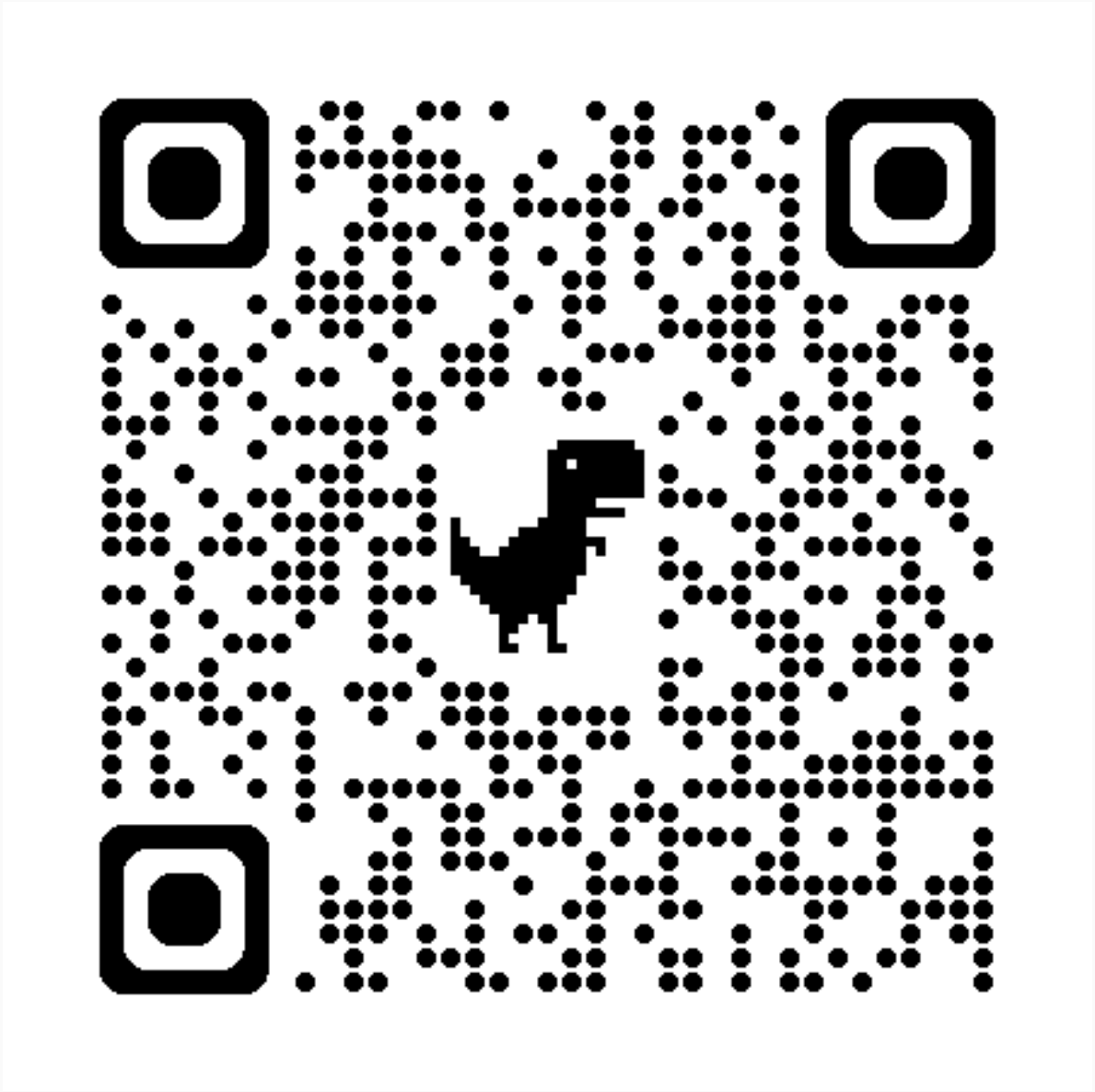
Variable Operations

Tai, Wei Hsuan

week 2

課程簡報



Outline

Recap

Basic Structure of C++ Program

- Header file.
- Namespace.
- Main function.

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

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

Remember!

- Every statement ends with a semicolon **(;)**.
- Programs always start from the main() function.
- All functional code needs to be inside a function.
- All functions must be enclosed in curly braces **{}**.

Variable Introduction

What is Variable?

Variables can be thought of as containers that hold data which can be changed during program execution.
Data is actually stored in the computer's memory, and variables provide a way to access and manipulate that data.
You can consider variables as labels for specific memory locations where data is stored.

Common Data Types

Table 1: Common C++ Data Types

Type	Typical Range	Size (bytes)
`int`	$-2^{31} \sim 2^{31} - 1$	4
`long long`	$-2^{63} \sim 2^{63} - 1$	8
`float`	$\pm 1.2 \times 10^{-38} \sim 3.4 \times 10^{38}$	4
`double`	$\pm 2.3 \times 10^{-308} \sim 1.7 \times 10^{308}$	8
`char`	$-2^{7} \sim 2^{7} - 1$	1
`bool`	0 or 1	1

Variable Operations

Operation Types

- Assignment Operators
- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators

Assignment Operators

Table 2: Assignment Operators

Operator	Description	Example
`=`	Assign	`a = b`
`+=`	Add and assign	`a += b (a = a + b)`
`-=`	Subtract and assign	`a -= b (a = a - b)`
`*=`	Multiply and assign	`a *= b (a = a * b)`
`/=`	Divide and assign	`a /= b (a = a / b)`
`%=`	Modulus and assign	`a %= b (a = a % b)`

Assignment Operators(Cont.)

```
1    int a=5, b=2;
2    a+=b;
3    a*=b;
```

Arithmetic Operators

Table 3: Arithmetic Operators

Operator	Description	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
%	Modulus (Remainder)	a % b
++	Increment	++a or a++
--	Decrement	--a or a--

Arithmetic Operators(Cont.)

```
1    int a=5, b=2;
2    cout<<a+b;
3    cout<<a%b;
4    cout<<++a;
5    cout<<b--;
```

You can also combine assignment and arithmetic operators:

```
1    int a=5, b=2;
2    int c=a+b, d=b++, e=a/2;//e=2
```

If the result of division is not an integer, it will be rounded down to the nearest integer.

Comparison Operators

Table 4: Comparison Operators

This operators always return a boolean value (true or false).

Operator	Description	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

Comparison Operators(Cont.)

```
1    int a=5, b=2;
2    cout<<(a==b);
3    cout<<(a!=b);
4    cout<<(a>b);
```

We will use lots of comparison operators in the condition statements(10/17).

Logical Operators

Table 5: Logical Operators

This operators always return a boolean value (true or false).

Operator	Description	Example
`&&`	Logical AND	`a && b`
`||`	Logical OR	`a || b`
`!`	Logical NOT	`!a`

For `and` and `or`, exactly you can directly use `and` and `or` instead of `&&` and `||`.

Logical Operators(Cont.)

A	B	!A	A and B	A or B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Logical Operators(Cont.)

```
1    int a=5, b=2;
2    cout<<((a>b) && (a!=b));
3    cout<<((a>b) || (a==b));
4    cout<<!(a>b);
```

Challenge: Determine the result of the following expression:

```
( (1 || 0) && (1 || !1) ) || ( !(0 && 1) && 1 )
```

Bitwise Operation

Basic concept of Bitwise

- Data is stored in binary format (0s and 1s).
- All the data would be converted into a number and stored in binary format.
- In the hardware, high voltage is represented as 1, and low voltage is represented as 0.
- Guess it: an empty disk and a full disk, which one is heavier?
- All operations in the computer will finally be converted into bitwise operations.

Exponential notation

We all learned multiplication in the elementary school. If we want to multiplicate a number by itself several times, we can use exponential notation to represent it.
As the following example:

$$2^4 = 2 \times 2 \times 2 \times 2 = 16$$

By definition, a^b means multiplying a by itself b times.

Concept of Number Systems

A number system represents a value as the sum of powers of its base. General form:

$$N = d_k \times b^k + d_{k-1} \times b^{k-1} + \cdots + d_1 \times b^1 + d_0 \times b^0$$

where b is the base, and d_i are the digits. Example in base 10 (decimal):

$$345_{10} = 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

Example in base 2 (binary):

$$1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11_{10}$$

Storage format of Integer

For the word, it can be converted into an integer by ASCII code.
We can store integers in binary format, storing binary data in memory is done using bits (0s and 1s). Each bit represents a power of 2, and the combination of bits represents the integer value.
For the aspect of physical, a bit is typically stored using a transistor or a capacitor, where a high voltage represents 1 and a low voltage represents 0.

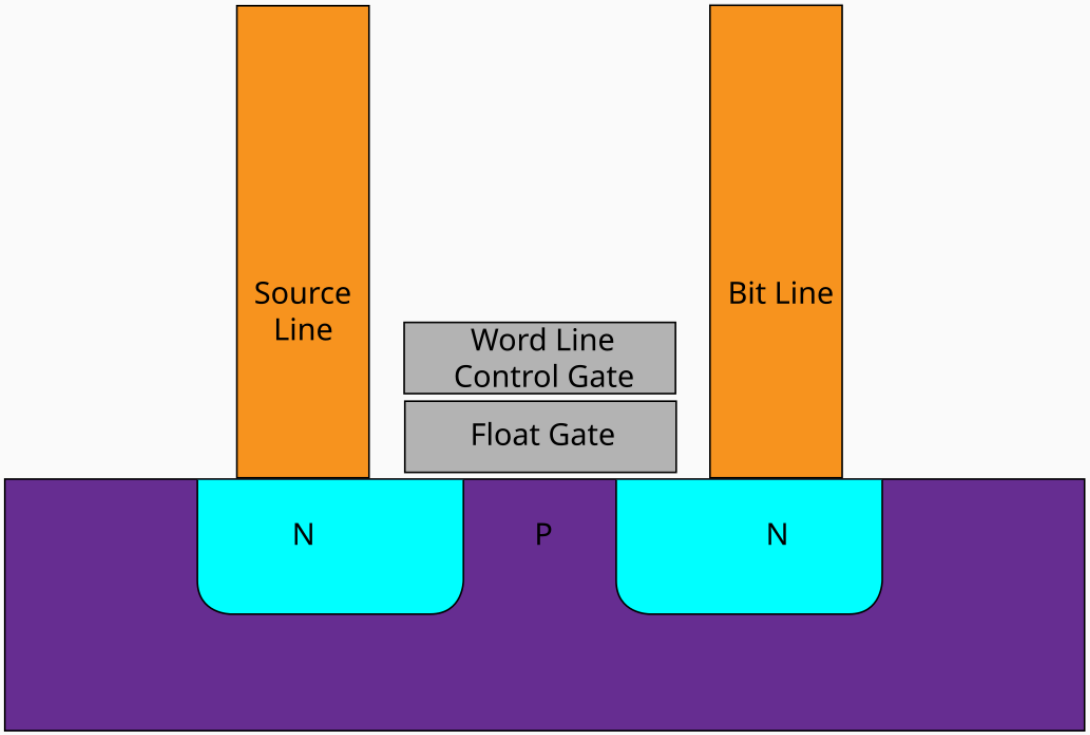


Figure 1: Storage principle of Flash memory

Source: Wikipedia

Basic concept of Bitwise operations

The operations we learned before are all operations on the entire variables. In this section, we will learn operations on each bit of the variable.

Varaibles would be converted into binary format first, and then each bit would be operated according to the rules of bitwise operations.

Bitwise Operators

- AND (`&`): Both bits must be 1 to result in 1.
- OR (`|`): At least one bit must be 1 to result in 1.
- XOR (`^`): Only one bit must be 1 to result in 1.
- NOT (`~`): Inverts the bits (0 becomes 1, and 1 becomes 0).
- Left Shift (`<<`): Shifts bits to the left, filling with 0s on the right.
- Right Shift (`>>`): Shifts bits to the right, filling with 0s on the left (for unsigned types).

In the following slides, assign `a=5` and `b=3` as examples.

AND, OR, XOR, NOT

First, convert `a` and `b` into binary format:

$$a = 5 = 0101_2$$

$$b = 3 = 0011_2$$

Bit Position	`a`	`b`	`and`	`or`	`xor`	`not a`
3	0	0	0	0	0	1
2	1	0	0	1	1	0
1	0	1	0	1	1	1
0	1	1	1	1	0	0

AND, OR, XOR, NOT(Cont.)

```
1        int a=5, b=3;
2        cout<<(a & b);    // 1
3        cout<<(a | b);    // 7
4        cout<<(a ^ b);    // 6
5        cout<<(~a);     // -6
```

Note: The result of NOT operation is negative because of the two's complement representation of negative numbers in binary.

Shift Operators

The shift operators are actually shift the bits to left or right.
Consider $a = 5 = 0101\_2$:

$$a << 1 = 1010_2 = 10$$

$$a >> 1 = 0010_2 = 2$$

Each bit shifted to the left is equivalent to multiplying the number by 2, and each bit shifted to the right is equivalent to dividing the number by 2 (discarding any remainder).

Practice

Practice

- Basic: a002, d827, d485
- Advanced: d060, a799, d068, d073, d460
- Weird: f987