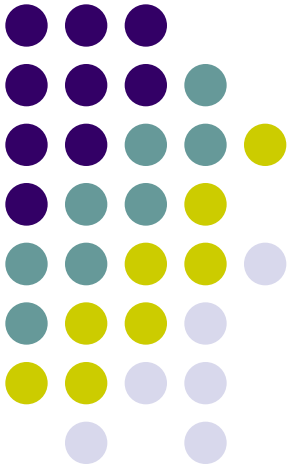
第二章
認識C++

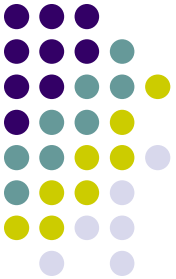
學習C++的基本語法

認識關鍵字與識別字的不同

學習程式碼偵錯的流程

學習如何提高程式的可讀性



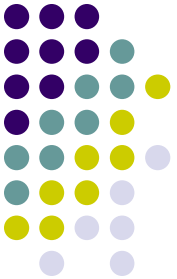


簡單的C++程式 (1/5)

```
01  // prog2_1, 簡單的 C++程式
02  #include <iostream>              // 含括 iostream 檔案
03  #include <cstdlib>               // 含括 cstdlib 檔案
04  using namespace std;             // 使用 std 名稱空間
05  int main(void)
06  {
07     int num;                      // 宣告整數 num
08     num=3;                        // 將 num 設值為 3
09     cout << "I have " << num << " apples."<< endl; //印出字串及變數內容
10     cout << "You have " << num << " apples, too." << endl;
11     system("pause");
12     return 0;
13  }
```

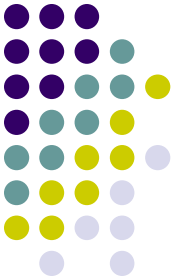
輸出如下：

```
I have 3 apples.
You have 3 apples, too.
```



簡單的C++程式 (2/5)

- C++的註解是以「//」記號開始，至該行結束來表示註解的文字。
- C++ 語言中，前面有「＃」符號的程式列，在編譯時會透過編譯器內建的前置處理器，在編譯其他程式碼之前就先被讀入。
  - #include <iostream>與#include <cstdlib>則是告訴編譯器把iostream與cstdlib檔案含括進來
- iostream為 input/output stream 的縮寫，所有跟輸入輸出有關的函式都定義在這。
- cstdlib 是標準函示庫 standard library 的縮寫。



簡單的C++程式 (3/5)

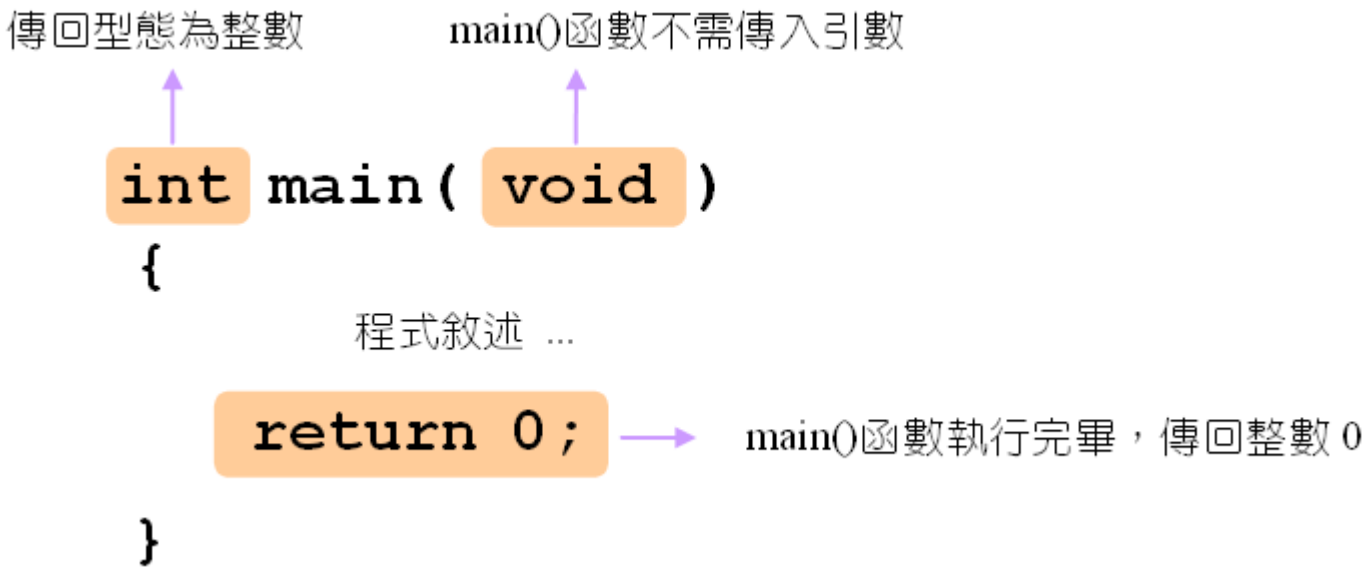
- using namespace std 是用來設定名稱空間(name space)為std 。
- cin、cout 等輸入及輸出功能的指令是定義在「std」命名空間中，所以要使用這些指令時的語法為「std::cin」、「std::cout」，每次使用都要重複輸入「std::」豈不是太麻煩了？只要使用 using 指令宣告要使用的命名空間，往後在此命名空間中的指令都可以省略命名空間的名稱。

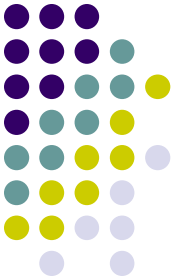
簡單的C++程式 (4/5)

- C++ 程式指定會由 main() 函式開始執行，所以每個程式都要有main() 函式，程式開始執行時會由 main() 函式區塊中的程式碼一列一列依序向下執行。
- C++ 語言規定 main() 函式的傳回值必需是整數，所以在 main() 函式的最後要以「return 整數」做為函式的結束指令，通常使用「return 0」即可，表示由main() 函式返回，也就是結束 C++ 程式的執行。

簡單的C++程式 (5/5)

- 在 C++ 語言中，函式中的程式碼必須被包含在一對大括號之中，即函式是以「{」開始，「}」結束。綜合上述規則，主程式區塊的寫法為：



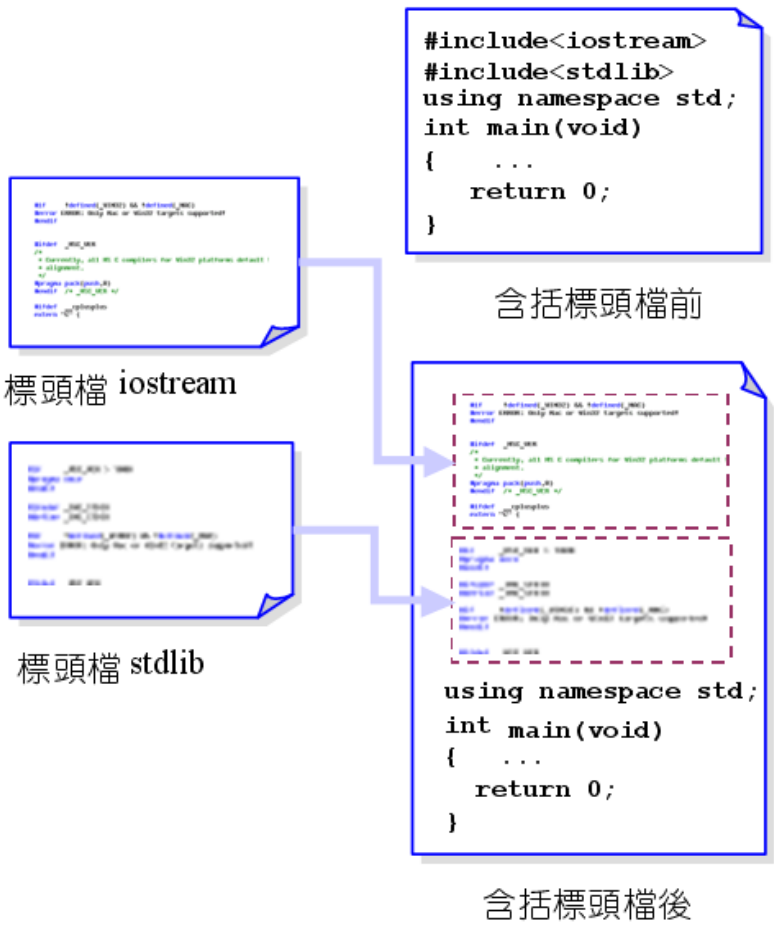


關於ANSI/ISO C++的標準

- C++標頭檔可能有下面四種型態
  - C語言的標頭檔：以「.h」結尾
  - C++語言的標頭檔：以「.h」結尾
  - ANSI/ISO C++新標準的標頭檔：沒有副檔名
  - 從C移植過來的標頭檔，沒有副檔名，字首加小寫的c
- 新版的ANSI/ISO C++必須利用using namespace來設定名稱空間為std

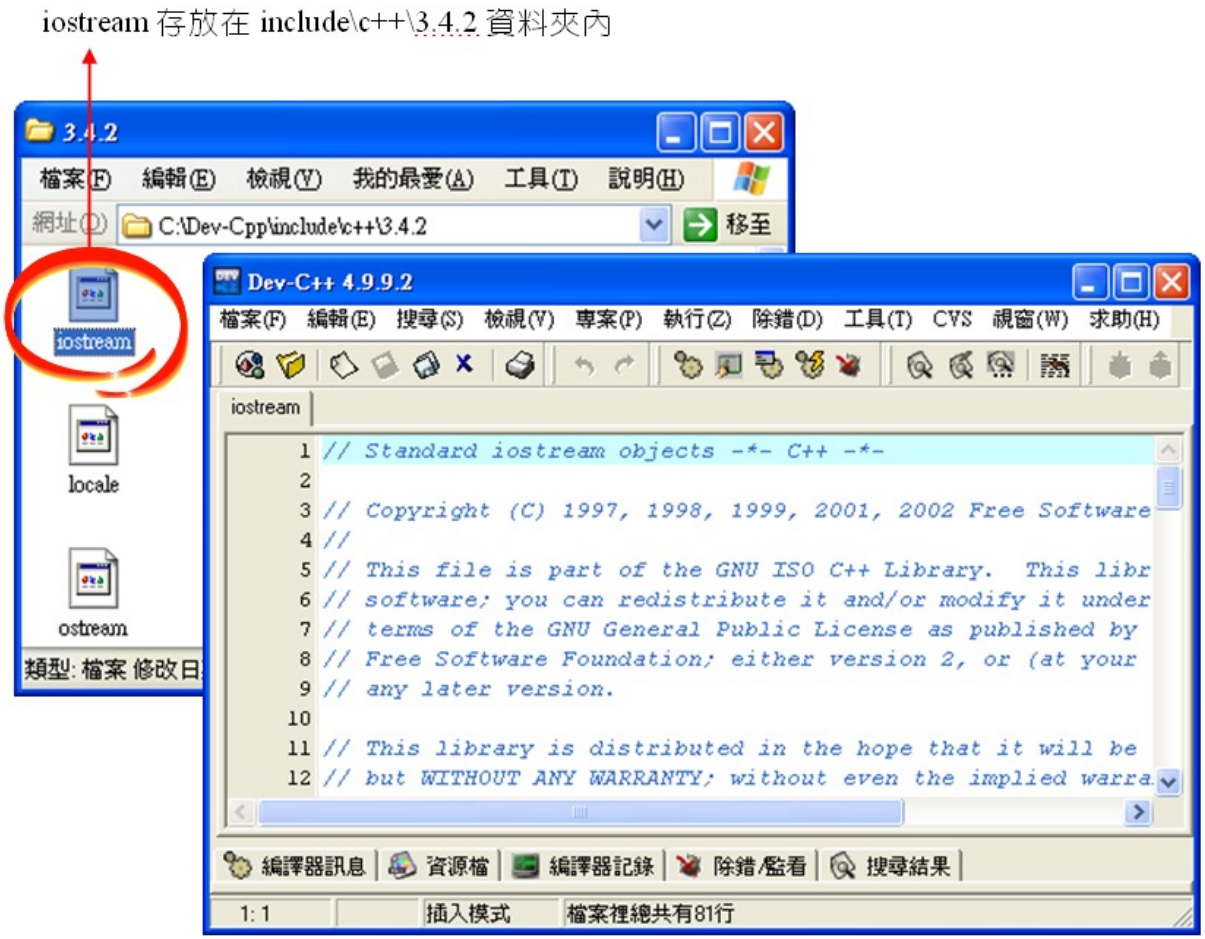
#include指令及標頭檔 (1/2)

- **標頭檔（header file）永遠被含括在程式碼的起頭處**
- 右圖為含括動作前後的比較

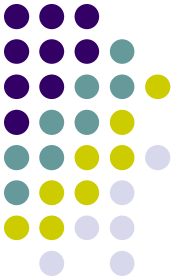


#include指令及標頭檔 (2/2)

- 標頭檔的內容

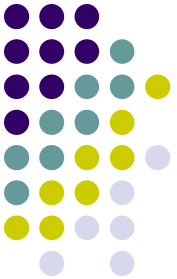


主函數、大括號、區塊與主體(1/4)



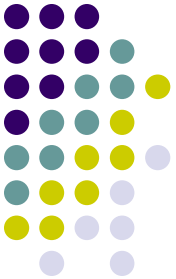
- 主函數main()
  - main() 是程式執行的開端
  - 每個C++程式必須有一個main()，而且只能有一個
- 大括號、區塊及主體
  - 區塊從左大括號（{）開始，到右大括號（}）結束
  - 指令敘述結束時，以分號「;」做結尾

主函數、大括號、區塊與主體(2/4)



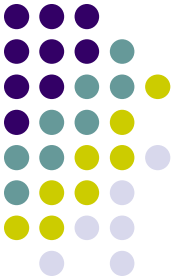
- C++ 程式執行時，電腦會開啟一個新的視窗來顯示執行結果，當程式執行完畢後，就自動關閉顯示結果的視窗。
- 由於電腦執行的速度非常快，使用者往往只見到螢幕閃了一下，什麼也沒有看見，視窗就關閉了，使用者根本不知道執行結果是什麼！

主函數、大括號、區塊與主體(3/4)



- 通常設計者會在主程式的倒數第二列（最後一列是「return 0」）加入暫停指令，讓程式在結束（即關閉視窗）前可以暫停一下，讓使用者觀察執行結果。
- C++ 暫停指令的語法為：
  - system("pause");

主函數、大括號、區塊與主體(4/4)

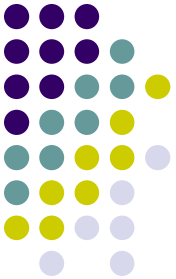


- 下面的範例說明什麼是程式區塊

```
01  // prog2_2, 程式的區塊
02  #include <iostream>              // 含括 iostream 檔案
03  #include <cstdlib>               // 含括 cstdlib 檔案
04  using namespace std;
05  int main(void)                   // main()區塊開始
06  {
07     int num=6;                    // 宣告整數 num
08     cout << "I have " << num << " apples." << endl;
09
10     system("pause");
11     return 0;
12  }                                // main()區塊結束
```

main()的區塊

```
/* prog2_2 OUTPUT---

I have 6 apples.
-------------------*/
```

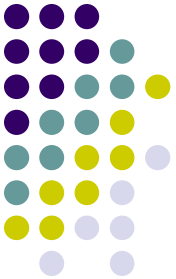


變數 (1/2)

- 變數的宣告

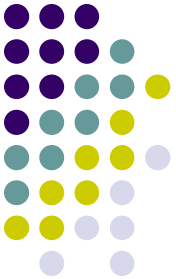
```
int num;                  // 宣告 num 為整數變數
int num,num1,num2;        // 同時宣告 num,num1,num2 為整數變數
```

- 變數的資料型態
  - 資料型態有char, int, bool, long, short, float與double等
  - 數值型態變數可分為有號（sign）或是無號（unsigned）



變數 (2/2)

- 變數名稱與限制
  - 通常會以變數所代表的意義來取名。
  - 自訂變數的名稱不能使用到關鍵字。
  - 變數名稱的字元可以是英文字母、數字或底線。
  - 名稱中不能有空白字元，且第一個字元不能是數字



變數的設值

- 方法1 在宣告的時候設值

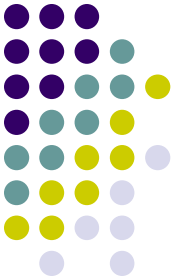
```
int num=9;                    // 宣告變數，並直接設值
```

- 方法2 宣告後再設值

```
int num1,num2;     // 宣告變數 num1,num2
char ch;           // 宣告字元變數 ch
num1=12;           // 設值給變數 num1
num2=38;           // 設值給變數 num2
ch ='w';           // 設值給字元變數 ch
```

- 方法3 在程式中的任何位置宣告並設值

```
int num;           // 宣告變數
   ...
num=9;             // 需要用到變數時，再行設定
```



為什麼要宣告變數

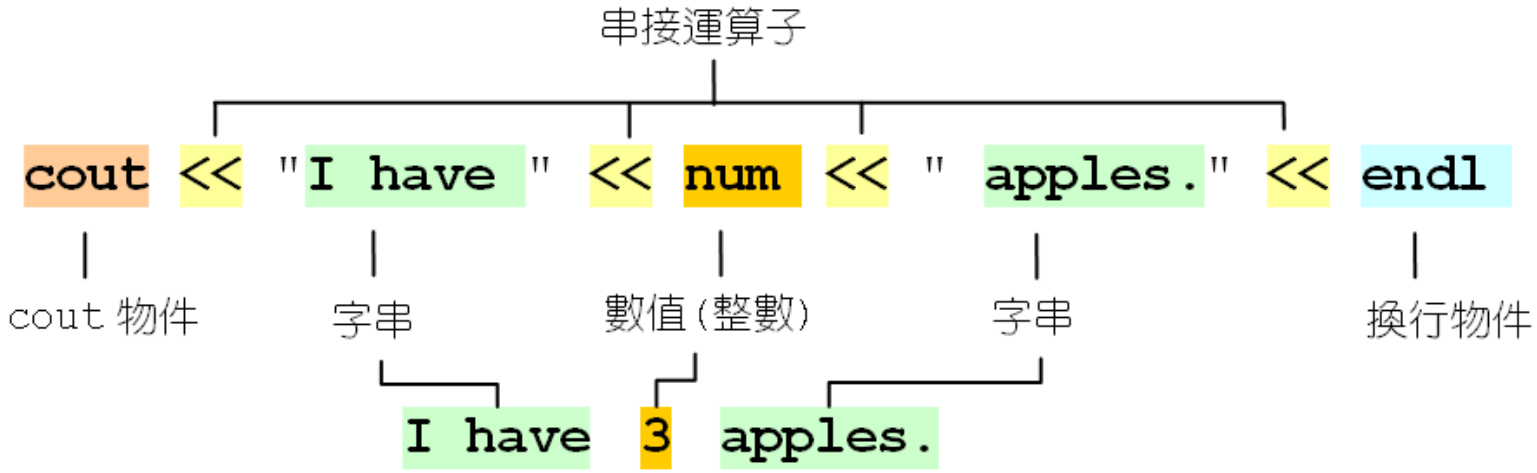
- 直譯式語言不需要宣告變數
- 編譯程式可找到錯誤的變數名稱，避免變數名稱誤用
- 將變數集中宣告時，在系統維護上也就容易得多

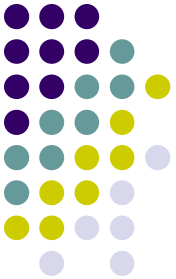
cout與串接運算子<<

- C++是採cout與「串接運算子<<」來輸出

```
cout << "I have " << num << " apples." << endl;
cout << "You have " << num << " apples, too." << endl;
```

- 以cout顯示字串：



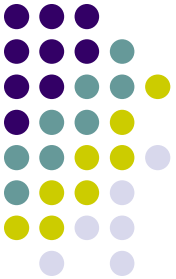


換行輸出的範例

- 下面是把字串換行輸出的範例

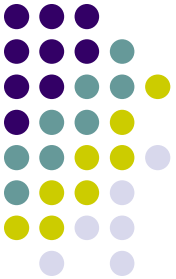
```
01  // prog2_3, endl 與"\n"的使用
02  #include <iostream>        // 含括 iostream 檔案
03  #include <cstdlib>         // 含括 cstdlib 檔案
04  using namespace std;
05  int main(void)
06  {
07     cout << "I love C++." << endl << "You love C++, too.\n";
08     cout << "We all love C++." << "\n";
09
10     system("pause");
11     return 0;
12  }
```

```
/* prog2_3 OUTPUT---

I love C++.
You love C++, too.
We all love C++.
--------------------*/
```



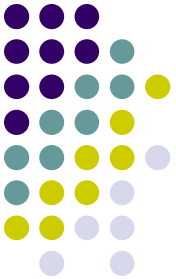
字串與值

- 輸出及換行指令
  - 輸出內容主要分為兩種型態：如果是字串，則必須在字串前後加上「”」符號，例如：
    - cout << "這是字串示範";
  - 如果是數值，則直接輸出數值即可：
    - cout << 1234;
  - 當然，也可以在數值前後加上「”」符號，將數值做為字串使用，例如：
    - cout << "1234";。



識別字 (1/2)

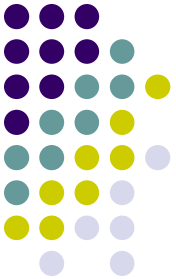
- 識別字是使用者用來命名變數或者是函數的文字
- 變數與函數的名稱均是識別字（identifier）
- 為識別字命名時，只要能代表變數的意義即可



識別字 (2/2)

- 識別字的習慣命名原則

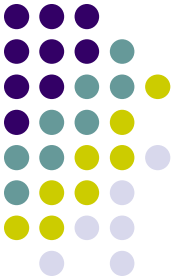
識別字	命名原則	範例
常數	全部字元皆由英文大寫字母及底線組成	PI MIN_NUM
變數	英文小寫字母開始，若由數個英文單字組成，則後面的英文字由大寫起頭，其餘小寫	radius rectangleArea myAddressBook
函數	英文小寫字母開始，若由數個英文單字組成，則後面的英文字由大寫起頭，其餘小寫	show addNum mousePressed
類別	英文大寫字母開始，若由數個英文單字組成，則後面的英文字由大寫起頭，其餘小寫	Cbbb CWin MaxSize



關鍵字

- 關鍵字（keyword）是編譯程式本身所使用的識別字
- C++使用的關鍵字：

```
asm         auto        bool        break       case
catch       char        class       const       const_cast
continue    default     delete      dynamic_cast
double      do          else        enum        explicit
extern      false       float       for         friend
goto        if          inline      int         long
mutable     namespace   new         operator    protected
private     public      register    reinterpret_cast
return      short       signed      sizeof      static_cast
static      struct      switch      template    this
throw       true        try         typedef     typeid
typename    union       unsigned    using       virtual
void        volatile    wchar_t     while
```

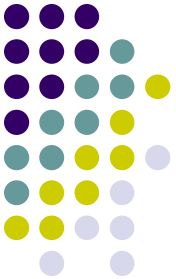


錯誤的分類

- 錯誤分為
  - 語法錯誤（syntax error）

    語法錯誤就是語法不符合C++的規定
  - 語意錯誤（semantic error）。

    語法正確，但執行結果不符合要求

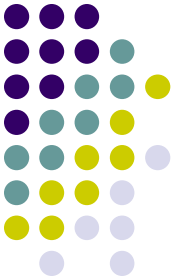


語法錯誤

- 下面是有語法錯誤的程式

```
01  // prog2_4, 有錯誤的程式
02  #include <iostream>          // 含括 iostream 檔案
03  #include <cstdlib>           // 含括 cstdlib 檔案
04  using namespace std;
05  int main(void)
06  {
07     int num;                  // 宣告整數 num
08     num=2;                    // 將 num 設值為 2
09     cout << "You have " << num <<" books."<< endl; 印出字串及變數內容
10     cout << "I want " << num << " books.  << endl;
11     system("pause")
12     return 0;
13  )
```

```
/* prog2_4 OUTPUT 除錯後的結果 ---

You have 2 books.
I want 2 books.
--------------------------------*/
```

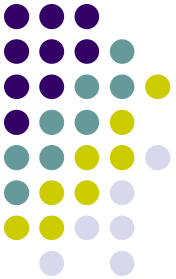


語意錯誤

- 執行結果不符合要求，可能犯了語意錯誤

```
01  // prog2_5, 語意錯誤的程式
02  #include <iostream>          // 含括 iostream 檔案
03  #include <cstdlib>           // 含括 cstdlib 檔案
04  using namespace std;
05  int main(void)
06  {
07      int num1=35;             // 宣告整數變數 num1，並設值為 35
08      int num2=28;             // 宣告整數變數 num2，並設值為 28
09
10      cout<<"I have "<<num1<<" books."<<endl;
11      cout<<"You have "<<num2<<" books."<<endl;
12      cout<<"We have "<<(num1-num2)<<" books."<<endl;
13      system("pause");
14      return 0;
15  }
```

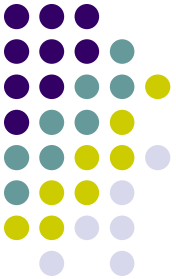
```
/* prog2_5 OUTPUT---

I have 35 books.
You have 28 books.
We have 7 books.
--------------------*/
```



程式碼請用固定字距 (1/2)

- 下面的程式碼是用固定字距

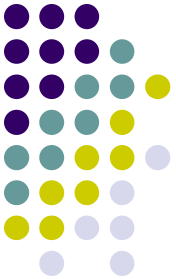
```
// 使用固定字距的程式碼，字型為 Courier New
#include <iostream>        // 含括 iostream 檔案
#include <cstdlib>         // 含括 cstdlib 檔案
using namespace std;
int main(void)
{
   cout << "We all love C++." << "\n";
   system("pause");
   return 0;
}
```



程式碼請用固定字距 (2/2)

- 下面的程式碼是用比例字距

```
// 使用非固定字距，且斜體字的程式碼，字型為 Times New Roman
#include <iostream>          // 含括 iostream 檔案
#include <cstdlib>           // 含括 cstdlib 檔案
using namespace std;
int main(void)
{
    cout << "We all love C++."<<"\n";
    system("pause");
    return 0;
}
```

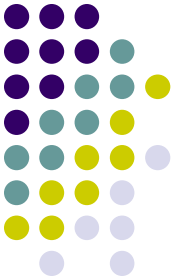


將程式碼縮排 (1/2)

- 有縮排的程式碼可提高可讀性

```
01  // prog2_6, 有縮排的程式碼
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int num1=12;
08     int num2=5;
09     cout << num1 << "+" << num2 << "=" << num1+num2 << endl;
10     system("pause");
11     return 0;
12  }
```

```
/* prog2_6, prog2_7 OUTPUT---

12+5=17

-----------------------------*/
```

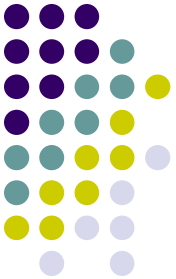


將程式碼縮排 (2/2)

- 下面的例子因為撰寫風格的關係，閱讀起來較為困難

```
01  // prog2_7, 沒有縮排的程式碼
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {int num1=12;
07  int num2=5;
08  cout<<num1<<"+"<<num2<<"="<<num1+num2<<endl;
09  system("pause");
10  return 0;}
```

```
/* prog2_6, prog2_7 OUTPUT---
12+5=17
-----------------------------*/
```



註解

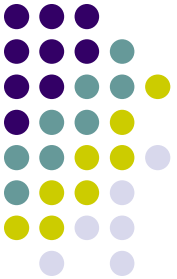
- 註解有助於程式的閱讀與偵錯。
- 以「/*」開始，「*/」結尾，將欲註解的文字括起來

```
// prog2_7, examples
// created by Wien Hong
```

以「//」符號註解

```
/* This paragraph demonstrates the capability
   of comments used by C++   */
```

於「/*」和「*/」符號之間的文字均是註解



使用註解的目的

- 在程式碼起始處加入一段說明文字
- 將變數、函數、類別或是程式碼的作用寫出
- 在除錯的過程中，避免重複輸入，浪費時間


-The End-