# 最新C程式語言

蔡明志 編著

## 1 C程式概觀

# 1.1 C程式語言

- **控制結構**

  C語言的程式流程控制結構使得電腦科學理論及實務上的要求得以實現；諸如結構化程式(Structure Programming)、由上而下設計法則(Top-down design)、以及模組化(Modular) 等等，這些原則將因C語言而更形自然。

- **效率良好：**

  C是種十分簡潔的語言，它能充分利用現代電腦能力上的諸多功能，譬如近似組合語言(Assembly language)的控制指令，而且還能充分掌握記憶體(Memory)的相關資訊。C語言的簡潔也相對使得執行結果更有效率。

- **可攜性：**

  所謂可攜性就是指在某種系統上開發的程式僅需少數的修改，甚至原封不動便能於另外一種系統上執行。可攜性愈高將可使程式開發的成本大為降低，C語言在這個方面一向居於領導地位，由於設計上的得當，使得C程式極容易於各系統間移植。

```
/* ch1 outline.c */

#include <stdio.h>
#include <stdlib.h>
#define Max 10
int square(int);
int main( )
{
    int i;
    int total;
    int square_total = 0;
    total = 0;
    for (i=1; i<=MAX; i++)
    {
        total += i;
    }
    printf("The sum of 1+2+...+10 is %d.\n", total);
    square_total = square(total);
    printf( "Its square is %d.\n", square_total);
    system("PAUSE");
    return 0;
}

int square(int value)
{
    return(value * value);
}
```

- 基本上，一個典型的C程式是由幾個主要部份組成，包括前端處理程式的指令(Preprocessor directive)、函式（函數）原型宣告、main( )主程式、主程式中的敘述、其它函式(function)定義與主體等等。我們以底下的圖形來做一說明：

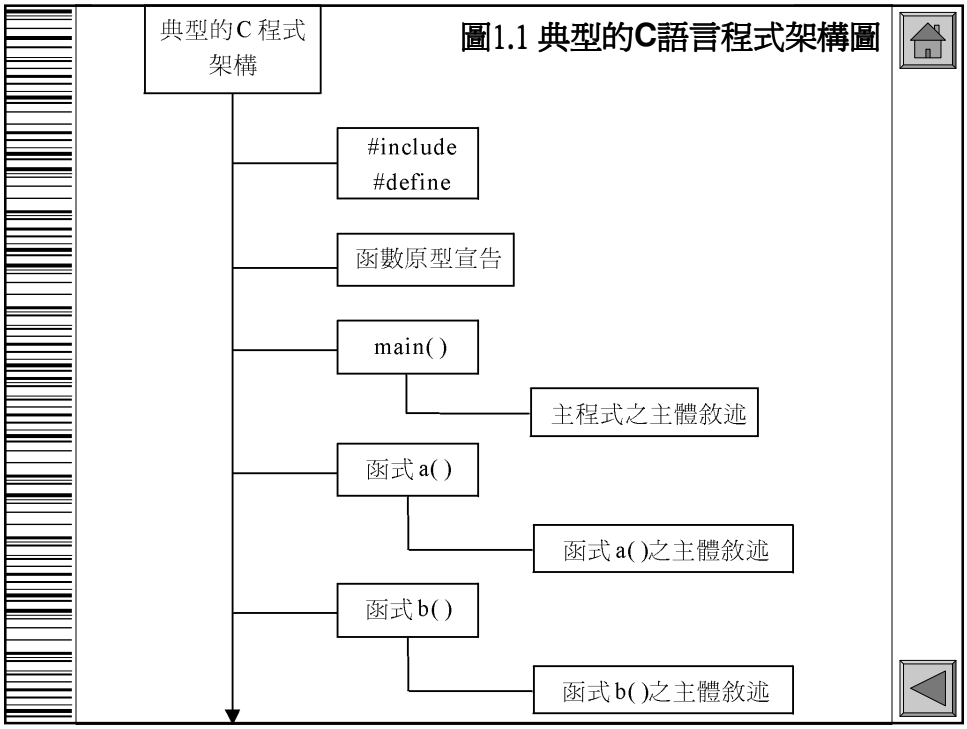
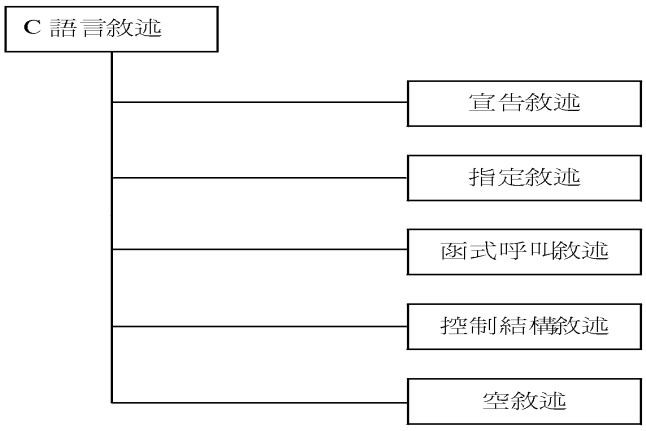



圖1.1 典型的C語言程式架構圖

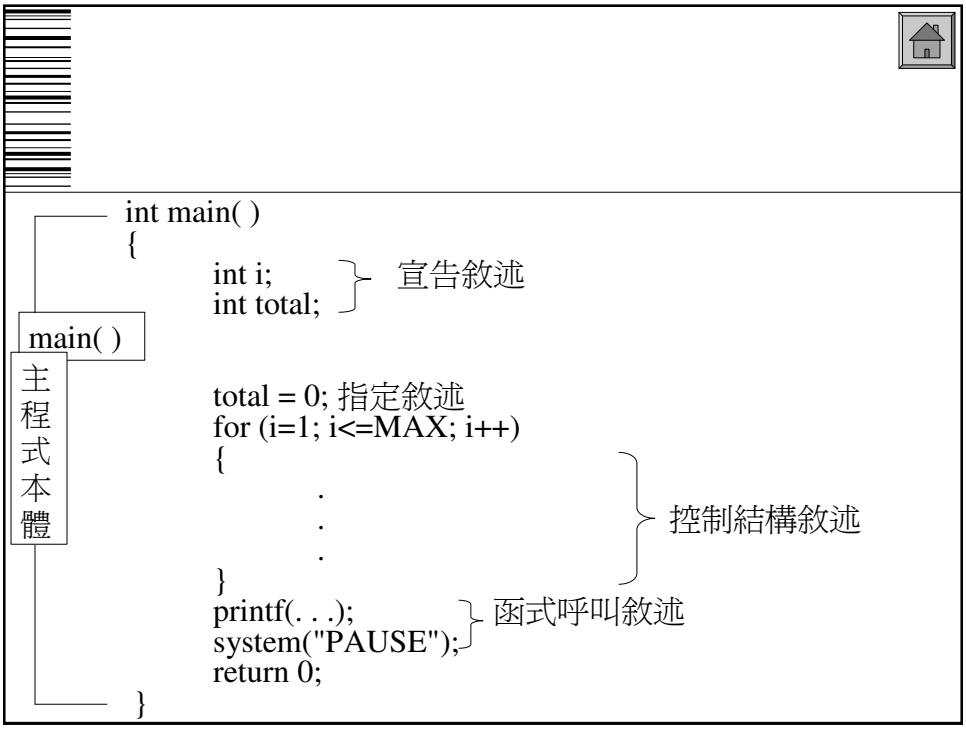
- 我們可以把程式outline.c中的敘述一一與上圖對照：

```
#include <stdio.h>
#include <stdlib.h>      } 前端處理程式的指令
#define MAX 10
int square(int);  ——→  函式原型宣告
int main( )       ——→  C語言主程式
{
        .
        .                } 主體敘述部份
        .
}
```

- C語言的敘述共有五種，如下圖所示：



- 主程式main( )中由大括弧({、})包圍起來的部份都是main( ) 函式主體的敘述，根據上圖它們分別為：



- 位於主程式main( )上方的int square(int)；則是函式原型宣告的一例，它指出有個函式square( )乃為自行定義，這個函式即位於main( )主式的下方：

```
int square(int value)    函式定義
{
  return(value * value);          } 函式主體
}
```

執行結果：

```
The sum of 1+2+...+10 is 55.
Its square is 3025.
```

# 1.3 從一個簡單的範例談起

```
/* ch1 first.c */

#include <stdio.h>                  /* header file */
#include <stdlib.h>
int main( )                         /* function of main( ) */
{
    printf("Hello, ");              /* function statement */
    printf("world.\n");
    printf("Learning C now.\n");
    system("PAUSE");
    return 0;
}
```

- **/*...*/註解**

  就如第一例所出現的 /* ch1 first.c */ 以及散落程式之間的類似部份，都是屬於C程式的註解(comment)。在C程式檔案中，凡是位於 /* 和 */ 之間的所有內容都是註解，它們會被編譯程式忽略；註解的目的是在補充程式的意圖，以便日後自己或他人能正確而方便地理解程式。不論是初學或是專業的程式設計者，都應該養成使用註解的習慣。

- #include <stdio.h>

  這一條並非C語言的命令,而是屬於C語言前置處理程式(preprocessor)所管理的指令(Directive),前一節程式中的 #define也是這類指令,詳細的內容將於後面章節中討論。簡單地說,這條假指令的目的就是於該處引進(include)檔案stdio.h的全部內容,其結果正如同我們於該處鍵入stdio.h檔案內容一般。

  檔案stdio.h乃系統所附的檔案,這類檔案泛稱為標頭檔(Header file),由其附加檔案.h即可看出它的意義。stdio.h的實際意義為標準輸入輸出標頭檔(Standard input/output header file),

檔案中定義了許多重要的常數(Constant)以及函式原型宣告(Function prototype declaration)或語法(Syntax)的宣告,譬如程式中使用到的printf( )函式的語法,是放在stdio.h的標頭檔裏,而system()庫存函式的語法則放在stdlib.h標頭檔中。

系統所提供的函式則稱為庫存函式(library function),如printf()和system()，往後您會看到更多。當我們在程式有使用庫存函式時,需載入其所對應的標頭檔,因為標頭檔有宣告其語法,以便在編譯時期就能判斷呼叫此庫存函式的語法是否正確。

- **main( )**

  由main( )後面跟著的一對小括弧即可看出其乃為一函式，整個程式的執行動作將由main( )函式中的各個敘述依序引發；main的確是個相當平凡的名字，不過它卻是唯一的選擇，所有的C程式不論擁有多少函式，它總會先行尋找main( )這個函式做為進入點(entry point)，然後開始執行。

  位於main之後的小括弧內通常包含許多必須傳給函式的訊息(例如數學上的函數sin(x)，其中的x就是傳給sin( )函數的訊息)。這些訊息謂之為參數(Parameter)，參數的個數不限止於一

個，而在我們的簡單範例中，並不需要傳遞任何訊息給main( )，所以括弧內是空著的；特別要注意，即使不需參數，小括弧也絕對不能省略。

```
{
        •
        •
        •
}
```

大括弧與區塊
(函式本體)

函式main( )之後的敘述被一對大括弧所包圍，它指出了那些命令該是屬於main( )的部份；大括弧可標示函式本體的開頭與結尾，必須注意到，僅有大括弧擁有此種能力，小括弧( )與中括弧[ ]都沒有辦法。

大括弧也可用於集結程式中的一些敘述使之成為一個單位，或稱之為區塊(block)，這方面正如同Pascal或Modula-2等語言中的begin和end。

- **printf( )函式**

接下來位於main( )函式內的即為三條printf( )函式呼叫，我們不難猜出這個函式的目的，它應該會把後面的文字內容顯示在螢幕(screen)上。 再強調一次，printf後面出現括弧即告知編譯程式其乃為一函式，程式中所看到的三個printf( )函式內都出現一群被雙引號 (" ") 圍起來的文字；在C語言中，凡是由雙引號所包含的內容都屬於字串(string，即文字串列)的一部份。在這裡，printf( )會接收一個字串做為參數，然後把它們列印到螢幕上。

輸出結果：

```
Hello, world.
Learning C now.
```

您可以和程式一一對照，printf( )果然忠實地把字串訊息印至螢幕；但以乎有些奇怪，輸出結果第一列最後印出world.之後為何會跑到下一列，而第二條printf( )中的 \n又為何沒有顯現出來?

事實上，在C語言中，字串內的 \n(反斜線後跟著一個特殊字母)乃屬於特殊的單一字元，換句話說，\n會被視為一個字元，而它的作用也並非程式撰寫時所看到的文字。就拿我們的例子來說，\n所代表的是換行字元(Newline character)，而它的作用將使得輸出的位置轉移到下一列，它的效果有點類似Pascal中的writeln( )程序(Procedure)。由於換行字元\n的影響，使得輸出結果變成這樣，第三條printf( )中最後的\n也有著同樣的功能，亦即使游標(cursor)移到下一列的開頭。

- **system("PAUSE");函式**

  程式的執行是一行接一行的，執行到最後結束時便跳離程式，所以當程式有輸出結果時，便看不到其結果，為了避免這種情形發生，利用system("PAUSE");函數是最佳的選擇，其目的是將輸出結果留在螢幕上，待使用者按任意鍵後才跳離。

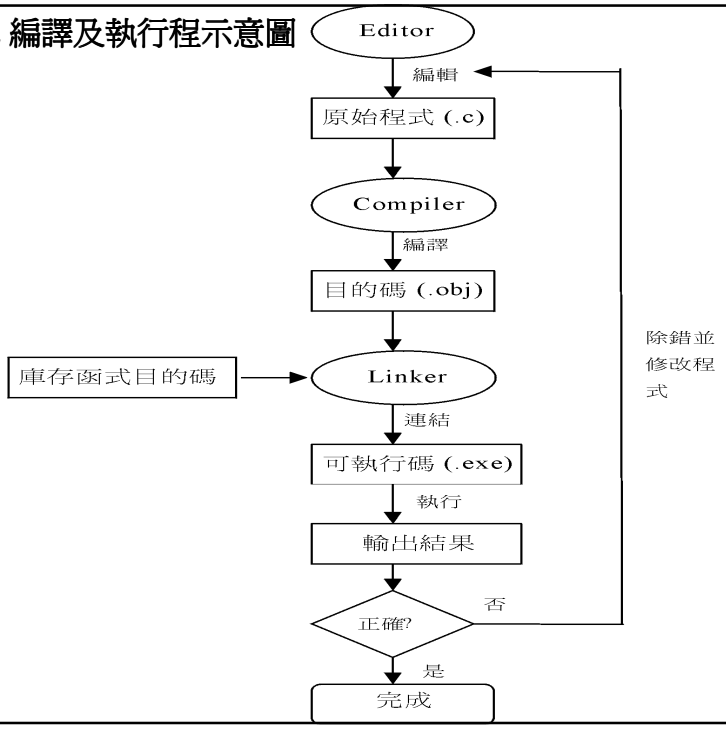
  這一函式對有些編譯程式如Dev-C++是很有用的，但對Visual C++而言，此函式是多餘的，因為Visual C++編譯程式在程式結束後，則會自動出現press any key to continue在螢幕上，待使用者按任意鍵後才跳離。

  由於system函式的語法放在stdlib.h的標頭檔，所以必須載入stdlib.h，因此#include <stdlib.h>是不可或缺的。

- **return 0;敘述**

  由於main函式的資料型態為int，故程式需要有一行return敘述與之匹配。

圖1.2 編譯及執行程示意圖




# 1.5 進一步的範例

```
/* ch1 variable.c */

#include <stdio.h>
#include <stdlib.h>
int main( )
{
    int num;
    int square;

    num = 10;
    square = num * num;

    printf("Square of number %d is %d.\n",num,square);
    system("PAUSE");
    return 0;
}
```

- **int num;**
  **int square;** } **宣告敘述**

宣告敘述(Declaration statement)是C語言重要特色之一，本例中的int num；即說明了兩件重要的事情：

1. 在程式的main( )函式中某處，存在著一個名為num的變數(variable)。
2. 這個num變數的型態(type)乃為整數(integer)，亦即不帶有小數部份的數值。

位於int num之後有個分號(;)，C語言的分號目的在於指出敘述的結束；例如int square;是一條敘述，函式呼叫printf( )也是一種敘述形式，所以後面也要有分號。至於註解或是前端處理程式的指令(例如#include)並非C語言的敘述，所以無需分號；此外，函式main( )因其僅為函式定義，仍然不屬於任何一種敘述，因此main( )後面沒有分號存在。

特別要提出來的是：分號也屬於敘述的一部份，而單獨只有分號的敘述稱為空敘述，表示不做任何事。有時只是為了程式的美觀易懂，或讓程式空轉而已。

- num = 10;

  宣告了變數num之後，就可以指定數值給它，上面這條敘述的意思就是把數值10指定給變數num，因而num便擁有數值10。這裡的符號 '=' 並不是一般的數學符號"等於"(事實上"等於"在C語言中有另外的符號代表)；正確的說法應該是指定(Assign)符號，而其作用方向則為從右到左，譬如像底下這樣：

  10 = num;

  那就不僅沒有意義，而且會被視為錯誤(Error)。在前面那條指定敘述後面同樣必須以分號終結。

- square = num * num;

  這是相當簡單的數學運算，其中的*代表的就是數學上的相乘，把num的值乘以num後，再把結果指定給square，注意到num的值並不會遺失或改變。

  由簡單的數學可知：

  10 * 10 = 100

  所以num此時的值是10，square則為100

- **printf( )函式呼叫**

別急著看printf( )的參數內容，我們先將執行結果顯示出來，如下所示：

```
Square of number 10 is 100.
```

printf( )呼叫是這樣寫的：

```
printf("Square of number %d is
%d.\n", num, square);
```

前面幾個字沒有問題，直到%d的時候，螢幕上卻出現10，而下一次的%d竟也出現100，最後的num，square又是什麼東西?您應該還記得，函式中可允許很多個參數，這些參數必須利用逗點(，)加以分隔；像本例中即有三個參數：

1. 字串參數："Square of ...%d.\n"。
2. 整數參數：num。
3. 整數參數：square。

函式printf( )第一個參數中的 %d並沒有顯現，它的意思是說該處將有一個整數資料出現，而顯現方式則為十進位形式(decimal)。至於實際取代的資料則由第二個參數(即num)決定，其值為10，所以%d之處便顯示出10；第二個%d也是這樣，但它會取出接下來的參數內容，亦即square，它的值經由前一條指定敘述設定為100。

```
/* ch1 bad_style.c */
#include <stdio.h>
#include <stdlib.h>
/* header file */
int main( )
/* function main( ) */
{   printf
   ("Hello, ");   /* function
         Statement */
         printf("world.\n")
         ;

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

# 1.6 變數宣告

```
/* ch1 many_var.c */

#include <stdio.h>
#include <stdlib.h>
int main( )
{
    int i , j, k;
    i = 10;
    j = 20;
    k= 30;

    printf("i + j + k = %d.\n", i+j+k);
    system("PAUSE");
    return 0;
}
```

- 我們希望宣告i，j，k三個int型態的變數，當然也可以寫成：

```
int i;
int j;
int k;
```

但是似乎有些累贅，我們還能採用例子中的宣告方式，把它們濃縮為同一條敘述：

```
int i, j, k;
```

i + j + k = 60.

```
/* ch1 declare.c*/

#include <stdio.h>
#include <stdlib.h>
int main( )
{
        int i=10,j=20;
        int k=30;

        printf("i + j + k = %d.\n", i+j+k);
        system("PAUSE");
        return 0;
}
```

- 宣告時順便加上設定敘述，這是一種良好的習慣，可以避免誤用了某些未經初始化的變數。上面的宣告可以濃縮為：

  int i=10, j=20, k=30;

  變數間仍然以逗號分隔，不過要注意到，下面這種宣告方式：

  int i, j, k = 30;

並不會使i, j, k三個變數都擁有設定值30，其中只有k會有這種影響，i和j所擁有的仍然是記憶體內的垃圾值；為了使程式看起來清晰，最好避免上面這種寫法，而把它寫為兩列：

```
int i, j;
int k = 30;
```

程式declare.c的執行結果與前例完全一樣：

```
i + j + k = 60.
```

- 為變數所選取的名稱最好要能明確地表現出變數的用途，譬如以weight代表某人的體重就遠比單純的w要好得多；如果變數名稱仍然無法充分表達，就只好藉助註解的幫忙：

```
int weight;  /* the weight of a child */
```

這項原則並非一成不變，有些意義上並非如此重要的變數則可儘量簡化，例如註標值(index)的表達用i, j, k就已十分清楚，實在不需要取個冗長的名稱index、position...等等。C語言在變數命名上是相當自由的。

- 變數的名稱可以是文字、數字、以及底線(_)，但是數字不能出現於第一個字。下面所列均為正確的變數名稱：

  正確的變數宣告

```
int number1, number2;
int width_of_screen;
int FirstName;
```

  至於底下的宣告則均為錯誤：

```
int 3man;                /*數字不可開頭*/
int we'll;               /*不允許標點符號*/
int color-screen;        /*連字號無法接受*/
```

- C語言的變數名稱原則上是區分大小寫的，例如FirstName與firstname將被視為兩個不同的變數。

  此外，變數名稱的長度也有所限制，視編譯程式而定；舉例來說，如在一個僅允許8個字元長度的編譯系統而言

```
int computer1;
```

  以及

```
int computer2;
```

  會被視為相同的變數，因為超過8個字元以後的字元都將忽略。

- 善用大寫字母與底線符號。譬如前面的例子，MyName或my_name任何一種方式都要比單純的myname和MYNAME來得好。本書中乃採用小寫字母為主，再配合底線符號來建構所有的變數名稱；至於完全大寫的名稱則保留給符號常數(symbol constant)，譬如第一節outline.c程式中的

  ```
  #define MAX 10
  ```

  在此的MAX即為符號常數的一例，它的值是無法再行改變的。

- 變數名稱不可與C語言的關鍵字相衝突，例如int這個字就是關鍵字，我們不可能寫出

  ```
  int int;
  ```

  如此的宣告將被視為錯誤。

# 1.7 關鍵字

- 這裡所謂的關鍵字(keyword)就是C語言中的字彙，對C來說，這些字十分特別，它們早已賦予特殊的意義，您不可以再使用它們做為變數或常數的名稱。

- 許多關鍵字是用來指定型態，例如我們常見的int型態，也有一些關鍵字乃用以控制程式流程，像是if、while、goto...等等，底下我們就列出標準C的關鍵字：

**表1.1 ANSI C 語言關鍵字**

| | | | |
|---|---|---|---|
| Auto | break | case | char |
| const | continue | default | do |
| double | else | enum | extern |
| float | for | goto | if |
| int | long | register | return |
| short | while | signed | sizeof |
| static | struct | switch | typedef |
| union | unsigned | void | volatile |