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

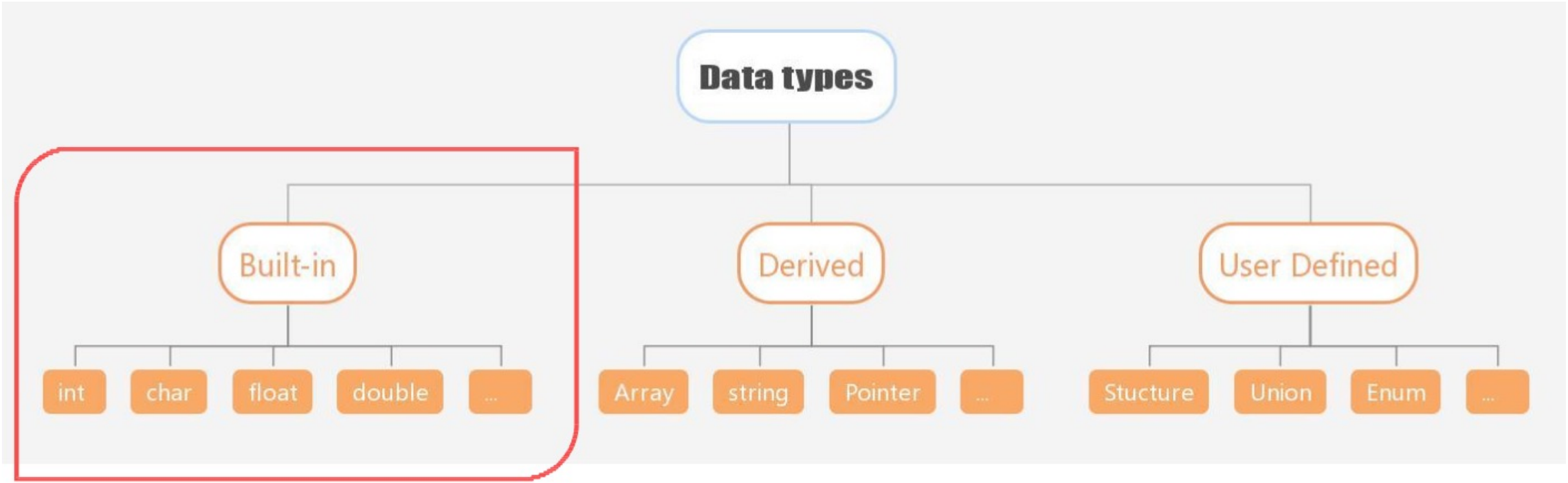
2 Knowledge Points

2.1 Fundamental Data Types

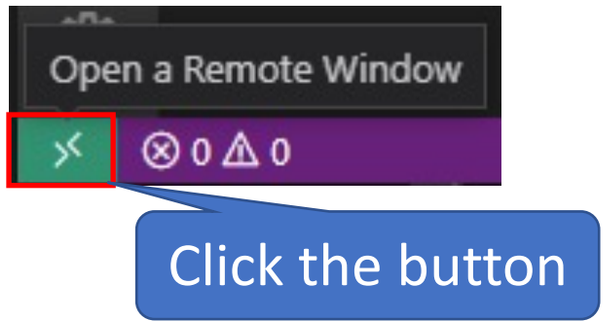
2.2 Arithmetic Operators and Assignment Operators

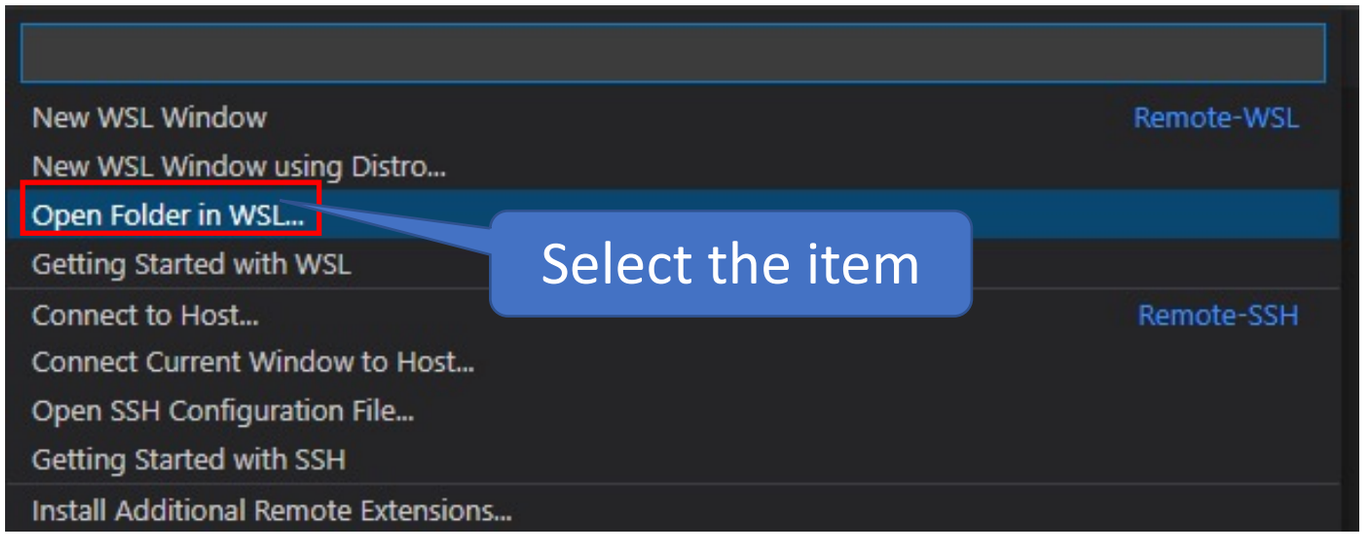
2.3 Input and Output

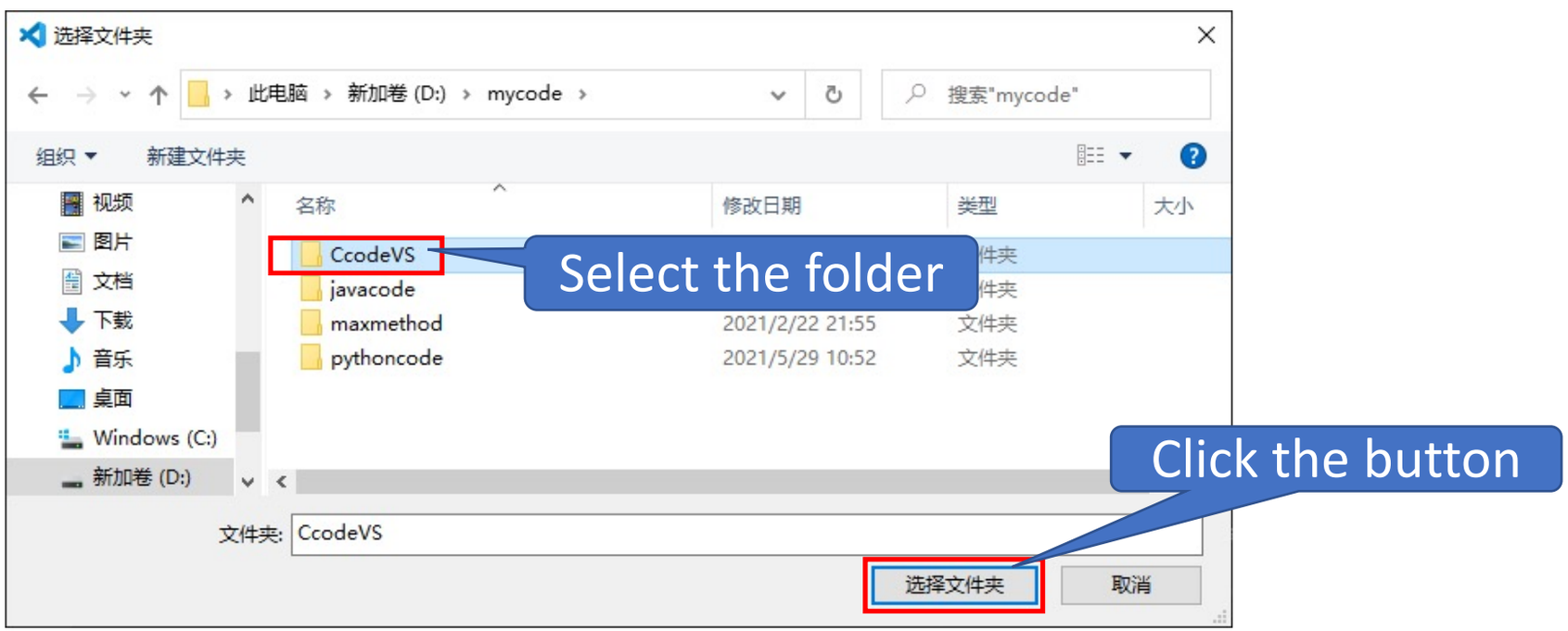
2.1 Data types

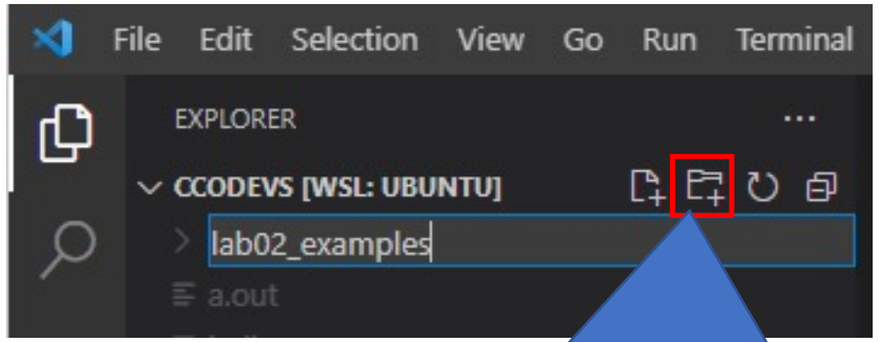


Example: Write a C program to find Size of fundamental data types.
(All examples are written in VScode and compiled under WSL)

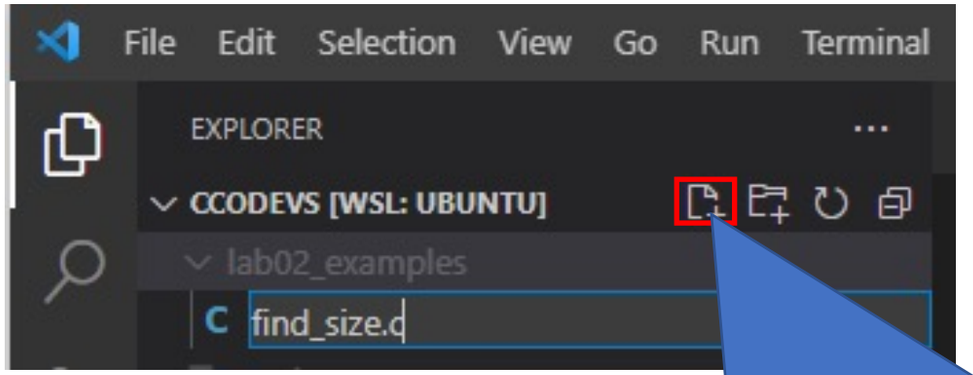




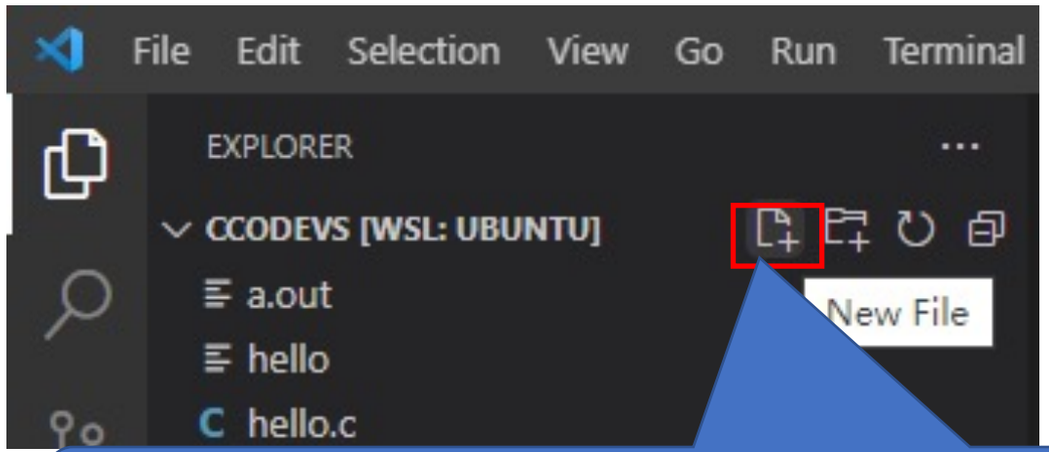




You can choose New Folder button and input a new folder name. The new folder is created in the current directory.



Click the New File button and create a new file in the directory you just created.



You can also choose New File button to create a new file in the current directory

Type the codes as follows:

lab02_examples > C find_size.c > ...

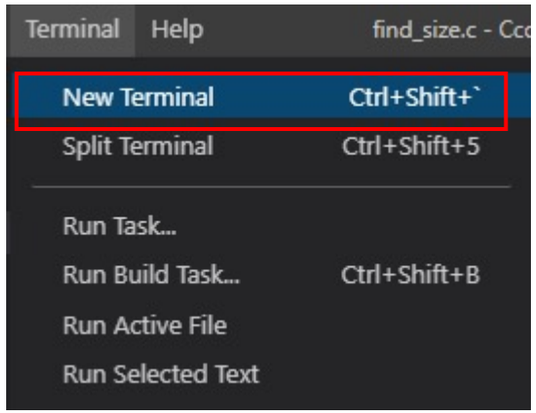
```c
#include <stdio.h>
#include <stdbool.h>

int main()
{
    printf("\n\nFind Size of the fundamental data typ
    printf("-------------------------------------------\n");
    printf("The sizeof(char) is:        %ld bytes\n", sizeof(char));
    printf("The sizeof(short) is:       %ld bytes\n", sizeof(short));
    printf("The sizeof(int) is:         %ld bytes\n", sizeof(int));
    printf("The sizeof(long) is:        %ld bytes\n", sizeof(long));
    printf("The sizeof(long long) is:   %ld bytes\n", sizeof(long long));
    printf("The sizeof(float) is:       %ld bytes\n", sizeof(float));
    printf("The sizeof(double) is:      %ld bytes\n", sizeof(double));
    printf("The sizeof(long double) is: %ld bytes\n", sizeof(long double));
    printf("The sizeof(bool) is:        %ld bytes\n", sizeof(bool));

    return 0;
}
```

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

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

```
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



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 > find_size.cpp > ...

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

int main()
{
    cout << "\n\nFind Size of the fundamental data types:\n";
    cout << "----------------------------------------------\n";
    cout << "The size of(char) is:         " << sizeof(char) << endl;
    cout << "The size of(short) is:        " << sizeof(short) << endl;
    cout << "The size of(int) is:          "  << sizeof(int) << endl;
    cout << "The size of(long) is:         " << sizeof(long) << endl;
    cout << "The size of(long long) is:    " << sizeof(long long) << endl;
    cout << "The size of(float) is:        " << sizeof(float) << endl;
    cout << "The size of(double) is:       " << sizeof(double) << endl;
    cout << "The size of(long double) is:  " << sizeof(long double) << endl;
    cout << "The size of(bool) is:         "    << sizeof(bool) << endl;

    return 0;
}
```

```
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 > integer_limit.cpp > main()

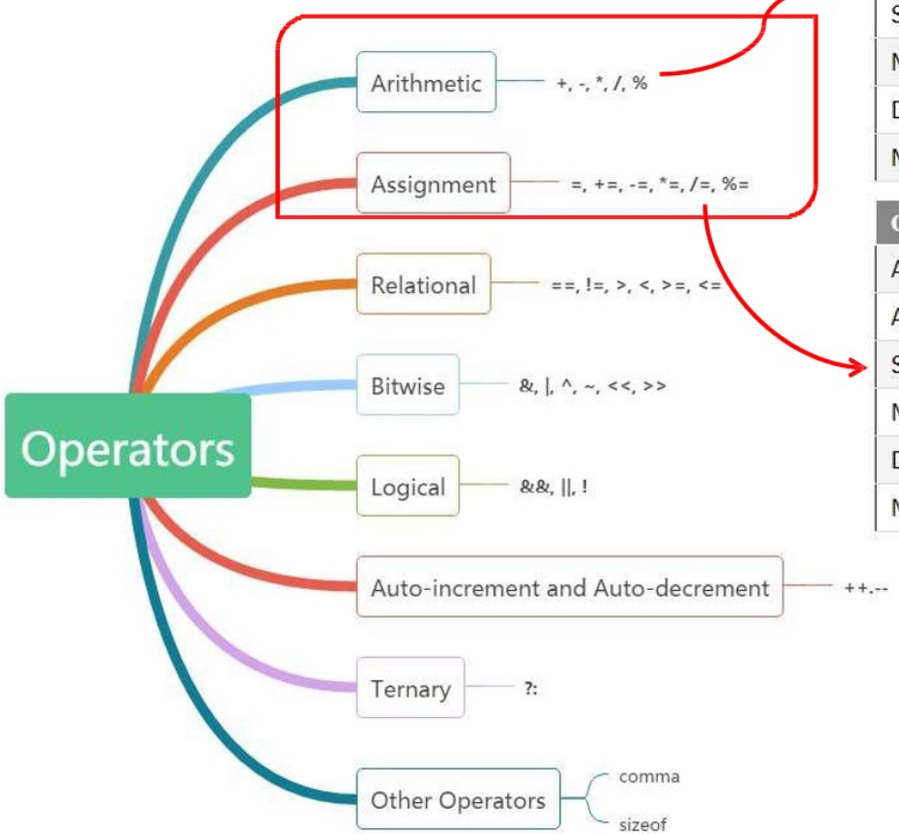
```cpp
#include <iostream>
#include <climits>

int main()
{
    using namespace std;

    int n_int = INT_MAX; //max value for integer number
    short n_short = SHRT_MAX;
    long n_long = LONG_MAX;
    long long n_llong = LLONG_MAX;

    cout << "int is " << sizeof (int) << " bytes." << endl;
    cout << "short is " << sizeof n_short << " bytes." << endl;
    cout << "long is " << sizeof n_long << " bytes." << endl;
    cout << "long long is " << sizeof n_llong << " bytes." << endl;
    cout << endl;

    cout << "Maximum values: " << endl;
    cout << "int: " << n_int << endl;
    cout << "short: " << n_short << endl << "long: " << n_long << endl << "long long: " << n_llong << endl << endl;

    cout << "Minimum int value = " << INT_MIN << endl;
    cout << "Bits per byte = " << CHAR_BIT << endl;

    return 0;
}
```

```
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

Example Program of Arithmetic Operators:

lab02_examples > add.cpp > main()

```cpp
#include <iostream>

using std::cout;
using std::endl;

int main()
{
    int a = 1234567890;
    int b = 1234567890;
    int sum = a+b;

    cout<< a <<" + "<< b <<" = "<< sum <<endl;

    return 0;
}
```

Run the program in your terminal and see what happens:

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ add.cpp
maydlee@LAPTOP-U1MO0N2F:/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?

lab02_examples > add_float.cpp > ...

```cpp
#include <iostream>

using std::cout;
using std::endl;

int main()
{
    float a = 1234567.0;
    float b = 1.0;
    float sum = a+b;

    cout<< a <<" + "<< b <<" = "<<sum<<endl;

    return 0;
}
```

Run the program in your terminal and see what happens:

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ add_float.cpp
maydlee@LAPTOP-U1MO0N2F:/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.

lab02_examples > C printf_demo.c > main()

```c
#include <stdio.h>

int main()
{
    int a = 5;
    char b = 'A';
    float c = 70.1f;
    double d = 129.6;

    printf("a = %d, b = %d, c = %d, d = %d\n", a, b, c, d);

    return 0;

}
```

The format control strings don't match the types of variables, what will be the output?

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ gcc printf_demo.c
printf_demo.c: In function 'main':
printf_demo.c:10:34: warning: format '%d' expects argument of type 'int', but argument 4 has type 'double' [-Wformat=]
   10 |     printf("a = %d, b = %d, c = %d, d = %d\n", a, b, c, d);
      |                                  ~^                  ~
      |                                   |                  |
      |                                   int                double
      |                                  %f
printf_demo.c:10:42: warning: format '%d' expects argument of type 'int', but argument 5 has type 'double' [-Wformat=]
   10 |     printf("a = %d, b = %d, c = %d, d = %d\n", a, b, c, d);
      |                                            ~^              ~
      |                                             |              |
      |                                             int            double
      |                                            %f
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
a = 5, b = 65, c = 187040176, d = 0
```

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:

lab02_examples > C scanf_demo.c > ...

```c
#include <stdio.h>

int main()
{
    printf("Please input an integer, a character and a float:\n");
    int a;
    scanf("%d", &a);
    printf("a = %d\n", a);

    getchar();    //discard the newline or space symbol
    char b;
    scanf("%c", &b);
    printf("b = %c\n", b);

    float c;
    scanf("%f", &c);
    printf("c = %f\n", c);

    return 0;
}
```

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];

lab02_examples > cout_demo.cpp > main()

```cpp
#include <iostream>

int main()
{
    using namespace std;
    int a = 5;
    char b = 'A';
    float c = 70.1f;
    double d = 129.6;

    cout << "a = " << a << ",b = " << b << ",c = " << c << ",d = " << d << endl;

    return 0;

}
```

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

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(fmtflags, fmtflags)

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.

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

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

lab02_examples > cout_set.cpp > main()

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

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

    cout.precision(2);
    cout << 123.356 << "\t";
    cout.precision(5);
    cout << 3897.678483 << endl;

    return 0;

}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ cout_set.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
56.800000++456.770000

123.36  3897.67848
```

- ***Using iomanip manipulators***

#include <iomanip>

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

lab02_examples > cout_manip.cpp > ...

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

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

    cout << setprecision(2) << 123.356 << endl;
    cout << setprecision(5) << 3897.6784385 << endl;

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ cout_manip.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
56.800000  456.770000
123.36
3897.67844
```

Some Standard Manipulators

Manipulator	Calls
boolalpha	setf(ios_base::boolalpha)
noboolalpha	unset(ios_base::noboolalpha)
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)
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::base-field)
hex	setf(ios_base::hex, ios_base::base-field)
oct	setf(ios_base::oct, ios_base::base-field)
fixed	setf(ios_base::fixed, ios_base::floatfield)
scientific	setf(ios_base::scientific, ios_base::floatfield)

lab02_examples > cout_manip.cpp > main()

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

int main()
{
//    cout.setf(ios_base::fixed, ios_base::floatfield);

    cout << fixed << 45.2 << endl;
    cout << 56.8 << setw(12) << 456.77 << endl;

    cout << setprecision(2) << 123.356 << endl;
    cout << setprecision(5) << 3897.6784385 << endl;

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ cout_manip.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
45.200000
56.800000  456.770000
123.36
3897.67844
```

cout.unsetf( )

lab02_examples > cout_unset.cpp > main()

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

int main()
{
//    cout.setf(ios_base::fixed, ios_base::floatfield);

    cout << fixed << 45.2 << endl;
    cout.unsetf(ios_base::fixed);
    cout << 56.8 << setw(12) << 456.77 << endl;

    cout << setprecision(2) << 123.356 << endl;
    cout << setprecision(5) << 3897.6784385 << endl;

    return 0;
}
```

Whatever you use setf() or manipulators, you can use unsetf() to cancel the settings.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ g++ cout_unset.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/mycode/CcodeVS/lab02_examples$ ./a.out
45.200000
56.8        456.77
1.2e+02
3897.7
```

Show the significant figures not precision

4. *cin*

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

lab02_examples > cin_demo.cpp > main()

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

int main()
{
    cout << "Please input an integer, a character and a float\n";
    int a;
    cin >> a;
    cout << "a = " << a << endl;

    char b;
    cin >> b;
    cout << "b = " << b << endl;

    float c;
    cin >> c;
    cout << "c = " << c << endl;

    return 0;
}
```

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
```

Use blank space key to separate the data

Use Enter key to separate the data

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

3 Exercises

1. Write a program to produce the output as shown below.

```
Result:
x value y value Expressions     Result
10 |    5 |     x=y+3           |x=8
10 |    5 |     x=y-2           |x=3
10 |    5 |     x=y*5           |x=25
10 |    5 |     x=x/y           |x=2
10 |    5 |     x=x%y           |x=0
```

2. Write a program that asks the user to enter the number of seconds as an integer value(use type long, or, if available, long long) and then displays the equivalent time in days, hours, minutes and seconds. Use symbolic constants to represent the number of hours in the day, the number of minutes in an hour, and the number of seconds in a minute. The output should look like this:

```
Enter the number of seconds:31600000
31600000 seconds = 365 days, 17 hours, 46 minutes, 40 seconds
```

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

```
Please Enter a Character :  A
Please Enter an Integer Value :  20
Please Enter Float Value :  30.678

The variables you entered were:
The Character Value that you Entered is :  A
The Integer Value that you Entered is : 20
The Float Value that you Entered is :  30.678
```

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