第二章

C語言的基本結構

前言

- 在第一章中，相信讀者已經熟悉如何編譯C語言程式。在本章中，我們將透過一個非常簡單的C語言程式來說明C語言的程式結構。並且從本章開始，我們將實際撰寫每一個C語言範例程式。

大綱

- 2.1最簡單的C程式範例
- 2.2註解（comment）
- 2.3前置處理指令－#include
- 2.4C程式的進入點main(…)函式
- 2.5敘述（statement）
- 2.6自由格式與空白字元
- 2.7深入C語言的文法【補充】
- 2.8本章回顧

2.1 最簡單的C程式範例

- 下面是一個簡單的C程式範列，請逐字將之輸入到『.c』的檔案中，慢慢培養屬於自己的程式風格。（若您的IDE已經幫您建立了某些預設內容，請先將它刪除後再輸入。）
- 範例2-1：ch2_01.c

2.1 最簡單的C程式範例（續）

```
1  /*****************************/
2  /*  檔名:ch2_01.c            */
3  /*  功能:簡單的C程式範例     */
4  /*****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  main(void)
10 {
11  //printf(“Hello Word.\n”);/*C++註解方式*/
12  printf("歡迎使用C語言!\n");
13  printf("這是一個簡單的C程式.\n");
14  system("pause");
15 }
```

2.1 最簡單的C程式範例（續）

- 執行結果：

```
歡迎使用C語言!
這是一個簡單的C程式.
請按任意鍵繼續 . . .
```

- 範例說明：
  - 雖然範例2-1是一個只有15行的C語言程式，但是卻說明了C程式的基本結構如下，我們將分別加以說明。

2.1 最簡單的C程式範例（續）

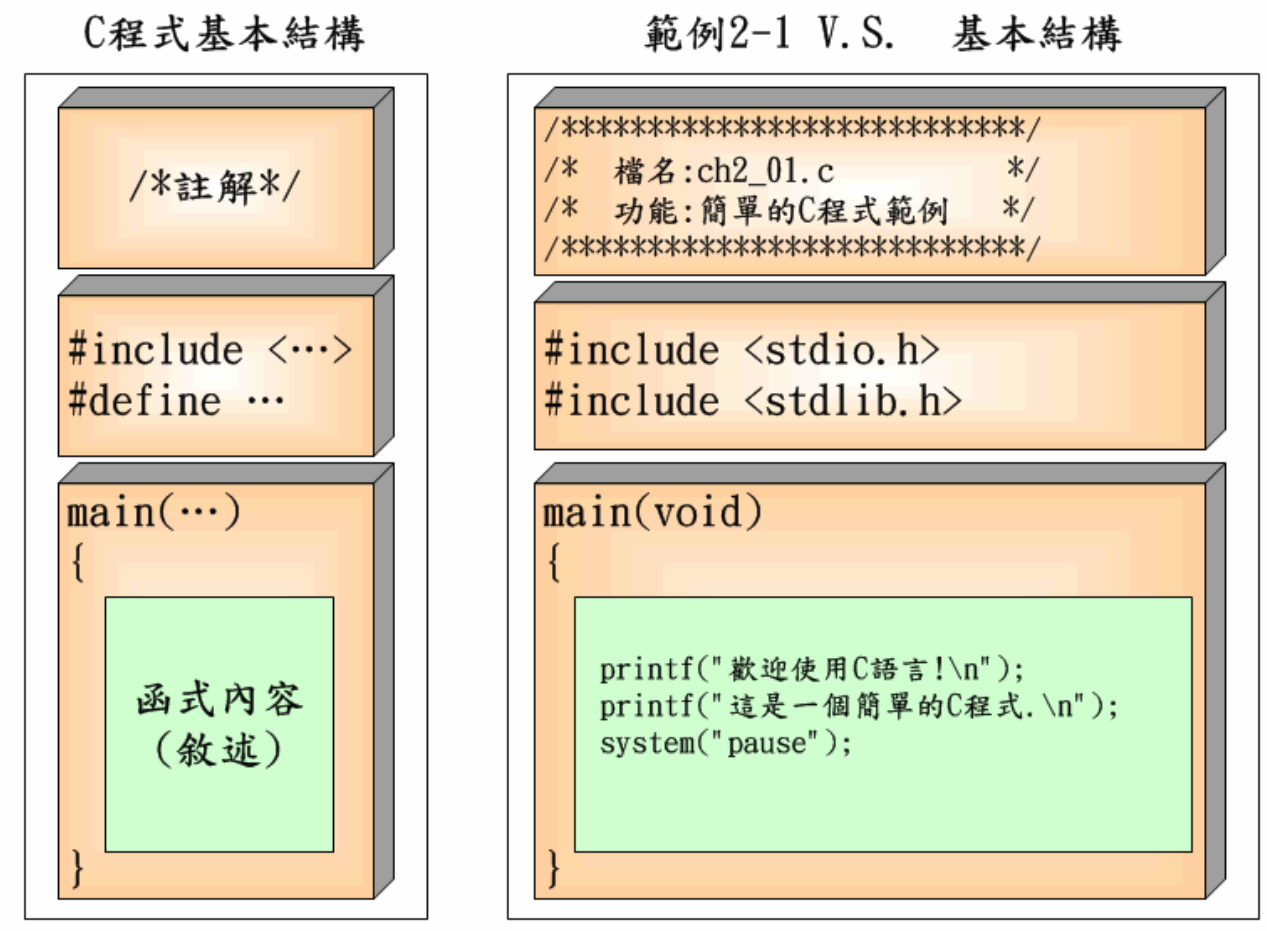


圖2-1 C程式基本結構

2.2 註解（comment）（續）

- C的註解符號爲『/*....*/』，在『/*』到『*/』之間的所有文字都將被編譯器忽略（事實上，註解將被前置處理器刪除後才輸入給編譯器），換句話說，沒有這些註解並不會影響程式的正確性。所以這些文字可以當作說明該程式或程式片斷之用，善用註解文字將有助於日後維護程式時，快速了解程式功用。）

2.2 註解（comment）（續）

- 由於『/*...*/』具有換行功能，也就是可以將註解跨行描述，因此範例2-1的註解可以改寫如下。

```
/*****************************
 *   檔名:ch2_01.c           *
 *   功能:簡單的C程式範例    *
 *****************************/
```

2.2 註解（comment）（續）

- 【錯誤用法】：
  - 由於『/*...*/』註解格式在編譯器中，會從第一個遇到的『/*』開始視爲註解文字，直到遇到第一個『*/』爲止，因此並不允許使用巢狀註解文字，例如下列的程式碼中，就犯了這個錯誤。

```
/*********外部註解*********
 *         /*內部註解*/         *
*********外部註解*********/
```

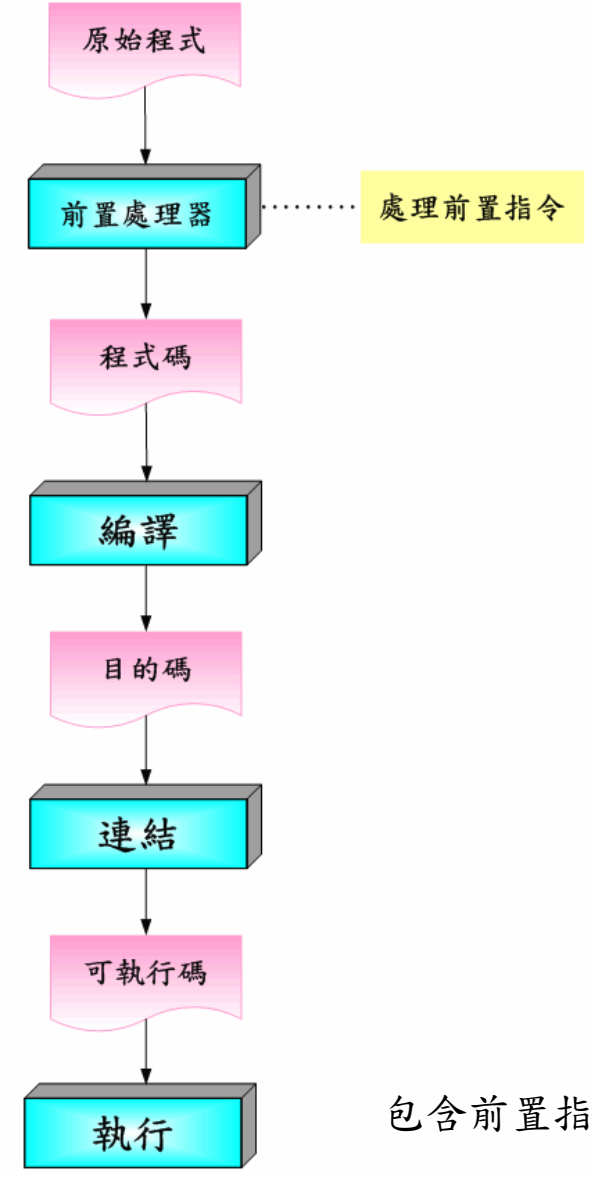
2.3 前置處理指令－#include

- 在C語言中，前置處理指令是前面出現「#」符號的指令，嚴格說起來，前置處理指令並非C語言的指令，因爲這些指令會在程式進行編譯之前，先被前置處理器（preprocessor）置換成某些程式碼，因此前置處理指令又稱爲假指令。在此，我們先說明#include這個前置處理指令，至於其他的前置處理指令，則留待後面章節中再做說明。

2.3 前置處理指令－#include（續）

- #include的功用在於引入標頭檔，所謂標頭檔就是放置某些已經撰寫完成函式內容的函式庫檔案，這些標頭檔可能是由編譯器所提供，也可能是自行撰寫的函式庫。我們必須先引入標頭檔，才可以使用標頭檔內所提供的函式。
  - 例如：stdio.h標頭檔中定義了printf()與scanf()函式的內容，stdlib.h標頭檔中定義了system()函式的內容。因此，若我們在程式中要使用這些函式的話，就必須使用#include <stdio.h>及#include <stdlib.h>將這些函式庫檔案引入。

2.3 前置處理指令－#include （續）



包含前置指令的編譯過程

2.3 前置處理指令－#include（續）

- #include引入標頭檔可分爲下列兩種格式：
  - #include <xxx.h>
    - xxx.h爲C編譯器提供的標頭檔，並且存放在編譯器內定的目錄中，使用此種格式，前置處理器會自動到內定目錄中找到標頭檔。

2.3 前置處理指令－#include（續）

ANSI C函式庫標頭檔

函式庫標頭檔名稱	函式種類
<stdio.h>	標準輸入與輸出
<ctype.h>	字元分類測試
<string.h>	字串處理與轉換
<math.h>	數學函式
<stdlib.h>	標準函式庫，提供各類基本函式
<assert.h>	例外偵測，有助於除錯
<stdarg.h>	引數串列的測試
<setjmp.h>	非區域跳躍
<signal.h>	訊號偵測
<time.h>	提供各類時間函式
<limits.h> 及 <float.h>	float.h提供浮點數的精確位數定義，limits.h提供某些極限值的定義

2.3 前置處理指令－#include（續）

- 【函式庫的功能】：
  引入的函式庫會在前置處理時被載入，通常函式庫會記錄許多的函式，除此之外，函式庫也可能定義了某些符號常數。
  - 例如數學的無限大在電腦中根本無法完全實現，因此只能用可儲存的最大值來代表，例如INT_MAX代表在int資料型態下的最大值2147483647，而INT_MAX的定義也被包含在limits.h之中。

2.3 前置處理指令－#include（續）

- #include "ooo.h"
  - ooo.h不是編譯器提供的標頭檔，所以程式設計師必須標明該檔案所在目錄，以便前置處理器取得該檔案。

2.4 C程式的進入點main(...)函式（續）

- C語言屬於模組化設計的一種語言，而C語言的模組則是以函式來加以表示。換句話說，C程式是由各個不同功能的函式所組成，並且函式與函式之間可以透過呼叫以及回傳值方式加以聯繫，一個函式的基本定義格式如下：

```
函式回傳值型態　函式名稱(傳入引數)
{
    函式內容（敘述群）
}
```

2.4 C程式的進入點main(...)函式（續）

- 函式的基本格式爲『函式名稱()』，由於函式可以於被呼叫時接受呼叫者傳入引數，因此這些引數必須在『()』內加以宣告。

- 函式也可以回傳資料給呼叫者，所以我們也必須在函式名稱前面宣告回傳值的資料型態（關於資料型態請見第三章）。

- 函式的內容則是包含在『{』與『}』之間。

2.4 C程式的進入點main(...)函式（續）

- main()函式是命令列(Console Mode)C程式的進入點，換句話說，當我們在命令列式的作業系統中執行由C所撰寫的應用程式時，會先從main()函式開始執行。

```
main()
{
 .......
}
```

```
void main()
{
 .......
}
```

```
main(void)
{
 .......
}
```

```
void main(void)
{
 .......
}
```

2.4 C程式的進入點main(...)函式（續）

- 除了上述的介紹之外，main函式還具有以下兩個特點：
  - (1) 唯一性：在C函式中，任何函式都具有唯一性，main函式也是如此。
  - (2) 必要性：爲了讓作業系統能夠找到程式進入點，因此不可省略或缺少main函式。

2.5 敘述（statement）（續）

- C語言是模組化設計，並利用區塊來撰寫程式內容，區塊的符號爲『{}』，不論是函式、迴圈、決策都是使用區塊符號來包裝內容。
  - 而區塊內容則是由敘述（statement）所組成，
  - 每一個敘述後面必須加入『;』做爲結束。

```
#include <stdio.h>
#include <stdlib.h>

main(void)
{
 printf("歡迎使用C語言!\n");
 printf("這是一個簡單的C程式.\n");
 system("pause");
}
```

2.5 敘述（statement）（續）

- 前兩個敘述都是呼叫printf()函式，
『()』內的字串則是傳入printf()函式的引數，
而printf()函式則已經定義在<stdio.h>標頭檔中。

- 最後的敘述是呼叫system()函式，『()』內的字串“pause”則是傳入system()函式的引數，代表要作業系統執行pause指令，而system()則定義於<stdlib.h>標頭檔中。

- 執行結果中的『請按任意鍵繼續．．．』其實就是執行**system("pause")**的效果，它會等待使用者按下任意鍵之後才會繼續後面的動作，您可以開啓一個Dos環境，並且單獨輸入pause指令，看看會有什麼結果。

2.5 敘述（statement）（續）

- C語言的敘述除了函式呼叫之外，還有以下幾種類型，我們將在後面章節中加以說明：
  - 算式敘述 (Expression Statement)
  - 複合敘述 (Compound Statement)
  - 選擇敘述 (Selection Statement)
  - 迴圈敘述 (Iteration Statement)
  - 標籤敘述 (Labeled Statement)
  - 跳躍敘述 (Jump Statement)

2.6 自由格式與空白字元

- C語言採用自由格式撰寫，換句話說，您可以去除程式中各敘述間的所有空白字元(spaces、tabs…等等)及換行符號（carriage、return），編譯器仍會正確編譯程式，例如：您可以將範例2-1中main的內容改寫如下：

```
main(void){ printf("歡迎使用C語言!\n"); printf("這是一個簡單的C程式.\n");system("pause");}
```

- 雖然省略空白字元以及換行符號能夠使得程式行數減少，但並不會加速程式的執行效率。

2.6 自由格式與空白字元（續）

- 【重點提示】：
  『“』內的空白字元則不會被編譯器忽略，因爲『”』在C語言中，是用來表示字串。

- 【常見的錯誤】：
  C程式和Unix/Linux作業系統一樣，對於字元的大小寫是有所區別的，因此您不能將main(void)改寫爲Main(void)或MAIN(VOID)。

2.7 深入C語言的文法【補充】

- C語言程式基本上仍舊是一個純文字檔案，因此是由眾多字元所構成，這些字元將會被編譯器的掃描程式分割爲句元(Token)。這些句元可能代表的是關鍵字、保留字、識別字、運算子、變數名稱或資料值。
- 眾多句元又可以組合成敘述(statement)，敘述的結尾則是『;』。而眾多的敘述則成爲函式內容。
- 編譯器的剖析程式(parser)會針對C程式的敘述是否符合C語言的文法(Grammar)加以判斷，若不符合文法，則會產生錯誤訊息。

2.8 本章回顧

- 在本章中，我們學習到C程式的基本結構。本章所學習到的內容如下：
  - (1) C程式的基本結構包含3大部分：
    - 程式註解、前置處理指令、函式及敘述
  - (2) 在C程式中，可使用『/*...*/』做爲註解符號。
  - (3) 純文字模式的C程式進入點爲 main( )函式。
  - (4) 一般敘述以「;」做結尾。
  - (5) C語言的輸出函式printf( )的簡單使用方法如下（我們將在第四章中，說明printf()的進階使用方法）
    - printf("輸出內容");

2.8 本章回顧（續）

- (6) C語言執行作業系統環境的指令可以透過system()來執行，將想要執行的指令包裝爲字串當作引數傳送給system()函式即可，格式如下。（並非所有的作業系統指令都可以用這個方式來執行，實際上還必須視作業系統與編譯器提供了哪些指令。）
  - system("作業系統指令");

- 在C程式結構中，除了上述的3大部分之外，還包含其他細節，例如：全域變數的宣告應該出現在其他函式宣告之外。這些細節，我們將於後面章節中分別加以介紹。

本章習題