A

C 語言的基礎

A-1 C 語言的程式架構

C 程式的程式架構是由「#」字元開頭的指令、函數、全域變數宣告和 main()主程式所組成。範例程式 ChA-1.c ，如下所示：

```
01: /* 程式範例: ChA-1.c */
02: #include <stdio.h>
03: #include <stdlib.h>
04: int total;                      /* 全域變數的宣告 */
05: /* 函數: 傳回1加到n */
06: int one2N(int n) {
07:     int i, total = 0;           /* 區域變數的宣告 */
08:     for ( i = 1; i <= n; i++ ) total+=i; /* 迴圈敘述 */
09:     return total;
10: }
11: /* 主程式 */
12: int main() {
13:     total = one2N(10);          /* 呼叫函數計算1加到10 */
14:     printf("從1到10的總和: %d\n", total);
15:     system("PAUSE");
16:     return 0;
17: }
```

上述 C 程式架構的說明，如下所示：

- **程式註解**：第 1 、 5 和 11 列是程式註解，註解可以出現在程式檔案的任何地方，在此的註解文字是說明 C 程式檔案名稱和函數用途。
- **標頭檔**：第 2~3 列是以「#」字元開頭的含括指令 include 含括的標題檔，標頭檔是 C 語言函式庫的函式宣告，如果在程式中需要使用這些函式庫，就需要含括指定的標頭檔，這是由「C 的前置處理器」（The C PreProcessor）來處理。

- **全域變數**：第 4 列是全域變數 total 的宣告。
- **函數**：第 6~10 列是函數，宣告同名的區域變數 total 來計算 1 加到參數值，換句話說，在此函數存取的 total 變數是區域變數，並不是全域變數。
- **主程式**：第 12~17 列的 main()函數是 C 程式的主程式，這是 C 語言應用程式執行時的進入點，也就是說執行 C 程式是從此函數的程式碼開始。

A-2 C 語言的變數與資料型態

變數在 C 程式的目的是用來儲存程式執行中的暫存資料，例如：運算結果。資料型態可以指定變數儲存的資料種類，例如：整數、浮點數或字元等。

A-2-1 資料型態與變數宣告

在 C 語言提供多種基本資料型態，不過，不同電腦系統和 C 語言編譯程式的資料型態範圍可能不同，以本書 Dev-C++ 的 GCC 編譯程式為例，其基本資料型態的範圍，如下表所示：

資料型態	說明	位元	範圍
unsigned char	無符號字元	8	0 ~ 255
unsigned short	無符號短整數	16	0 ~ 65535
unsigned int	無符號整數	32	0 ~ 4294967295
unsigned long	無符號長整數	32	0 ~ 4294967295
signed char	字元	8	-128 ~ 127
signed short	短整數	16	-32768 ~ 32767

資料型態	說明	位元	範圍
signed int	整數	32	-2147483648 ~ 2147483647
signed long	長整數	32	-2147483648 ~ 2147483647
float	單精度浮點數(6)	32	1.175494e-038 ~ 3.402823e+038
double	雙精度浮點數(15)	64	2.225074e-308 ~ 1.797693e+308

C 語言的變數在使用前一定需要宣告使用的資料型態。例如：整數變數宣告的範例，如下所示：

```
int balance;
```

上述程式碼宣告一個整數變數，資料型態為整數 int ，名稱為 balance 。如果需要同時宣告多個變數，請使用「,」逗號分隔，如下所示：

```
int i, j, balance;
```

A-2-2 常數宣告

「常數」（Symbolic Constants 或 Named Constants）是指變數在設定初始值後，就不會變更其值。 C 語言支援常數修飾子 const 和使用前置處理器（Preprocessor）的 #define 指令來定義常數，如下所示：

```
#define PI 3.1415926
```

上述程式碼宣告圓周率常數 PI ，其中 PI 是一個識別字。另一種方法的常數宣告是在宣告變數前使用 const 常數修飾子，如下所示：

```
const double e = 2.71828182845;
```

A-2-3 指定敘述

「指定敘述」（Assignment Statements）是在程式執行中存取變數值，如果宣告變數時沒有指定初值，也可以使用指定敘述即「=」等號來指定或更改變數值。一些指定敘述的範例，如下所示：

```
int size, size1, size2;
size = 35;
size1 = 57;
```

A-3 C 語言的基本輸出入與運算子

程式的輸入與輸出是指如何取得使用者鍵盤或滑鼠輸入的資料，在執行後以指定格式在螢幕上顯示執行結果。運算子就是在建立 C 語言的運算式，以便執行所需的運算。

A-3-1 基本輸出入

C 語言的標準輸入與輸出是文字輸入與輸出模式，循序由一行一行所組成的文字串流（Text Stream），每一行的最後是使用新行字元（即 '\n' 字元）來結束。

格式化資料輸出

函數 printf()使用格式字串輸出指定資料型態的變數資料，內含「%」符號開始的格式字元，如下所示：

```
printf("a(d) = %d\n", a);
printf("b(d) = %d c(d) = %d\n", b, c);
```

上述程式碼使用格式字元 %d 輸出整數變數 a 、 b 和 c ，格式字元 %f 輸出浮點數， %c 是字元， %s 是字串。

格式化資料輸入

scanf()函數是以格式字元來判斷輸入的資料屬於哪一種資料型態。例如：整數 int 的格式字元是 %d ，如下所示：

```
scanf("%d", &cels);
```

上述第 1 個參數是格式字元 %d ，表示輸入的資料是整數，第 2 個參數使用「&」取址運算子取得變數的記憶體位址，變數 cels 儲存的值就是使用者輸入的整數值。

A-3-2 C 語言的運算子

在 C 語言指定敘述的右邊是一個「**運算式**」（Expressions），它是由「**運算子**」（Operators）和「**運算元**」（Operands）所組成。一些運算式的範例，如下所示：

```
a + b - 1
a >= b
a > b && a > 1
```

上述運算式變數 a 、 b 和數值 1 都屬於運算元，「+」、「-」、「>=」、「>」和「&&」為運算子， C 語言的運算子是使用 1 到 2 個字元所組成的符號。

C 語言的運算子分成很多種，在同一個運算式如果使用多種運算子，為了讓運算式能夠得到相同的運算結果，運算式是以運算子預設的優先順序進行運算， C 語言各運算子的優先順序（愈上面愈優先），如下表所示：

運算子	說明
()	括號
!、-、++、--	邏輯運算子 NOT 、算數運算子負號、遞增和遞減
*、/、%	算術運算子的乘、除法和餘數
+、-	算術運算子加和減法
<<、>>	位元運算子左移、右移
>、>=、<、<=	關係運算子大於、大於等於、小於和小於等於

運算子	說明
==、!=	關係運算子等於和不等於
&	位元運算子 AND
^	位元運算子 XOR
|	位元運算子 OR
&&	邏輯運算子 AND
||	邏輯運算子 OR
?:	條件控制運算子
=、op=	指定運算子

A-4 C 語言的流程控制指令

流程控制可以配合運算式的條件來執行不同程式區塊，或重複執行指定區塊的程式碼，流程控制指令主要分為兩類，如下所示：

- **條件控制**：條件控制是一個選擇題，分為單選或多選一，依照運算結果來決定執行哪一個程式區塊的程式碼。
- **迴圈控制**：迴圈控制是重複執行程式區塊的程式碼，擁有結束條件可以結束迴圈的執行。

A-4-1 條件控制指令

C 語言的條件控制敘述可以分為單選（if）、二選一（if/else）或多選一（switch）幾種方式，此外還提供條件運算子（?:）可以建立單行程式碼的條件控制。

if 是否選條件敘述

if 條件敘述是一種是否執行的單選題，只是決定是否執行程式區塊內的程式碼。例如：如果 score 大於等於 60 ，就執行程式區塊的 if 條件敘述，如下所示：

```
if ( score >= 60 ) {
    printf("成績及格......\n");
    printf("成績為%d\n", score);
}
```

if/else 二選一條件敘述

單純 if 條件只能選擇執行或不執行程式區塊的單一選擇，更進一步，如果屬於排它情況的兩個程式區塊，就可以使用 if/else 二選一條件敘述。例如：一個 if/else 條件敘述的範例，如下所示：

```
if ( score >= 60 )
    printf("成績及格%d\n", score);
else
    printf("成績不及格%d\n", score);
```

上述程式碼因為成績有排它性， 60 分以上為及格分數，以下為不及格，所以只會執行其中之一的程式碼。

?:條件運算式

C 語言提供「條件運算式」（Conditional Expressions），可以使用條件運算子?:在指定敘述以條件來指定變數值，其功能如同 if/else 條件，使用「?」符號代替 if ，「:」符號代替 else 。例如：一個條件敘述運算子的範例，如下所示：

```
hour = (hour >= 12) ? hour-12 : hour;
```

上述程式碼使用條件敘述運算子指定變數 hour 的值，如果條件不等於 0 ， hour 變數值為 hour-12；等於 0 是 hour 。

if/else if 多選一條件敘述

C 語言提供兩種方法建立多選一條件敘述。第一種方法是 if/else 條件的擴充，重複使用 if/else 條件建立多選一的條件敘述，如下所示：

```
if ( score >= 80 )
    printf("學生成績A: %d\n", score);
else
    if ( score >= 70 )
       printf("學生成績B: %d\n", score);
    else
       printf("學生成績C: %d\n", score);
```

上述程式碼使用 if/else 條件，每次判斷一個條件，如果等於 0 就重複使用 if/else 條件再進行下一次的判斷。

switch 多選一條件敘述

第二種方法是 switch 多條件程式敘述，只需依照符合條件，就可以執行不同程式區塊的程式碼，例如：switch 條件敘述範例，如下所示：

```
switch (grade) {
   case 'A':
      printf("學生成績A超過80分\n");
      break;
   case 'B':
      printf("學生成績B超過70分");
      printf(", 小於80分\n");
      break;
   case 'C':
      printf("學生成績C超過60分");
```

```
        printf(", 小於70分\n");
        break;
    default:
        printf("學生成績D小於60分\n");
        break;
}
```

上述 switch 條件只有一個關係運算式，每一個 case 條件的比較相當於是一個「==」運算子，如果符合，就執行 break 指令之前的程式碼，每一個條件需要使用 break 指令跳出條件敘述。

最後 default 指令並非必要指令，這是一個例外條件，如果 case 條件都沒有符合，就執行 default 程式區塊。

A-4-2 迴圈控制指令

C 語言支援計數、前測和後測等多種迴圈控制敘述，能夠設計出各種重複執行多次程式區塊的程式碼。

for 計數迴圈

for 迴圈稱為「計數迴圈」（Counting Loop），因為迴圈預設擁有計數器，可以從一個值執行到另一個範圍值。例如：計算 1 加到 5 的總和，每次遞增 1 ，如下所示：

```
for ( i = 1; i <= 5; i++ ) {
    printf("數字: %d ", i);
    total += i;
}
```

相反的情況，如果是從 5 加到 1 ，計數器使用 i-- 表示每次遞減 1 ，如下所示：

```
for ( i = 5; i >= 1; i-- ) {
  ..........
}
```

前測式 while 迴圈敘述

while 迴圈敘述不同於 for 迴圈，需要在程式區塊自己處理計數器的增減， while 迴圈是在程式區塊的開頭檢查結束條件，如果條件不等於 0 才進入迴圈執行。例如：改為使用 while 迴圈計算 1 加到 10 的總和，如下所示：

```
while ( i <= 10 ) {
    total += i;
    i++;
}
```

上述 while 迴圈計算從 1 加到 10 的總和，變數 i 是計數器變數，如果符合 i <= 10 條件，就進入迴圈執行程式區塊，迴圈的結束條件為 i > 10 。

後測式 do/while 迴圈敘述

do/while 和 while 迴圈敘述的差異是在迴圈結尾的檢查條件， do/while 迴圈是先執行程式區塊的程式碼後才測試條件，所以 do/while 迴圈的程式區塊至少會執行一次。例如：改為使用 do/while 迴圈計算 1 加到 10 ，如下所示：

```
do {
    total += i;
    i++;
} while ( i <= 10 );
```

上述迴圈的第 1 次執行需要執行到迴圈結尾，才會檢查 while 條件是否不等於 0 ，如果不等於 0 就繼續執行迴圈，可以計算從 1 加到 10 的總和，迴圈的結束條件為 i > 10 。

巢狀迴圈

巢狀迴圈是在迴圈內擁有其他迴圈，例如：在 for 迴圈內擁有 for 、while 和 do/while 迴圈，同樣的， while 迴圈內也可以擁有 for 、 while 和 do/while 迴圈。

在 C 語言的巢狀迴圈可以擁有二或二層以上。例如：在 for 迴圈內擁有 while 迴圈，如下所示：

```
for ( i = 1; i <= 9; i++ ) {
    ......
    j = 1;
    while ( j <= 9 ) {
       .....
       j++;
    }
}
```

A-5 C 語言的函數

C 語言的模組單位是「函數」（Functions），函數是一個獨立的程式單元，我們可以使用函數將大工作分割成一個個小型工作，也可以重複使用以前已經建立的函數或直接呼叫 C 語言標準函式庫的函數。

A-5-1 建立 C 語言的函數

C 語言的函數是由函數名稱和程式區塊所組成，參數列是函數的使用介面，如果傳回值型態為 void 表示函數沒有傳回值，如果省略，預設傳回值型態為 int（例如：主程式 main()預設的傳回值是 int）。例如：沒有傳回值的函數，如下所示：

```
void writeString(int rows) {
   int i;
    for ( i = 1; i <= rows; i++ )
       printf("歡迎使用C/C++!\n");
}
```

上述函數傳回值的資料型態為 void ，表示沒有傳回值，在「{」和「}」括號內是函數的程式區塊，函數名稱為 writeString ，在括號內定義傳入的參數列。

在 C 語言的程式碼呼叫函數需要使用函數名稱，例如：函數 writeString()擁有參數列，在呼叫時需要加上參數列（或稱為引數），如下所示：

```
writeString(5);
```

A-5-2 函數的傳回值

如果函數的傳回值型態不是 void ，而是指定資料型態 int 或 char 等，就表示函數擁有傳回值，在函數的程式區塊是使用 return 關鍵字傳回函數值。例如：擁有傳回值的函數範例，如下所示：

```
int n2N(int start, int end) {
    int i;
    int total = 0;
    for ( i = start; i <= end; i++ )
        total += i;
    return total;
}
```

上述 n2N()函數的傳回值型態為 int ，可以計算傳入參數 start 到 end 的總和，使用 return 關鍵字傳回函數執行結果的 total 變數值。如果函數擁有傳回值，在呼叫時可以使用指定敘述來取得函數的傳回值，如下所示：

```
total = n2N(start, end);
```

上述程式碼的 total 變數可以取得函數的傳回值。