新北市立海山高中

Dev C++
簡 明 教 材

翁鴻仁 教師編授

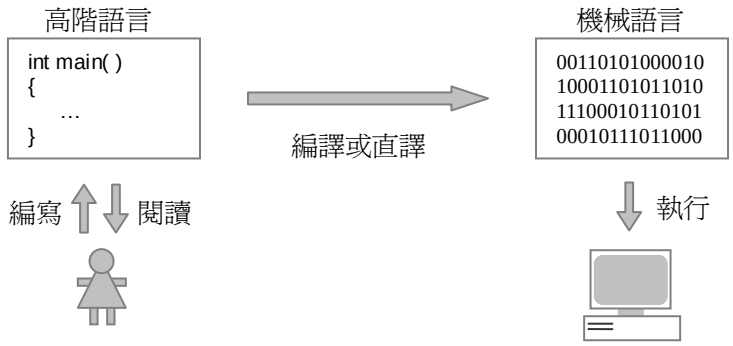
目 錄

第一章 緒論

1-1 爲什麼要學 C++程式語言

C 程式語言是在 1970 年代由貝爾實驗室(AT&T)的 Dennis Ritchie 和 Ken Thompson 所發展出來的，因爲其具有結構化、模組化與可攜性的特色，因此廣受程式設計者的愛好與使用。到了 1980 年代，爲了解決大型程式開發所遭遇到的問題，貝爾實驗室的 Bjarne Stroustru 將物件導向的設計概念導入 C 程式語言裡，並逐漸發展成爲今日的 C++程式語言。

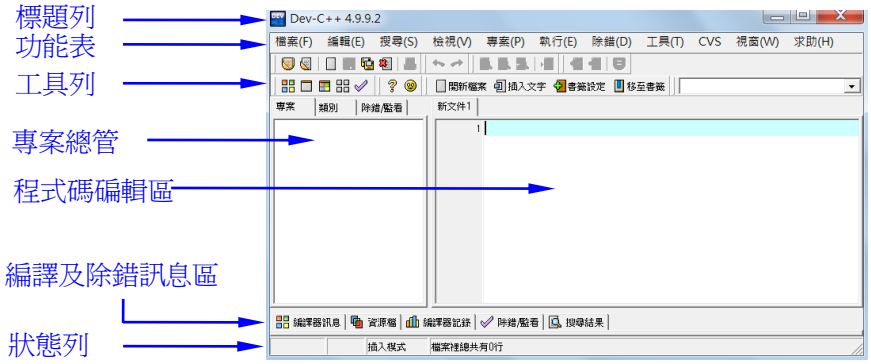
電腦所能接受的指令是由 0 與 1 所組成的機械語言，而 C++程式語言爲高階語言，其語法較接近人們平常講話的自然語言，電腦是無法理解的。因此，當程式設計者編寫完高階語言後，必須翻譯成電腦可讀取的機械語言，電腦才能依程式指令執行。而翻譯的方式有兩種：一種是將全部的程式翻譯完之後再交給電腦執行，此種方式稱爲編譯，而負責翻譯者稱爲編譯器(complier)。另一種翻譯的方式則是，翻譯一行程式指令之後，就立刻執行，然後才翻譯下一行程式敘述，此種一邊翻譯一邊執行的方式稱爲直譯，而翻譯者稱爲直譯器(interpreter)。



1-2 Dev-C++整合開發環境

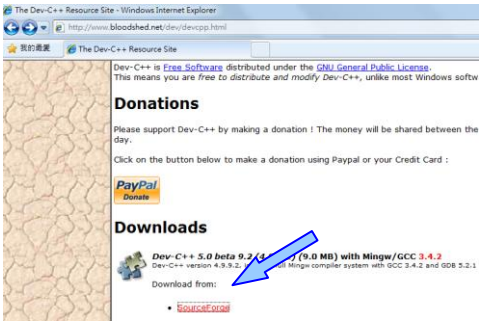
1. Dev-C++整合開發環境

Dev-C++中的 Dev 是 Develop 的縮寫，是由 Bloodshed 公司專爲 C++程式語言所發展的整合開發環境(IDE；Integrated Development Environment)，其可提供程式設計者在視窗化的作業環境下進行程式碼的編寫、編譯與執行，其視窗畫面如下：



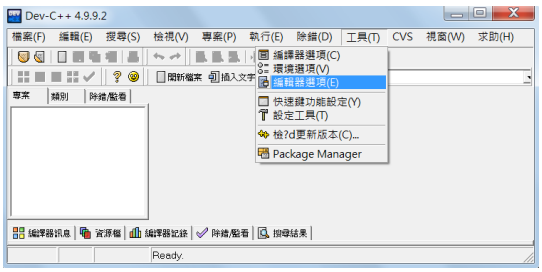
2. Dev-C++整合開發環境的安裝

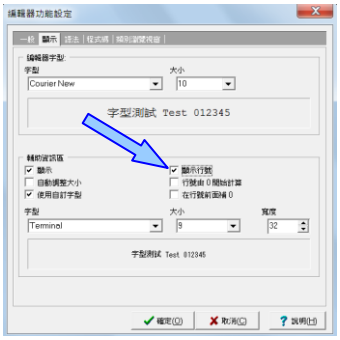
進入網頁「http://www.bloodshed.net/dev/devcpp.html」，並下載 Dev-C++ 安裝檔 devcpp-4.9.9.2_setup.exe。下載後將游標移至該檔案，並連按滑鼠左鍵兩下即可進行安裝。



1-3 編輯程式前的準備

請在功能表列中點選「工具/編輯器選項」；即可開啓「編輯器功能設定」視窗；接著選擇「顯示」頁面，並在「顯示行號」的核選方框內打鈎，再按「確定」鈕。顯示行號僅需設定一次。





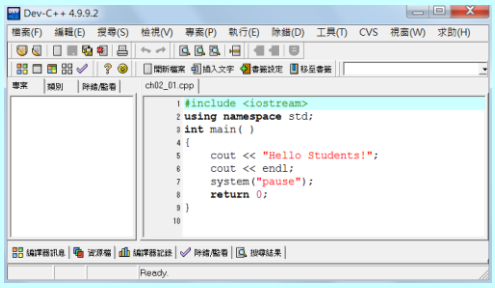
第二章 程式設計的基本概念

2-1 我的第一個 C++程式

步驟 1：請在「程式碼編輯區」內鍵入下列文字敘述；

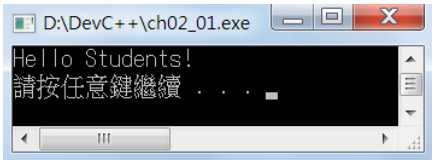
程式碼

```
#include <iostream>
using namespace std;
int main( )
{
      cout << "Hello Students!";
      cout << endl;
      system("pause");
      return 0;
}
```

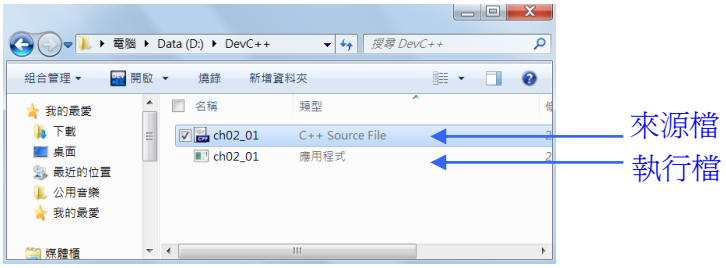


步驟 2：點選功能表中的「執行/編譯並執行」，或是工具列上的「」，或直接按「F9」快捷鍵；可

以看到程式的執行結果如下圖。



步驟 3：請先關閉 Dev-C++整合開發環境及執行結果的視窗畫面。接下來在 Windows 作業系統中，由「電腦」或「檔案總管」找出所儲存的目錄，即可發現在此目錄底下除了原先已建立附檔名爲「.cpp」的 Dev-C++來源檔外，還包括有另一新建立的同名執行檔，其附檔名爲「.exe」。



其中，Dev-C++來源檔必須用 Dev-C++整合開發環境來開啓和執行；而執行檔則可在 Windows 作業系統中連點滑鼠左鍵兩下來執行。

2-2 我的第二個 C++程式

1. 我的第二個 C++程式

下列程式示範如何做簡單的數值運算，並在螢幕上顯示運算結果。

程式碼

```
/*    加法示範    */                          // 註解
#include <iostream>
using namespace std;
int main( )
{
      int x;                                  // 宣告 x 爲整數變數
      x=14;                                   // 設定 x 的初值
      cout << "x="     << x     << endl;      // 印出「x=14」後換行
      cout << "x+3=" << x+3 << endl;          // 印出「x+3=17」後換行
      system("pause");
      return 0;
}
```

位元、位元組與字元：

位元(bit)指的是電腦裡最小的記憶單元，在此記憶單元可存放 0 與 1 兩種數位訊號。位元組(byte)亦是電腦的記憶單元，其由而 8 個位元所組成，因爲每個位元都可以存放兩種訊號，所以一個位元組可存放的訊號有 2^8=256 種組合。每一種組合即分別對應二進位的一個數值，而一個數值就代表

一個字元(character)。例如組合 00100001 即爲二進位表示，換成十進位數值爲 65，代表的字元爲大寫的英文字母「A」；00100010 即爲十進位的數值 66，代表的字元爲大寫的英文字母「B」，依此類推。上述用以指示字元的數值，稱爲 ASCII 碼(讀音同 ask key)。

2. C++程式語言的保留字

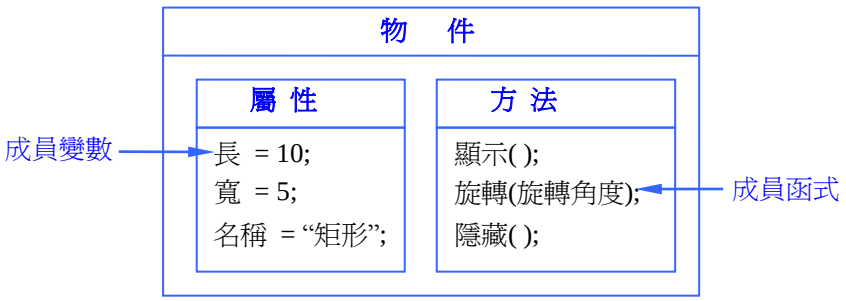
保留字的意義就是指令，命令程式做一些動作，有其特殊的作用，不可以當作變數名稱，初學者因認識的指令不多，所以往往會有誤用情況，以下列出 C++程式語言的保留字(keyword)。

and	and_eq	asm	auto	bitand
bitor	bool	break	case	catch
char	class	compl	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	export
extern	false	float	for	friend
goto	if	inline	int	long
mutable	namespace	new	not	not_eq
operator	or	or_eq	private	protected
public	register	reinterpret_cast	return	short
signed	sizeof	static	static_cast	struct
switch	template	this	throw	true
try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile
wchar_t	while	xor	xor_eq	

2-3 輸出物件與輸入物件

1. 物件的基本概念

在物件導向的程式語言中，將物件視爲一個獨立的個體，就像是一個獨立的人一樣。例如將矩形視爲一種物件，此物件有自己的特徵，例如長、寬與名稱；而且此個體可以表現出自己的行爲，例如顯示、旋轉和隱藏等。在電腦術語中，長、寬與名稱稱爲屬性(property)，顯示、旋轉和隱藏則被稱爲方法(method)，而不管是屬性或是方法，都屬於物件的成員(member)。當成員是一個變數時，這個變數被稱爲成員變數(data member)；而當成員是一個函式時，這個函式則被稱爲成員函式(member function)。



2. 輸出物件 cout

假設要解決的問題與螢幕的顯示有關，因此操作的主體是關於螢幕的輸出，使用的物件就是「cout」物件，而使用 cout 必須引入標頭檔<iostream>。有關 cout 的語法(syntax)如下：

cout 語法	<iostream>
cout << 變數名稱;	// 輸出變數所儲存的值
cout << 字串;	// 輸出字串的內容
cout << 運算式;	// 輸出運算的結果
cout << 格式指令;	// 輸出格式指令所指定的效果
cout << 字串 << 變數名稱 << …;	// 連續輸出

3. 輸出格式設定

(1)跳脫字元：

cout 的程式敘述之一

```
cout << “ABC\n”;                    // 輸出「ABC」後，換行
cout << “A\t” << “B\t” << “C\t\n”;  // 輸出「A   B   C」後，換行
```

詳細的跳脫字元格式如下表：

跳脫字元

\n	// 游標移至下一列	\t	// 游標移至下一定位點
\\	// 輸出反斜線 \	\”	// 輸出雙引號”

(2)格式指令：

cout 的程式敘述之二

```
cout << endl;                       // 換行顯示
```

詳細的格式指令如下表：

cout 的格式指令

endl	// 本行結束		
left	// 靠左對齊	right	// 靠右對齊

(3)格式函式：

利用標頭檔<iomanip>中的格式函式來設定輸出的格式：

cout 的程式敘述之三

```
cout << setw(5) << “ABC”;           // 輸出「  ABC」
cout << setw(5) << left << “ABC”;   // 輸出「ABC  」
cout << right;                      // 還原靠右對齊的內定值
```

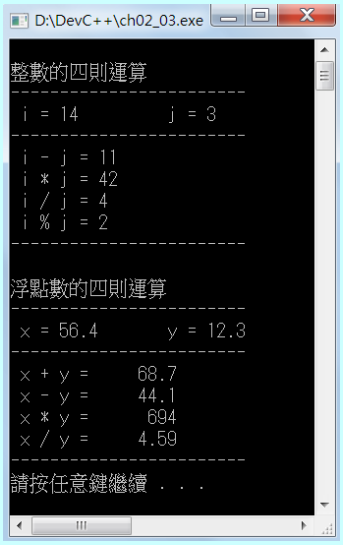
☞ 範例：四則運算程式

使用整數和浮點數變數來設計一個程式，其可在螢幕上顯示此兩數之和，兩數之積，兩數之差，兩數之商，以及兩數相除之後的餘數。

程式碼

```
/*    算術運算子的示範    */
#include <iostream>
```

```
#include <iomanip>   // 使用 setw( )須引入
using namespace std;

int main( )
{
      int i,j;          // 宣告 i 和 j 爲整數變數
      float x,y;      // 宣告 x 和 y 爲浮點數變數

      i=14; j=3;                // 變數的初始化
      x=56.4; y=12.3;

      cout << "\n 整數的四則運算\n";
      cout << "------------------------\n";
      cout << " i = " << i << "\t\tj = " << j <<endl;
      cout << "------------------------\n";
      cout << " i - j = "    <<    i-j    << endl;
      cout << " i * j = "    <<    i*j    << endl;
      cout << " i / j = "    <<    i/j    << endl;        // 取 x 除以 y 的商
      cout << " i % j = " <<    i%j << endl;        // 取 x 除以 y 的餘數
      cout << "------------------------\n";

      cout << "\n 浮點數的四則運算\n";
      cout << "------------------------\n";

      cout.precision(3);                                  // 只顯示至百分位

      cout << " x = " << setw(8) << left << x;
      cout << "\ty = " << setw(8) << y << endl << right;        // 還原靠右對齊內定值
      cout << "------------------------\n";
      cout << " x + y = "    <<    setw(8) << x+y    << endl;
      cout << " x - y = "    <<    setw(8) << x-y    << endl;
      cout << " x * y = "    <<    setw(8) << x*y    << endl;
      cout << " x / y = "    <<    setw(8) << x/y    << endl;
      cout << "------------------------\n";

      system("pause");
      return 0;
}
```



4. 輸入物件 cin

想要擷取鍵盤鍵入的資料，需用標頭檔<iostream>中的輸入物件「cin」，其語法說明如下：

cin 語法	<iostream>

```
cin >>  變數名稱;                    // 輸入數值至變數中
cin >>  變數 1 >>  變數 1 >>  …;      // 連續輸入
```

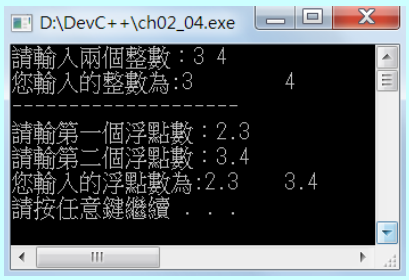
☞ 範例：顯示鍵入的數值

設計一個程式，可分別輸入兩個整數和浮點數，並在螢幕上顯示使用者所輸入的數值。

【程式碼】

程式碼

```
/* 輸入物件 cin 的示範 */
#include <iostream>
using namespace std ;
int main( )
{
      int i,j;
      float x,y;

      cout << "請輸入兩個整數：";
      cin >> i >> j;                          // 連續輸入兩個整數
      cout << "您輸入的整數為:";
      cout << i << "\t" << j << endl;

      cout << "--------------------\n";

      cout << "請輸第一個浮點數：";   cin >> x;        // 分別輸入兩個浮點數
      cout << "請輸第二個浮點數：";   cin >> y;
      cout << "您輸入的浮點數為:";
      cout << x << "\t" << y << endl;
.
      system("pause");
      return 0 ;
}
```



第三章 資料型態

3-1 數值資料與運算子

1. 數值型態

型態	記憶體大小	說明	範圍
int	4 byte	整數	-32767~32767
long	4 byte	長整數	-2147483647~2147483647
float	4 byte	單精確度浮點數	-3.402823e38~-1.401298e-45,0, +1.401298e-45~+3.402823e38

宣告數值資料的語法如下：

型態宣告的語法

```
型態  變數名稱;                          // 宣告變數名稱所儲存內容的資料型態
```

程式敘述如下：

```
float x, y ;                             // 宣告 x 和 y 皆爲浮點數
int counter = 1;                         // 宣告一個整數變數並初始化
int i=1, j=0 ;                           // 宣告多個整數變數並初始化
```

2. 算式運算子(Arithmetic operator)

符號	意義	程式敘述	結果
+	加法運算	1+2	3
-	減法運算	10-4	6
*	乘法運算	3*2	6
/	除法運算	6/3	2
%	取得餘數	18%4	2
++	遞增運算	int x=5; x++;	6
--	遞減運算	int x=5; x--;	4

3. 複合運算子(Compound operator)

利用運算子的組合而形成複合運算子，可以簡化程式敘述。例如：

符號	意義	程式敘述	等同敘述	結果
=	指定變數的值	int x; x=1+2	int x; x=3	3
+=	相加後再指定	int x=3; x+=2;	int x=3; x=x+2;	5
-=	相減後再指定	int x=3; x-=2;	int x=3; x=x-2;	1
*=	相乘後再指定	int x=8; x*=2;	int x=8; x=x*2;	16
/=	相除後再指定	int x=18; x/=3;	int x=18; x=x/3;	6

4. 關係運算子(Relational operator)

符號	意義	程式敘述	結果
==	等於判別	(1 == 2)	0
!=	不等於判別	(7%2 != 0)	1
>	大於判別	(2/3 > 4/5)	0
<	小於判別	(2*3 < 2*5)	1
>=	大於或等於	(10/2 >= 15/3)	1
<=	小於或等於	(2*2 <= 2/2)	0

5. 邏輯運算子(Logical operator)

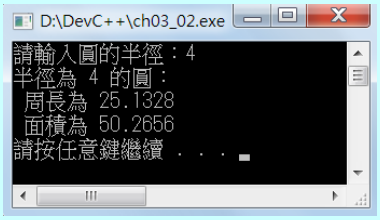
符號	意義	說明	程式敘述	結果
!	不是(not)	0 轉 1，1 轉 0	!0	1
&&	且(and)	兩者爲真才爲真	int x=5; (x>0 && x<=10)	1
||	或(or)	有一爲真就是真	int x=18; !(x%3 || x%6)	1

☞ 範例：圓的性質程式

請設計一個程式，由使用者給定圓半徑的值之後，程式會自動顯示圓的性質，如直徑、圓周長與圓面積。

程式碼

```
/* 圓的性質 */
#include <iostream>
using namespace std ;
int main( )
{
      const float PI=3.1416;                  // 宣告圓周率爲一常數
      float radius, perimeter, area;          // 宣告半徑,周長,面積

      cout << "請輸入圓的半徑：";              // 輸入
      cin >> radius;

      perimeter = 2*PI*radius;                // 運算
      area = PI*radius*radius;

      cout << "半徑為 " << radius << " 的圓：\n";     // 輸出
      cout << " 周長為 " << perimeter << endl;
      cout << " 面積為 " << area << endl;

      system("pause");
      return 0 ;
}
```



☞ 範例：計算成績總分與平均程式

設計一學生成績計算的程式，可讓同學輸入各科成績，並自動計算總分及平均分數。

程式碼

```
/* 計算成績總分與平均 */
#include <iostream>
#include <iomanip>
using namespace std ;
int main( )
{
      short i=0;                  // 科目別
      int score_subject01;   // 第 1 科成績
      int score_subject02;   // 第 2 科成績
      int score_subject03;   // 第 3 科成績
      float sum;                  // 總分

      cout << "請輸入第" << i+1 << "科成績：";        // i=0，輸入第 1 科成績
      cin >> score_subject01;
      i++;

      cout << "請輸入第" << i+1 << "科成績：";        // i=1，輸入第 2 科成績
      cin >> score_subject02;
      i++;

      cout << "請輸入第" << i+1 << "科成績：";        // i=2，輸入第 3 科成績
```

```
        cin >> score_subject03;
        i++;                                                    // i=3，總科目數爲 i

        /*      成績計算        */
        sum = score_subject01 + score_subject02 + score_subject03;
        average = sum / i;

        /*      輸出統計結果        */
        cout << "----------------\n";
        cout << "  總分為" << sum << "分\n";
        cout << fixed << setprecision(2);                       // 固定浮點數的小數位數
        cout << "  平均為"   << average << "分\n";

        system("pause");
        return 0 ;
}
```

3-2 字元與字串資料

1. 字元資料型態

請參考下列的程式敘述：

字元的宣告與初始化

```
char a;                         // 宣告 a 爲一字元變數
a = 'A';                        // 將符號「A」所對應的 ASCII 碼(65)指定給字元變數 a
char b = 'A';                   // 宣告並指定字元變數 b 的值爲 65
```

☞ 範例：字元轉 ASCII 程式

設計一個字元轉 ASCII 程式。

【程式碼】

程式碼

```
/*   求字元的 ASCII 碼程式   */
#include <iostream>
using namespace std;
int main( )
{
        short asciicode;
        char a;

        cout << "請輸入字元：";
        cin >> a;

        asciicode = int(a);                     // 強制將字元轉換爲整數

        cout << "字元  ( " << a << " )的";
        cout << "ASCII 碼為  " << asciicode << endl;

        system("pause");
        return 0;
}
```

2. 字串資料型態

字串語法如下：

字串資料型態宣告的語法 **<iostream>**

```
string  字串變數名稱
```

程式敘述參考如下：

字串的宣告與初始化

```
string str1;                    // 宣告 str1 爲字串型態
string str2 = “ABC”;            // 宣告 str2 爲字串，並指定初值
```

例如：

字串的操作

```
cin >> str1;                    // 將輸入的文字存入 str1 字串內
cout << str1;                   // 將 str1 字串儲存的文字輸出至螢幕
str1 = str1 + “ABC”;            // 將文字” ABC” 附加至 str1 的最後面
str1 += “ABC”;                  // 功效同 str1 = str1 + “ABC”
```

3. 字串物件

請參考下列的程式碼：

```
string str3 = “ABCDEF”;         // 宣告 str3 爲一字串，並指定初值
cout << str3[0] << str3[3];     // 輸出「AD」
str3[2] = ‘c’;                  // 字串內容由「ABCDEF」變爲「ABcDEF」
```

請看下列的程式敘述：

```
string str1 = “ABCDEF”;         // 宣告 str1 爲一字串，並指定初值
str1.erase(3);                  // 刪除索引值 3(含)之後的字元
cout << str1;                   // 輸出「ABC」
```

程式敘述請參考如下：

```
string str1 = “I love you!”;
int position=0;
cout << str1.size( );           // 輸出字串大小「11」
position = str1.find(“love”);   // position = 2
position = str1.find(“like”);   // position = -1
```

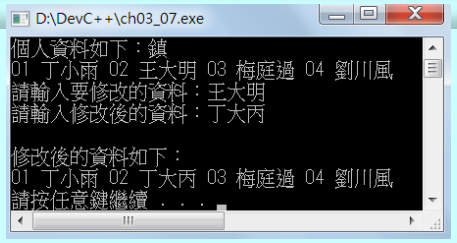
程式敘述請參考如下：

```
string str1, str2;
str1 = “I love you!”;
str2.replace(2,4,”forget”);     // str2=” I forget you!”
```

☞ 範例：個人資料的查詢與修改

從已存在的個人資料中，搜尋欲某一特定的文字，並將文字修改成正確的文字，再於螢幕上顯示出正確的個人資料。

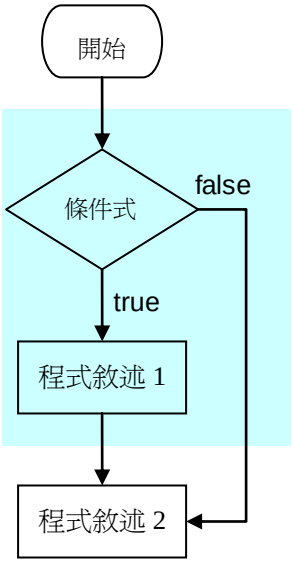
程式碼




```
/* 個人資料的查詢與修改 */
#include <iostream>
using namespace std ;
int main( )
{
      string txt_line, txt_search, txt_replace;
      int position ;                                  // 字元在字串的位置

      txt_line = "01 丁小雨 02 王大明 03 梅庭過 04 劉川風";

      cout << "個人資料如下：\n" << txt_line << endl;
      cout << "請輸入要修改的資料："; cin  >> txt_search;
      cout << "請輸入修改後的資料："; cin  >> txt_replace;

      position = txt_line.find(txt_search);                    // 尋找字串並取得啓始位置

      txt_line.replace(position, txt_search.size(),txt_replace);     // 舊字串置換成新字串

      cout << "\n 修改後的資料如下：\n";
      cout << txt_line << endl;

      system("pause");
      return 0 ;
}
```

第四章 程式的流程判斷

4-1 單向選擇



【程式碼】

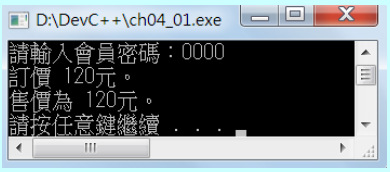
```
…
if (條件式)
{
   程式敘述 1;
}
程式敘述 2;
…
```

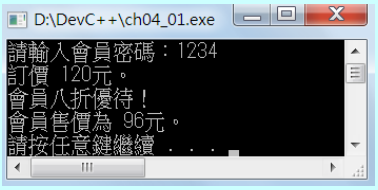
☞ 範例：會員特惠價程式

請設計一個顯示物品售價的程式，讓使用者輸入密碼以確認會員身份；如果是非會員就顯示原價，而會員則顯示八折優待後的價格。

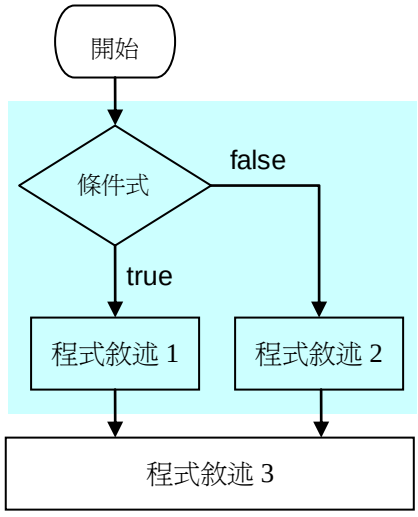
程式碼

```
/*  會員特惠價程式   */
#include <iostream>
using namespace std;
int main( )
{
      float price;
      bool flag;                                  // 設定旗標，1 爲會員，0 爲非會員
      string password;

      price = 120;                                // 原價 120 元
      cout << "請輸入會員密碼：";
      cin >> password;
      cout << "訂價  " << price << "元。\n";

      if ( password == "1234" )                   // 判別密碼是否正確
          flag = 1;

      if (flag)                                   // flag=1 爲會員
      {
          cout << "會員八折優待！\n 會員";
          price *= 0.8 ;
      }
      cout << "售價為  " << price << "元。";
      cout << endl;

      system("pause");
      return 0;
}
```





4-2 雙向選擇



【程式碼】

```
…
if (條件式)
{
   程式敘述 1;
}
else
{
   程式敘述 2;
}
程式敘述 3;
…
```

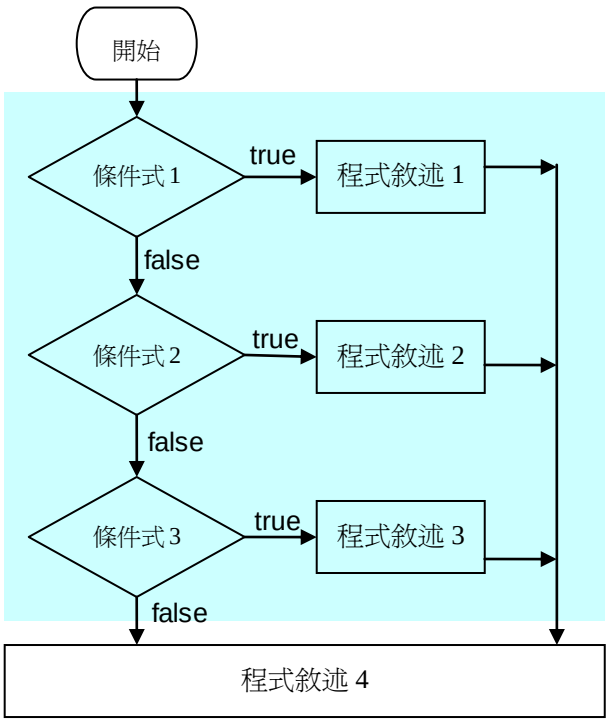
☞範例：閏年判斷程式

請設計一個可以判斷使用者所輸入的西元年份是否爲閏年的程式。平年有西元 1、2、3、5、6、7、9、…、100、200、300、500、600、…等；閏年有 4、8、12、…、400、800、…、2000、…。

程式碼

```
/*   閏年判斷程式   */
#include <iostream>
using namespace std;
int main( )
{
      int year;

      cout << "請輸入西元年份：";
      cin >> year;

      if ( year%4 != 0 )                                  // 不被 4 整除者爲平年
          cout << "西元" << year << "爲平年。\n";
      else                                                // 被 4 整除者要再進一步判斷
      {
          if ( (year%100==0) && (year%400!=0) )
              // 被 4 整除中，能被 100 整除但不被 400 整除者，爲平年
              cout << "西元" << year << "爲平年。\n";
          else
              cout << "西元" << year << "爲閏年。\n";
      }

      system("pause");
      return 0;
}
```

4-3 巢狀選擇



【程式碼】

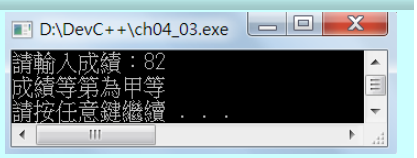
```
…
if (條件式 1)
{
   程式敘述 1;
}
else if (條件式 2)
{
   程式敘述 2;
}
else if (條件式 3)
{
   程式敘述 3;
}
程式敘述 4;
…
```

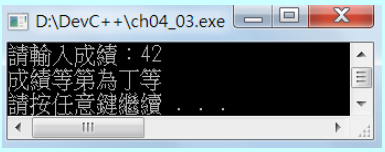
☞ 範例：成績等第判斷程式

請設計一個可以判定學生成績等第的程式。成績大於 90 分者爲優等，介於 80 至 90 分者爲甲等，70 至 80 分者爲乙等，60 至 70 分者爲丙等，小於 60 分者則判定爲丁等。

程式碼

```
/*   成績等第判定程式   */
#include <iostream>
using namespace std;
int main( )
{
      int score;

      cout << "請輸入成績：";
      cin >> score;

      if ( score < 0 || score > 100 )                  // 檢查有無輸入錯誤
          cout << "輸入錯誤，本程式結束\n";

      else if ( score >= 90 )                          // 90 至 100 分
          cout << "成績等第為優等\n";

      else if ( score >= 80 )                          // 80 至 89 分
          cout << "成績等第為甲等\n";

      else if ( score >= 70 )
          cout << "成績等第為乙等\n";                  // 70 至 79 分

      else if ( score >= 60 )
          cout << "成績等第為丙等\n";                  // 60 至 69 分

      else                                             // 處理 60 分以下的成績
          cout << "成績等第為丁等\n";

      system("pause");
      return 0;
}
```





第五章 函式

5-1 函式的基本概念

函式的程式敘述可以寫在主程式之前，編譯器編譯到函式時就會將函式名稱記起來，等到編譯至主程式的程式敘述時，若遇到曾經編譯過的函式名稱，編譯器就知道要回頭去找函式的內容；而在主程式中，告訴編譯器去執行函式內容的行爲稱作呼叫(call)函式。

☞ 範例：求各式形狀的面積程式

請設計一個求平方的函式，並實際使用這個函式。

程式碼

```
/*    函式寫在主程式之前   */
#include <iostream>
using namespace std ;

int sqr ( int x )                                  // 定義函式
      {
          return x * x;                            // 函式內容
      }

int main( )
{
      int x = 3;
      cout << sqr( x );                            // 呼叫函式
      system("pause");
      return 0;
}
```

5-2 函式的遞迴

函式自己呼叫自己的程式設計方法稱之爲遞迴(recusive)。

1. 數值的遞迴

在數學上，將一系列的數字聚集起來即形成一個集合，數學上記爲{ a_n }，n 表 1、2、3、…的自然數，an 表此集合內的第 n 項元素。例如奇數的集合爲{1,3,5,7,9,…, a_n}，第 1 項元素爲 1，即 $a_1 = 1$，第 2 項元素爲 3，即 $a_2=3$；以此類推，第 n 項元素的值爲 2n - 1，即 $a_n = 2n - 1$。

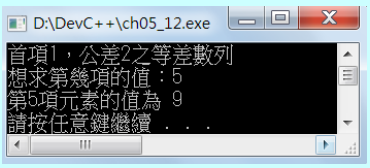
由數學歸納法可以將上列各式寫成遞迴表示式如下：

$$\begin{cases} a_1 = 1； & \text{初始部分} \\ a_n = 2 + a_{n-1}，n >= 2； & \text{遞迴部分} \end{cases}$$

而在使用遞迴函式時就如同上述表示法一樣簡單，只要將數學公式「抄寫」在函式內即可；

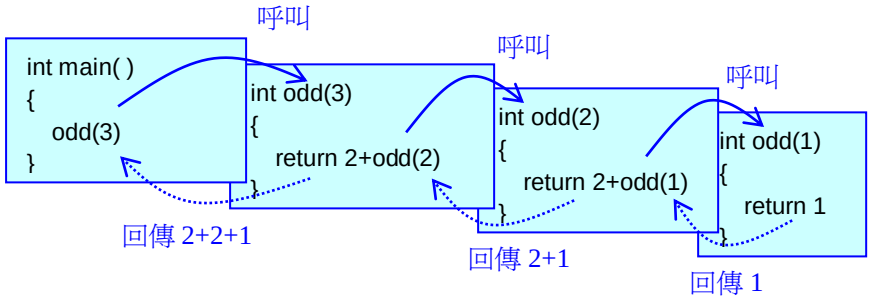
程式碼

```
/*   累加型數列之遞迴函式示範   */
#include <iostream>
using namespace std;

int odd( int n )
{
      if ( n==1 )
            return 1;                              // 邊界條件
      else
            return 2 + odd(n-1);                   // 遞迴公式
}

int main( )
{
      int n;
      cout << “首項 1，公差 2 之等差數列” << endl;
      cout << "想求第幾項的值：";
```



```
    cin >> n;
    cout << "第" << n << "項元素的值爲 ";
    cout << odd(n) << endl;                          // 呼叫遞回函式

    system("pause");
    return 0;
}
```

在主程式呼叫遞迴函式所產生的效果如下圖之說明：每呼叫函式一次就會產生一個新的記憶體空間以存放被呼叫的函式。如圖之最右側，最後一次呼叫函式時必須給一個確定的回傳值，如此才能終止再次呼叫函式而不至於陷入無窮的遞迴呼叫。



2. 巨集指令

在主程式之前以井字號「#」爲首之指令稱爲前置處理指令(processor)。常見的前置處理指令有巨集(macro)指令#define；其語法如下：

巨集指令#define 的語法

```
#define  巨集名稱   巨集定義;
#define  巨集名稱(引數 1,引數 2,…)  巨集定義;
```

其功用是在程式敘述中，將常用的敘述片段或特殊的文字定義，寫在程式碼的前言部分，以使程式更容易閱讀與更改。例如下列的程式碼：

```
1   #include <iostream>
2   using namespace std;
3
4   #define NUM 3;
5   #define sqr(x)   x*x;                  // 定義求平方函式 sqr(x)
6
7   int main( )
8   {
9       int x = NUM;                       // 即 int x = 3;
10      cout << sqr(x);                    // 即 cout << x*x
11  }
```

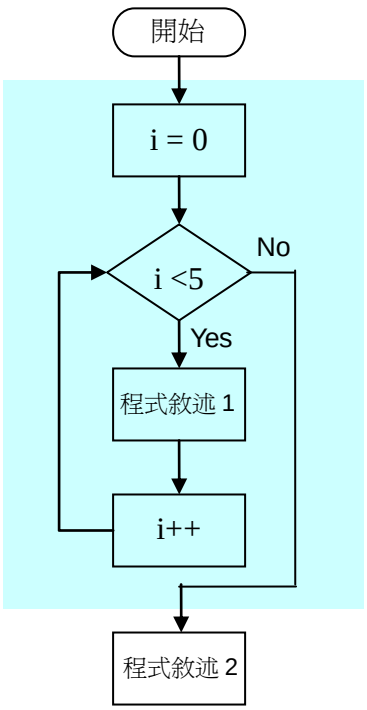
第六章 程式的流程控制

6-1 計次迴圈 for

for 迴圈利用一索引(index)來計算迴圈執行的次數，給定索引變數的初值與終值，並指定從初值起每執行一次迴圈時索引值的增加量，就可以得知迴圈執行的次數；同時，也可利用索引變數來處理一些運算。

1. 單層迴圈

下列程式碼寫出可以執行五次的迴圈，而索引值從 0 至 4；



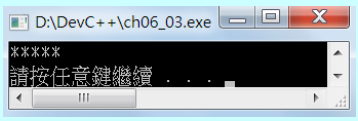
【程式碼】

```
…
for ( i=0; i<5; i++)
{
    程式敘述 1
}
程式敘述 2
```

程式碼參考如下：

程式碼

```
/*   for 迴圈示範之一   */
#include <iostream>
using namespace std;

int main( )
{
      for ( int i=0; i<5; i++)                  // 重覆執行迴圈 5 次
            cout << "*";                        // 每次印 1 個「*」

      cout << endl;                             // 印完最後一個「*」後游標跳下一行

      system("pause");
      return 0;
}
```



2. 巢狀迴圈

巢狀迴圈指的是在一層迴圈內又有另一層迴圈。首先使用的迴圈稱作外迴圈，索引變數習慣定爲 i；而被外迴圈包圍的迴圈稱作內迴圈，習慣上內迴圈的索引變數設爲 j。

程式碼

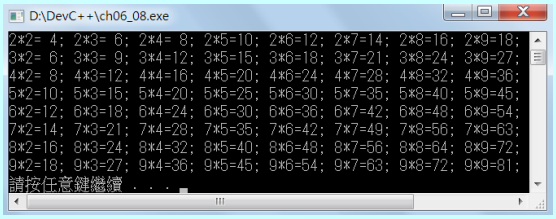
```
/*   巢狀迴圈示範   */
#include <iostream>
using namespace std;

int main( )
{
      for ( int i=0; i<3; i++)
      {
            for ( int j=0 ; j< 4; j++)
                  cout << " i= " << i << "\tj=" << j << endl;
      }

      system("pause");
      return 0;
}
```

☞ 範例：九九乘法表

請設計一可以列印從 2 開始的九九乘法表的程式。

程式碼

```
/*   九九乘法表程式   */
#include <iostream>
#include <iomanip>
using namespace std;

int main( )
{
      for (int i=2; i<10; i++)                    // 外迴圈 i=2~9
      {
            for (int j=2 ; j<10; j++)             // 內迴圈 j=2~9
            {
                  cout << i << "*" << j ;
                  cout << "=" << setw(2) << i*j << "; ";  // 執行了 64 次
            }
            cout << endl;                         // 執行了 8 次
      }

      system("pause");
      return 0;
}
```



第七章 陣列

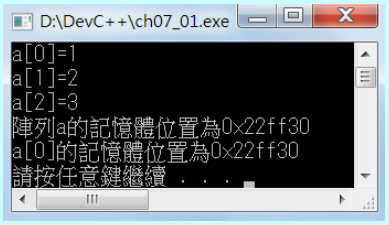
在程式語言中，將多個相同資料型態的資料聚集在一起稱作陣列(array)。

7-1 一維陣列

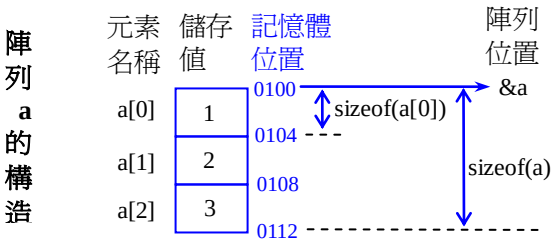
數值陣列的示範：

程式碼

```
/*   整數陣列示範   */
#include <iostream>
using namespace std;

int main( )
{
      int a[3];                          // 宣告整數陣列 a，其具有 3 個元素
      a[0]=1;                            // 第 1 個元素值爲 1
      a[1]=2;                            // 第 2 個元素值爲 2
      a[2]=3;                            // 第 3 個元素值爲 3

      for (int i=0; i<3; i++)                          // 輸出整數陣列 a 全部的元素
            cout << "a[" << i << "]=" << a[i] << endl;

      cout << "陣列 a 的記憶體位置為" << a << endl;       // 輸出陣列 a 的位置
      cout << "a[0]的記憶體位置為" << &a[0] << endl;      // 輸出元素 a[0]的位置

      system("pause");
      return 0;
}
```



上述程式利用敘述「int a[3]」宣告了整數陣列 a，而此陣列包含有 3 個元素。所以宣告陣列之後，變數 a 即代表陣列，不可再被宣告爲變數；既然變數 a 代表陣列，那把「a」印出來會不會得到陣列內所有元素的值呢？答案是不會，利用「cout << a;」的敘述，輸出的是陣列 a 的記憶體位置，也就是陣列中第 1 個元素 a[0]的記憶體位置。



宣告新陣列的同時亦可以初始化陣列的元素值，舉例如下：

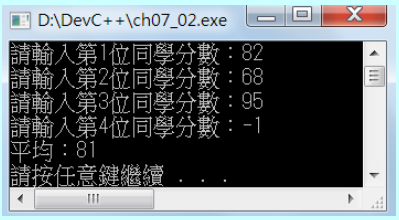
陣列元素的宣告與初始化

```
int a[3] = {1, 2, 3};          // a[0]=1, a[1]=2, a[2]=3  宣告陣列並初始化
int a[3] = {0};                // a[0]=0, a[1]=0, a[2]=0  陣列所有元素初值皆爲 0
```

下列的程式示範了如何計算全班的平均成績。

程式碼

```
/*   計算全班平均成績程式   */
#include <iostream>
using namespace std;

#define NUM_MAX 50                      //全班人數上限

int main( )
{
      int score[ NUM_MAX-1 ];           // 索引值從 0 開始
      int sum=0, num=NUM_MAX;           // 全班人數 num 暫定爲人數上限 NUM_MAX

      for (int i=0; i<NUM_MAX; i++)     // 輸入成績
      {
           cout << "請輸入第" << i+1 << "位同學分數：";
           cin >> score[i];

           if (score[i] < 0)
           {
               num = i;                 // 第(i+1)個的成績爲負值，故全班人數應爲 i
               break;
           }

           sum += score[i];             // 總分累加
       }

        cout << "平均：" <<    sum/num << endl;

      system("pause");
      return 0;
}
```



7-2 二維陣列

1. 二維陣列

宣告一個可以存放 5x4 個整數資料的程式敘述爲「int a[5][4];」，二維陣列的名稱記做「a」，其元素有 5x4 個，列索引分別爲 0, 1, 2, 3, 4，行索引分別爲 0, 1, 2, 3；而所對應的陣列元素如下之右表。

座號	國文	英文	數學
1	90	86	75
2	60	59	80
3	75	68	62
…	…	…	…
5	80	66	84

	第 1 行	第 2 行	第 3 行	第 4 行
	座號	國文	英文	數學
第 1 列	a[0][0]	a[0][1]	a[0][2]	a[0][3]
第 2 列	a[1][0]	a[1][1]	a[1][2]	a[1][3]
第 3 列	a[2][0]	a[2][1]	a[2][2]	a[2][3]
第 4 列	…	…	…	…
第 5 列	a[4][0]	a[4][1]	a[4][2]	a[4][3]

二維陣列的宣告語法如下：

二維陣列的宣告語法

```
型態  陣列名稱  [ n ] [m]                  // n,m 皆爲正整數;n 表總列數,m 表總行數
```

宣告陣列的同時，也可使用大括號來指定其初值，例如：

二維陣列元素的宣告與初始化

```
int a[2][3]= { {90, 86, 75}, {60, 59, 80} };        // a[0][0]=90, a[0][1]=80, a[0][2]=75
                                                    // a[1][0]=60, a[1][1]=59, a[1][2]=80
int b[3][4] =
{
   {1, 90, 86, 75},                 // b[0][0]=1, b[0][1]=90, b[0][2]=86, b[0][3]=75
   {2, 60, 59, 80},                 // b[1][0]=2, b[1][1]=60, b[1][2]=59, b[1][3]=80
   {3, 75, 68, 62},                 // b[2][0]=3, b[2][1]=75, b[2][2]=68, b[2][3]=62
};                                  // 結尾用分號
```

2. 矩陣與二維陣列

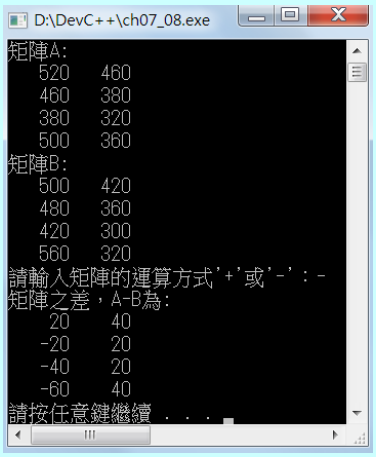
在電腦的程式語言，常使用二維陣列來表示矩陣。下列程式可計算矩陣的加減：

☞ 範例：二維矩陣的加法與減法程式

請設計一程式，可以計算二維矩陣的加減法。

程式碼

```
/*   矩陣的加減   */
#include <iostream>
#include <iomanip>
using namespace std;

int main( )
{
      const int row=4, column=2;                    // 矩陣的列數與行數
      int a[row][column] = {{520,460}, {460,380}, {380,320}, {500,360}};
      int b[row][column] = {{500,420}, {480,360}, {420,300}, {560,320}};
      int c[row][column];
      char op;                                      // 矩陣運算方式'+'或'-'

      cout << "矩陣 A:\n";                          // 輸出 A 矩陣
      for (int i=0; i<row; i++)
      {
            for (int j=0; j<column; j++)
                  cout << setw(6) << a[i][j];
            cout << endl;
      }

      cout << "矩陣 B:\n";                          // 輸出 B 矩陣
      for (int i=0; i<row; i++)
      {
            for (int j=0; j<column; j++)
                cout << setw(6) << b[i][j];
            cout << endl;
      }

      cout << "請輸入矩陣的運算方式'+'或'-'：";
      cin >> op;

      for (int i=0; i<row; i++)          // 矩陣運算
```



```
    {
        for (int j=0; j<column; j++)
        {
            if (op == '+')
                c[i][j] = a[i][j] + b[i][j];
            else if (op == '-')
                c[i][j] = a[i][j] - b[i][j];
        }
    }

    if (op == '+')                                  // 輸出提示文字
        cout << "矩陣之和，A+B 為:\n";
    else if (op == '-')
        cout << "矩陣之差，A-B 為:\n";
    else
        exit(1);                                    // 強制結束程式

    for (int i=0; i<row; i++)                       // 輸出 C 矩陣
    {
        for (int j=0; j<column; j++)
            cout << setw(6) << c[i][j];
        cout << endl;
    }

    system("pause");
    return 0;
}
```

師父領進門，修行在個人！

The master teaches the trade, but the apprentice's skill is self-made.