

CT3407301/CT3407701  
C/VB程式與應用

C/VB Programming and Applications

為什麼要學習程式寫作?

Why Programming ?

為什麼要學習程式寫作？

- 程式寫作能力 == 解決問題之能力 + 表達能力
  - 分析、拆解、表達問題
  - 以電腦程式語言的語法描述解法
- 可以隨心所欲地應用電腦解決問題
  - 電腦硬體 ← 電腦軟體 ← 程式
- 其它
  - 建立自己的第二專長、了解電腦應用程式背後的邏輯、充份利用電腦之功能(運算、儲存、傳輸)...

- It has often been said that a person does not really understand something until he teaches it to someone else.
- **Actually a person does not really understand something until after teaching it to a computer.**

- Donald Knuth

為什麼學 C/C++

Why C/C++

為什麼學 C/C++

- 所有現代的程式語言 (C#, Java, JavaScript, etc.) 的語法都是基於 C/C++ 。
  - 如果你有興趣寫手機程式或遊戲
    - Android → Java
    - iPhone / iPad → Objective C (一種 C 語言)
    - 網頁 → JavaScript
- 可以介紹到重要的程式設計觀念

## 課程目的

學習電腦程式之寫作以

- 強化解析問題與電腦應用之能力;
- 提升資料/資訊處理的能力與速度;
- 進一步了解電腦程式運作之原理;
- 訓練表達能力;
- 期望將來能充份運用電腦協助解決工程問題。

## 課程大綱

- 基本程式架構與輸入輸出
- 檔案操作
- 變數與算術運算
- 程式流程控制
- 迴圈與陣列
- 函式與標準函式庫
- 多維陣列、指標、與動態記憶體配置
- 變數之儲存等級與函式呼叫之參數傳遞
- 函式覆載與遞迴函式

## 課本

無指定課本，以兩班的上課講義為主。

<http://140.118.105.174/Courses/CVB/> (also on BB)

<http://hungming.ct.ntust.edu.tw/2014spring/ct3407/ct3407.htm>

但請各位同學自己選定一本自己最能接受的 C++ 書籍。

儘量不要找 Visual C++、視窗程式設計 (或至少要有介紹 C++ 程式語法的書籍) 的書，因為其內容大部份都不會用到 (我們沒有要寫視窗程式) ...

## 使用之開發環境

Microsoft Visual Studio 2012 Express

<http://www.microsoft.com/zh-tw/download/details.aspx?id=34673>

## 學期成績

30%: 期中考

30%: 期末考

30%: 作業

10%: 課堂參與

## 作業成績

作業遲交一星期以九折計算。

遲交超過一星期以零分計算。

抄襲以**零分**計算?!

## 答疑時段?

- 助教: 洪詩晴
- Email: m10205318@mail.ntust.edu.tw

## Lecture 1

基本程式架構  
變數與算術運算

## 今日大綱

- Visual Studio 2012 Express 使用
- 第一個程式
- 程式如何被執行
- 變數 (variables)

Microsoft Visual Studio 2012 Express 使用

## 操作 (I)

- 檔案 → 新增 → 專案 → Win32 主控台應用程式 → 輸入名稱 → 選擇檔案所在位置 → 確定
- 進入 win32 應用程式精靈後: 下一步 → 主控台應用程式/空專案 → 完成
- 在「原始程式檔」上按右鍵 → 加入 → 新增項目 → C++ 檔 (.cpp) → 輸入名稱 “*ex01.cpp*” → 新增
- 現在可以開始寫程式囉，完成後按 **CTRL-F5** 即可開始執行。

第一個程式

The first C++ program

## 1-1.cpp

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

int main() {
    // 輸出 "Hello World" 至螢幕上
    cout << "Hello World";

    /* 函式執行完畢，回傳 0 */
    return 0;
}
```

此被稱為**原始程式碼 (source code)**

## 觀察

- 程式由許多的敘述 (statement) 組成
- C/C++ 的敘述以分號 (semi-colon) 作為結尾
- 有兩行我們看得懂的部份 (有中文的部份) 為給人看的註解 (comment, remark)
- 有幾行看似英文的敘述即為程式碼的部份
- 程式在執行時，**由上而下逐列執行**。

## #include <iostream>

- 目的：將 **iostream** 這個標頭檔的內容「含括」(include) 進來。
  - 標頭檔 (header files)：定義了一些可使用的函式 (function) 或是物件 (object)
  - **iostream**: 定義了以「串流」(stream) 進行輸入與輸出的物件
    - **cout**: console **output** - 從螢幕輸出字元/字串
    - **cin**: console **input** - 從鍵盤輸入字元/字串
    - **console**: 控制台、主控台

## using namespace std;

- 目的：告訴電腦在尋找一**函式**或**物件**時，可在 **std** (**standard**, 標準) 這個命名空間內搜尋。
- 說明：現在對此不清楚沒有關係，我們本學期的程式大多都會需要此行敘述。

## 主程式區塊 main program block

```
int main() {
    ... 主程式內容 ...
    return 0;
}
```

{ ... } 定義了一個**區塊 (block)**

`int main() {...}` 定義了主程式區塊，此部份以後會再更詳細的介紹，目前僅需要記得整個 C/C++ 的主程式定義如上所示即可。

## C/C++ 程式註解

### 單列註解

範例：`// output "Hello World" to console`  
說明：`//` 之後所有的字都被認為是註解的一部份直到換行為止

### 多列註解

範例：`/* return value 0 */`  
說明：

- `/*` 表示註解區塊的開始
- `*/` 表示註解區塊的結束
- 可用來註解多列

```
/*
這是一個註解
這裡還是註解
這列還是註解
*/
```

```
/* 這也是一列註解 */
```

```
// 這也是一列註解
```

```
// 這也是一列註解 //
```

## 為什麼要寫註解？

Why write **comments** or **remarks**?

- 讓 **programmer** 提醒自己程式某片段程式的功能
- 讓別人比較有機會看得懂自己寫的程式
- 讓程式變得比較好維護
- 讓自己以後有機會看懂自己的程式

```
cout << "Hello World";
```

- 目的：將“Hello World”這個「字串」(string) 輸出至螢幕上。
- 說明：
  - **cout**: console output, 為一個「物件」(object), 用來將接收到的資料輸出至螢幕上。
  - **<<**: 串流運算子 (stream operator), 用來將右邊 (RHS, Right Hand Side) 的資料傳送到左邊 (LHS, Left Hand Side) 的物件。
  - “Hello World”: 為字串資料, 在C/C++中, **字串**必需使用**雙引號**“”括起來。

## 1-1.cpp

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

int main() {
    // 輸出 "Hello World" 至螢幕上。
    cout << "Hello World";

    /* 函式執行完畢, 回傳0。 */
    return 0;
}
```

## 變數 Variable

電腦具有處理大量資料的能力, 但是資料必需先存在**記憶體**中, 而變數就是用來幫助我們存取記憶體的**工具**...

## 變數 Variable

- 記憶體是一連串 0 與 1 的組合 → bit
- 電腦的記憶體單位
  - Bit → 0, 1
  - Byte → 8 bits →  $2^8=256$  → 0 - 255 or -128 - 127
- 在高階電腦程式語言裡, 我們使用變數來**利用記憶體資源**, 用來進行資料的儲存與計算。
- 一個變數具有:
  - **名稱** (identifier / 識別項): 用來區別不同的資料存放位置
  - **型別** (type / 型態): 代表該變數所儲存資料的意義

## 1-2.cpp

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

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

    b = 4;
    c = a + b;
    cout << "a+b=" << c;

    return 0;
}
```

宣告一個變數 **a**, 並初始其值為 3

宣告兩個變數 **b** 與 **c**

令 **b** 變數的值为 4

令 **c** 變數的值为 **a+b** 的計算結果

輸出字串 "a+b=" 以及變數 **c** 的值

**a+b=7**

## 說明

- 所有變數在使用**前**必需先被宣告 (declaration)
  - `int a = 3;` ← 宣告一個變數, 名稱為 a, 並賦予初始值為 3
  - `int b, c;` ← 宣告兩個變數, 名稱為 b 和 c。
  - 在 C 語言時代, 所有變數都必需在**所有可執行敘述之前**宣告完成。在 C++, 變數宣告只要在**使用前**宣告即可, 沒有一定要放在程式的開頭。
- `b = 4;`
  - 此行敘述 (statement) 執行了指定 (assignment) 的動作, 將右側的運算元 (operand) 之值指定給左側的運算元。
  - 指定 (=) 為一種二元運算子 (binary operator) – 需要兩個運算元的動作或計算, 不要把它和「相等與否」(==) 混在一起。
  - 亦即, 將值 4 指定給變數 b、或令 b 的值为 4。

## 說明

- `c = a + b;`
  - 此行敘述中共有兩個二元運算子, 第一個為 +、第二個為 =
  - 由於運算子之優先順序 (precedence) 的緣故, + (加法運算) 先執行、= (指定) 運算後執行。
  - 此行敘述即將 a + b 的值計算完後, 將其和 (sum) 指定給 c。
- `cout << "a+b=" << c;`
  - 此行敘述 (statement) 中, << 為資料流插入運算子, 目的在螢幕上輸出 **字串** "a+b="與變數 c 之值。
  - 如果需要輸出許多資料時, 再繼續加 << 和要輸出的資料。  
e.g. `cout << "a=" << a << ", b=" << b << ", c=" << c;`

## 變數名稱、識別項 name, identifier

- 用來區別不同的變數, 且大小寫視為不同 (case sensitive)
- 文字、數字、與底線符號的組合
- 第一個字元必需為文字或是底線
  - 正確: `myName`, `_myName2`, `my_Name5`,
  - 錯誤: `4Name`, `4_name`, `my-Name`
- 變數名稱不可為**保留字 (reserved words)**, 如 C/C++ 語言中的指令 `while`, `for`, `if`, ...
- 慣例:
  - 名稱盡量取得有意義: `myName`, `my_name`
  - 由底線開頭的名稱保留給各函式庫內部之變數
  - 名稱全部大寫保留給常數變數

## 正確或錯誤的變數名稱?

name  
 8dogs  
 myGoodFriend  
 MyDog  
 CSI  
[www.ntust.edu.tw](http://www.ntust.edu.tw)  
 ABcdEFG  
 Iam\_28\_yearsOld  
 13Ghosts

## 1-2.cpp

```

#include<iostream>
using namespace std;

int main() {
    int a=3;        // 變數宣告時可以給定初始值
    int b, c;      // 同一個型別的變數可以一起宣告

    b = 4;
    c = a + b;
    cout << "a+b=" << c;

    return 0;
}
  
```

## 基本變數型態 basic 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	準確
整數	<code>short int **</code>	2	-32768 ~ 32767	準確
	<code>int</code>	4	-2 <sup>31</sup> ~ 2 <sup>31</sup> -1	準確
	<code>long int ***</code>	4	-2 <sup>31</sup> ~ 2 <sup>31</sup> -1	準確
浮點數	<code>float</code>	4	-3.4x10 <sup>38</sup> ~ +3.4x10 <sup>38</sup>	6-7 位有效數字
	<code>double</code>	8	-1.7x10 <sup>308</sup> ~ +1.7x10 <sup>308</sup>	14-15 位有效數字
	<code>long double</code>	8	-1.7x10 <sup>308</sup> ~ +1.7x10 <sup>308</sup>	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。

\*\* 可以簡寫為 `short`, `int` 可省略不寫

\*\*\* 可以簡寫為 `long`, `int` 可省略不寫

37

<https://www.youtube.com/watch?v=9bZkp7q19f0>

The first Video that breaks YouTube's counter...

2<sup>31</sup>-1=2,147,483,647

<http://www.buffintopost.com/2014/12/03/gangnam-style-youtube-n-6261332.html?rcid=fbkinkushrmp0000063>

38

### 1-3.cpp

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

int main() {
    int a = 3;           // a 為整數變數
    double b = 3.14159; // b 為雙精度浮點數

    float c = 3.14f;    // c 為單精度浮點數
    short d = 123;      // d 為短整數

    b = c * d;          // 隱性型別轉換
    c = b * a;          // 隱性型別轉換，會有警告

    return 0;
}
```

39

### 說明

- 在1-3.cpp中，宣告了四個變數 a, b, c, d，分別有自己的名稱、型別、與初始值。
- b = c \* d;**
  - 先計算 c \* d (float \* short)，在此由於兩個運算子 c 與 d 的型別不同，電腦會先進行型別轉換使其一致 (皆為 float) 後再相乘，此動作稱為「隱性型別轉換」(implicit type conversion)。
  - 計算得到結果後，將其計算結果存入 b 變數 (double)。此時又發生一次隱性型別轉換為 double 後存入 b 中。由於 double 的精度較高，編譯器不會發出警告。
- c = b \* a;**
  - 先計算 b \* a (double \* int)，同樣地會發生「隱性型別轉換」，將 int 轉換為 double 後再進行計算。
  - 計算結果 (double) 要存入變數 c (float)，由於 c 變數的精度較低，編譯器會發出警告。

40

### 有沒有隱性型別轉換? 幾次? 會不會有警告?

```
short a=1;
int b=2;
long c=3;
float d = 4.0;
double e = 5.0;

b = c + a;
d = d * e;
d = a * e;
e = e / a;
c = d + a;
a = b / c;
e = e + d;
```

意義	變數型態	記憶體量
字元	char	1
	short	2
整數	int	4
	long	4
	float	4
浮點數	double	8
	long double	8

資料會有損失時才會發出警告 (e.g. 1.34 → 1; 1.5714 → 1.57)

41

### 1-4.cpp

```
#include<iostream>
using namespace std;
int main() {
    char a = 'A';           宣告一個變數 a，並初始為 'A' 字元

    cout << a << endl;      輸出 a 變數的值
    cout << (int) a << endl; 以整數方式輸出 a 變數的值
    a = 67;                 將 67 存入 a 變數
    cout << a << endl;      輸出 a 變數的值
    cout << sizeof(a) << endl; 輸出 a 變數所佔記憶體大小
    cout << sizeof(char) << endl; 輸出 char 型別所佔記憶體大小
    return 0;
}
```

A  
65  
C  
1  
1

42

### 說明

- 字元型別為一特殊的整數型別，其實質資料為一個整數，此例來說以美國國家標準碼 (ASCII, American Standard Code for Information Interchange) 存入與輸出。
- cout << (int) a << endl;
  - 將變數 a 「強制型別轉換」或「顯性型別轉換」為 int 型別後輸出，因此會看到輸出一整數 65 (代表 A)
- a = 67;
  - 將 67 存入變數 a，此處會發生隱性型別轉換，將 67 轉換為字元型別後存入 a。在 C/C++ 中直接寫出來 (e.g. 67) 的整數資料為 int 型別。
- sizeof() 用來取得小括弧內的變數或是型別名稱所需的記憶體空間大小，單位為 bytes。

## 字元

- 何謂字元 (character)
  - 它其實是一種整數，但它的意義是給人看的「符號」。
  - 每個電腦上看到的「符號」背後都依循某一種編碼標準
    - 英文字母: ASCII (American Standard for Information Interchange)
    - 正體中文: Big-5、UTF-8 (萬國碼)
    - 簡體中文: GB-2312

## 基本變數型態 basic variable type

- 整數型態 (integer)
  - 三種: short int、int、long int  $\leftarrow \rightarrow$  short、int、long
    - 記憶體用量與 CPU & 編譯器 (compiler) 有關
    - 一般來說 int: 4 bytes (32 bits  $\rightarrow 2^{32}=4,294,967,296 \sim 40$ 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。

## 1-5.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)

## 基本變數型態 basic variable type

- 實數, 浮點數 (real or floating point numbers) 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  $\leq$  double
    - long double: 佔用記憶體量  $\geq$  double
  - IEEE-754 規定了浮點數在記憶體裡的儲存方式。
  - Example:
    - float a;  $\leftarrow$  宣告變數 a, 型別是浮點數
    - double b = 3.0e8;  $\leftarrow$  宣告變數 b, 並初始化為  $3 \times 10^8$

## 1-6.cpp

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

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

    return 0;
}
```

宣告一倍精度浮點數 a  
並初始化為 3.14159..宣告一單精度浮點數 b 並初始化為  $3 \times 10^8$ 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

## 1-7.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;
}
```

宣告一布林變數 a，並初始為 false  
 宣告一布林變數 b，並初始為 true  
 宣告一布林變數 c，並初始為 100

輸出變數a的值  
 輸出變數b的值  
 輸出變數c的值

在 C/C++ 中，非 0 值為真 (true)，0 為假 (false)

0  
1  
1

## 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**。

## 1-8.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`

## 字串 (string)

- 在 C++ 裡有兩種字串
  - C-字串，之後會介紹，由 C 語言繼承過來。
  - C++字串，方便易用，但不能用在 C 語言。
- C++字串型別為 `string`、需要 `#include <string>`
- 記得在 C++ 裡字串前後需要加雙引號 "

## 1-9.cpp

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

int main() {
    string weather="今天天氣很好\n";
    cout << weather;

    weather="明天會下雨\n";
    cout << weather;
    return 0;
}
```

需有這一行才能使用 C++ 的字串功能

其中，`\n` 為一特別的字元，會在印出時將列印位置移到下一列的開頭。

宣告 weather 為字串型別的變數

## 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?
- 如何確定讓一個變數的值不會在程式執行時被修改?

## 隨堂練習

- 請寫一個程式,
  - 宣告 **三個變數** 分別儲存你的出生年、月、日
  - 逐列印出自己的姓名、學號、班級, 並利用之前宣告的 **三個變數** 輸出你的出生年月日,
  - 並印出其它你/妳想要透露給我或是助教的資訊 (e.g. 想要 all pass ...)
- 完成後上傳至 Moodle