SoftTip 尖端電腦NO.1099

TURBO

C專業程式設計師訓練教材（進階）

- TURBO C
- LATTICE
- C86
- MICROSOFT.

磐圃電腦軟體研究開發部門　編譯

全華科技圖書股份有限公司　總經銷

C 專業程式設計師訓練教材

序

本教材針對已具備電腦及其他程式語言如：BASIC、PASCAL基礎，而想要成為 C專業程式設計師，或欲擁有此能力的讀者而編。

本教材雖分成初階與進階兩冊，但章節的安排主要是按學習者功能需要劃分而非絕對依照內容的深淺難易。在各章的中心主題之下，我們採取下列的主要方針：

1. 務必先讓讀者學會簡單的程式如何撰寫，再介紹深入的詳細內容與特性。

2. 在討論 C的功能特性的同時，一併探討程式設計的技巧、風格、並提倡結構化的設計規劃方式。

3. 著重於如何將程式設計得更強大、更有效、更精簡、更易維護。

我們認為：每一個具專業水準的 C程式，都是 "軟體藝術" 的高度結晶。本教材的主要精神，亦即要訓練出 "發揮軟體藝術" 的 C程式設計人材。

閱讀本教材來學習 C，我們希望您學會 C，並且成為一個講究 "軟體藝術" 的專業設計師。

C 專業程式設計師訓練教材/進階

目　錄

第一章　函數與模組

□ 當作程序使用的函數.....1-3
□ 函數的宣告及定義.....1-6
□ 參數與傳回值.....1-12
□ 函數的呼叫.....1-20
□ 指向函數的指標.....1-26
□ 儲存體配置與遞迴.....1-31
□ 程式模組的設計.....1-33
□ C 的程式庫函數.....1-40
□ 習題.....1-51

第二章　運算子

□ 資料計算表示法.....2-3
□ 運算子組成式與運算元.....2-4
□ 優先權與函數參考.....2-6
□ 選擇運算與位址運算.....2-9
□ 算術運算.....2-11
□ 邏輯運算與比較運算.....2-16
□ 位元運算.....2-21
□ 指定運算式.....2-25
□ 資料形態與循序運算.....2-29
□ 運算的種類.....2-35
□ 習題.....2-43

第三章　計算表示式

□ 資料形態的自動轉換.....3-3

□ 控制的資料形態轉換.....3-7
□ 運算子的優先順序及組合.....3-10
□ 表示式的撰寫及解釋.....3-18
□ 使用敘述中的表示式.....3-27
□ 習題.....3-32

第四章

□ 結構的與非結構的控制.....4-3
□ 反覆敘述：while、for、do.....4-4
□ 岔斷敘述：break及 continue.....4-8
□ 條件敘述：if...else及 switch.....4-11
□ 敘述標示及 goto.....4-20
□ 使用程式敘述中的控制.....4-24
□ 習題.....4-29

第五章 C 應用系統的特色

□ float 和 double 資料形態.....5-3
□ float 和 double 的轉換.....5-6
□ 資料元素的大小與範圍.....5-8
□ 常數值的設定.....5-10
□ 聯集與位元欄位.....5-13
□ 列舉的資料形態.....5-15
□ 資料形態指定字 void.....5-18
□ 結構與聯集的設定.....5-19
□ 函數的參數與值.....5-20
□ 識別名稱種類的規定.....5-23
□ 習題.....5-24

第六章

□ 前置處理器的操作.....6-3
□ 符號代換.....6-4
□ 條件編譯.....6-14

□ 檔案包含過程及列的編號.....6-16
□ 組織軟體的發展.....6-18

第七章 C 函數庫

□ 函數的規定與形態.....7-3
□ 字串處理函數.....7-5
□ strlen.....7-6
□ strcmp.....7-7
□ strncpy.....7-8
□ strncat.....7-9
□ strcmp.....7-10
□ strcpy.....7-11
□ strcat.....7-12
□ 儲存體配置函數.....7-13
□ malloc.....7-14
□ calloc.....7-15
□ free.....7-15
□ 資料轉換函數.....7-16
□ itoa.....7-17
□ atoi.....7-17
□ ftoa.....7-18
□ atof.....7-19
□ 分類函數.....7-20
□ isupper.....7-20
□ islower.....7-21
□ isalpha.....7-21
□ isdigit.....7-22
□ isalnum.....7-23
□ isprint.....7-23
□ isspace.....7-24
□ tolower.....7-25
□ 數學函數.....7-26
□ toupper.....7-26
□ frexp.....7-27
□ ldexp.....7-28

□ abs.....7-29
□ ceil.....7-30
□ floor.....7-30
□ sqrt.....7-31
□ log.....7-32
□ exp.....7-33
□ pow.....7-34
□ sin.....7-35
□ cos.....7-36
□ tan.....7-36
□ asin.....7-37
□ acos.....7-38
□ atan.....7-39
□ 程式庫函數摘要.....7-40

第八章　輸入與輸出函數

□ 組織與規定.....8-3
□ 資料存取函數.....8-5
□ fopen.....8-9
□ fclose.....8-10
□ ftell.....8-11
□ fseek.....8-12
□ ferror.....8-13
□ 非格式化的輸入與輸出函數.....8-14
□ fgetc.....8-14
□ getchar.....8-15
□ fputc.....8-16
□ putchar.....8-17
□ fgets.....8-18
□ fputs.....8-19
□ fread.....8-20
□ fwrite.....8-21
□ 格式化的輸入與輸出函數.....8-23
□ fscanf.....8-24

□ scanf.....8-25
□ sscanf.....8-26
□ fprintf.....8-27
□ printf.....8-28
□ sprintf.....8-29
□ 輸入格式.....8-31
□ 輸出格式.....8-38
□ 無緩衝區的輸入與輸出函數.....8-45
□ 輸入/輸出函數摘要.....8-47

第九章　C 程式的測試與維護

□ 測試與維護的週期.....9-3
□ 測試的環境.....9-5
□ 缺點的防止.....9-8
□ 軟體的文件.....9-12
□ 軟體的程式庫.....9-15

第一章 函數與模組

大家準備！好戲就要開鑼了！

函數是 C程式的個別演員，通常在模組或更大的組合中合併使用，它是程式中最小的完整個體。本章所討論的主題包括 ：

- 當作程序 (procedures) 使用的函數
- 函數的宣告及定義
- 參數與傳回值
- 函數的呼叫
- 指向函數的指標
- 儲存體配置與遞迴
- 程式模組的設計
- C 的程式庫函數

□ 當作程序使用的函數

和大多數的程序化語言一樣，C 也採用 "單一控制路徑" 的概念。這種型態的語言所產生的程式，是由一個程序連接而成。程序具有下列特點 ：

- 必須由其他程序的外顯 (explicit) 動作來呼叫。
- 可以自呼叫程序接收資訊，也可以將資訊傳回給呼叫程序。
- 可以呼叫其他程序。
- 程式中所有的動作都由單一的程序控制執行。
- 任何時間裡，只能有一個程序控制程式的動作。
- 程序中的動作由" 可執行敘述 (executable statements)" 指揮執行，可執行敘述則依照某特定順序來操作。
- 任何時間裡，只有一個可執行敘述發生作用。
- 被呼叫的程序必須記錄其呼叫程序的相關資訊，並且可以將控制權傳回給該呼叫程序。

有些語言，像 Pascal、FORTRAN，及 PL/1，具有多種型態的程序。各種程序根據其是否可以修改被呼叫時所接收的資訊，以及是否可以傳回單一資訊值來區分。C 語言的程

序型態只有一種，即函數。C 的函數具有下列特點，以下各節將分別討論之：

- 每一個 C 函數可以有一個或數個傳遞資訊的變數，稱為參數(parameters)，其有效範圍為區域性的。
- 用以呼叫函數的敘述，藉著引數 (arguments)來指定參數 (如果有的話)的值。函數中的參數所接收的，為這些引數的拷貝。
- 函數跳離時，可以提供一個值給它的呼叫函數，稱為傳回值 (return value)。
- 函數的呼叫，或稱函數參考(function references)，可以當作 C敘述之表示式的組成部份。其值為被呼叫函數所傳回的值。
- 除了參數以外，函數可以宣告具有特定有效範圍,這些變數在函數被呼叫時，即可使用。
- C 的函數只能有一個進入點，但可以有一個以上的出口點。呼叫函數並不接收有關出口點的特別指示。
- C 的函數可以呼叫其本身，或被其所呼叫的函數叫用，此過程稱為遞迴 (recursion)。

函數之外，C 語言只允許宣告識別字、資料、及資料的組織。所有的程式操作及可執行敘述，都必須包含在函數中

。任何 C程式的執行，都由一個特定的函數開始。在大多數的 C環境中，此函數命名為 main。其它函數則在被呼叫時才執行。

圖 1.1說明 C函數，以及用以呼叫函數之敘述的形式。圖中 evaluate 函數因為出現在呼叫函數的敘述中，所以會被呼叫到。此函數有一參數 parm 。當 evaluate 接收控制權時 parm 的值即等於呼叫函數的引數值－也就是 arg最新的值。返回時，呼叫函數中的 value變數，將接收 evaluate 函數中 datum的最新值。

```
1
2           int value ;       /* result of evaluate        */
3
4           int arg ;         /* argument for evaluate     */
5
6                             /* declaration of evaluate   */
7
8           int evaluate() ;
9           . . .
10
11                            /* invoke evaluate function */
12
13          value = evaluate( arg ) ;
14          . . .
15
16                            /* define evaluate function */
17
18      int vealuate( parm )
19
20      int parm ;
```

(續下頁)

(承上頁)

```
21
22      {
23
24          int datum ;      /* variable in evaluate      */
25          . . .
26
27          return datum ;  /* return value of evaluate */
28
29      }
30
```

圖 1.1 ：函數範例及呼叫該函數的敘述

□ 函數的宣告及定義

C函數的宣告與變數的宣告非常相似，但是，函數的名稱所遵循的一些限制，則與資料名稱無關。在 C的敘述中，如果有一個名稱，後面緊跟著括弧(" (")， 則被視為函數。函數名稱的使用情形只有四種：

1 . 在宣告或定義該函數時

2 . 呼叫該函數時

3 . 取得指向函數的指標值時

4 . 在呼叫某函數時作為引數之用

第 3及第 4種情形，只有在已經定義或宣告的函數範圍之內，才可能發生。

函數宣告的方法，是指定其傳回值的形態，並在識別字後面加上一對括弧（"（）"）。

大多數的 C應用系統中，函數的參數及傳回值只可以是資料元素及指標。函數宣告的形式如下：

```
extern int evaluate ( ) ;
```

在此例中，evaluate 被宣告為函數名稱（該函數於別處定義），其傳回值為整數形態 int。

函數定義的啓始形式與函數宣告相同。此外，函數定義還具有下列的特點：

- 函數定義必須置於其它函數之外。
- 函數定義必須單獨存在；不可和其它宣告或定義合併在一起。
- 函數名稱後面的括號中，必須放置參數名稱，各名稱以逗號分隔，而函數被呼叫時，其引數出現的順序，必須遵照此參名稱的順序。
- 右括弧後面所接的，是各參數的宣告敘述，每一敘述以分號結尾。
- 宣告敘述後面為函數的主體，由宣告敘述及可執行敘述組成一個區塊。此區塊的前後必須加上大括號（{ }），即使只有一個敘述，也不可省略。

C 函數的定義，形式如下 ：

```
1
2          int evaluate ( parm )
3
4          int parm ;
5
6       ┌──{
7       |
8       |      int datum
9       |
10      |      . . .
11      |
12      |
13      |      return datum ;
14      |
15      └───}
16
17
```

此例中，evaluate 宣告為函數名稱，其傳回值為整數形態 int。此函數有一參數 parm，形態為 int。函數定義中另外還有一個 int形態的變數 datum用以儲存 return 敘述所傳回的值。構成函數的敘述區塊，其寫法與其它的敘述區塊相同；區塊後面不必加上分號。

當傳回值的宣告表示式為複合形式時，函數定義有時候也會變得比較複雜，例如 ：

```
1
2          int ( * ( * vector ( prm ) )[ ] ) ( )
3
4          int * prm ;
5
6       ┌──{
7       |
8       |    . . .
9       |
10      └──}
```

此例中，vector函數有一個參數 prm 為指向整數的指標。vector函數傳回指向某一指標陣列的指標，陣列中的各指標則指向傳回值為 int形態的函數。

函數宣告與函數定義中可以使用的儲存體種類指定字，只有 static 和 extern。 這兩個指定字只影響函數的使用範圍，不影響其記憶體配置。指定 static 時，表示函數的有效範圍為其所屬的模組。指定extern時，則為總體性的有效範圍。如果兩者都未指定，但是，函數名稱先前已宣告或定義過，則其有效範圍為先前所建立的有效範圍。在預設情形下，函數名稱具有總體性的有效範圍。在函數名稱已取得總體性的有效範圍之後再指定 static， 或是在函數名稱已取得模組性有效範圍之後再指定extern，都是錯誤的。

在函數定義中，不需要儲存體種類的指定字，也不需要指明資料形態。如果在定義函數時省略傳回值的資料形態，則函數的傳回值為 int形態。函數中不可以同時省略儲存體種類的指定字和資料形態 ;兩者中必須有一個存在。否則，在函數之外出現的名稱，後面加上左括弧時，可能會被解釋

為函數定義的開端，而在函數之內，這樣的敘述結構則會呼叫所指定的函數。

如果敘述中所呼叫的函數未曾宣告或定義過，則代表該函數具有總體性的有效範圍，而其傳回值為 int形態。假定 myzlplk為未宣告的函數名稱，而出現下列敘述時 ：

```
coccyx = myzlplk( gluteus, maximus ) ;
```

其結果除了呼叫 myzlplk之外，如同在目前區塊中的第一個可執行敘述之前,出現下列敘述 ：

```
extern int myzlplk( ) ;
```

任何函數宣告中的有效範圍及其傳回值的資料形態，包括隱含式的宣告，必須與後續的宣告及函數的定義一致。

在一般使用的 C語言中，資料和函數的識別字屬於同一等級。同樣的名稱只能代表函數或資料，不能同時代表兩者。一個識別字被宣告為資料或資料形態後，不可再用來呼叫函數。而被宣告為函數名稱的識別字，也不可再作其它用途。

函數名稱的有效範圍，始於其宣告或定義敘述之後。而定義的有效範圍，則延伸至其所屬之模組的結尾。如果宣告敘述出現在函數外面，則其有效範圍一直延伸到該模組的結尾。如果宣告是隱含式的，則其有效範圍雖然必須與同一模組中，同一函數的宣告或定義一致，但只限定在其所屬的區塊內。