

Lecture 2

變數 (續)
算術運算與布林運算

回顧

- 第一個程式
 - 基本架構 → 五行必寫的公式
 - 註解的寫法、`cout`、`<<`、`endl`
- 變數 (variables)
 - 變數名稱 – 限制?
 - 變數型別 (type)
 - 字元型別 – `char`
 - 整數型別 – `short`, `int`, `long`
 - 浮點數型別 – `float`, `double`

變數型態 variable type

意義	變數型態	記憶體量*	數值範圍	準確度 (Accuracy)
字元	<code>char</code>	1	-128 ~ +127	精確
整數	<code>short int</code>	2	-32768 ~ 32767	精確
	<code>int</code>	4	$-2^{31} \sim 2^{31}-1$	精確
	<code>long int</code>	4	$-2^{31} \sim 2^{31}-1$	精確
浮點數	<code>float</code>	4	$-3.4 \times 10^{38} \sim +3.4 \times 10^{38}$	6-7 位有效數字
	<code>double</code>	8	$-1.7 \times 10^{308} \sim +1.7 \times 10^{308}$	14-15 位有效數字
	<code>long double</code>	8	$-1.7 \times 10^{308} \sim +1.7 \times 10^{308}$	14-15 位有效數字

* 單位為 bytes。記憶體量與所使用之作業系統、硬體、以及編譯器等皆有關係，本表則依據 win32 作業系統之 Visual C++ 9.0。

2-1.cpp

```
#include<iostream>
using namespace std;
int main() {
    char a = 'A';

    cout << a << endl;
    cout << (int) a << endl;
    a = 67;
    cout << a << endl;
    cout << sizeof(a) << endl;
    cout << sizeof(char) << endl;
    return 0;
}
```

```
A
65
C
1
1
```

變數型態 variable type

- 整數型態 (integer)
 - `short int` / `int` / `long int` ← → `short` / `int` / `long`
 - 長度與 CPU & 編譯器 (compiler) 有關
 - 一般來說 `int`: 4 bytes (32 bits → $2^{32}=4,294,967,296 \sim 40$ 億)
 - `short (int)`: 短整數, 佔用記憶體量 \leq `int`
 - `long (int)`: 長整數, 佔用記憶體量 \geq `int`
 - Modifier (修飾字): **unsigned vs. signed**
 - **unsigned int/ signed short...**
 - 有號數或無號數
 - 當資料不可能為負值時, 可使用無號數, 使可代表的數值大小擴大一倍。

2-2.cpp

```
#include<iostream>
using namespace std;
int main() {
    short int a = 32767;
    unsigned short b = 65535;

    cout << a << ", " << b << endl;
    a = a + 1;
    b = b + 1;
    cout << a << ", " << b << endl;

    return 0;
}
```

```
32767, 65535
```

```
-32768, 0
```

這裡所看到的情況稱為溢位 (overflow)

變數型態 variable type

- 實數, 浮點數(real number, floating point)型態
 - float, double, long double
 - 佔用記憶體量與 CPU & 編譯器(compiler) 有關
 - 目前一般來說 double: 8 bytes (64 bits)
 - float: 佔用記憶體量 <= double
 - double: 佔用記憶體量 >= double
 - IEEE-754 規定了浮點數在記憶體裡的儲存方式。
- Example:
 - float a; ← 宣告變數 a, 型別是浮點數
 - double b = 3.0e8; ← 宣告變數 b, 並初始化為 3×10^8

2-3.cpp

```
#include<iostream>
using namespace std;
int main() {
    double a = 3.14159265358979323846;
    float b = 3.e8;

    cout << a << endl;
    cout << b << endl;

    return 0;
}
```

```
3.14159
3e+008
```

變數型態 variable type

- 布林值 (boolean) 型態
 - bool
 - 用來儲存布林或邏輯運算的結果: 真 (true) 或是假 (false)
 - 之後會與決策判斷的指令一起使用

2-4.cpp

```
#include<iostream>
using namespace std;
int main() {
    bool a = false;
    bool b = true;
    bool c = 100;

    cout << a << endl;
    cout << b << endl;
    cout << c << endl;

    return 0;
}
```

```
0
1
1
```

在 C/C++ 中, 非 0 值為真 (true), 0 為假 (false)

常數變數 (constant variable)

- 在宣告變數時, 對於整數可使用修飾字 unsigned/signed 將一個整數或字元型態的變數宣告為無號數或有號數。
- 另一個修飾字 const, 可用於前述的所有型別, 讓所宣告出來的變數成為常數變數。
- Example:
 - const double a = 3.14159;
- 在宣告常數變數時, 必需同時進行初始化的動作。
- 程式內若試圖對常數變數進行修改, 將導致編譯錯誤。

2-5.cpp

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

int main() {
    int a=3, c ;
    const int b=4;

    c = a + b;
    // b = 5;
    cout << "a+b=" << c;

    return 0;
}
```

回顧一下

- 變數名稱 – 如何命名、命名限制
- 變數型態
 - 字元
 - char a;
 - a = 65; ← → a='A';
 - 整數
 - int b;
 - b = 10;
 - 浮點數
 - double c = 3.14159e15;
 - 布林數
 - bool T = true; bool F = false;

修飾字：

- const
- unsigned, signed

回答以下問題

- 我今天想要儲存 3.141592654, 應該選用何種變數型別?
- 今天有一個資料是 512, 那些變數型別可以使用?
- 今天我想要儲存一個字元 'q', 可以使用何種變數型別?
- -4.2 可以使用何種變數型別儲存?
- 我要怎麼確定讓一個變數的值不會在程式被修改?

算術運算

Arithmetic operations

算術運算

arithmetic operations

- 我們在 1-2.cpp 中已看過了兩個運算子 (+, =) 以進行算術運算。

```
int a=3;
int b, c;
b = 4;
c = a + b;
cout << "a+b=" << c;
```

- 在 C/C++ 的算術運算中有以下重點：
 - 算術運算子與運算元
 - 算術運算子的優先順序
 - 型別轉換

2-6.cpp

```
#include <iostream>
using namespace std;
int main() {
    int a=14;
    int b=4;

    cout << "a= " << a << endl;
    cout << "b= " << b << endl;
    cout << "a+b=" << a+b << endl;
    cout << "a-b=" << a-b << endl;
    cout << "a*b=" << a*b << endl;
    cout << "a/b=" << a/b << endl;
    cout << "a%b=" << a%b << endl;

    return 0;
}
```

試試看

- 請將以上程式的 a 改成 3, a/b 的結果?
- 將以上程式的變數形別改成 char, 結果?
- 將以上程式的變數形別改成 double, 結果?

二元算術運算子

=: 指定 (assignment) 3 + 2 = 5
 +: 相加 (addition) 3 - 2 = 1
 -: 相減 (subtraction) 3 * 2 = 6
 *: 相乘 (multiplication) 3 / 2 = 1 (因為是整數)
 /: 相除 (division) 3 % 2 = 1
 %: 取餘數 (modulus)

運算的順序上與數學相同：先乘除後加減，% 視為除法運算

- 3 + 2 * 6 = 15
- 3 * 6 + 3 = 21
- 3 * 6 / 2 = 9

也同於數學運算式，可用括號 (...) 來改變運算的順序

- (3 + 2) * 6 = 30
- 3 * (6 + 3) = 27
- (3 * 6) / 2 = 9

2-7.cpp

```
#include <iostream>
using namespace std;
int main() {
    int a=2;
    int b;

    b = a + 3 * a;
    cout << "b=" << b << endl;

    b = b + 3 * a;
    cout << "b=" << b << endl;

    cout << "b+3 is " << b+3 << endl;
    cout << "b=" << b << endl;
    return 0;
}
```

2-8.cpp

```
#include<iostream>
... 32766 32767 -32768 -32767 -32766 ...

int main() {
    short a=-30000;          (-30000 - 10000) - (-32768) = 7232
    short b=10000;          32768 + (-7232) = 25536
    short c = a - b;        此種數字超出範圍的狀況，專業術語稱為
    short d = a - b / 1000; Underflow (向下溢位)
    short e = ( a - b ) / 1000; 若是加上去超出範圍，稱為
                                Overflow (溢位)

    cout << "a=" << a << endl;
    cout << "b=" << b << endl;
    cout << "c=" << c << endl;
    cout << "d=" << d << endl;
    cout << "e=" << e << endl;
    return 0;
}
```

一元運算子 (Unary Operation)

- 一元運算子即僅需一個運算元即可進行計算的運算子
 - ++: 將運算元的值遞增 (increment)，在算術運算時即為+1
 - --: 將運算元的值遞減 (decrement)，在算術運算時即為-1
- 範例：
 - A=3;
 - A++; ← A變為4
 - A--; ← A的值變為3
- 但是！

2-9.cpp

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

int main() {
    int A=5;

    cout << A++ << endl;
    cout << ++A << endl;
    cout << A++ << endl;
    cout << A << endl;

    return 0;
}
```

5
7
7
8

前置 (prefix) 與後置 (postfix)

- 前置
 - ++a, --b, ++c, ...
 - 在這行敘述執行之**前**完成該運算
- 後置
 - a--, b++, ...
 - 在這行敘述執行之**後**再進行該運算



隨堂練習

- 請寫一程式，定義變數 p 、 E 、 b 、 h 、 x 、 L ，分別給予一適當的值，並根據下式計算出 $w(x)$ (單號)、 $\theta(x)$ (雙號)。(假設此問題為矩形樑斷面!)

$$w(x) = -\frac{p x^2 (6 L^2 - 4 x L + x^2)}{24 E I}$$

$$\theta(x) = -\frac{p x (3 L^2 - 3 x L + x^2)}{6 E I}$$