# C/C++ Program Design

LAB 2

# CONTENTS

❑ Objectives

❑ Knowledge points

❑ Exercises

# 1 Objectives

- Master Fundamental Data types

- Master Arithmetic Operators and Assignment Operators
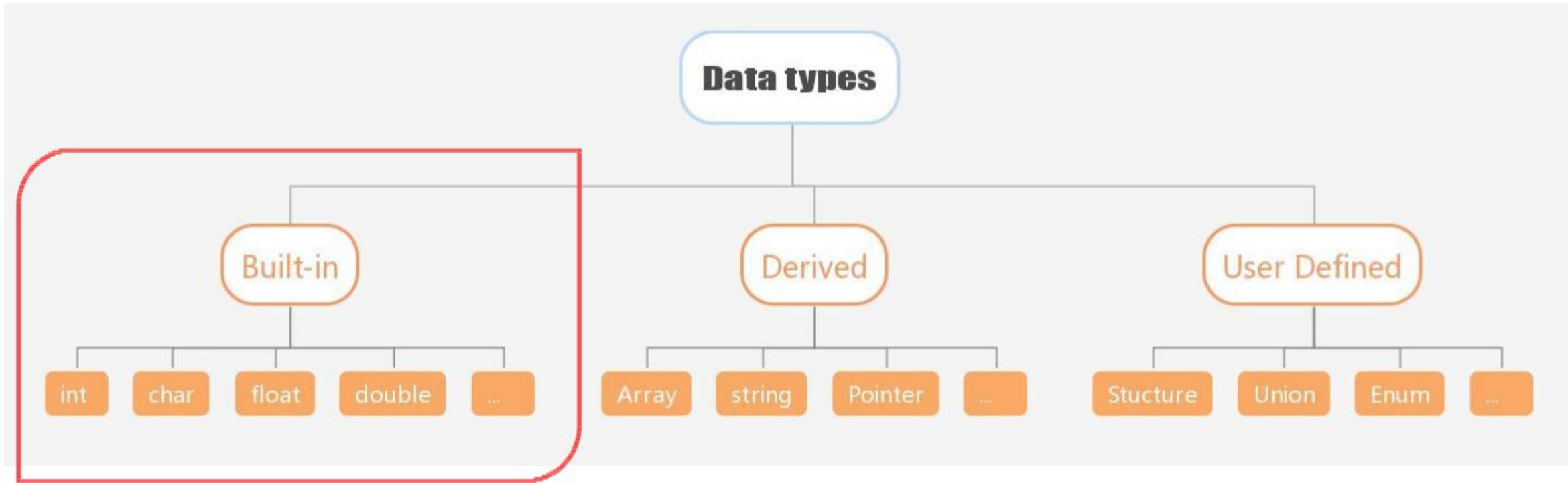
- Master Keyboard Input and Terminal Output

# 2 Knowledge Points

2.1  Fundamental Data Types

2.2  Arithmetic Operators and Assignment Operators

2.3  Input and Output

# 2.1 Data types

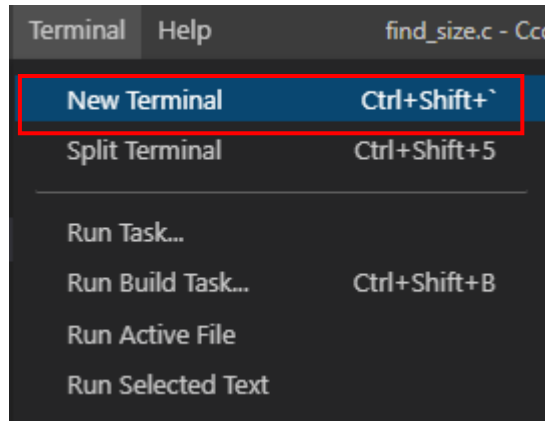**sizeof** operator returns the size, in bytes, of a type or a variable.

```c
lab02_examples > C find_size.c > ...
 1    #include <stdio.h>
 2    #include <stdbool.h>
 3
 4    int main()
 5    {
 6        printf("\n\nFind Size of the fundamental data typ         ");
 7        printf("------------------------------------        -----\n");
 8        printf("The sizeof(char) is:         %ld bytes\n", sizeof(char));
 9        printf("The sizeof(short) is:        %ld bytes\n", sizeof(short));
10        printf("The sizeof(int) is:          %ld bytes\n", sizeof(int));
11        printf("The sizeof(long) is:         %ld bytes\n", sizeof(long));
12        printf("The sizeof(long long) is:    %ld bytes\n", sizeof(long long));
13        printf("The sizeof(float) is:        %ld bytes\n", sizeof(float));
14        printf("The sizeof(double) is:       %ld bytes\n", sizeof(double));
15        printf("The sizeof(long double) is: %ld bytes\n", sizeof(long double));
16        printf("The sizeof(bool) is:         %ld bytes\n", sizeof(bool));
17
18        return 0;
19    }
```

If you use %d, the compiler will give a warning.

using the **sizeof** operator with a type name, enclose the name in parentheses; while using the operator with the name of the variable, parentheses are optional.

```
find_size.c: In function 'main':
find_size.c:8:42: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long unsigned int' [-Wformat=]
    8 |     printf("The sizeof(char) is:       %d bytes\n", sizeof(char));
      |                                        ~^                ~~~~~~~~~~~~~
      |                                         |                 |
      |                                        int              long unsigned int
      |                                        %ld
```

open the Terminal window to input the commands: Terminal ->New Terminal

| Terminal | Help | | find_size.c - Cco |
| --- | --- | --- | --- |
| New Terminal | | Ctrl+Shift+` | |
| Split Terminal | | Ctrl+Shift+5 | |
| Run Task... | | | |
| Run Build Task... | | Ctrl+Shift+B | |
| Run Active File | | | |
| Run Selected Text | | | |

compile and link

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ gcc find_size.c -o find_size
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./find_size

Find Size of the fundamental data types:
-------------------------------------------
The sizeof(char) is:        1 bytes
The sizeof(short) is:       2 bytes
The sizeof(int) is:         4 bytes
The sizeof(long) is:        8 bytes
The sizeof(long long) is:   8 bytes
The sizeof(float) is:       4 bytes
The sizeof(double) is:      8 bytes
The sizeof(long double) is: 16 bytes
The sizeof(bool) is:        1 bytes
```

the output

# Example: Write a C++ program to find size of fundamental data types.

```
lab02_examples > C find_size.cpp > ...
1    #include <iostream>
2    using namespace std;
3
4    int main()
5    {
6        cout << "\n\nFind Size of the fundamental data types:\n";
7        cout << "------------------------------------------------\n";
8        cout << "The size of(char) is:        " << sizeof(char) << endl;
9        cout << "The size of(short) is:       " << sizeof(short) << endl;
10       cout << "The size of(int) is:         " << sizeof(int) << endl;
11       cout << "The size of(long) is:        " << sizeof(long) << endl;
12       cout << "The size of(long long) is:   " << sizeof(long long) << endl;
13       cout << "The size of(float) is:       " << sizeof(float) << endl;
14       cout << "The size of(double) is:      " << sizeof(double) << endl;
15       cout << "The size of(long double) is: " << sizeof(long double) << endl;
16       cout << "The size of(bool) is:        " << sizeof(bool) << endl;
17
18       return 0;
19   }
```

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ find_size.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ls
a.out  find_size.c  find_size.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
```

```
Find Size of the fundamental data types:
------------------------------------------------
The size of(char) is:        1
The size of(short) is:       2
The size of(int) is:         4
The size of(long) is:        8
The size of(long long) is:   8
The size of(float) is:       4
The size of(double) is:      8
The size of(long double) is: 16
The size of(bool) is:        1
```

# The value range of an integer

```
lab02_examples  > C integer_limit.cpp > main()
  1    #include <iostream>
  2    #include <climits>
  3
  4    int main()
  5    {
  6        using namespace std;
  7
  8        int n_int = INT_MAX; //max value for integer number
  9        short n_short = SHRT_MAX;
 10        long n_long = LONG_MAX;
 11        long long n_llong = LLONG_MAX;
 12
 13        cout << "int is " << sizeof (int) << " bytes." << endl;
 14        cout << "short is " << sizeof n_short << " bytes." << endl;
 15        cout << "long is " << sizeof n_long << " bytes." << endl;
 16        cout << "long long is " << sizeof n_llong << " bytes." << endl;
 17        cout << endl;
 18
 19        cout << "Maximum values: " << endl;
 20        cout << "int: " << n_int << endl;
 21        cout << "short: " << n_short << endl << "long: " << n_long << endl << "long long: " << n_llong << endl << endl;
 22
 23        cout << "Minimum int value = " << INT_MIN << endl;
 24        cout << "Bits per byte = " << CHAR_BIT << endl;
 25
 26        return 0;
 27    }
```

CHAR_BIT: the number of bits in a byte.

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ integer_limit.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
int is 4 bytes.
short is 2 bytes.
long is 8 bytes.
long long is 8 bytes.

Maximum values:
int: 2147483647
short: 32767
long: 9223372036854775807
long long: 9223372036854775807

Minimum int value = -2147483648
Bits per byte = 8
```
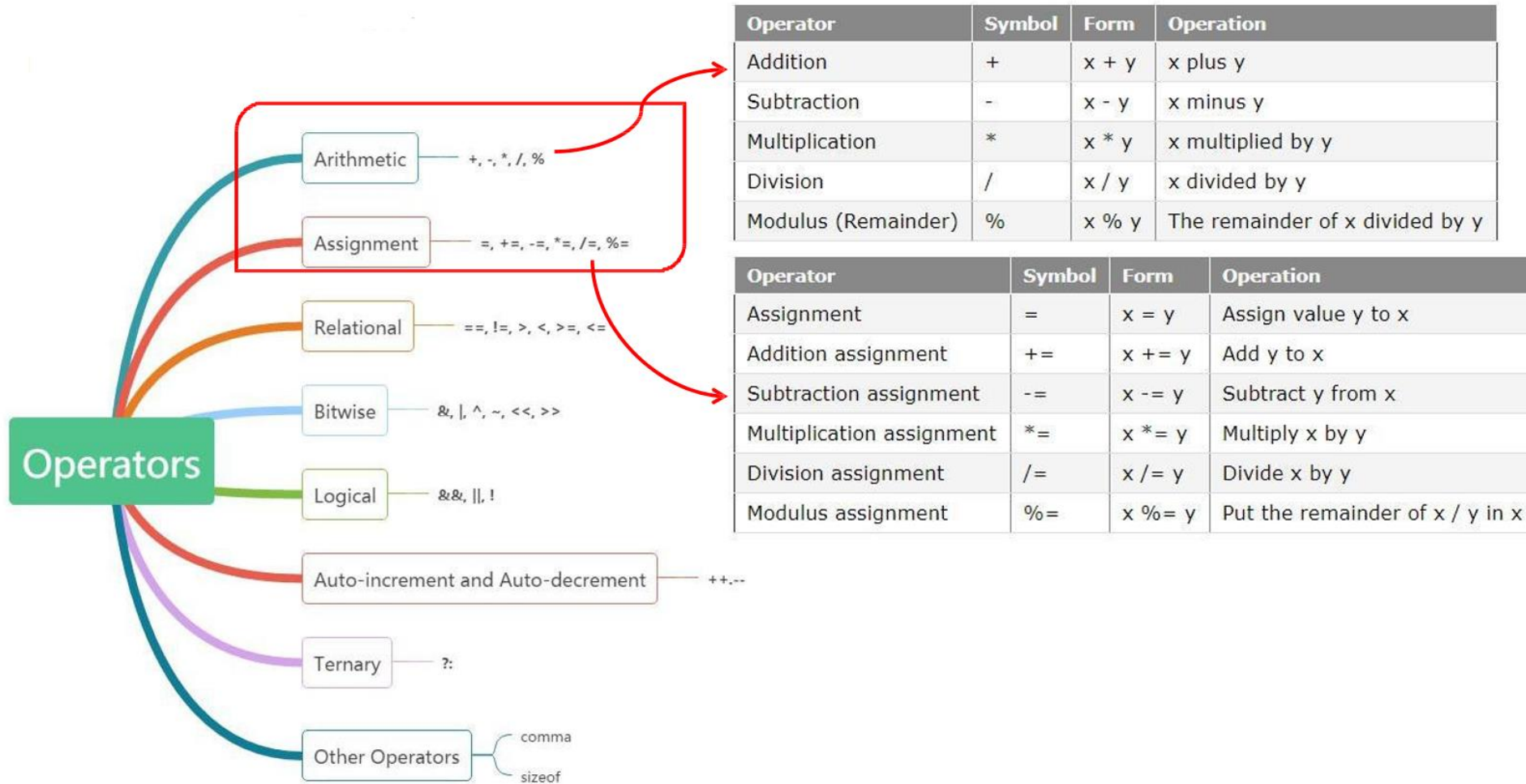
# Floating point precision

Floating-point type also has a range of values. Besides that, it has a significant figures. Normally, the system guarantees the 6 significant figures of type float variable, and 15 significant figure of the type double variable. Floating-point numbers have precision limitations when they are evaluated.

# 2.2 Arithmetic Operators

| Operator | Symbol | Form | Operation |
|---|---|---|---|
| Addition | + | x + y | x plus y |
| Subtraction | - | x - y | x minus y |
| Multiplication | * | x * y | x multiplied by y |
| Division | / | x / y | x divided by y |
| Modulus (Remainder) | % | x % y | The remainder of x divided by y |

| Operator | Symbol | Form | Operation |
|---|---|---|---|
| Assignment | = | x = y | Assign value y to x |
| Addition assignment | += | x += y | Add y to x |
| Subtraction assignment | -= | x -= y | Subtract y from x |
| Multiplication assignment | *= | x *= y | Multiply x by y |
| Division assignment | /= | x /= y | Divide x by y |
| Modulus assignment | %= | x %= y | Put the remainder of x / y in x |

**Operators**

- Arithmetic — +, -, *, /, %
- Assignment — =, +=, -=, *=, /=, %=
- Relational — ==, !=, >, <, >=, <=
- Bitwise — &, |, ^, ~, <<, >>
- Logical — &&, ||, !
- Auto-increment and Auto-decrement — ++,--
- Ternary — ?:
- Other Operators
  - comma
  - sizeof

# Example Program of Arithmetic Operators:

```
lab02_examples > G+ add.cpp > ⦿ main()
  1    #include <iostream>
  2
  3    using std::cout;
  4    using std::endl;
  5
  6    int main()
  7    {
  8        int a = 1234567890;
  9        int b = 1234567890;
 10        int sum = a+b;
 11
 12        cout<< a <<" + "<< b <<" = "<< sum <<endl;
 13
 14        return 0;
 15    }
```

Run the program in your terminal and see what happens:

```
maydlee@LAPTOP-U1MOON2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ add.cpp
maydlee@LAPTOP-U1MOON2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
1234567890 + 1234567890 = -1825831516
```

Why the result is negative? Can we use **unsigned int** or **long** type to solve the this problem?

```cpp
lab02_examples > G add_float.cpp > ...
 1    #include <iostream>
 2
 3    using std::cout;
 4    using std::endl;
 5
 6    int main()
 7    {
 8        float a = 1234567.0;
 9        float b = 1.0;
10        float sum = a+b;
11
12        cout<< a <<" + "<< b <<" = "<<sum<<endl;
13
14        return 0;
15    }
```

Run the program in your terminal and see what happens:

```
maydlee@LAPTOP-U1MOON2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ add_float.cpp
maydlee@LAPTOP-U1MOON2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
1.23457e+06 + 1 = 1.23457e+06
```

The result looks like no addition performed. Why?
Using fixed-point notation can we get the right result?

# 2.3 Keyboard input and terminal output

1. Formatting output with *printf*

*printf* (*format-control-string, other-arguments*)

*format-control-string* describes the output format, which consists of conversion specifiers, field widths, precisions and literal characters with percent sign(%).

| Conversion specifier | Description |
|---|---|
| d | Display as a *signed decimal integer.* |
| i | Display as a *signed decimal integer.* [*Note:* The i and d specifiers are *different* when used with scanf.] |
| o | Display as an *unsigned octal integer.* |
| u | Display as an *unsigned decimal integer.* |
| x or X | Display as an *unsigned hexadecimal integer.* X causes the digits 0-9 and the *uppercase* letters A-F to be used in the display and x causes the digits 0-9 and the *lowercase* letters a-f to be used in the display. |
| h, l or ll (letter "ell") | Place *before* any integer conversion specifier to indicate that a short, long or long long integer is displayed, respectively. These are called **length modifiers**. |
| e or E | Display a floating-point value in *exponential notation.* |
| f or F | Display floating-point values in *fixed-point notation* (F is supported in the Microsoft Visual C++ compiler in Visual Studio 2015 and higher). |
| g or G | Display a floating-point value in either the *floating-point form* f or the exponential form e (or E), based on the magnitude of the value. |
| L | Place before any floating-point conversion specifier to indicate that a long double floating-point value should be displayed. |

# 2.Reading Formatted input with *scanf*

*scanf* (*format-control-string, other-arguments*)

**format-control-string** describes the formats of input, **other-arguments** are **pointers** to variables in which the input will be stored.

| Conversion specifier | Description |
| --- | --- |
| *Integers* | |
| d | Read an *optionally signed decimal integer*. The corresponding argument is a pointer to an int. |
| i | Read an *optionally signed decimal, octal or hexadecimal integer*. The corresponding argument is a pointer to an int. |
| o | Read an *octal integer*. The corresponding argument is a pointer to an unsigned int. |
| u | Read an *unsigned decimal integer*. The corresponding argument is a pointer to an unsigned int. |
| x or X | Read a *hexadecimal integer*. The corresponding argument is a pointer to an unsigned int. |
| h, l and ll | Place *before* any of the integer conversion specifiers to indicate that a short, long or long long integer is to be input, respectively. |
| *Floating-point numbers* | |
| e, E, f, g or G | Read a *floating-point value*. The corresponding argument is a pointer to a floating-point variable. |
| l or L | Place before any of the floating-point conversion specifiers to indicate that a double or long double value is to be input. The corresponding argument is a pointer to a double or long double variable. |
| *Characters and strings* | |
| c | Read a *character*. The corresponding argument is a pointer to a char; no null ('\0') is added. |
| s | Read a *string*. The corresponding argument is a pointer to an array of type char that's large enough to hold the string and a terminating null ('\0') character—which is automatically added. |

***Note:*** When inputting data, prompt the user for one data item or a few data items at a time. Avoid asking the user to enter many data items in response to a single prompt.

# Example:

```c
lab02_examples > C scanf_demo.c > ...
1    #include <stdio.h>
2
3    int main()
4    {
5        printf("Please input an integer, a character and a float:\n");
6        int a;
7        scanf("%d", &a);
8        printf("a = %d\n", a);
9
10       getchar();    //discard the newline or space symbol
11       char b;
12       scanf("%c", &b);
13       printf("b = %c\n", b);
14
15       float c;
16       scanf("%f", &c);
17       printf("c = %f\n", c);
18
19       return 0;
20   }
```

If you omit the statement, what will be the output?

When you input data with keyboard, the **white space**,(such as space, new line and tab) is the valid separator.

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ gcc -o scanf_demo scanf_demo.c && ./scanf_demo
Please input an integer, a character and a float:
34 A 54.7
a = 34
b = A
c = 54.700001
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./scanf_demo
Please input an integer, a character and a float:
12
a = 12
x
b = x
21.3
c = 21.299999
```

Use blank space key  to separate the data

Use Enter key  to separate the data

# 3. *cout*

cout << variable1(expression1) [<< variable2 << variable n];

```cpp
lab02_examples > C+ cout_demo.cpp > main()
1    #include <iostream>
2
3    int main()
4    {
5        using namespace std;
6        int a = 5;
7        char b = 'A';
8        float c = 70.1f;
9        double d = 129.6;
10
11       cout << "a = " << a << ",b = " << b << ",c = " << c << ",d = " << d << endl;
12
13       return 0;
14
15   }
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    **TERMINAL**

maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ cout_demo.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
a = 5,b = A,c = 70.1,d = 129.6

cout is a little bit smart, it can recognizes the type of
the variable and print the exact value of the variable.

Floating-point types are displayed with a total of six digits, except that trailing zeros aren't displayed. In particular, E notation is used if the exponent is 6 or larger or -5 or smaller.

***C++*** provides two methods to control the **output formats**
- ***Using member functions of ios class***
- ***Using iomanip manipulators***

- ***Using member functions of ios class***

1. cout.setf(): The setf() function has two prototypes, the first one is:
   cout.set(fmtflags flags);

std::ios_base::setf

| | |
|---|---|
| fmtflags setf( fmtflags flags ); | (1) |
| fmtflags setf( fmtflags flags, fmtflags mask ); | (2) |

**Formatting Constants**

| Constant | Meaning |
|---|---|
| ios_base::boolalpha | Input and output bool values as true and false. |
| ios_base::showbase | Use C++ base prefixes (0,0x) on output. |
| ios_base::showpoint | Show trailing decimal point. |
| ios_base::uppercase | Use uppercase letters for hex output, E notation. |
| ios_base::showpos | Use + before positive numbers. |

https://blog.csdn.net/baishuiniyaonulia/article/details/79144033

- ***Using member functions of ios class***

The second  one is:

    cout.set(fmtflags flags,fmtflags mask);

Arguments for setf(long, long)

| Second Argument | First Argument | Meaning |
|---|---|---|
| ios_base::basefield | ios_base::dec | Use base 10. |
| | ios_base::oct | Use base 8. |
| | ios_base::hex | Use base 16. |
| ios_base::floatfield | ios_base::fixed | Use fixed-point notation. |
| | ios_base::scientific | Use scientific notation. |
| ios_base::adjustfield | ios_base::left | Use left-justification. |
| | ios_base::right | Use right-justification. |
| | ios_base::internal | Left-justify sign or base prefix, right-justify value. |

## Using member functions of ios class
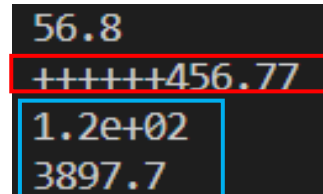
2. cout.width(len)          //set the field width
3. cout.fill(ch)            // fill character to be used with justified field
4. cout.precision(p)        // set the precision of floating-point numbers

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

int main()
{
    cout << 56.8 << endl;
    cout.width(12);
    cout.fill('+');
    cout << 456.77 << endl;

    cout.precision(2);
    cout << 123.356 << endl;
    cout.precision(5);
    cout << 3897.678485 << endl;

    return 0;
}
```

```
56.8
+++++456.77
1.2e+02
3897.7
```
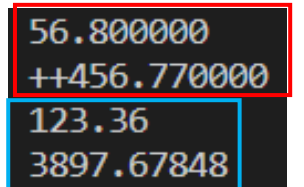
significant digits

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

int main()
{
    cout.setf(ios_base::fixed, ios_base::floatfield);
    cout << 56.8 << endl;
    cout.width(12);
    cout.fill('+');
    cout << 456.77 << endl;

    cout.precision(2);
    cout << 123.356 << endl;
    cout.precision(5);
    cout << 3897.678485 << endl;

    return 0;
}
```

```
56.800000
++456.770000
123.36
3897.67848
```

precision of floating number

The effect of calling **setf()** can be undone with **unsef()**.

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

#include <iostream>
using namespace std;

int main()
{
    bool flag = true;
    float f = 0.20f;

    cout.setf(ios::showpoint);
    cout.setf(ios::boolalpha);
    cout << flag << endl;
    cout << f << endl;

    cout.unsetf(ios::boolalpha);
    cout.unsetf(ios::showpoint);
    cout << flag << endl;
    cout << f << endl;

    return 0;
}
```
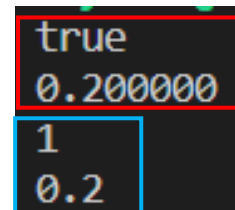
```
true
0.200000
1
0.2
```

# Standard Manipulators

C++ offers several manipulators to invoke setf(),automatically supplying the right arguments.

## Some Standard Manipulators

| Manipulator | Calls |
|---|---|
| boolalpha | setf(ios_base::boolalpha) |
| noboolalpha | unset(ios_base:: boolalpha) |
| showbase | setf(ios_base::showbase) |
| noshowbase | unsetf(ios_base::showbase) |
| showpoint | setf(ios_base::showpoint) |
| noshowpoint | unsetf(ios_base::showpoint) |
| showpos | setf(ios_base::showpos) |
| noshowpos | unsetf(ios_base::showpos) |
| uppercase | setf(ios_base::uppercase) |
| nouppercase | unsetf(ios_base::uppercase) |

| Manipulator | Calls |
|---|---|
| internal | setf(ios_base::internal, ios_base::adjustfield) |
| left | setf(ios_base::left, ios_base::adjustfield) |
| right | setf(ios_base::right, ios_base::adjustfield) |
| dec | setf(ios_base::dec, ios_base::basefield) |
| hex | setf(ios_base::hex, ios_base::basefield) |
| oct | setf(ios_base::oct, ios_base::basefield) |
| fixed | setf(ios_base::fixed, ios_base::floatfield) |
| scientific | setf(ios_base::scientific, ios_base::floatfield) |

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

int main()
{
    bool flag = false;
    double a = 2.3876;
    double b = 0.46e2;

    cout << boolalpha << flag << endl;
    cout << fixed << a << endl;
    cout << b << endl;

    cout << noboolalpha << flag << endl;
    cout.unsetf(ios::fixed);
    cout << a << endl;
    cout << b << endl;

    return 0;
}
```

```
false
2.387600
46.000000
0
2.3876
46
```

- ***Using iomanip manipulators***

  #include <iomanip>

  1. setw(p)    2. setfill(ch)    3. setprecision(d)

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

int main()
{
    cout.setf(ios_base::fixed, ios_base::floatfield);
    cout << 56.8 << setw(12) << setfill('#') << 456.77 << endl;

    cout << left;
    cout << setw(12) << setprecision(2) << 123.356 << endl;
    cout << setw(12) << setprecision(5) << 3897.6784385 << endl;

    cout << right;
    cout << setw(12) << setfill(' ') << 123.356 << endl;
    cout << setw(12) << setfill(' ') << 3897.6784385 << endl;

    cout.unsetf(ios_base::fixed);
    cout << 56.8 << setw(12) << setfill('$') << 456.77 << endl;

    return 0;
}
```

```
56.800000##456.770000
123.36######
3897.67844##
      123.35600
     3897.67844
56.8$$$$$$456.77
```

# 4. *cin*

cin >> variable1 [>> variable2 >> ...variable n];

```cpp
lab02_examples > G cin_demo.cpp > ⊗ main()
1    #include <iostream>
2    using namespace std;
3
4    int main()
5    {
6        cout << "Please input an integer, a character and a float\n";
7        int a;
8        cin >> a;
9        cout << "a = " << a << endl;
10
11       char b;
12       cin >> b;
13       cout << "b = " << b << endl;
14
15       float c;
16       cin >> c;
17       cout << "c = " << c << endl;
18
19       return 0;
20   }
21
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ cin_demo.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
Please input an integer, a character and a float
3 A 2.5
a = 3
b = A
c = 2.5
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
Please input an integer, a character and a float
10
a = 10
M
b = M
34.6
c = 34.6
```

white space, such as space, new line and tab is the valid separator.

Use blank space key to separate the data

Use Enter key to separate the data

# 3 Exercises

1.Compile and run the following program, what is the result?

You need to explain the reason to a SA to pass the test.

```
#include <stdio.h>

int main()
{
    char a = 127;
    unsigned char b = 0xff;
    unsigned char c = 0;

    a++;
    b++;
    c--;
    printf("a=%d\nb=%d\nc=%d\n",a,b,c);

    return 0;
}
```

## 2. Run the following source code and explain the result.

You need to explain the reason to a SA to pass the test.

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

int main()
{
    cout << fixed;
    float f1 = 1.0f;
    cout<<"f1 = "<<f1<<endl;

    float a = 0.1f;
    float f2 = a+a+a+a+a+a+a+a+a+a;
    cout<<"f2 = "<<f2<<endl;

    if(f1 == f2)
        cout << "f1 = f2" << endl;
    else
        cout << "f1 != f2" << endl;

    return 0;
}
```

3. Run the following source code and explain the result. Why the value of a and b are not equal? Explain the division operation with different types.

You need to explain the reason to a SA to pass the test.

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

int main()
{
    int a, b;
    double c, d;

    a = 19.99 + 21.99;
    b = (int)19.99 + (int)21.99;
    c = 23 / 8;
    d = 23 / 8.0;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "c = " << c << endl;
    cout << "d = " << d << endl;
    cout << "0/0= " << 0/0 << endl;

    return 0;

}
```

4. Write a .C program that asks the user to enter an integer value, a character, and a float value and then print them out. A sample run should look like this:

You should use **scanf** and **printf** functions for input and output.

```
Please input  a character :
T
Please input an integer:
45
Please input  a float:
89.3
The variables you entered were:
The character is:T
The integer is:45
The float is 89.300003
```

What happens when you are prompted to enter an integer, but you enter a float?