哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

# 高级语言程序设计
# High-level Language Programming

## **Lecture 4** Simple IO

Yitian Shao (shaoyitian@hit.edu.cn)
School of Computer Science and Technology

# Simple IO

*Course Overview*

- 4.1 Simple input and output
- 4.2 Manipulator operators
- 4.3 Single-character input and output
- 4.4 Different types of function output

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

- Data Stream
  - Data stream objects are used to perform basic input and output of data to and from various devices such as the keyboard and the screen.
  - A stream is a data communication object connected to an input or output device.
- cout: standard output stream associated with SCREEN
  - <<:The insertion operator is used to write data to cout.
- cin :standard output stream associated with KEYBOARD
  - >>: The extraction operator is used to read data from keyboad.

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

```
cout << "Output sentence"; // prints Output sentence on screen
cout << 120;               // prints number 120 on screen
cout << x;                 // prints the value of x on screen
```

- The << operator inserts the data that follows it into the stream that precedes it.
- In the examples above, it inserted the literal *string* Output sentence, the *number 120*, and the value of *variable x* into the standard output stream cout.
- Notice that the sentence in the first statement is enclosed in double quotes (") because it is a string literal, while in the last one, x is not.

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

- The double quoting is what makes the difference; when the text is enclosed between them, the text is printed literally;
- When they are not, the text is interpreted as the identifier of a variable, and its value is printed instead.

For example, these two sentences have very different results:

```
cout << "Hello";  // prints Hello
cout << Hello;    // prints the content of variable Hello
```

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

- Multiple insertion operations (<<) may be chained in a single statement:

```
cout << "This " << " is a " << "single C++ statement";
```

This statement would print the text “This is a single C++” statement

- Chaining insertions is especially useful to mix literals and variables in a single statement:

```
cout << "I am " << age << " years old and my zipcode is " << zipcode;
```

If *age* = 24 and the *zipcode* = 90064, the output would be:

```
I am 24 years old and my zipcode is 90064
```

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

- What cout does not do automatically is add line breaks at the end, unless instructed to do so.

  For example, take the following two statements inserting into cout:

```
cout << "This is a sentence.";
cout << "This is another sentence.";
```

  The output would :

```
This is a sentence.This is another sentence.
```

  If newline mark is required, the statements should be:

```
cout<<"This is a sentence." << "\n";
cout<<"This is another sentence.";
```

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

- Alternatively, the endl manipulator can also be used to break lines.

For example:

```
cout<<"This is a sentence." << endl;
cout<<"This is another sentence.";
```

This would print:

```
This is a sentence.
This is another sentence.
```

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

- In most program environments, the standard input by default is the keyboard, and the C++ stream object defined to access it is cin.
- For formatted input operations, cin is used together with the extraction operator, which is written as >> (i.e., two "greater than" signs).
- This operator is then followed by the variable where the extracted data is stored.

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

```
int age;
cin >> age;
```

- The first statement declares a variable of type int called age, and the second extracts from cin a value to be stored in it.
- This operation makes the program wait for input from cin; generally, this means that the program will wait for the user to enter some sequence with the keyboard.
- The keyboard are only transmitted to the program when the ENTER (or RETURN) key is pressed.

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

- The extraction operation on **cin** uses the type of the variable after the >> operator to determine how it interprets the characters read from the input

If it is an integer, the format expected is a series of digits, if a string a sequence of characters, etc.

```
// i/o example

#include <iostream>
using namespace std;

int main ()
{
  int i;
  cout << "Please enter an integer value: ";
  cin >> i;
  cout << "The value you entered is " << i;
  cout << " and its double is " << i*2 << ".\n";
  return 0;
}
```

```
Please enter an integer value: 702
The value you entered is 702 and its double is 1404.
```

Try it yourself on https://www.onlinegdb.com

# 4.1 Simple Keyboard Input & Output

- Extractions on cin can also be chained to request more than one datum in a single statement:

```
cin >> a >> b;
```

- This is equivalent to:

```
cin >> a;
cin >> b;
```

- In both cases, the user is expected to introduce two values, one for variable a, and another for variable b.

Try it yourself on https://www.onlinegdb.com

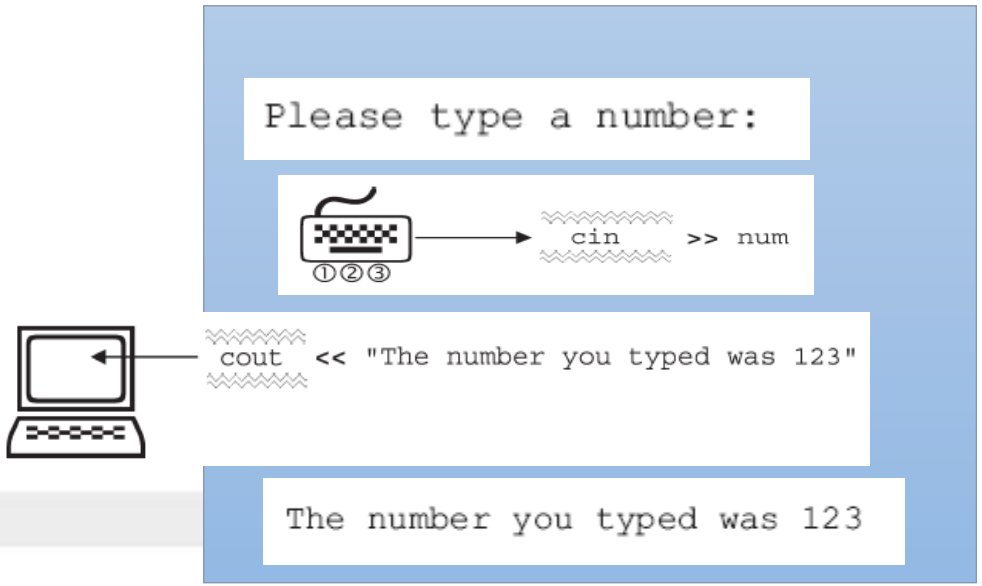
# 4.1 Simple Keyboard Input & Output

Example:
Read a number from the keyboard and store it in the variable num.

```
#include <iostream>
using namespace std ;

int main()
{
    int num ;

    cout << "Please type a number" ;
    cin >> num ;
    cout << "The number you typed was" << num << endl ;
}
```




Try it yourself on https://www.onlinegdb.com

# 4.2 Manipulators

- Manipulators are used to modify input and output data streams.
- Typical manipulators: endl, setw, setfill, fixed, setprecision
  - endl：skip to the start of a new line on the screen

```
cout << endl << endl << “endl can be used anywhere” << endl;
```

  - setw：set the width of a data field which is the number of columns that the data item occupies on the screen;
  - setfill：change the “padding” character from a space to any other character.

Try it yourself on https://www.onlinegdb.com

# 4.2 Manipulators: setw

- Example: How to use manipulators

#include <iomanip>

It is required for any manipulator, like setw, that has a value in parentheses.

```
#include <iostream>
#include <iomanip>
using namespace std ;

int main()
{
    int num1 = 123, num2 = 4567 ;

    cout << "Without setw:" << endl ;
    cout << num1 << num2 << endl ;
    cout << setw(4) << num1 << setw(7) << num2 << endl ;
}
```

```
Without setw:
1234567
With setw:
 123    4567
```

If the field width is set too small to display a value, the width is automatically expanded so that all the digits in the value are displayed

Try it yourself on https://www.onlinegdb.com

# 4.2 Manipulators: setfill

- Example: How to use manipulators

```
#include <iostream>
#include <iomanip>
using namespace std ;

int main()
{
    double num = 123.456 ;

    cout << setw(9) << setfill('*') << num << endl ;
    cout << setw(9) << setfill('0') << num << endl ;
    cout << setw(10) << num << endl ;
}
```

```
**123.456
00123.456
000123.456
```

Unlike *setw* ,which applies only to the next data item in the output stream, the *setfill* manipulator remains in effect for all subsequent the fill character remains 0.

Try it yourself on https://www.onlinegdb.com

# 4.2 Manipulators: setprecision

- Example: How to use manipulators

```
#include <iostream>
#include <iomanip>
using namespace std ;

int main()
{
    double num = 123.45678 ;

    cout << num << endl ;
    cout << setprecision(7) << num << endl ;
    cout << fixed <<  setprecision(2) << num << endl ;
}
```

```
123.457
123.4568
123.46
```

- setprecision： the *total* digits to reserve *before and after* the decimal point (default is 6)
- With fixed preceding, it sets the digits to reserve *after* the decimal point
- Both fixed and setprecision remain in effect for subsequent insertions into the output stream

Try it yourself on https://www.onlinegdb.com

# 4.3 Single-character input and output

- Character Input and Output
  - The following code segment inputs a character from the keyboard to the variable ch, ignoring whitespace characters.

```
char ch;
cin >> ch ; // Reads the next character,
            // whitespace characters are ignored
```

  - Whitespace: space, tab, newline, enter, etc.

Try it yourself on https://www.onlinegdb.com

# 4.3 Single-character input and output

- Example
  - cin >> ch

main.cpp

```
1  /*************************************************************
2  Program Example
3  How to achieve single-char input & output
4  *************************************************************
5  #include <iostream>
6
7  int main() {
8      char ch;
9
10     std::cout << "Please enter a character: ";
11     std::cin >> ch; // Reads the next character,
12                     // whitespace characters are ignored
13
14     std::cout << "You entered: " << ch << std::endl;
15
16     return 0;
17 }
18
```

Please enter a character:
a
You entered: a

Try it yourself on https://www.onlinegdb.com

# 4.3 Single-character input and output

- Character Input and Output
  - Using the manipulator ***noskipws***.(no skip whitespaces).

```
std::cout << "Please enter a character: ";
std::cin >> std::noskipws >> ch; // Reads the next character,
                                 // a whitespace character
                                 // may be read

std::cout << "You entered: " << ch << std::endl;
```

```
Please enter a character:
        a
You entered:
```

Try it yourself on https://www.onlinegdb.com

# 4.3 Single-character input and output

- Character Input and Output
  - Alternatively, the function *get( )* associated with the *cin* can be used.

```
#include <iostream>

int main() {
    char ch;

    std::cout << "Enter a string of characters including whitespaces: ";

    // Read characters until a newline character is encountered
    while (std::cin.get(ch) && ch != '\n') {
        // Output each character
        std::cout << ch;
    }

    std::cout << std::endl;

    return 0;
}
```

Enter a string of characters including whitespaces: a b c d
a b c d

Try it yourself on https://www.onlinegdb.com

# 4.3 Single-character input and output

- Character Input and Output
  - Similarly,  the output stream object cout has a member function *put( )* that can be used to display a character.

```
#include <iostream>

using namespace std;

int main() {
    char ch = 'A'; // Assuming the character to be displayed is 'A'

    cout << "Displaying the character: ";
    cout.put(ch); // Display the character ch

    return 0;
}
```

Try it yourself on https://www.onlinegdb.com

# 4.4 Different types of function output

- Example: output of type int

```cpp
#include <iostream>

// Class definition for Calculator
class Calculator {
public:
    // add function: Takes two integers and returns their sum
    int add(int num1, int num2) {
        return num1 + num2;
    }
};

// Main function
int main() {
    // Create a Calculator object
    Calculator calc;

    // Call the add function and output the result
    int result = calc.add(10, 20);
    std::cout << "The sum is: " << result << std::endl;

    return 0;
}
```

Try it yourself on https://www.onlinegdb.com

# 4.4 Different types of function output

- Example: output of type float

```cpp
#include <iostream>
using namespace std;

// Function to calculate the average of two float numbers
float calculateAverage(float num1, float num2) {
    return (num1 + num2) / 2.0;
}

int main() {
    float a = 5.5, b = 7.5;

    // Call the function and store the result
    float average = calculateAverage(a, b);

    // Output the result
    cout << "The average of " << a << " and " << b << " is: " << average << endl;

    return 0;
}
```

Try it yourself on https://www.onlinegdb.com

# 4.4 Different types of function output

- Example: output of type double

```cpp
#include <iostream>

// Function declaration
double multiply(double num1, double num2);

int main() {

    // Directly assigning two numbers
    double number1 = 9.69, number2 = 7.86;

    // Calling the multiplication function and outputting the result
    double product = multiply(number1, number2);
    std::cout << "The product of the two numbers is: " << product << std::endl;

    return 0;
}

// Function definition: Calculate the product of two numbers
double multiply(double num1, double num2) {
    return num1 * num2;
}
```

Try it yourself on https://www.onlinegdb.com

# HOMEWORK

# Homework 4

- 1. Write a program to input four numbers and display them in reverse order.

- 2.Write a program to input three floating-point numbers from the keyboard and to calculate (a) their sum and (b) their average. Display the results to three decimal places.

Try it yourself on https://www.onlinegdb.com

# Homework 4

- 3.Write a program to accept a temperature in degrees Fahrenheit and convert it to degrees Celsius. Your program should display the following prompt:

  Enter a temperature in degrees Fahrenheit:

  You will then enter a decimal number followed by the Enter key. The program will then convert the temperature by using the formula

  Celsius = (Fahrenheit - 32.0 ) * (5.0 / 9.0)

  Your program should then display the temperature:

  The temperature in degrees Celsius is:

Try it yourself on https://www.onlinegdb.com