

# Chapter 9

## C 的前置處理器

### 本章重點

- 9.1 C 語言的編譯過程
- 9.2 前置處理器
- 9.3 巨集代換指引
- 9.4 自訂標頭檔
- 9.5 條件式編譯指引
- 9.6 實例-書籍介紹
- 9.7 習題



### 9.1 C 語言的編譯過程

我們將整合開發環境下所編寫的程式碼稱為「原始程式」，其附檔名為 \*.c。當您在 Dev C++ 整合開發環境下執行功能表的 [執行(Z)/編譯並執行(O)] 時，原始程式分別經過下列前置(Preprocess)、編譯(Compiling)、組譯(Assembling)處理，無錯誤之後才能執行程式。

#### 1. 前置處理

前置處理即是在程式做翻譯之前要做的工作，主要是因為 C 語言在程式前面大都使用一些不屬於 C 語言的敘述，我們稱為「前置處理指引」(Preprocessor Directive)，編譯前必須將這些敘述交給前置處理器，將它擴充成 C 語言，再進行下一步的編譯處理。

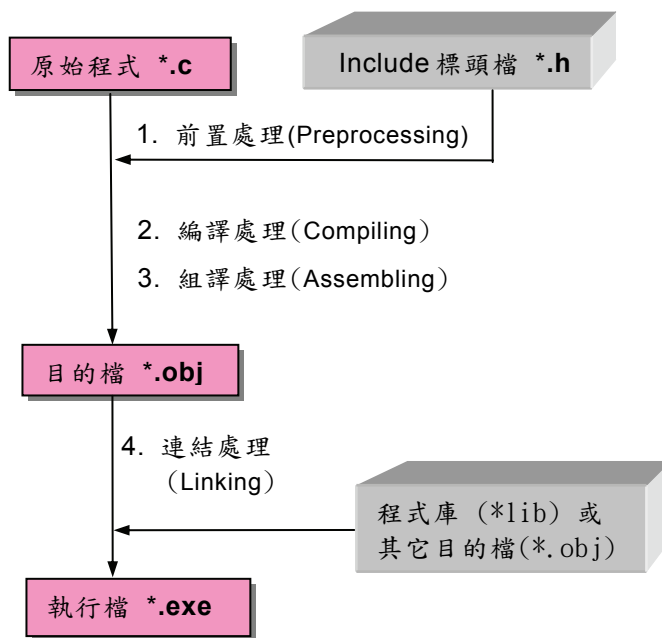
#### 2. 編譯處理

在編譯過程中，主要的工作是對原始程式做字彙分析(Lexical Analysis)、語法分析(Parsing)、語意分析(Semantic analysis)、產生中間碼(Intermedia code generation)、程式碼最佳化(Optimization)、產生組合語言程式碼(Code generation)六大步驟。前三步驟若有發生錯誤，編譯器會停止編譯，此時必須將發生錯誤的地方更正，再重新編譯一直到無錯誤為止，此時編譯器會進行第 4-6 步驟，將程式中所有敘述透過 Code generation 轉成更低階的組合語言。至於字彙分析是將程式中所有敘述拆成有意義的字串，我們將這些獨立字串稱為「Token」。譬如：total=price\*qty; 分成 "total"、"="、"price"、"+"、"qty"、";" 六個 Tokens。語法分析就是檢查這些 Token 是否符合 C 語言的文法規則，譬如：是否漏打符號、括號、或括號不成對等錯誤發生。語意分析是檢查是否有拼錯字、變數是否有宣告等錯誤發生。

### 3. 組譯處理

組譯處理主要的工作是將上述編譯完成的組合語言程式，透過編譯器內部所提供的組譯器(Assembler)組譯成機器語言，我們將所得的機器碼稱為「目的檔」其附檔名為 \*.obj。

當程式編譯完成沒有發生錯誤，便可交給連結器(Linker)，連結相關檔案，而產生一個可執行檔，其附檔名為\*.exe。此時程式便可以直接執行。



## 9.2 前置處理器

C 語言的前置處理器(Preprocessor)是一個巨集處理器，在編譯器編譯原始程式之前會自動啟動，主要用來處理 C 程式中含有 # 符號開頭的敘

述，我們稱為「前置處理指引」(Preprocessor directive)敘述。透過檔案含入指引(`#include`)將程式中指定的標頭檔(Header File)含入到程式中，或是巨集代換指引(`#define`)將程式中經常使用的常數、字串、函式以巨集名稱取代，以及使用條件式編譯指引依據不同環境需求，在編譯時選擇編譯不同的敘述。由此可知，前置處理指引和一般 C 語言所提供的敘述最大的差異是，該指引最前面必須以 `#` 符號(pound sign)開頭，它可以出現在程式任何地方。習慣上，將前置處理指引放在 `main()`主函式或自定函式的前面，所以，前置處理指引的有效範圍是從指引所在處開始，一直到程式結束為止，除非使用`#undef` 指引才可使其有效範圍中途失效。前置處理指引不必像函式必須呼叫，當編譯器在編譯程式之前會自動啟動。至於 C 語言所提供的前置處理指引敘述有下列三種：

1. 檔案含入功能：`#include`
2. 字串置換和巨集定義：`#define`、`#undef`
3. 條件編譯：`#if...#elif...#else...#endif`，`#ifdef...#else...#endif`，`#ifndef...#else...#endif`



## 9.3 巨集代換指引

### 9.3.1 如何定義巨集

C 語言的 `#define` 巨集代換指引允許您使用有意義的名稱來代替特定的常數、字串、函式或數學公式。其語法如下：

語法 1：`#define` 巨集名稱 [替代內容]

語法 2：`#define` 巨集名稱(引數列) 運算式

功 能：巨集名稱可用來定義常用的常數、字串、簡單的數學公式或函式。

1. 巨集名稱：和一般變數的命名規則相同，習慣上名稱最好有意義且以大寫英文字母來命名，巨集名稱中間不允許有空白。如下敘述設定 PI 巨集名稱的值為 3.14。

```
#define PI 3.14
```

2. 替代內容：設定巨集名稱被替代的內容、可以指定數值或字串的資料。當程式中出現此指引時，在編譯前的前置處理器會將程式中每個巨集名稱都使用接在其後的替換內容取代，再進行程式的編譯。如下敘述設定 MSG 巨集名稱的替代內容為「Hello」。

```
#define MSG "Hello"
```

3. 巨集名稱前後至少空一格，此指引最後不可加『;』分號當結束符號。

### 9.3.2 巨集定義符號常數

所謂常數是指程式在執行過程中，其值保持不變，不像變數會改變，為避免程式執行時更改其值，且避免和變數產生混淆，將常數使用符號名稱是最佳的選擇。假設程式中有十多處使用到圓周率 3.14，碰到欲將 3.14 改成 3.1416，便要牽一髮動全身，稍一疏忽有圓周率的地方未改到，便會發生錯誤。若能善用巨集，將圓周率使用巨集名稱代替所有的 3.14，如此修改圓周率時，只要更改 #define 那一行的圓周率值，在 main() 主函式或函式內由於使用巨集名稱代替圓周率而不用變更，使得程式易維護且可讀性高。下表列出常用巨集替換指引寫法。

常用巨集替換指引寫法	程式中寫法
#define PI 3.14	printf("%.2f", PI);
#define BUFFER_SIZE 4096	char buff[BUFFER_SIZE]
#define TAX_RATE 0.05	total = price*qty*TAX_RATE
#define ARRAY_SIZE 20	char myArray[ARRAY_SIZE]
#define EQ ==	if (ch EQ 10)
#define EOF (-1)	while (ch=getchar() != EOF)

#define NULL '0'	if (ch==NULL)
#define msg "Please Enter your Name : "	printf(msg);
#define TRUE 1	return(TRUE)
#define YES 1	if(ans ==YES)
#define FALSE 0	return(FALSE)
#define NO 0	if(ans == NO)
#define TWO 2	if(keyin == TWO)
#define FOUR TWO*2	sum=FOUR+TWO
#define SECONDS_PER_DAY (60*60*24)	tot_scnds=TWO*SECONDS_PER_DAY

新的 ANSI C 標準中定義一個 `const` 修飾字，其效果和使用 `#define` 巨集定義符號常數相同，兩者寫法比較如下：

```
const float PI=3.1416;    /* 尾部必須加分號 */
#define PI 3.1416         /* 尾部不允許加分號 */
```

下面是使用巨集定義常數、數學公式、指令代換、函式代換的完整範例，在程式中能夠活用巨集，可使得程式易維護且增加程式的可讀性。在 `main()` 主函式中有使用到巨集的最後面備註，都寫出該巨集經過前置處理器處理過所轉換敘述的結果，請看下面例子的示範。

### 【程式碼】 FileName:macrol.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #define PI 3.1416                /* 定義常數 */
04 #define AREA(r) PI*r*r          /* 定義數學公式 */
05 #define INPUT scanf              /* 指令代換 */
06 #define OUTPUT printf           /* 指令代換 */
07 #define MSG "Please input radius : " /* 字串代換 */
08 #define FMT "The circle area is : %6.2f\n" /* 格式代換 */
09 #define BEEP "\007"              /* 控制字元發出嗶聲代換 */
10
```

```

11 int main(int argc, char *argv[])
12 {
13     int r;
14     OUTPUT(MSG);          /* printf("Please input radius :"); */
15     INPUT("%d",&r);       /* scanf("%d",&r); */
16     /* printf("The circle area is : %6.2f\n",3.1416*r*r); */
17     OUTPUT(FMT,AREA(r));
18     OUTPUT(BEEP);         /* printf("\007"); */
19     system("PAUSE");
20     return 0;
21 }

```



### 【執行結果】

## 9.3.3 巨集使用引數

巨集使用引數(Argument)看起來和函式很相似，由於引數都用小括號括起來。也可在巨集中再插入巨集變成巢狀巨集。譬如下列敘述都是典型的引數範例：

```

#define SQR ((x)*(x))          /* 某數的平方 */
#define CUBE (SQR(x)*(x))     /* 某數的立方 */
#define ABS(x) ((x)<0 ? -(x):(x)) /* 取絕對值 */
#define MAX2(x,y) ((x)>(y)?(x):(y)) /* 兩數取最大 */
#define MIN2(x,y) ((x)<(y)?(x):(y)) /* 兩數取最小 */
#define MIN3(x,y,z) MIN(x,MIN(y,z)) /* 三數取最小 */
#define MIN4(w,x,y,z) MIN(MIN(w,x),MIN(y,z)) /* 四數取最小 */
#define EQ(x,y) (x=y)

```

```
#define swap(x,y) (EQ(t,x),EQ(x,y),EQ(y,t)) /*記得 main()中要宣告 x, y, t */
```

撰寫上面含有引數的巨集時，加上小括號要格外小心，由於小括號會改變運算先後次序，稍不注意可能會發生意想不到的結果，撰寫時應留意。譬如：下表是含有引數的巨集，用來計算某數的平方，引數加小括號的位置不同而產生不同的運算結果：

引數 x	#define SQR(x) x*x	#define SQR(x) (x)*(x)
4	SQR(4)→4*4=16	SQR(4)→(4)*(4)=16
x=4	SQR(x+2)→ x+2*x+2=16	SQR(x+2)→(x+2)*(x+2)=6*6=36
x=4	SQR(++x)→++x*++x=5*6=30	SQR(++x)→(++x)*(++x)=5*6=30
4	160/SQR(4)→160/4*4=10	160/SQR(4)→160/(4)*(4)=160
引數 x	#define SQR(x) (x*x)	#define SQR(x) ((x)*(x))
4	SQR(4)→(4*4)=16	SQR(4)→((4)*(4))=16
x=4	SQR(x+2)→( x+2*x+2)=16	SQR(x+2)→((x+2)*(x+2))=6*6=36
x=4	SQR(++x)→(++x*++x)=5*6=30	SQR(++x)→(((++x)*(++x)))=5*6=30
4	160/SQR(4)→160/(4*4)=10	160/SQR(4)→160/(((4)*(4)))=10

### 9.3.4 巨集的副作用

由上表可看到在定義含有引數的巨集時，引數內加入小括號使用時要小心，以免巨集經過替換成敘述後，運算後的結果與預期不一樣即產生副作用(Side Effect)。譬如上表中求某數的平方：

```
#define SQR(x) x*x
printf("%d", SQR(2+6));
```

執行結果為 40，並不是您想要的  $8^2 = 64$ 。主要由於上面 printf()函式內的巨集經過前置處理後變成：



`printf("%d",  $\frac{2+6}{x} * \frac{6+2}{x}$ );` → `printf("%d",  $\frac{2+36}{2} * \frac{2}{2}$ );` → 結果顯示 40

若將上面定義巨集時，替換字元使用括號括住。寫成：

```
#define SQR(x) (x)*(x)
printf("%d", SQR(2+6));
```

上面 `printf()` 函式內的巨集經過前置處理後，小括號內的運算事先處理，替換後變成：

`printf("%d",  $(2+6) * (6+2)$ );` → `printf("%d",  $8*8$ );` → 結果顯示 64

所以，在引數前後加上小括號，才能得到正確的結果。

### 9.3.5 巨集與函式的差異

設計程式時，有很多地方可使用含有引數的巨集或是使用函式，到底應選擇哪種方式撰寫，我們可由時間(Time)和空間(Space)兩方面來考慮：

1. 由於巨集在編譯前先進行前處理時，會將巨集進行代換變成程式中的敘述。由上節附表中可知巨集引數稍微使用不當易引起副作用。至於函式則不會發生此種現象。
2. 若在程式中使用到 10 次巨集，在進行編譯前，前處理器會進行取代的動作，因此較費時一點。但執行時不必像呼叫函式時，必須跳到函式所在處，待執行完再返回原呼叫處的下一個敘述，因此巨集執行時間較快，但程式空間加長。
3. 若在程式中呼叫函式 10 次，只要在程式中留下一份函式的拷貝，呼叫函式時跳到函式所在處，待執行完再返回原呼叫處的下一個敘述，因此函式比巨集較花費時間，也就是說函式執行時間較慢，但程式長度不會加長。

4. 函式在編譯時會做參數檢查；巨集指引編譯前已變成敘述，所以不會做參數檢查。

本範例主要目的在熟悉含有引數巨集的巢狀用法。macro2.c 範例製作一個功能表選項，選 "1" 時，顯示兩數的最大值。選 "2" 時，顯示三數的最大值。選 "3" 時，顯示四數的最大值。選 "0" 時結束程式執行。

### 【程式碼】FileName:macro2.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #define ZERO 0
04 #define ONE 1
05 #define TWO 2
06 #define THREE 3
07 #define EQU ==
08 #define GREATER >
09 #define LESS <
10 #define MAX2(x,y) ((x)>(y) ? (x):(y))
11 #define MAX3(x,y,z) MAX2(x,MAX2(y,z))
12 #define MAX4(w,x,y,z) MAX2(MAX2(w,x),MAX2(y,z))
13
14 int main(int argc, char *argv[])
15 {
16     int a=10,b=8,c=30,d=15;
17     int ch;
18     printf("\n ===== Choose Maxinum =====");
19     printf("\n    1. Compare two numbers ");
20     printf("\n    2. Compare three numbers ");
21     printf("\n    3. Compare four numbers");
22     printf("\n    0. Quit :");
23     printf("\n =====");
24     printf("\n    Please Choose Number [0-3] : ");
25     scanf("%d",&ch);
26     if (ch EQU ZERO)
27     {
28         exit(0);
29     }
30     else
```

```
31     {
32         if (ch > 3 || ch < 1)
33         {
34             printf(" Error !!   Please Keyin 1-3 ");
35         }
36         else
37         {
38             switch (ch)
39             {
40                 case ONE:
41                     printf(" The MAX(%d,%d) is : %d \n",
42                         a, b, MAX2(a,b));
43                     break;
44                 case TWO:
45                     printf(" The MAX(%d,%d,%d) is : %d \n",
46                         a,b,c, MAX3(a,b,c));
47                     break;
48                 case THREE:
49                     printf(" The MAX(%d,%d,%d,%d) is : %d \n",
50                         a,b,c,d, MAX4(a,b,c,d));
51                     break;
52             }
53         }
54     }
55     printf("\n");
56     system("PAUSE");
57     return 0;
58 }
```



### 【執行結果】

```
C:\dev\ch09\macro2\macro2.exe

===== Choose Maximum =====
1. Compare two numbers
2. Compare three numbers
3. Compare four numbers
0. Quit :
=====
Please Choose Number [0-3] : 3
The MAX(10,8,30,15) is : 30
請按任意鍵繼續 . . .
```

由上可知，在編譯前經過前置處理器會將程式中的巨集直接轉換成所指定的敘述。和函式比較起來，使用巨集可以省掉程式執行時再去呼叫函式的時間，以縮短程式的執行時間，增加程式的可讀性及效率。因此定義巨集可以替代一些簡單的運算式或單行的簡單函式，而函式則適用於較複雜的功能。



## 9.4 自訂標頭檔

### 9.4.1 含入標頭檔

假若您寫了一些好用的巨集或函式，當您在撰寫新的程式時，需要用到一些裡面的巨集，難道需要再打一次嗎？答案是 不！您可以將這些巨集單獨存成一個附檔名\*.h 的標頭檔(Header File)，透過 C 語言本身所提供的 `#include` 檔案含入指引，在編譯前進行前處理，將指定的標頭檔含入到該程式中，便可直接套用。至於標頭檔的寫法有下列兩種方式：

```
#include <filename.h>      /* 使用角括號括住 */  
#include "filename.h"      /* 使用冒號括住 */
```

若 `#include` 後面是使用角括號括住標頭檔，是用來告訴前置處理器，到系統預設的資料夾去尋找指定的標頭檔。若使用雙冒號括住標頭檔，告知前置處理器先到您目前正在使用的資料夾去尋找指定的標頭檔，若找不到才會到系統預設的資料夾去尋找指定的標頭檔。兩者都會將找到的標頭檔內的所有敘述取代 `#include` 這行指引。

標頭檔一般都是用來存放 C 的符號常數或巨集代換，以提供給多個程式使用。所以，`#include` 指引能在一個原始檔案中載入另一個檔案。譬如：若您大部份時間都撰寫以數學為主的程式，那麼就可以將常用的數學公式做成巨集放在一個標頭檔內，以供日後設計新程式時，只要使用 `#include`

含入該標頭檔到此程式中，不必再重寫這些巨集。C 語言自己本身提供許多標準的標頭檔，下表列舉系統所提供常用的標準標頭檔所對應的內建函式，使用下列函式時要記得含入對應的標頭檔。關於這些函式的語法功能，請自行參閱附錄 C。

標準標頭檔	使用下列敘述必須含入左側的標頭檔
stdio.h	getchar()、getc()、gets()、putc()、puts()、putchar()、fopen()、fclose()、feof()、fgetc()、fgets()、fprintf()、sprintf()、fread()、fputc()、fputs()、fwrite()、fscanf()、fseek()
stdlib.h	abs()、rand()、srand()、atof()、atoi()、free()、malloc()、exit()
time.h	clock()、ctime()、time()
ctype.h	isdigit()、islower()、isupper()、tolower()、toupper()、ispunct()、isctrl()、isalnum()、isalpha()
string.h	strcat()、strcmp()、strcpy()、strlen()、strncpy()、strtok()、strncpy()、strncmp()、strncat()
math.h	sqrt()、pow()、exp()、log()、log10()、floor()、ceil()、fabs()、sin()、cos()、tan()、asin()、acos()、atan()

**[簡例 1]** 若 macro1.h 標頭檔與 C 語言的 main() 主函式放在相同資料夾下，則主函式 main() 含入標頭檔時，只須指定標頭檔檔案名稱即可，其寫法如下：

```
#include "macro1.h"
```

**[簡例 2]** 若自訂標頭檔 macro2.h 放在 C:\macro 資料夾下，且不與 C 語言的主程式放在相同資料夾下，則 main() 主函式含入自訂標頭檔時必須指定該標頭檔完整的實際路徑，其寫法如下：

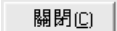
```
#include "c:\macro\macro2.h"
```

## 9.4.2 標頭檔的建立

將上一節 macro2.c 程式中第 3-12 行所定義的巨集存成一個標頭檔，檔名為 my.h。請按照下列步驟，學習如何在 Dev C++ 環境下建立標頭檔其檔名為 my.h。這個範例我們會使用兩個程式檔，分別是 my.h 為自訂標頭檔；headerfile.c 為主程式，並在 headerfile.c 含入 my.h。本例的執行結果與範例 macro2.c 相同。

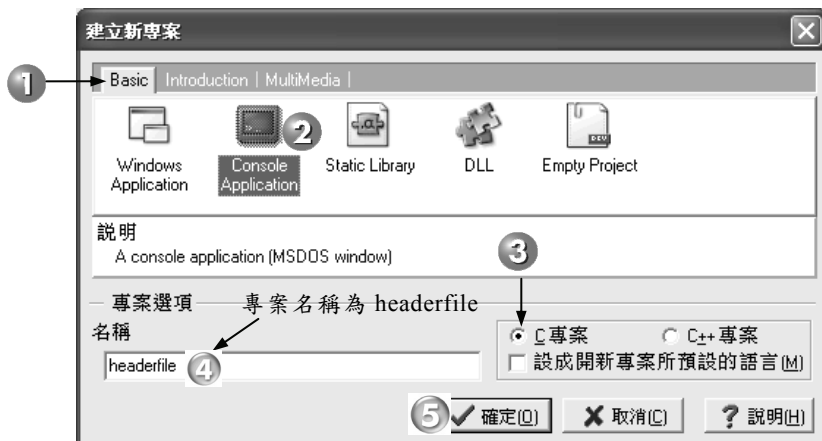



### 【分析】

**Step1** 以 Windows XP 作業系統的環境為主，執行 [開始/所有程式(P)/Bloodshed Dev-C++/Dev-C++]，此時即進入 Dev-C++ 整合開發環境內。接著請按  鈕，將「每日提示」視窗關閉。


**Step2** 請依下圖步驟指示，新增 Console Application 類型的 C 專案。

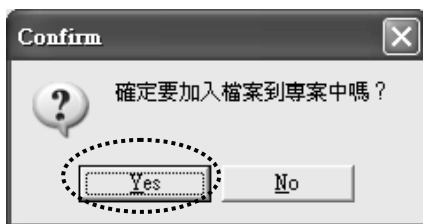
- ① 請執行功能表的 [檔案(F)/開新檔案(N)/專案(P)...] 指令開啟「建立新專案」的對話方塊。
- ② 在「建立新專案」對話方塊中切換到「Basic」標籤頁，然後選擇要新增的專案的類型為「Console Application」，再選取「C 專案」，專案名稱設為「headerfile」，再按  鈕。



- ① 接著出現「Create new project」對話方塊，請將 headerfile.dev 專案儲存在「C:\devC\ch09\headerfile」資料夾下，最後再按  鈕。
- ② 新增專案後，預設會有一個未存檔的 main.c 程式。請先執行功能表的【檔案(F)/關閉(C)】指令將 main.c 程式檔關閉。

**Step3** 依下圖步驟在 headerfile.dev 專案中新增 my.h 標頭檔，最後將 my.h 儲存在「C:\devC\ch09\headerfile」資料夾下。

- ① 請執行功能表的【檔案(F)/開新檔案(N)/原始碼(S)】指令新增檔案。
- ② 接著出現下面對話方塊詢問是否要將新增檔案加入到目前的專案中，請按  鈕。



- ③ 在新增的檔案中撰寫如下程式碼(my.h 標頭檔)。

【程式碼】 FileName:my.h

```
01 #define ZERO 0
02 #define ONE 1
03 #define TWO 2
04 #define THREE 3
05 #define EQU ==
06 #define GREATER >
07 #define LESS <
08 #define MAX2(x,y) ((x)>(y) ? (x):(y))
09 #define MAX3(x,y,z) MAX2(x,MAX2(y,z))
10 #define MAX4(w,x,y,z) MAX2(MAX2(w,x),MAX2(y,z))
```

- ④ 執行功能表的 [檔案(F)/儲存(S)] 指令開啟「儲存檔案」對話方塊，請將上述新增的檔案命名為「my.h」，並將該檔儲存在「C:\devC\ch09\headerfile」資料夾下。

**Step4** 重複上述 Step3，在 headerfile.dev 專案中再新增 headerfile.c。然後在 headerfile.c 撰寫如下程式碼，完成之後可執行功能表的 [執行(Z)/編譯並執行(O)] 指令觀看程式的執行結果。

【程式碼】 FileName:headerfile.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include "my.h"
04 int main(int argc, char *argv[])
05 {
06     int a=10,b=8,c=30,d=15;
07     int ch;
08     printf("\n ===== Choose Maxinum =====");
09     printf("\n    1. Compare two numbers ");
10     printf("\n    2. Compare three numbers ");
11     printf("\n    3. Compare four numbers");
12     printf("\n    0. Quit :");
13     printf("\n =====");
14     printf("\n    Please Choose Number [0-3] : ");
15     scanf("%d",&ch);
16     if (ch EQU ZERO)
17     {
18         exit(0);
19     }
20     else
21     {
22         if (ch > 3 || ch < 1)
23         {
24             printf(" Error !!    Please Keyin 1-3 ");
25         }
26     }
27     else
28     {
```



```

29         switch (ch)
30         {
31             case ONE:
32                 printf(" The MAX(%d,%d) is : %d \n",
33                     a, b, MAX2(a,b));
34                 break;
35             case TWO:
36                 printf(" The MAX(%d,%d,%d) is : %d \n",
37                     a,b,c, MAX3(a,b,c));
38                 break;
39             case THREE:
40                 printf(" The MAX(%d,%d,%d,%d) is : %d \n",
41                     a,b,c,d, MAX4(a,b,c,d));
42                 break;
43         }
44     }
45 }
46     system("PAUSE");
47     return 0;
48 }

```



### 【說明】

1. 第 3 行：含入 my.h 標頭檔，因為我們將 my.h 放在與 headerfile.c 相同路徑下，所以含入 my.h 不用指定完整的真實路徑。



### 【執行結果】

```

C:\dev\ch09\headerfile\headerfile.exe
===== Choose Maximum =====
1. Compare two numbers
2. Compare three numbers
3. Compare four numbers
0. Quit :
=====
Please Choose Number [0-3] : 2
The MAX(10,8,30) is : 30
請按任意鍵繼續 . . .

```



為節省篇幅，關於標頭檔與多個 C 程式檔的新增與建置，在後面章節我們將省略說明。



## 9.5 條件式編譯指引

### 9.5.1 如何控制程式的編譯流程

前面章節所介紹的原始程式，在程式執行前會先經過前置處理，接著編譯器便無條件照單全收將所有敘述編譯成機器碼。本節將介紹條件編譯指引，它允許此種指引加在程式中任何地方，編譯器碰到此類的指引會對程式中的敘述做選擇性的編譯。

C 語言提供 `#if`、`#elif`、`#else` 及 `#endif` 來控制「程式碼敘述」及「前置處理器」的編譯流程，也就是說可依條件來選擇要編譯的敘述區段，此種控制稱為「條件式編譯」(conditional compilation)，其使用方式類似 `if...else if...else` 選擇結構。其語法如下：

```
#if 條件運算式 1      /* 相當於 if */
    敘述 1             /* 當條件運算式 1 成立時，編譯敘述 1 */
#elif 條件運算式 2    /* 相當於 else if */
    敘述 2             /* 當條件運算式 2 成立時，編譯敘述 2 */
#elif 條件運算式 3    /* 相當於 else if */
    敘述 3             /* 當條件運算式 3 成立時，編譯敘述 3 */
    ...
#elif 條件運算式 N    /* 當條件運算式 N 成立時，編譯敘述 N */
    敘述 N
#else                 /* 相當於 else */
    敘述 N+1          /* 當上述條件運算式皆不立時會編譯敘述 N+1 */
#endif                /* 結束條件式編譯 */
```

上述條件式編譯的 `#elif` 和 `else if` 一樣可以省略或同時使用多個；而 `#endif` 不可以省略，它是用來結束 `#if` 敘述。條件式編譯和 `if...else if...else` 選擇結構的不同處是，條件式編譯可以根據條件運算式是否成立來選擇所要編譯哪一個程式敘述或前置處理器；而選擇結構是根據條件運算式是否成立來選擇要執行編譯完成的程式敘述。在程式中使用 `#undef` 條件編譯指引，便可取消原先使用 `#define` 定義的巨集。

範例	cond.c
本範例我們首先定義巨集 LEVEL 來表示會員的等級，接著使用 <code>#if</code> 、 <code>#elif</code> 、 <code>#else</code> 、 <code>#endif</code> 選擇所要編譯巨集 ROOM 的房間類型。	



#### 【程式碼】FileName:cond.c

```

01 #include <stdio.h>
02 #include <stdlib.h>
03 #define LEVEL 2  /* 只要更改數字，編譯器會依條件選擇編譯 */
04 #if LEVEL==1
05     #define ROOM "總統套房"
06 #elif LEVEL==2
07     #define ROOM "蜜月套房"
08 #elif LEVAL==3
09     #define ROOM "尊爵雙人房"
10 #else
11     #define ROOM "山水/風情標準房"
12 #endif

```

```
13
14 int main(int argc, char *argv[])
15 {
16     printf(" ===歡迎光臨-華福酒店===\n\n");
17     printf(" 您的會員等級是 %d\n", LEVEL);
18     printf(" 可以住的房間是 %s\n", ROOM);
19     printf("\n");
20     system("PAUSE");
21     return 0;
22 }
```



### 【說明】

1. 第 3 行：設定巨集 LEVEL 等於 2。
2. 第 4-5 行：假若 LEVEL 等於 1，則編譯第 5 行，並設定巨集 ROOM 的替代內容為「總統套房」字串。
3. 第 6-9 行：方式同第 4-5 行。
4. 第 10-11 行：若 LEVEL 不等於 1, 2, 3，則會編譯第 11 行，設定巨集 ROOM 的替代內容為「山水/風情標準房」字串。
5. 第 12 行：結束 #if。

## 9.5.2 如何判斷巨集是否定義

C 語言還可以使用 #ifdef、#ifndef、#else、#endif 來判斷巨集是否已經定義完成。其中#ifdef 用來判斷巨集是否已經定義；而#ifndef 用來判斷巨集是否還未定義。其語法如下。

語法 1：若巨集 A 定義完成則編譯  
敘述 1，否則編譯敘述 2

```
#ifdef 巨集 A
    敘述 1
#else
    敘述 2
#endif
```

語法 2：若巨集 A 還未定義則編譯  
敘述 1，否則編譯敘述 2

```
#ifndef 巨集 A
    敘述 1
#else
    敘述 2
#endif
```

譬如下面 `define1.c` 範例中第 3 行有定義 `IBM_NB` 巨集，由於第 11 行檢查 `IBM_NB` 巨集是否有定義，結果滿足條件，因此會編譯第 12 行，跳過第 14 行。接著第 16 行使用 `#undef` 取消 `IBM_NB` 巨集。第 18 行檢查 `IBM_NB` 是否未定義，由於條件成立編譯第 19 行，跳過第 21 行。

【程式碼】 `FileName:define1.c`

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #define IBM_NB 32000
04 #define ACER_NB 30000
05 #define MSG1 " \n IBM NoteBook's Price : %d "
06 #define MSG2 " \n IBM_NB 未定義 "
07 #define MSG3 " \n ACER NoteBook's Price : %d "
08
09 int main(int argc, char *argv[])
10 {
11     #ifdef IBM_NB                /* 是否有定義 IBM_NB 巨集，滿足條件 */
12         printf(MSG1, IBM_NB);    /* 編譯此行 */
13     #else
14         printf(MSG2, ACER_NB);   /* 跳過此行 */
15     #endif
16     #undef IBM_NB                /* 取消第 3 行 IBM_NB 巨集定義 */
17
18     #ifndef IBM_NB               /* 是否未定義 IBM_NB 巨集，滿足條件 */
19         printf(MSG2);            /* 編譯此行 */
20     #else
21         printf(MSG3, ACER_NB);   /* 跳過此行 */
22     #endif
23     printf("\n\n");
24     system("PAUSE");
25     return 0;
26 }
```



【執行結果】

```
C:\devC\ch09\define1\define1.exe

IBM NoteBook's Price : 32000
IBM_NB未定義

請按任意鍵繼續 . . .
```

#undef 可用來取消巨集的定義，其語法如下：

語法：#undef 巨集名稱



## 9.6 實例-書籍介紹

範例

book.dws

試使用#if、#elif、#else、#endif 條件式編譯指引，根據主函式中所定義的 ID 巨集 ID 值來選擇要含入 flash.h、java.h、office.h、vb.h 中哪個書籍資訊的標頭檔。

```
C:\devC\ch09\book\book.exe

====經典系列書籍====

書名：Visual Basic 2008 程式設計經典
等級：初階-中階
定價：580
種類：程式設計
請按任意鍵繼續 . . .
```

**【提示】**

- ① 本範例分別將四本書的相關資訊存入 flash.h、java.h、office.h、vb.h 標頭檔，這些標頭裡面分別定義 BNAME、LEVEL、PRICE、SERIES 巨集名稱來表示書籍的書名、等級、單價、種類。
- ② 在 book.c 主程式中含入 selectbook.h 標頭檔，然後在 selectbook.h 中使用 #if、#elif、#else、#endif 指令根據 main() 主函式內巨集 ID 的設定值來選擇要到底要含入 flash.h、java.h、office.h、vb.h 哪個標頭檔。

**【標頭檔】** FileName:flash.h

```
01 #define BNAME "Flash CS3 程式設計經典"
02 #define LEVEL "初-中階"
03 #define PRICE 500
04 #define SERIES "多媒體網頁程式設計"
```

**【標頭檔】** FileName:java.h

```
01 #define BNAME "Java SE 6 程式設計經典"
02 #define LEVEL "初-進階"
03 #define PRICE 550
04 #define SERIES "程式設計"
```

**【標頭檔】** FileName:office.h

```
01 #define BNAME "Office XP 實戰"
02 #define LEVEL "初階"
03 #define PRICE 480
04 #define SERIES "辦公室應用"
```

**【標頭檔】** FileName:vb.h

```
01 #define BNAME "Visual Basic 2008 程式設計經典"
02 #define LEVEL "初階-中階"
03 #define PRICE 580
04 #define SERIES "程式設計"
```

### 【標頭檔】 FileName:selectbook.h

```
01 #if ID==1                /* 若 ID 等於 1 則含入 flash.h */
02     #include "flash.h"
03 #elif ID==2              /* 若 ID 等於 2 則含入 java.h */
04     #include "java.h"
05 #elif ID==3              /* 若 ID 等於 3 則含入 office.h */
06     #include "office.h"
07 #else                    /* 若 ID 不等於 1~3，則含入 vb.h */
08     #include "vb.h"
09 #endif
```

### 【程式碼】 FileName:book.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #define ID 4              /* 設定巨集 ID 為 4 此行為彈性*/
04 #include "selectbook.h"  /* 含入 selectbook.h 標頭檔 */
05
06 int main(int argc, char *argv[])
07 {
08     printf(" ====經典系列書籍==== \n\n");
09     printf(" 書名：%s \n", BNAME);
10     printf(" 等級：%s \n", LEVEL);
11     printf(" 定價：%d \n", PRICE);
12     printf(" 種類：%s \n", SERIES);
13     system("PAUSE");
14     return 0;
15 }
```

## 9.7 習題

### 一. 選擇題

( ) 1. 定義巨集必須使用下列哪一個指令？

- (1) #include      (2) #define      (3) #if      (4) #elif



- ( ) 2. 定義巨集的替代內容，請問可以使用幾行敘述？  
(1) 1                      (2) 2                      (3) 3                      (4) 無限
- ( ) 3. 要含入標頭檔必須使用下列哪一個指令？  
(1) #include              (2) #define              (3) #if                      (4) #elif
- ( ) 4. 巨集可以定義在那些位置？  
(1) 標頭檔函式定義中間              (2) 沒有限制  
(3) C 程式檔最後面                      (4) C 程式檔最前面
- ( ) 5. #define CUBE(x) x\*x\*x  
.....  
printf(“%d”, CUBE(2));  
上述程式會印出多少？  
(1) 2                      (2) 4                      (3) 6                      (4) 8
- ( ) 6. #define CUBE(x) x\*x\*x  
.....  
int n=3;  
printf(“%d”, CUBE(++n));  
上述程式會印出多少？  
(1) 27                      (2) 60                      (3) 120                      (4) 以上皆非
- ( ) 7. 以下何者正確？(複選)  
(1) 巨集定義最後必須有分號做結尾  
(2) 巨集名稱與替代內容或運算式必須使用空白來區隔  
(3) 巨集的執行速度快，所以可以完全使用巨集來代替函式  
(4) 巨集定義必須寫在一行
- ( ) 8. 判斷巨集是否有定義可以使用？  
(1) #if                      (2) #ifdef                      (3) #ifndef                      (4) #else
- ( ) 9. 下列那些不是條件式編譯所使用的敘述？  
(1) #if                      (2) if                      (3) #elif                      (4) #endif
- ( ) 10. 若巨集要傳入引數，最好使用什麼符號將引數括住？  
(1) ()                      (2) []                      (3) \*\*                      (4) /\* \*/

- ( ) 11. 編譯器主要做那些工作？(複選)
- (1) 前置處理    (2) 編譯    (3) 組譯    (4) 解碼
- ( ) 12. 當編譯器做組譯處理時會產生？
- (1) \*.c    (2) \*.exe    (3) \*.obj    (4) \*.html
- ( ) 13. 前置處理指引允許放在？(複選)
- (1) 任可位置    (2) main 函式之前    (3) 自訂函式之前
- ( ) 14. 巨集之後必須？
- (1) 加上;    (2) 加上()    (3) 加上[]    (4) 不可加上任可字元
- ( ) 15. 下面哪些函式不屬於 stdio.h 標頭檔內所宣告的函式？
- (1) getc    (2) fputs    (3) isdigit    (4) fread

## 二. 簡答題

1. 試簡述巨集與函式的差異？
2. 試簡述 #if、#elif、#else、#endif 與選擇結構的差異。
3. C 語言在編譯過程中，主要做了那些工作。

## 三. 程式設計

1. 定義攝氏轉華氏以及華氏轉攝氏的計算溫度巨集。公式如下：  
攝氏轉華氏公式： $\text{華氏溫度} * 9/5 + 32$   
華氏轉攝氏公式： $(\text{攝氏溫度} - 32) * 5/9$
2. 撰寫巨集 SEARCH\_MIN3 可找出三個數值中的最小值；撰寫巨集 SEARCH\_MAX3 可找出三個數值中的最小值。