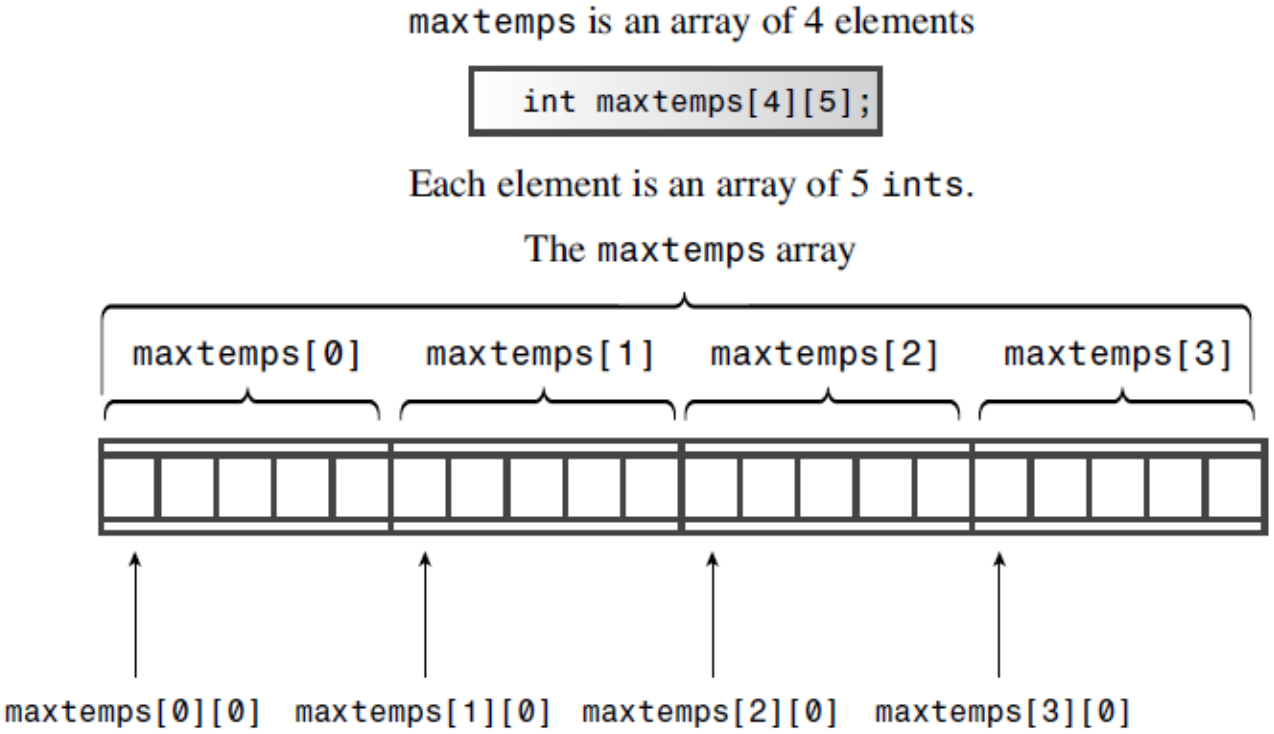
- Using unadorned cin for input
  - When to stop?
    - ✓ A sentinel character
  - See program example 15
    - ✓ The program omit the spaces
    - ✓ Program and operating system both work
- cin.get(char) to the rescue
  - See program example 16
    - ✓ Read the space
    - ✓ Declare the argument as a reference

Example: Nested Loops and Two-Dimensional Arrays

- Example:

int maxtemps[4][5];

- See program example 17



The maxtemps array viewed as a table:

		0	1	2	3	4
maxtemps[0]	0	maxtemps[0][0]	maxtemps[0][1]	maxtemps[0][2]	maxtemps[0][3]	maxtemps[0][4]
maxtemps[1]	1	maxtemps[1][0]	maxtemps[1][1]	maxtemps[1][2]	maxtemps[1][3]	maxtemps[1][4]
maxtemps[2]	2	maxtemps[2][0]	maxtemps[2][1]	maxtemps[2][2]	maxtemps[2][3]	maxtemps[2][4]
maxtemps[3]	3	maxtemps[3][0]	maxtemps[3][1]	maxtemps[3][2]	maxtemps[3][3]	maxtemps[3][4]

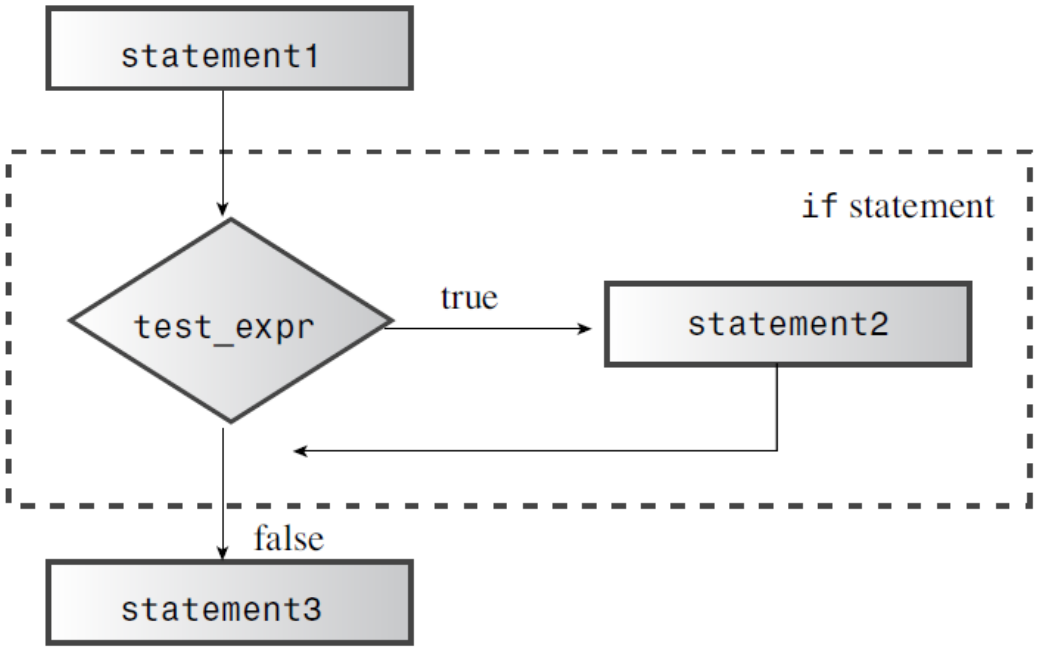
Branching Statements

The **if** Statement

- One of the keys to designing intelligent programs is to give them the ability to make decisions
  - Looping
  - if statement

- See program example 1

```
statement1
if (test_expr)
      statement2
statement3
```

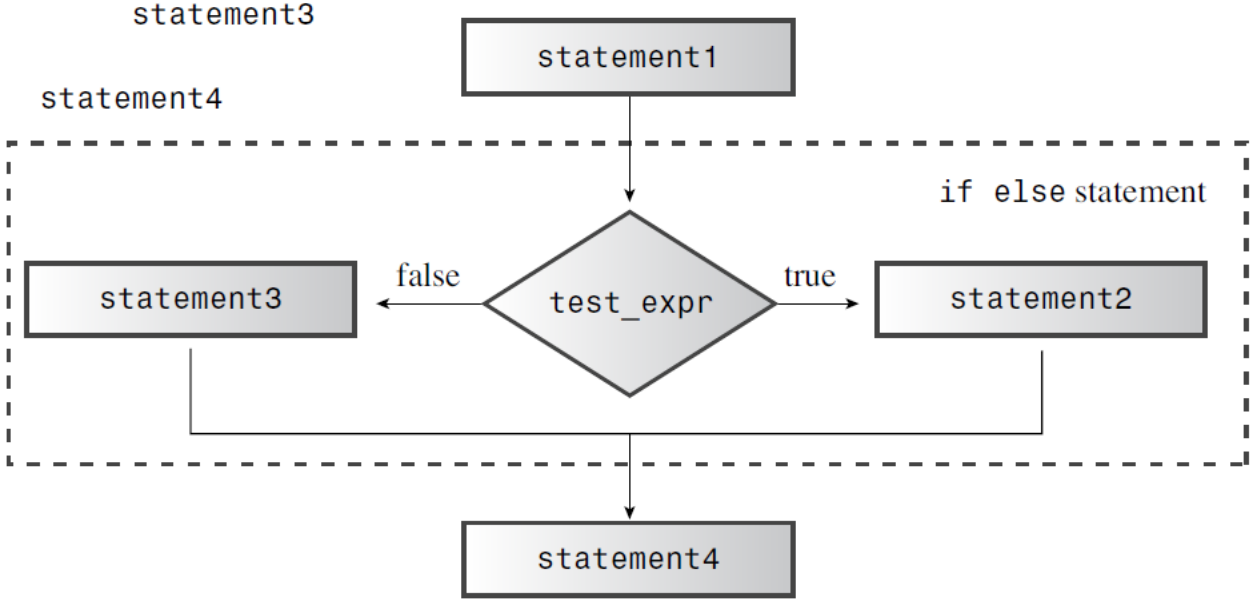


More than one selections

- The if else Statement
  - Decide which of two statements or blocks is executed
  - Must use braces to collect statements into a single block
  - Remember the conditional compilation #if, #else
- The if else if else Construction
- See program example 2

```
statement1
if (test_expr)
      statement2
else
      statement3

statement4
```



Logical Expressions

The Logical OR Operator: ||

- Three operators
  - Logical OR, written ||
  - Logical AND, written &&
  - Logical NOT, written !

- The logical OR operator: ||
  - || has a lower precedence than the relational operators
  - The || operator is a sequence point
  - C++ won't bother evaluating the expression on the right if the expression on the left is true

The Value of `expr1 || expr2`		
	`expr1 == true`	`expr1 == false`
`expr2 == true`	`true`	`true`
`expr2 == false`	`true`	`false`

AND Operator: && NOT Operator: !

- AND Operator
  - Lower precedence than the relational operators
  - Acts as a sequence point
  - C++ doesn't bother evaluating the right side in some cases
- See program example 3
- NOT Operator
  - Exclamation point
  - If expression is true, or nonzero, then !expression is false
  - If expression is false, then !expression is true

The Value of `expr1 && expr2`

	`expr1 == true`	`expr1 == false`
`expr2 == true`	true	false
`expr2 == false`	false	false

Logical Operator Facts

- Precedence
  - The NOT(!) operator has a higher precedence than any of the relational or arithmetic operators
  - The AND operator has a higher precedence than the OR operator
  - Use parentheses to tell the program the interpretation you want
- Alternative Representations

Operator	Alternative Representation
&&	and
||	or
!	not

- The cctype library of character functions
  - A handy package of character-related functions

The ?: Operator

- Conditional operator
  - More concise

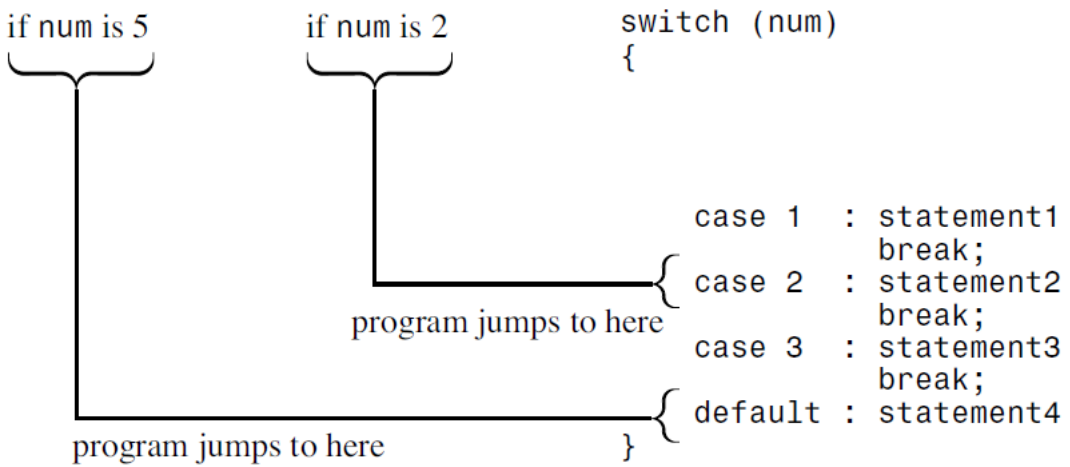
```
int c;
if (a > b)
    c = a;
else
    c = b;
```

```
int c = a > b ? a : b;
```

The switch Statement

- Acts as a routing device that tells the computer which line of code to execute next
- You must use the break

```
switch (integer-expression)
{
      case label1 : statement(s)
      case label2 : statement(s)
     ...
      default     : statement(s)
}
```



- See program example 4

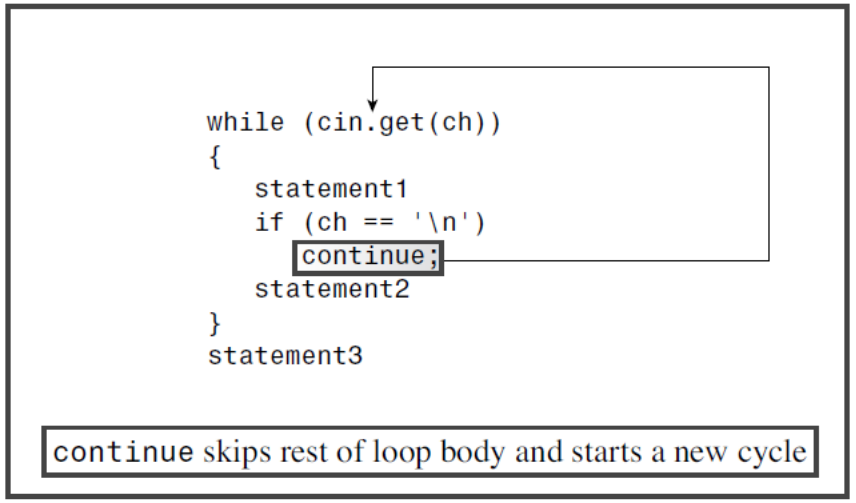
More About switch

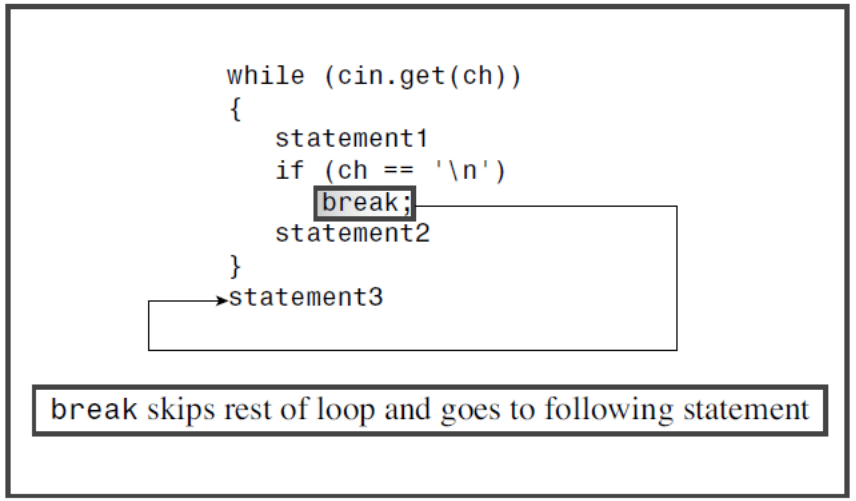
- Using enumerators as labels
  - See program example 5

- switch and if else
  - Let a program select from a list of alternatives
  - A switch statement isn't designed to handle ranges
  - Each switch case label must be a single value
  - Also that value must be an integer
  - A switch statement can't handle floating-point tests

The break and continue Statements

- The break and continue statements enable a program to skip over parts of the code
  - break causes program execution to pass to the next statement following the switch or the loop
  - continue statement is used in loops and causes a program to skip the rest of the body of the loop and then start a new loop cycle
- See program example 6





Example: Number-Reading Loops

- What happens if the user responds by entering a word instead of a number?

```
int n;
cin >> n;
```

- See program example 7
  - The preceding example doesn't attempt to read any input after non-numeric input
- See program example 8

Simple File Output

- Main steps for using file output
  - Include the fstream header file
  - Create an ofstream object
  - Associate the ofstream object with a file (C-style) using open()
  - Use the ofstream object in the same manner you would use cout
  - Use the close() method to close the file

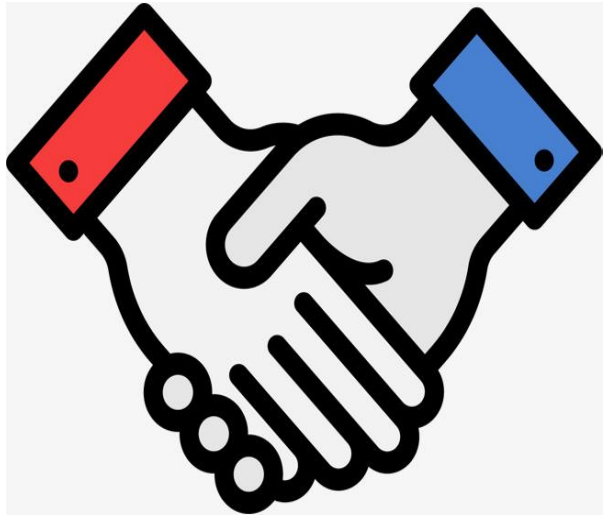
- See program example 9

Simple File Input

- Main steps for using file input
  - Include the fstream header file and account for the std
  - Declare one or more ifstream variables, or objects
  - Associate a ifstream object with a file using open()
  - Use the close() method to close the file
  - Use >> operator, get(), getline(), …… method
- See program example 10
  - What happens if you attempt to open a non-existent file for input?
  - exit(EXIT_FAILURE);
  - Communicate with the operating system
  - Terminate the program

Summary

- Loops
  - Increment/decrement operators: ++; --
  - Rational expressions: 6
  - for, while, do while
- Branch statements
  - if; if else; if else if else; switch
- The Logical Operator
  - OR, AND, NOT
- Jump operations
  - break and continue
- File fstream
  - Simple File Output: ofstream
  - Simple File Input: ifstream



Thanks

zhengf@sustech.edu.cn