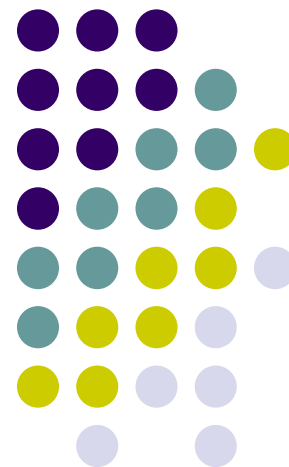
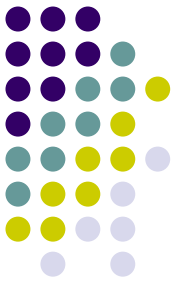


# 第一章

## 認識C++

認識C++的歷史  
瞭解程式的規劃與實作  
撰寫第一個C++程式  
學習程式碼的編譯與執行





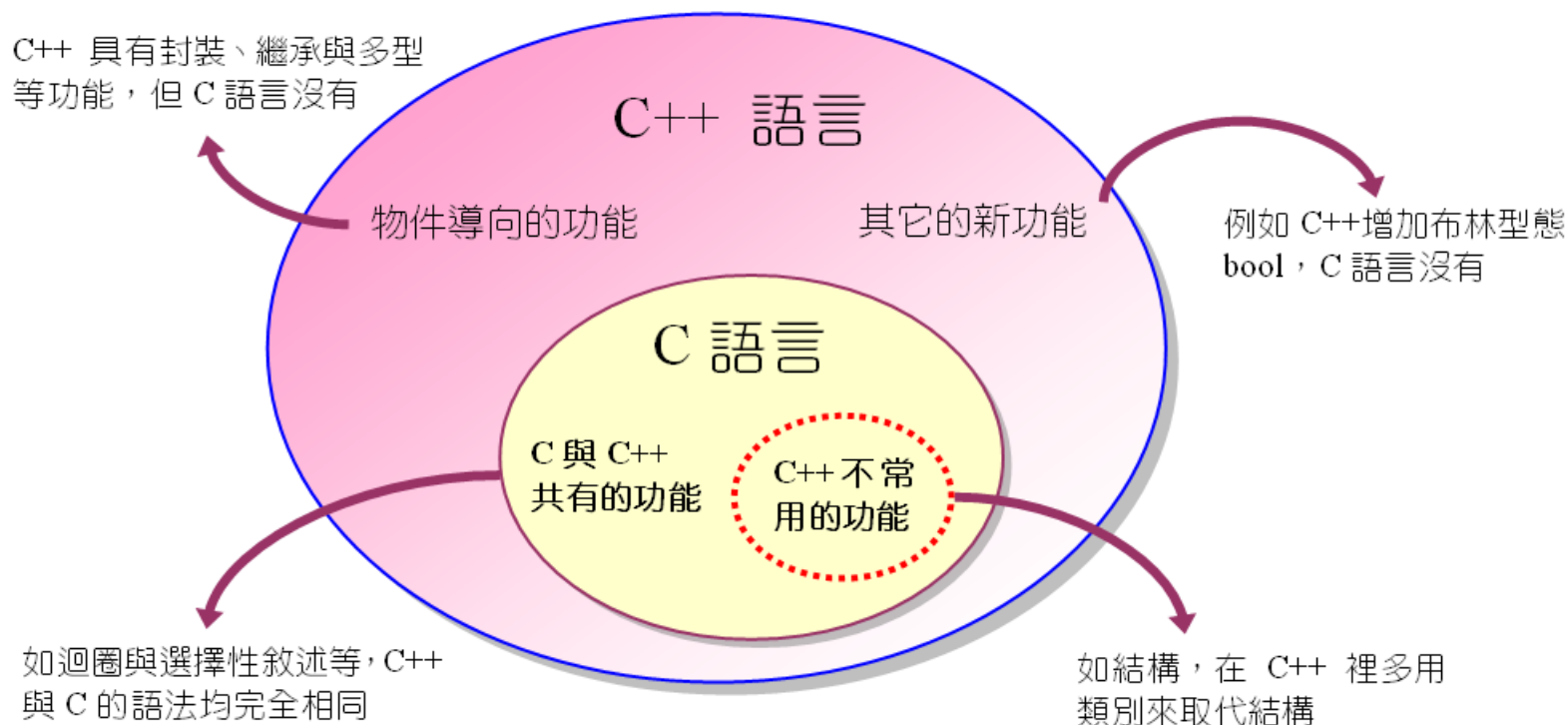
# 緣起 C/C++

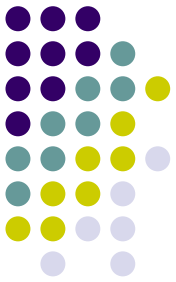
- C語言是於1972年所發展出來的語言
- C++ 於1980年開始發展
- 將物件導向（object oriented）加入C語言中
- 稱為「C with classes」程式語言



# C++ 的特色 (1/3)

- C++ 具有下列的幾項特色：
- 向下相容C語言





# C++ 的特色 (2/3)

## 具物件導向技術的語言

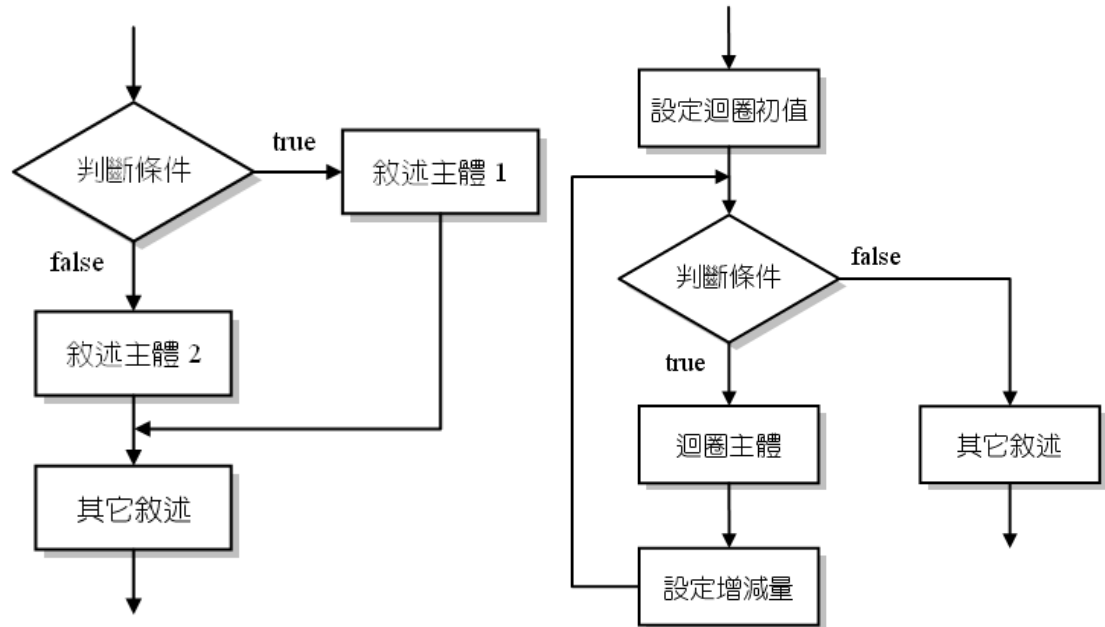
物件導向程式設計（OOP）具有的相關技術

「封裝」

「繼承」

「多型」

## 靈活的程式流程控制





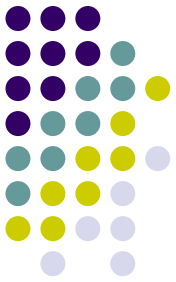
# C++ 的特色 (3/3)

- **高效率且可攜性佳的語言**

- C++在執行前必須先經過編譯，**執行效率遠高於直譯式語言**
- 可在修改少許程式碼的情況下，便能**在不同的作業系統執行**

- **程式碼的再利用**

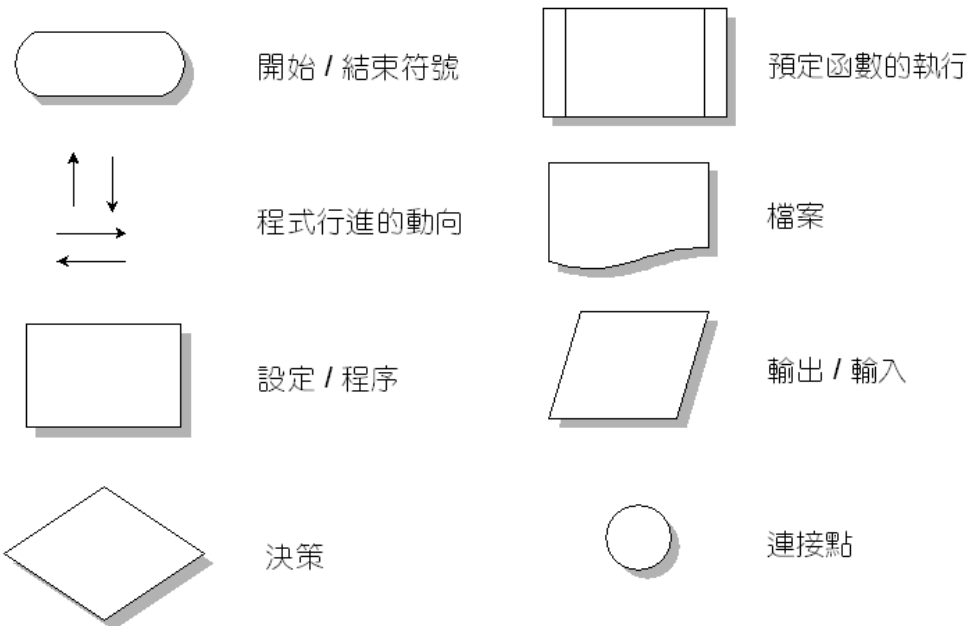
- 可依據原有類別所提供的功能，再加上新的需求，進而設計出新的類別，便是**程式碼的「再利用」**
- 程式碼的再利用可**減少程式碼的開發時間與費用**



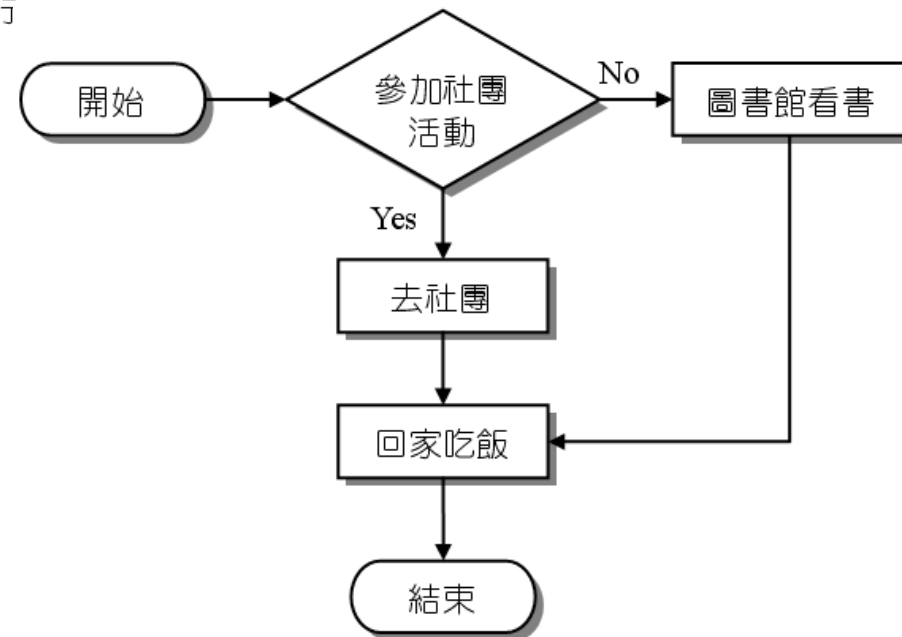
# 規劃與實作 (1/2)

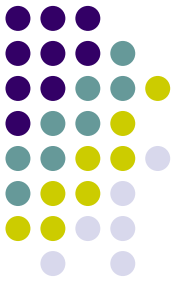
## 1. 規劃程式

### 常用的流程圖符號



### 繪製流程圖的範例





# 規劃與實作 (2/2)

2. 撰寫程式碼及註解

3. 編譯程式

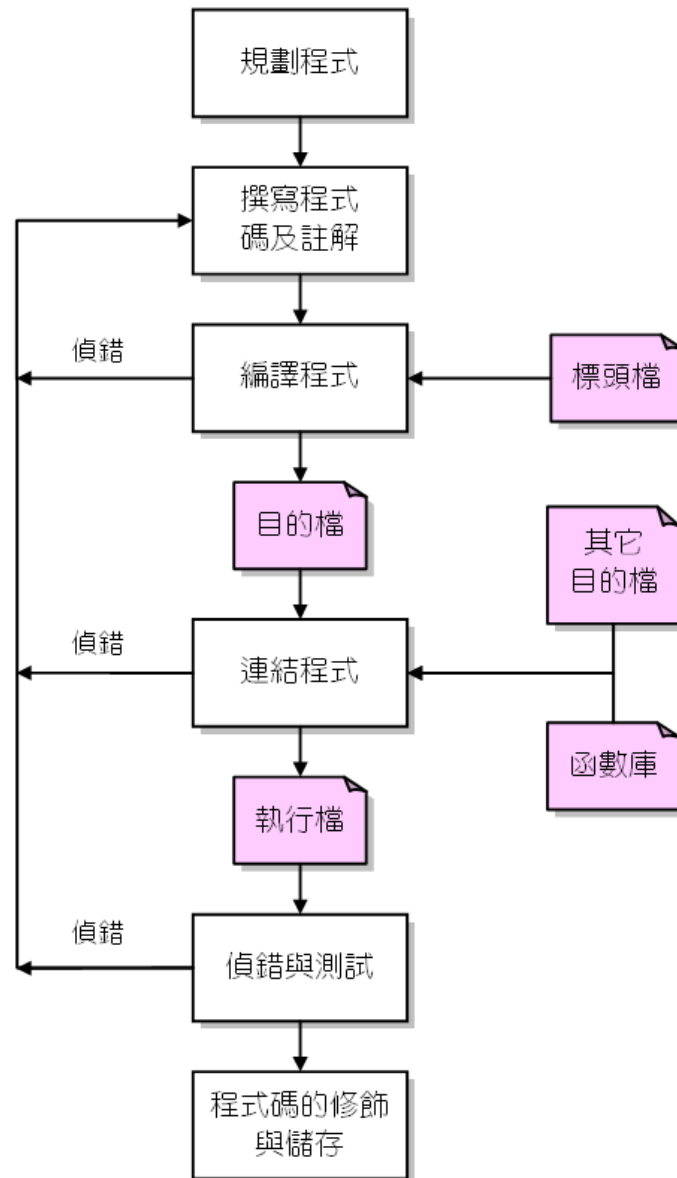
4. 連結程式

5. 偵錯與測試

語意錯誤 (semantic error)

語法錯誤 (syntax error)

6. 程式碼的修飾與儲存





# 程式碼的編輯到執行 (1/2)

- 第一個C++程式

```
01 // prog1_1, 第一個 C++ 程式
02 #include <iostream>
03 #include <cstdlib>
04 using namespace std;
05 int main(void)
06 {
07     cout << "Hello, C++" << endl;    // 印出 Hello, C++
08     system("pause");
09     return 0;
10 }
```





# 程式碼的編輯到執行 (2/2)

- 下面的視窗為鍵入程式碼之後的情形：

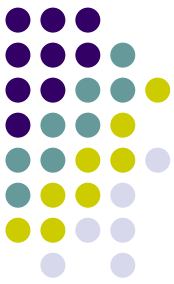
```
1 // prog1_1, 第一個C++程式
2 #include <iostream>
3 #include <cstdlib>
4 using namespace std;
5 int main(void)
6 {
7     cout << "Hello, C++" << endl; // 印出Hello C++
8     system("pause");
9     return 0;
10 }
```

Dev-C++ 4.9.9.2 interface details: The window title is 'Dev-C++ 4.9.9.2'. The menu bar includes 檔案(F), 編輯(E), 搜尋(S), 檢視(V), 專案(P), 執行(Z), 除錯(D), 工具(T), CVS, 視窗(W), and 求助(H). The toolbar contains icons for file operations, editing, and execution. The status bar at the bottom shows '10: 2', '修改', '插入模式', and '檔案裡總共有10行'.

程式碼經過編譯與  
執行後的執行結果

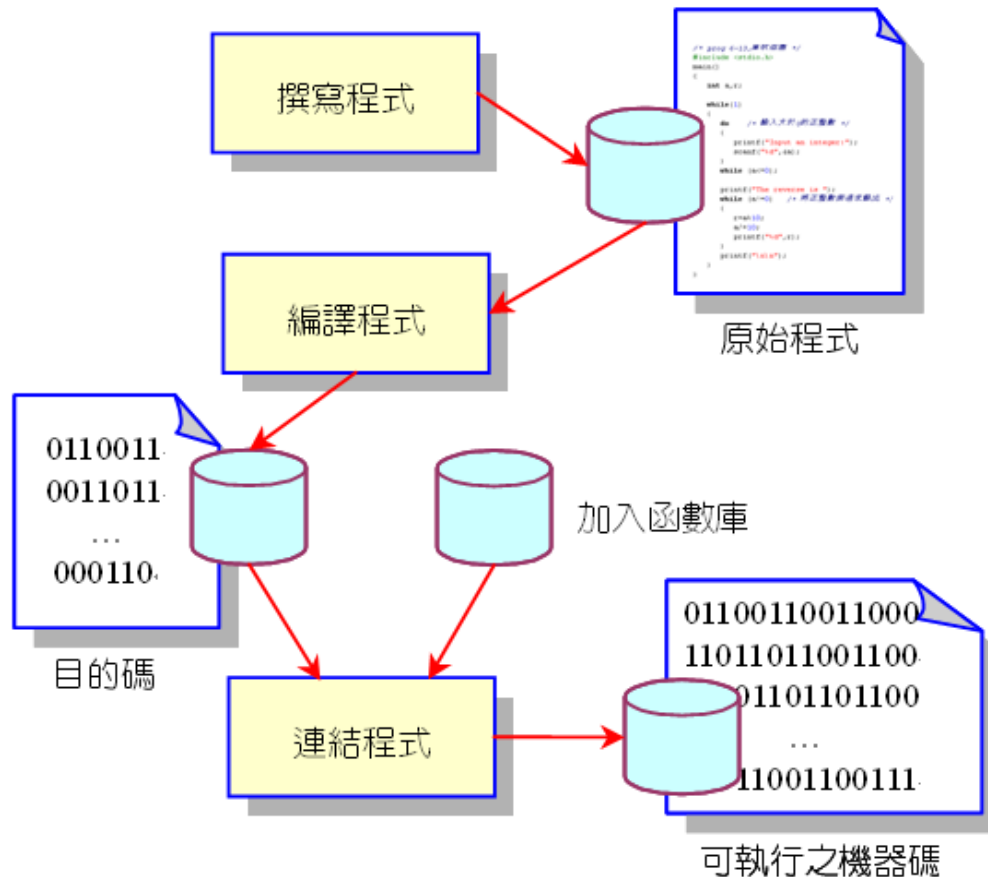
```
C:\Documents and Settings\Admini...
Hello, C++
請按任意鍵繼續 . . .
```

Command prompt window details: The title bar shows 'C:\Documents and Settings\Admini...'. The output displays 'Hello, C++' followed by a prompt '請按任意鍵繼續 . . .' (Press any key to continue).



# 編譯與執行過程的解說

## 原始程式編譯及連結的過程



# 第二章

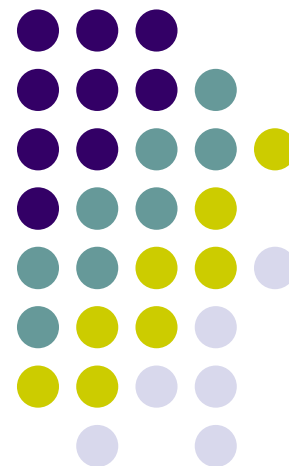
## 認識C++

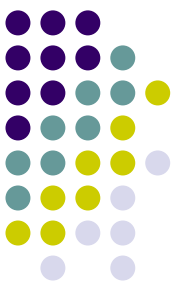
學習C++的基本語法

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 (1/2)

```
01 // prog2_1, 簡單的 C++ 程式
02 #include <iostream>           // 包括 iostream 檔案
03 #include <cstdlib>            // 包括 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 (2/2)

- main() 函數的傳回值型態及引數型態的設定

傳回型態為整數

main()函數不需傳入引數

```
int main( void )  
{
```

程式敘述 ...

```
return 0;
```

main()函數執行完畢，傳回整數 0

```
}
```



# 關於ANSI/ISO C++的標準

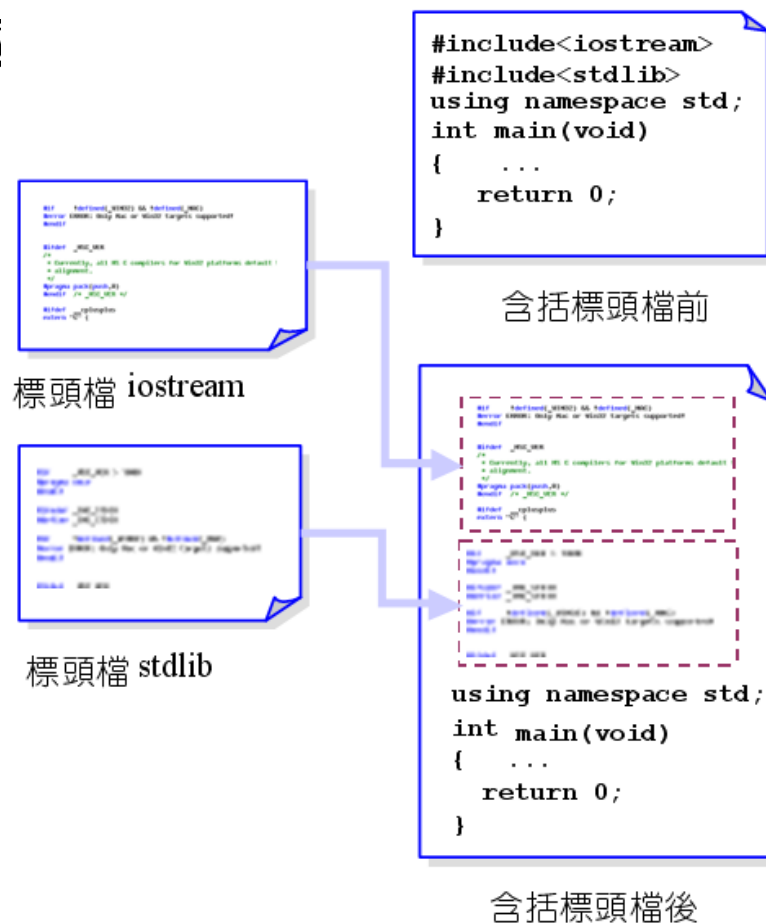
- C++標頭檔可能有下面四種型態
  - C語言的標頭檔：以「.h」結尾
  - C++語言的標頭檔：以「.h」結尾
  - ANSI/ISO C++新標準的標頭檔：沒有副檔名
  - 從C移植過來的標頭檔，沒有副檔名，字首加小寫的c
- 新版的ANSI/ISO C++必須利用using namespace來設定名稱空間為std



# #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

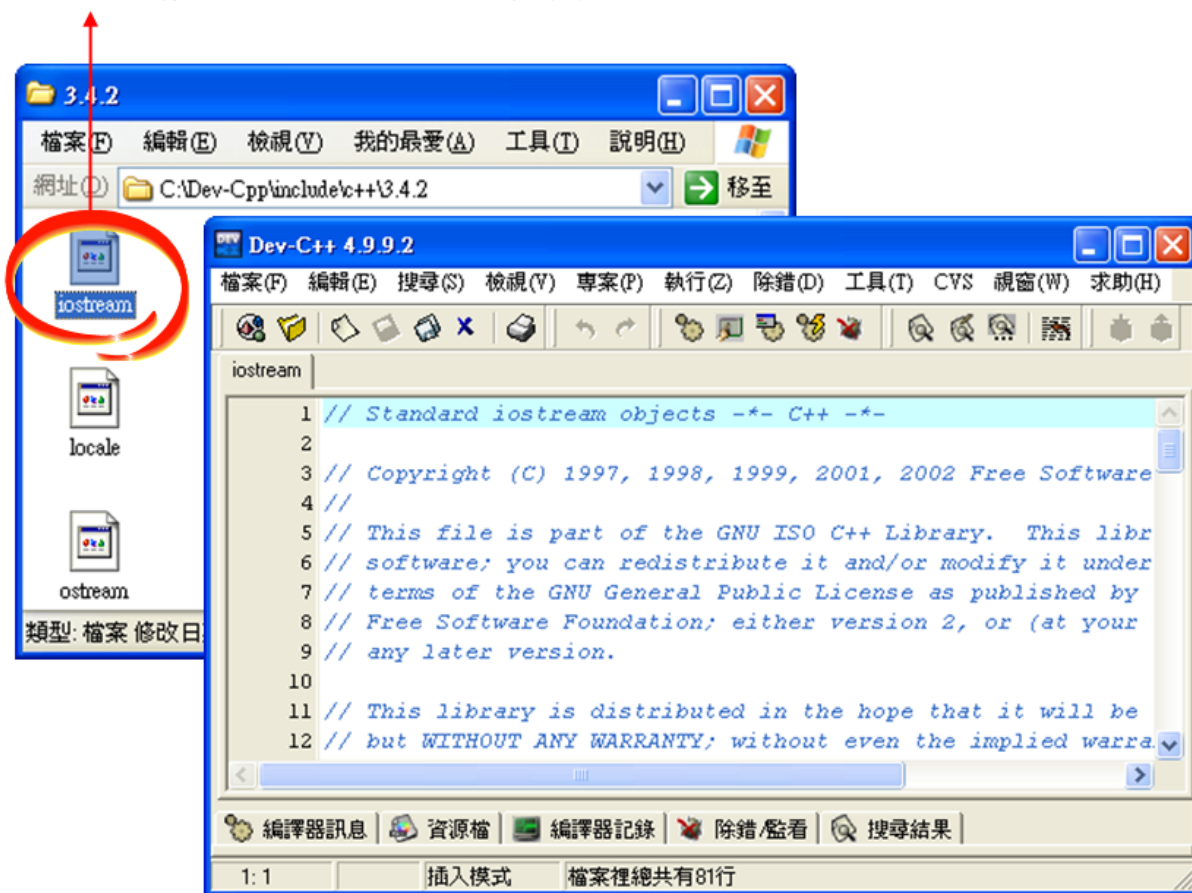
- 右圖為含括動作前後的比較



# #include指令及標頭檔 (2/2)

## 標頭檔的內容

iostream 存放在 include\c++\3.4.2 資料夾內







# 主函數、大括號、區塊與主體(1/2)

- 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(2/2)

- 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)

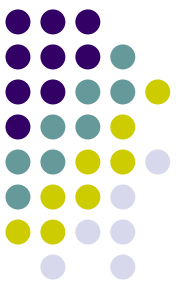
- 變數名稱與限制

- 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
```



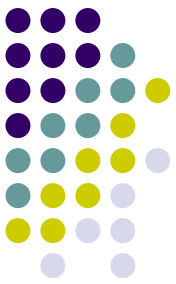
# 為什麼要宣告變數

- 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



# 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

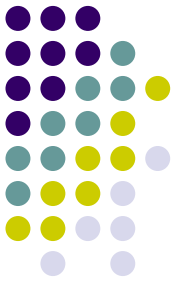


# 識別字 (2/2)

- 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：

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

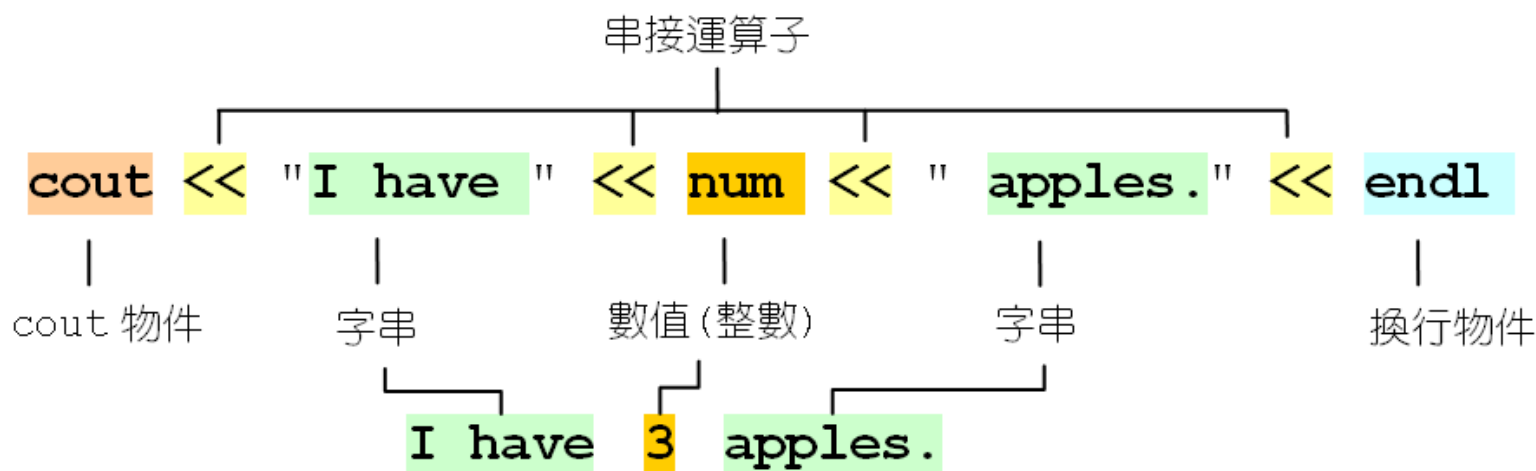


# cout與串接運算子<<

- 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>           // 包括 iostream 檔案
03 #include <cstdlib>             // 包括 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++.

**-----\*/**



# 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>           // 包括 iostream 檔案
03 #include <cstdlib>            // 包括 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>           // 包括 iostream 檔案
03 #include <cstdlib>             // 包括 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.

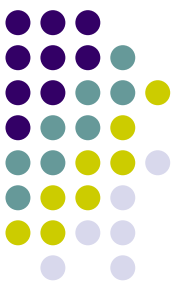
**-----\*/**



# 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>           // 包括 iostream 檔案  
#include <cstdlib>           // 包括 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)

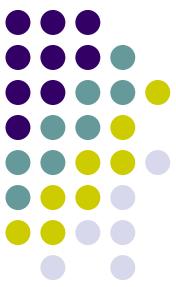
- 有縮排的程式碼可提高可讀性

```
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

**-----\*/**

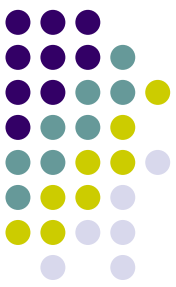


# 將程式碼縮排 (2/2)

- 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
-----*/
```

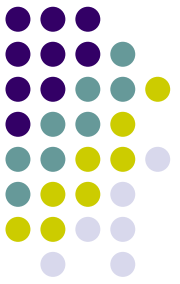


# 註解

- 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

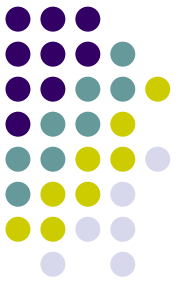
```
// prog2_7, examples      }    以「//」符號註解  
// created by Wien Hong
```

```
/* This paragraph demonstrates the capability  }  於「/*」和「*/」符號  
   of comments used by C++    */              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-