第四章

基本的輸出與輸入

前言

- 電腦除了內部的資料處理之外，必須藉由輸出與輸入裝置與外界產生互動。
  - 輸出與輸入裝置包含很多類型，例如：檔案、螢幕、印表機、鍵盤等等。

- 在本章中，我們先介紹最常見的螢幕與鍵盤（即標準輸出入裝置：stdout與stdin），透過螢幕電腦可以將運算結果輸出，讓使用者看到運算結果；透過鍵盤的輸入，則可以取得使用者的輸入，使得程式根據輸入而產生變化。

- 在C語言中，對於螢幕與鍵盤的輸出入提供了不少的函式，其中最常用的則爲輸出函式printf( )及輸入函式scanf( )。本章將針對這兩個函式詳加介紹。

大綱

- 4.1 printf()-C的格式化輸出函式
  - 4.1.1 簡單的printf()使用方法
  - 4.1.2 printf()語法
  - 4.1.3 魔術『%』
  - 4.1.4 在printf()中使用跳脫字元
- 4.2 scanf()-C的格式化輸入函式
  - 4.2.1 scanf()語法
  - 4.2.2 魔術『%』
- 4.3 ANSI C所提供的其他輸出入函式
  - 4.3.1 getchar()與putchar()
  - 4.3.2 puts()與gets()
  - 4.3.3 fgets()與fputs()

大綱

- 4.4 非ANSI C所提供的輸出入函式
- 4.5 深入探討C語言的輸出入
  - 4.5.1　沒有I/O的C語言
  - 4.5.2　I/O的轉向
- 4.6 本章回顧

4.1 printf()-C的格式化輸出函式

- C語言最常見的輸出函式爲printf( )，
- 由<stdio.h>函式庫所提供。
- printf()除了可以輸出簡單的字串之外，還可以先將要輸出的資料做格式化之後，再輸出到螢幕上。
  - 在前面章節中，我們已經使用過printf()函式了，使用printf()最簡單的方法，就是將字串（用雙引號『"』括起來）直接當做printf( )函式的引數，螢幕即可輸出字串。

4.1.1 簡單的printf()使用方法

- 範例4-1：ch4_01.c

```
1  /*************************
2      檔名:ch4_01.c
3      功能:簡單的printf()
4   *************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  printf("您好.");
12  printf("歡迎學習C語言.");
13  /*  system("pause");  */
14 }
```

4.1.1 簡單的printf()使用方法

- 執行結果：

```
您好.歡迎學習C語言.
```

- 範例說明：
  - 由執行結果很容易會發現一個現象，輸出的兩個字串都列印在同一行。
  - 這是由於並未指定輸出換行字元的緣故。
  - 由於『換行』其實就是對游標的一種控制，算是一種跳脫字元，而非一個可見文字。在C語言中，則將換行控制字元指定爲『\n』，若printf( )看到字串中出現『\n』時，就會自動將螢幕游標移往下一行。

4.1.1 簡單的printf()使用方法

- 範例4-2：ch4_02.c

```
1  /*************************
2      檔名:ch4_02.c
3      功能:\n的練習
4   *************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  printf("您好.\n");
12  printf("歡迎學習C語言.");
13  /*  system("pause");  */
14 }
```

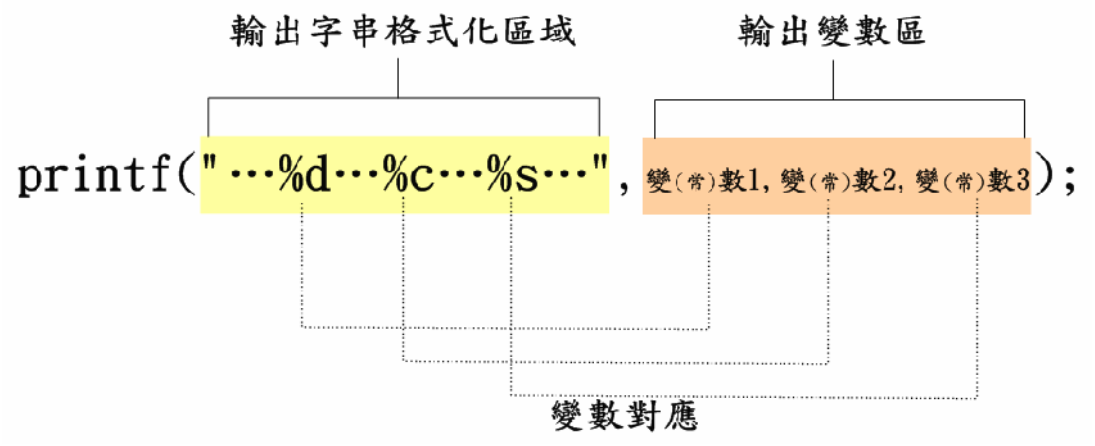
4.1.2 printf()語法

- printf()除了簡單輸出字串的方法之外，也提供了格式化輸出功能，並且我們可以透過控制符號，來設計我們想要的輸出格式，printf()就會自動把資料輸出成符合我們要求的格式。
- printf()語法：

```
#include <stdio.h>                /* 使用printf()必須載入stdio.h標頭檔 */
int printf(const char *format[,argument,…]);
```

  - 【註】：printf()函式會回傳一個整數，但一般使用時大多不會去接收這個回傳的整數。
  - printf()函式的引數分爲兩區域：輸出字串格式化區域與輸出變數區。

4.1.2 printf()語法



printf函式示意圖

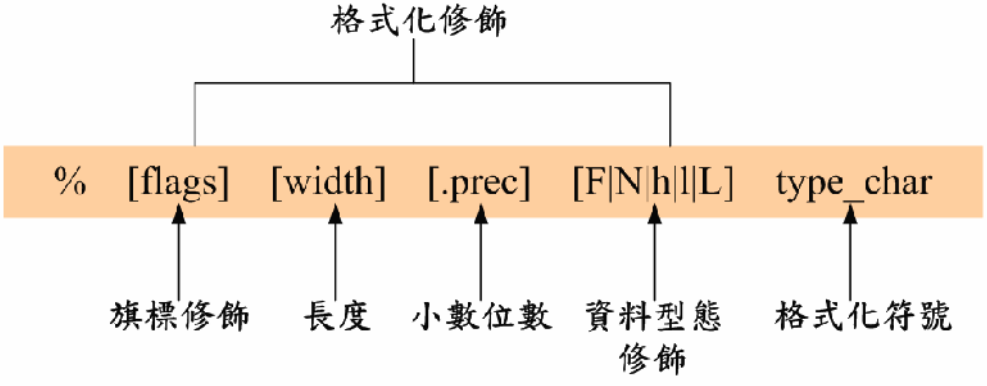
- 輸出字串格式化區域：本區域內容由一般字元與『%』字元組成，一般字元會忠實地顯示於螢幕上（跳脫字元則產生效果），而『%』字元代表一個『%』符號後面跟隨某些已經定義好的字元，例如：『%c』、『%d』等等。

- 輸出變數區：本區域存放0個以上的變數，視輸出字串格式化區域中，包含有多少個『%』字元而決定本區域的變數數目。並且這些變數將與『%字元』以一對一的方式對應格式。

4.1.3 魔術『%』

- 在輸出字串格式化區域中，凡是以『%』爲開頭的字元，其實就是代表某種特殊的資料規格，例如：%d代表將相對應的變數，以整數的方式來表現。事實上，這個特殊的『d』稱之爲格式化符號。

  - 『%』後面所接的格式，並不只單一個字元，其完整格式如下，我們將在本節中說明各參數的意義及組合變化。



『%』格式示意圖

4.1.3　魔術『%』

參數	選擇／必要	功能
[flags]	可選擇	用做正負號、左右對齊之控制。
[width]	可選擇	指定欄位寬度（可以與.prec搭配使用）
[.prec]	可選擇	指定小數位數及字串長度
[F|N|h|l|L]	可選擇	重新修飾資料型態
type_char	必要	配合變數資料型態，用來指定輸出格式，例如：%c、%d、%f等等。

- 【註】：若使用[ ]者，代表該選項可有可無。

4.1.3 魔術『%』

- type_char（格式化符號）
  - type_char（格式化符號）是最重要的一個參數，不可省略。
  - 我們應該依據對應的變數或常數的資料型態決定要選用哪一種的格式化符號，例如：輸出浮點數時使用『f』格式化符號，也就是『%f』。各種格式化符號細節如下表：

分類	%type_char	傳入之參數型態	輸出格式
字元類	%c	char	字元
	%s	char *（即字串）	字串
	%%	無	輸出『%』字元

分類	%type_char	傳入之參數型態	輸出格式
整數類	%d、%i	int	十進位整數（signed int）
	%o	int	八進位整數（unsigned int，前面不含0）
	%u	int	無正負號十進位整數（unsigned int）
	%x	int	小寫十六進位的整數（unsigned int）
	%X	int	大寫十六進位的整數（unsigned int）
浮點數類	%f	Float冂、double	浮點數（有效位數6位數）
	%e	float、double	以科學符號e表示的浮點數（有效位數6位數）
	%E	float、double	以科學符號E表示的浮點數（有效位數6位數）
	%g	float、double	以輸入值的精確度自動決定使用%f或%e輸出數值，小數以後多餘的0捨棄。
	%G	float、double	以輸入值的精確度自動決定使用%f或%E輸出數值，小數以後多餘的0捨棄。
指標類	%p	指標	指標位址
	%n	int指標	回傳%n前一字元的輸出位置（也就是字元數）

4.1.3 魔術『%』

- 範例4-3：ch4_03.c

```
1  /*****************************
2      檔名:ch4_03.c
3      功能:printf()格式化符號
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   int ch1 = 65;
10  char ch2='A';
11  char *str1="Hello!";
12  int data1 = 123456;
13  int data2 = -45678;
14  float f1,f2,sum;
15  int count;
16  f1 = 3.1415926;
17  f2 = 1.41421;
18  sum = f1 + f2;
```

```
19
20    printf("%%c格式 %c\n",ch1);
21    printf("%%c格式 %c\n",ch2);
22    printf("%%s格式 %s\n",str1);
23    printf("----------------------\n");
24    printf("%%d格式 %d  %%i格式 %i\n",data1,data2);
25    printf("%%o格式 %o\n",data1);
26    printf("%%u格式 %u  %%u格式 %u\n",data1,data2);
27    printf("%%x格式 %x  %%X格式 %X\n",data1,data1);
28    printf("----------------------\n");
29    printf("%%f格式 %f\n",f1);
30    printf("%%e格式 %e  %%E格式 %E\n",f2,f2);
31    printf("----------------------\n");
32    printf("%%g格式 %g  %%G格式 %G\n",sum,sum);
33    printf("%%p格式 %p\n",str1);
34    printf("123%n\n",&count);
35    printf("conut爲%d\n",count);
36    /*  system("pause");  */
    }
```

4.1.3 魔術『%』

- 執行結果：

```
%c格式 A
%c格式 A
%s格式 Hello!
----------------------
%d格式 123456   %i格式 -45678
%o格式 361100
%u格式 123456   %u格式 4294921618
%x格式 1e240   %X格式 1E240
----------------------
%f格式 3.141593
%e格式 1.414210e+000   %E格式 1.414210E+000
----------------------
%g格式 4.5558   %G格式 4.5558
%p格式 00401210
123
conut爲3
```

4.1.3 魔術『%』

- 範例說明：
  - (1) 連續出現的『%%』，會視爲顯示『%』字元。
  - (2) ch1是數字65，但若使用『%c』來顯示的話，則會依照ASCII的'A'來顯示字元。
  - (3) 『%d』與『%i』都可以顯示整數（正負整數），一般我們常用的是『%d』。
  - (4) 『%u』只會顯示正整數，若填入負整數，則會將之視爲正整數來顯示，例如：-45678的2's補數表示法爲『1111 1111 1111 1111 0100 1101 1001 0010』，所以視爲正數時，則爲『4294921618』，因此若是負數，請用『%d』或『%i』。
  - (5) 『%x』與『%X』都會依照16進制來表示數值，其差別在於『a~f』的大小寫。
  - (6) 『%e』與『%E』都會以科學記號方式表示浮點數，其差別在於科學記號『E』的大小寫。
  - (7) 『%g』與『%G』會刪除小數點末端多餘的0。
  - (8) 『%p』會將該變數在記憶體的位址印出。
  - (9) 第36行，由於只列印3個字元，並使用『%n』指定count的值，因此count的值爲『3』。

4.1.3 魔術『%』

- [width][.prec]（資料寬度及小數點位數）
  - width爲輸出資料的寬度，prec則爲精準度（precision），其實也就是小數點的位數。此兩個參數可有可無，並且可以互相搭配產生多種變化。請見以下說明：
    - (1) 當格式化符號屬於整數類型（例如：%d、%X）或字元型態（%c）時，prec參數就不會發生作用。
    - (2) 當格式化符號爲浮點數類型（例如：%f、%g）時，prec就可以用來指定小數點後面要輸出多少位數。
    - (3) 當格式化符號屬於字串類型（例如：char *）時，輸出格式會以prec來切割字串。

[width]	輸出格式
n	最少輸出n個字元，若輸出的值少於n個字元，則前面會補上空白，使得寬度維持在n。
0n	最少輸出n個字元，若輸出的值少於n個字元，則前面會補上0，使得寬度維持在n。
*	間接使用星號指定欄寬。

範例4-4：ch4_04.c

```
/*************************
    檔名:ch4_04.c
    功能:printf()[width]參數
 *************************/
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
 int data1=1234;
 float data2 = 45.67;
  printf("        ==>123456789\n");  /* 對齊資料之用 */
 printf("          ---------\n");
 printf(" %%3d格式==>%3d\n",data1);
 printf(" %%6d格式==>%6d\n",data1);
 printf("%%03d格式==>%03d\n",data1);
 printf("%%06d格式==>%06d\n",data1);
 printf(" %%*d格式==>%*d\n",5,data1);
 printf("==================================\n");
 printf("        ==>12345678901234567890\n");   /* 對齊資料之用 */
 printf("          ----------\n");
 printf(" %%15f格式==>%15f\n",data2);
 printf("%%015f格式==>%015f\n",data2);
 printf("  %%*f格式==>%*f\n",12,data2);
 /*  system("pause");  */
}
```

4.1.3 魔術『%』

- 執行結果：

```
          ==>123456789
              ---------
 %3d格式==>1234
 %6d格式==>   1234
%03d格式==>1234
%06d格式==>001234
 %*d格式==> 1234
==================================
            ==>12345678901234567890
                --------------------
 %15f格式==>        45.669998
%015f格式==>00000045.669998
  %*f格式==>    45.669998
```

4.1.3　魔術『%』

- 範例說明：
  - (1) 由於data1共有4位數『1234』，所以%3d與%03d都只會顯示完整的數值。當指定超過4位數時，就會產生變化，例如：%6d會在數值前面補上2個空白，%06d會在數值前面補上2個0。而*d則由輸入的『5』，決定了輸出5位數。
  - (2) data2是浮點數（預設小數位數爲6），當指定爲%15f時，會在前面補上6個空白（15-2-1-6=6），當指定爲%015f時，會在前面補上6個0，而*f則由輸入的『12』，決定了輸出12個數字（含小數點）。

4.1.3 魔術『%』

[.prec]	輸出格式
無（預設值）	小數點位數之系統預設值如下：%d、%i、%o、%u、%x、%X：預設值爲0。%f、%e、%E：預設爲輸出小數點6位。%g、%G：預設爲輸出全部小數點位數（但最後多餘的0會去除）。%s：預設輸出'\0'之前的所有字元。%c：無作用。
.0	%d、%i、%o、%u、%x、%X：使用系統的精確度。%f、%e、%E：不輸出小數部分。
.n	%s：輸出n個字元，若原始字串超過n個字元，則會被裁減。%f、%e、%E：輸出小數點n位（會保留小數最後面多餘的0）。%g、%G：輸出至少n個數字。
*	設定位數精準度。

4.1.3 魔術『%』

- 範例4-5：ch4_05.c

```
1  /*****************************
2      檔名:ch4_05.c
3      功能:printf()[.prec]參數
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int data1=1234;
12  float data2 = 45.6789;
13  double data3 = 78.900000;
14  char *str1="Hello!";
15
16  printf("   %%d格式==>%d\n",data1);
17  printf("   %%f格式==>%f\n",data2);
18  printf("   %%g格式==>%g\n",data3);
19  printf("   %%s格式==>%s\n",str1);
```

```
20  printf("==================================\n");
21   printf("        ==>12345678901234567890\n");  /*對齊資料之用*/
22   printf("           ------------------\n");
23   printf("%%.0d格式==>%.0d\n",data1);
24   printf("%%.0f格式==>%.0f\n",data2);
25   printf("==================================\n");
26   printf("        ==>12345678901234567890\n");  /*對齊資料之用*/
27   printf("           ------------------\n");
28   printf("%%.3s格式==>%.3s\n",str1);
29   printf("%%.1f格式==>%.1f\n",data2);
30   printf("%%.5f格式==>%.5f\n",data2);
31   printf("%%.1g格式==>%.1g\n",data3);
32   printf("%%.5g格式==>%.5g\n",data3);
33   printf("==================================\n");
34   printf("        ==>12345678901234567890\n");  /*對齊資料之用*/
35   printf("           ------------------\n");
36   printf("%%.*f格式==>%.*f\n",3,data2);
37   /*  system("pause");  */
38  }
```

4.1.3 魔術『%』

- 執行結果：

```
    %d格式==>1234
    %f格式==>45.678902
    %g格式==>78.9
    %s格式==>Hello!
==================================
          ==>12345678901234567890
              ------------------
%.0d格式==>1234
%.0f格式==>46
==================================
          ==>12345678901234567890
              ------------------
%.3s格式==>Hel
%.1f格式==>45.7
%.5f格式==>45.67890
%.1g格式==>8e+001
%.5g格式==>78.9
==================================
          ==>12345678901234567890
              ------------------
%.*f格式==>45.679
```

4.1.3 魔術『%』

- 範例說明：
  - (1) 第16~19行沒有設定[.prec]參數，所以會照預設值來顯示，例如浮點數的小數位數爲6位。
  - (2) 『.0d』與『.0f』都不會輸出小數位數。
  - (3) 『.3s』會將"Hello"剪裁爲3個字元輸出。『.1f』只會輸出一位小數。『.5f』會輸出五位小數。『.1g』只會輸出一位小數。『.5g』會輸出五位小數，但小數後面多餘的0會被刪除。
  - (4) 『.*』會依照輸入指定小數位數，在本例中，指定小數位數爲『3』位。

4.1.3 魔術『%』

- 範例4-6：ch4_06.c

```
1  /*****************************
2      檔名:ch4_06.c
3      功能:[width][.prec]參數
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  float data1=123.4;
12  float data2 = 45.678;
13  float data3 = 0.92;
14
15  printf("data1==>%09.5f\n",data1);
16  printf("data1==>%09.5f\n",data2);
17  printf("data1==>%09.5f\n",data3);
18  /*  system("pause");  */
19 }
```

4.1.3 魔術『%』

- 執行結果：

```
data1==>123.40000
data1==>045.67800
data1==>000.92000
```

- 範例說明：
  - (1) 這是[width]與[.prec]參數合作的範例，可以用來對齊浮點數資料。
  - (2) 『09.5f』代表共有9個字元（含小數點），並且小數位數固定爲5位。不足的整數部分前面會補0，不足的小數位數後面也會補0。

4.1.3 魔術『%』

- [flags]（正負旗標修飾）
  - flags參數可有可無，並且除了可以單獨設定某一個參數之外，還可以重複設定其他參數值。此外，除了『-』號之外，其餘參數值只會對於數字資料發生作用。參數值說明如下：

表：printf()-[flags]參數

4.1.3 魔術『%』

[flags]	輸出格式
-	與width合作定義對齊方式。由於系統內定的資料對齊爲靠右對齊，未達欄位寬度時，會在左邊補上空白或0（如之前的範例），而將flags參數設爲『-』，則可以將對齊方式改爲靠左對齊，並以空白補齊右邊的空位。
+	將flags參數設爲『+』，則不論是正數或負數都會在左邊出現『+』或『-』。
空白（balnk）	輸出爲正數時，數值前面以空白代替正號『+』，若同時出現『+』與空白參數值，則『+』優先權較高（也就是正數仍會印出『+』）。
#	要求完整呈現所有的數值位數。當遇上『%o』時（也就是%#o）會在數值前加上一個0，以代表8進位。當遇上『%x』時（也就是%#x）會在數值前加上一個0x，以代表16進位。若遇上『%f、%e、%E、%g、%G』，即使設定了不含小數位數（例%#.0f），仍舊會保留小數點。當遇上『%g、%G』時還會將原本要去除的小數尾數0給保留下來。

4.1.3 魔術『%』

- 範例4-7：ch4_07.c

```
1  /*****************************
2      檔名:ch4_07.c
3      功能:[flags]參數
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   int var1=5555,var2=6666,var3=7777;
10  float data1=123.4,data2 =-45.678,data3 = 0.92000;
11    printf("%-10f<==data1\n",data1);
12  printf("%-10f<==data2\n",data2);
13  printf("%-10f<==data3\n",data3);
14  printf("---------------------\n");
15  printf("data1==>%+.5f\n",data1);
16  printf("data2==>%+.5f\n",data2);
17  printf("data3==>%+.5f\n",data3);
```

4.1.3 魔術『%』

```
18  printf("--------------------\n");
19   printf("data1==>% .5f\n",data1);
20   printf("data2==>% .5f\n",data2);
21   printf("data3==>% .5f\n",data3);
22   printf("--------------------\n");
23   printf("data1==>%#g\n",data1);
24   printf("data2==>%#g\n",data2);
25   printf("data3==>%#g\n",data3);
26   printf("--------------------\n");
27   printf("var1==>%#x\n",var1);
28   printf("var2==>%#x\n",var2);
29   printf("var3==>%#x\n",var3);
30   printf("--------------------\n");
31   printf("%-#10g<==data1\n",data1);
32   printf("%-#10g<==data2\n",data2);
33   printf("%-#10g<==data3\n",data3);
34   /*  system("pause");  */
35  }
```

4.1.3 魔術『%』

- 執行結果：

```
123.400002<==data1
-45.678001<==data2
0.920000  <==data3
----------------------
data1==>+123.40000
data2==>-45.67800
data3==>+0.92000
----------------------
data1==> 123.40000
data2==>-45.67800
data3==> 0.92000
----------------------
data1==>123.400
data2==>-45.6780
data3==>0.920000
----------------------
var1==>0x15b3
var2==>0x1a0a
var3==>0x1e61
----------------------
123.400   <==data1
-45.6780  <==data2
0.920000  <==data3
```

4.1.3 魔術『%』

- 範例說明：
  - (1) 第14~16行，由於設定了『-』，因此原本10個字元會靠左對齊，並且不夠的字元會以空白加以補齊。
  - (2) 第18~20行，由於設定了『+』，因此不論是正數或負數都會印出正負號。
  - (3) 第22~24行，由於設定了空白『 』，因此，負號會印出負號，正號則會出現空白一格。
  - (4) 第26~28行，由於設定了『#』，因此，所有的位數都會完整印出，並且小數尾端的0也被保留。
  - (5) 第30~32行，由於設定了『#』，因此16進制會以『0x』爲開頭表示。
  - (6) 第34~36行，由於同時設定了『-』與『#』，因此不但會保留小數尾端的0，同時會靠左對齊，並且不夠的字元會以空白加以補齊。

4.1.3　魔術『%』

- [F|N|h|l|L]（資料型態修飾）
  - [F|N|h|l|L]資料型態修飾可以用來指定printf()函式應該使用哪一種資料型態來顯示傳入的變數值，例如：short int。參數值說明如下：

4.1.3 魔術『%』

[F|N|h|l|L]	格式化符號	輸出格式
F	%p、%s	遠指標（far pointer）
N	%n	近指標（near pointer）
h	%d、%i、%o、%u、%x、%X	short int
	%c、%C	單位元組字元
	%s、%S	單位元組字串
l	%d、%i、%o、%u、%x、%X	long int
	%e、%E、%f、%g、%G	double
	%c、%C	單位元組字元
	%s、%S	單位元組字串
L	%d、%i、%o、%u、%x、%X	long double
	%e、%E、%f、%g、%G	__int64（若支援）

- 【註】：本書的語法規則中，若使用『|』者，代表『或』。

4.1.3 魔術『%』

- 範例4-8：ch4_08.c

```
1  /*****************************
2      檔名:ch4_08.c
3      功能:[FlNlhlllL]參數
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  int var1=0x281a820e;
12
13  printf("var1==>%#8x\n",var1);
14  printf("var1==>%#8hx\n",var1);
15  /*  system("pause");  */
16 }
```

4.1.3 魔術『%』

- 執行結果：

```
var1==>0x281a820e
var1==>   0x820e
```

- 範例說明：
  - (1) var1宣告爲int（4個位元組），並指定其值爲『281a820e』（16進制表示法）。
  - (2) 第14行使用參數『h』指定重新以short int來顯示資料（即2個位元組），所以顯示結果恰好剩下後半部（820e）。

4.1.4 在printf()中使用跳脫字元

- 在前面我們曾經介紹ASCII定義了許多跳脫字元，這些跳脫字元無法由鍵盤以圖文方式輸入，但是卻會產生某些效果，例如：\n（換行）、\a（響鈴）。而我們可以利用輸出函式來展現其效果，例如：使用printf()函式，並將跳脫字元加入其中，如下所示。
  - printf("....\n\n");
  - 上述printf函式會在字串最後產生換兩行的效果。跳脫字元表如下所列：

跳脫字元	效果
\a	響鈴（Bell）
\b	退格一格（Backspace）
\f	通知印表機，資料由下一頁開始列印（Format）
\n	換行(Newline)
\r	回到該行第一格(Carriage Return)
\t	水平跳格，與tab鍵作用相同(Tab)
\v	垂直跳格（Vertical tab）
\0	空字元(Null)
\\	印出斜線符號『//』
\'	印出單引號『'』
\"	印出雙引號『"』
\?	印出問號『?』
\0	0爲8進制數字（最多3位數）
\xH	xH爲小寫16進制數字（最多2位數）
\XH	XH爲大寫16進制數字（最多2位數）

4.1.4 在printf()中使用跳脫字元

- 範例4-9：ch4_09.c

```
1  /*****************************
2      檔名:ch4_09.c
3      功能:跳脫字元
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  printf("123456789012345678901\n");
12  printf("Hello\n");
13  printf("\tHello\n");
14  printf("\t\tHello\n");
15  printf("\t\rHello\n");
16  /*  system("pause");  */
17 }
```

4.1.4 在printf()中使用跳脫字元

- 執行結果：

```
12345678901234567890
Hello
        Hello
                Hello
Hello
```

- 範例說明：
  - (1) 第12行，最後面『\n』會產生換行。
  - (2) 第13行，『\t』會間隔8個字元。
  - (3) 第14行，『\t\t』會間隔16個字元。
  - (4) 第15行，『\t』會間隔8個字元，但是又遇到『\r』所以退回第一個字元位置。

4.2 scanf()-C的格式化輸入函式

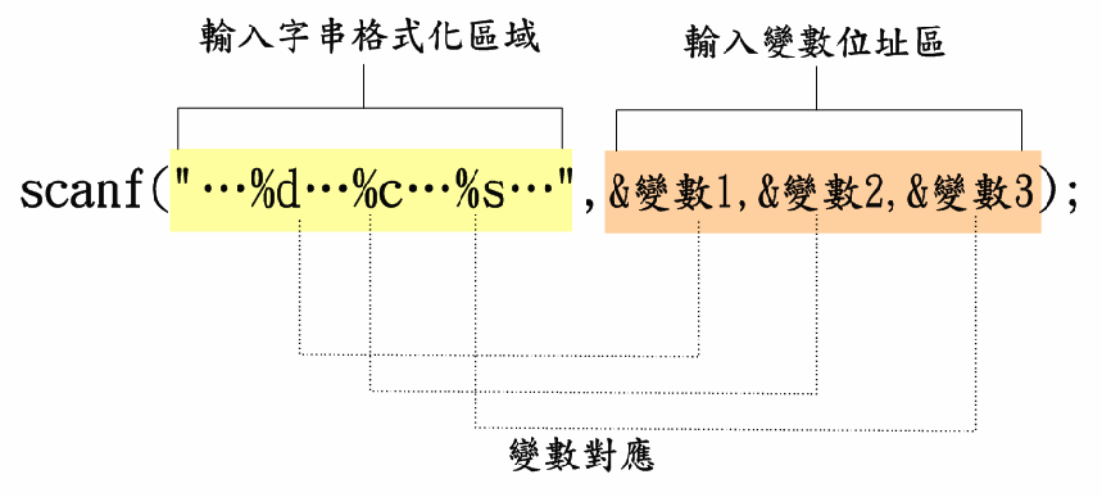
- 相對於printf()輸出函式，C所提供的格式化輸入函式則是scanf()，使用scanf，同樣必須載入<stdio.h>函式庫。

- 4.2.1 scanf()語法
  - scanf()在<stdio.h>函式庫中的定義如下：

```
#include <stdio.h>                    /*使用scanf()必須載入stdio.h標頭檔*/
int scanf(const char *format[,address,…]);
```

4.2.1 scanf()語法

- scanf()函式的參數分爲兩區域：輸入字串格式化區域與輸入變數位址區。

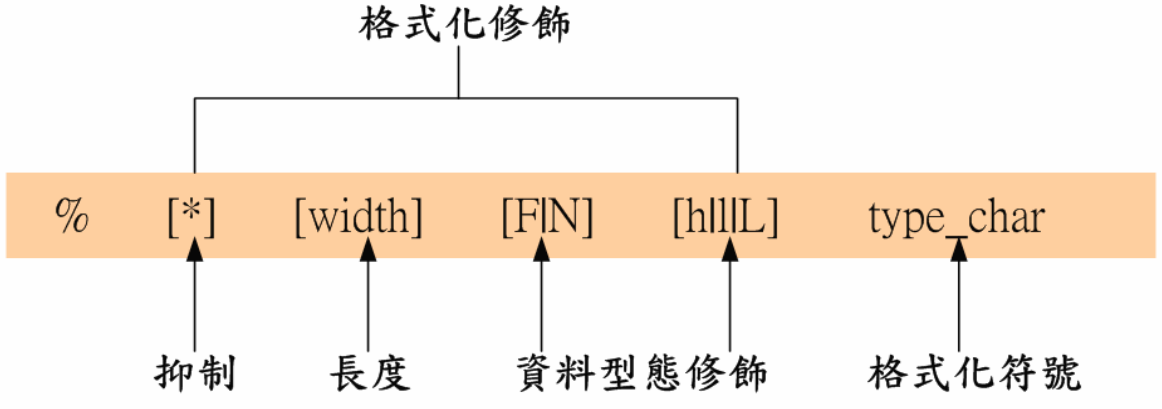


scanf函式示意圖

- 輸入字串格式化區域：本區域將決定輸入變數的資料格式，並且與輸入變數位址區的變數一一對應。
- 輸入變數位址區：由於scanf要求把輸入值填入變數在記憶體中的位址，因此必須提供變數在記憶體中的位址，取出變數記憶體位址使用的語法爲『&變數名稱』（若是字串指標，則直接指定即可）。

4.2.2 魔術『%』

- 和printf()類似，scanf也使用『%』爲開頭（稱爲格式化符號）來表示某種特殊的資料規格，用以格式化輸入資料。例如：%d代表將輸入的字元，以整數型態指定給對應的變數位址。
  - scanf()的格式化符號與printf()格式化符號的語法有一點點的不一樣，其完整格式如下。



『%』格式示意圖

4.2.2 魔術『%』

參數	選擇／必要	功能
[*]	可選擇	抑制格式化符號，使得雖然scanf可取得相對應的資料，但是卻不會真的存入記憶體位址。
[width]	可選擇	指定輸入欄位寬度。
[F|N]	可選擇	重新修飾資料型態。
[h|l|L]	可選擇	重新修飾資料型態。
type_char	必要	配合變數資料型態，用來指定輸入的數值要以哪一種格式填入變數位址，例如：%c、%d、%f等等。

4.2.2 魔術『%』

- 範例4-10：ch4_10.c

```
1  /***************************
2      檔名:ch4_10.c
3      功能:scanf()格式化符號
4   ***************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   char ch1;
10  int data1;
11  float f1;
12  char *str1;
13  printf("請輸入字元,字串,整數,浮點數(使用Enter加以間隔)\n");
14  scanf("%c%s%d%f",&ch1,str1,&data1,&f1);
15
16  printf("======正在輸出=======\n");
17  printf("ch1=%c\n",ch1);
18  printf("str1=%s\n",str1);
19  printf("data1=%d\n",data1);
20  printf("f1=%f\n",f1);
21  /*  system("pause");  */
22 }
```

4.2.2 魔術『%』

- 執行結果：

```
請輸入字元,字串,整數,浮點數(使用Enter加以間隔)
T
Hello
123
567.123
═══正在輸出═══
ch1=T
str1=Hello
data1=123
f1=567.122986
```

- 範例說明：
  - (1) 當要求輸入多個變數值的時候，scanf會自動以空白、『\t』、『\n』字元等當做分隔點。本例中，我們使用的是【Enter】所造成的『\n』字元。
  - (2) str1由於是字串，並且以指標方式加以宣告，因此本身就是一個位址（詳見指標一章），因此不需要使用『&』來取變數位址。其餘的整數、浮點數、字元變數則需要使用取址運算子『&』來取得變數位址。
  - (3) 在執行結果中粗體文字爲使用者透過鍵盤輸入的資料。

4.2.2 魔術『%』

- [width]（指定輸入資料的寬度）
  - width參數會告知scanf()函式，應該讀取多少長度的輸入資料。
- 範例4-11：ch4_11.c

```
1  /***************************
2      檔名:ch4_11.c
3      功能:[width]參數
4   ***************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   int data1;
10   printf("請輸入整數:");
11  scanf("%4d",&data1);
12  printf("======正在輸出=======\n");
13  printf("data1=%d\n",data1);
14  /*  system("pause");  */
15 }
```

4.2.2　魔術『%』

- 執行結果：

```
請輸入整數:56789
=======正在輸出=======
data1=5678
```

- 範例說明：
  - 雖然我們輸入了『56789』，但由於scanf()函式中指定了[width]參數爲『4』，所以變數data1實際取得的值爲前4位數。也就是『5678』。而最後一個『9』可能會造成標準輸入緩衝區中的垃圾，此時，應該使用fflush(stdin)函式來加以清除。

4.2.2 魔術『%』

- [*]（抑制格式化符號）
  - width參數會使得輸入的資料無法存入變數所佔用的記憶體空間。
- 範例4-12：ch4_12.c

```
1  /*************************
2      檔名:ch4_12.c
3      功能:[*]參數
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   int data1=1234;
10  printf("請輸入整數data1的值:");
11  scanf("%*d",&data1);
12  printf("======正在輸出=======\n");
13  printf("data1=%d\n",data1);
14  /*  system("pause");  */
15 }
```

4.2.2 魔術『%』

- 執行結果：

```
請輸入整數:56789
=======正在輸出=======
data1=1234
```

- 範例說明：
  - 雖然我們輸入了『56789』，但是由於設定了『*』，所以scanf()並不會將之存入變數data1的記憶體位址中，所以data1的值仍是『1234』。

4.2.2 魔術『%』

- [F|N]、[h|l|L]（資料型態修飾）
  - scanf()的[F|N]、[h|l|L]與printf()的[F|N|h|l|L]類似，也是用來修飾資料型態，F可以改寫遠指標的預設值，N可以改寫近指標的預設值。而h 代表short、l代表long或double、L代表long double，請看以下的範例，就很容易可以瞭解這些參數的作用。

4.2.2 魔術『%』

- 範例4-13：ch4_13.c

```
1  /*****************************
2      檔名:ch4_13.c
3      功能:[hlllL]參數
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9  int var1=0,var2=0;
10
11
12   printf("請用16進制輸入var1的值:");
13   scanf("%x",&var1);
14   printf("請用16進制輸入var2的值:");
15   scanf("%hx",&var2);
16   printf("var1==>%#x\n",var1);
17   printf("var2==>%#x\n",var2);
18   /*  system("pause");  */
   }
```

4.2.2 魔術『%』

- 執行結果：

```
請用16進制輸入var1的值:5a4f3eb2
請用16進制輸入var2的值:5a4f3eb2
var1==>0x5a4f3eb2
var2==>0x3eb2
```

- 範例說明：
  - 雖然我們兩次都輸入了『5a4f3eb2』，但是由於第16行設定了『h』參數，所以只會將後半部以short int方式指定給var2，在列印結果中，可以很明顯地看到這個現象。而事實上，C語言在宣告變數時，並不會自動做初始變數的動作，因此若將第11 行改寫爲『int var1,var2;』，則可以明顯看到var2後半部2個位元組被置換，而前半部則不會改變。

4.3ANSI C所提供的其他輸出入函式

- 除了printf()與scanf()之外，ANSI C語言的<stdio.h>函式庫還提供了許多好用的輸出入函式，在本節中，我們將這些函式的語法列出，並以範例來加以說明。
- 4.3.1　getchar()與putchar()

函式	引入標頭檔	說明
getchar()	<stdio.h>	由標準輸入裝置（鍵盤）取得單一字元（輸入完畢後，需要按【Enter】鍵）
putchar()	<stdio.h>	輸出單一字元到標準輸出裝置（螢幕）。

- 語法

```
#include <stdio.h>
int getchar(void);      /* 回傳值為輸入字元，失敗時回傳EOF */
int putchar(int c);     /* 輸出字元c，失敗時回傳EOF */
```

4.3.1 getchar()與putchar()

- 範例4-14：ch4_14.c

```
1
2   /*********************************
3       檔名:ch4_14.c
4       功能:getchar函式與putchar函式
5    *********************************/
6   #include <stdio.h>
7  #include <stdlib.h>
8  void main(void)
9  {
10  char ch1='k';
11  printf("原始字元是");
12  putchar(ch1);
13  printf("\n請輸入一個字元:");
14  ch1=getchar();
15  printf("您所輸入的字元:%c\n",ch1);
16  /*  system("pause");  */
    }
```

4.3.1 getchar()與putchar()

- 執行結果：

```
原始字元是k
請輸入一個字元:e
您所輸入的字元:e
```

- 範例說明：
  - (1) 第14行透過putchar(ch1)輸出單一字元ch1，此時ch1的內容是'k'。
  - (2) 執行結果的第一個『e』是我們輸入的字元，由getchar()函式將之傳給ch1變數，第二個『e』則是putchar()函式輸出ch1字元。

4.3.2 puts()與gets()

函式	引入標頭檔	說明
puts()	<stdio.h>	輸出字串到標準輸出裝置（螢幕），並且自動產生一個換行。
gets()	<stdio.h>	從標準輸出裝置（鍵盤）讀入一個字串。

- 語法：

```
#include <stdio.h>
int puts(const char *s);    /* 將字串s輸出到螢幕上 */
char *gets(char *s);        /* 將輸入的字串儲存到s  */
```

4.3.2 puts()與gets()

- 範例4-15：ch4_15.c

```
1  /*************************
2      檔名:ch4_15.c
3      功能:puts函式
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9   char *str1="Hello!";
10  char *str2="Welcome!";
11   puts(str1);
12  puts(str2);
13  /*  system("pause");  */
14 }
```

4.3.2 puts()與gets()

- 執行結果：

```
Hello!
Welcome!
```

- 範例說明：
  - 第11行與第12行分別利用puts()函式輸出字串str1、str2。請注意，當puts()函式執行完畢時，會自動換行，所以字串中不必出現\n換行字元。

4.3.2 puts()與gets()

- 範例4-16：ch4_16.c

```
1  /**************************
2      檔名:ch4_16.c
3      功能:gets函式
4   **************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   char str1[]="";
10  char *str2="您所輸入的字串如下:";
11   printf("請輸入一個字串:");
12  gets(str1);        /*不建議使用gets*/
13  puts(str2);
14  puts(str1);
15  /*  system("pause");  */
16 }
```

4.3.2 puts()與gets()

- 執行結果：

```
請輸入一個字串:I love C language
您所輸入的字串如下:
I love C language.
```

- 範例說明：
  - 第15行的gets(str1)可以接收一個字串輸入，並將之儲存在str1字串中。不過，我們強烈不建議使用gets()輸入字串，因爲gets()並不會考慮輸入字串是否大於它所擁有的記憶體配置空間，所以當使用者所輸入的字串大於所配置的記憶體時，將可能會出錯。
  - 所以我們建議使用fgets()來代替gets()函式，請見下一個範例。

4.3.3 fgets()與fputs()

函式	引入標頭檔	說明
fgets()	<stdio.h>	從檔案或鍵盤讀入一個字串。
fputs()	<stdio.h>	輸出字串到檔案或螢幕上，不會自動產生換行。

- fgets( )－從檔案串流中，讀出一段文字
  - 語法：

```
標頭檔：#include <stdio.h>
語法：char *fgets(char *s, int size, FILE *stream);
功能：由檔案中讀取一個字串。
```

  - 語法說明：
    - (1)回傳值：當回傳值等於string，代表讀取成功；當回傳值等於NULL時，則代表檔案讀取指標已經讀到檔案盡頭。
    - (2)s：存放讀入的字串。
    - (3)size：讀入字串長度+1。
    - (4)stream：一個已開啓的檔案指標，代表要讀取的檔案串流。若stream指定爲stdin，則目標會改爲鍵盤，也就是從鍵盤中讀取一個字串。

4.3.3 fgets()與fputs()

- 範例4-17：ch4_17.c

```
1  /*************************
2      檔名:ch4_17.c
3      功能:fgets函式
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   char str1[]="";
10  char *str2="您所輸入的字串如上";
11   printf("請輸入一個字串:");
12  fgets(str1,80,stdin);       /*使用fgets取代gets*/
13  printf("%s",str1);
14  printf("%s",str2);
15  /*  system("pause");  */
16 }
```

4.3.3 fgets()與fputs()

- 執行結果：

```
請輸入一個字串:I love C language
I love C language
您所輸入的字串如上
```

- 範例說明：
  - (1)第15行的fgets()可以接收檔案中的字串輸入，並將之儲存在str1字串中。由於我們將檔案指定爲標準輸入裝置stdin，所以fgets()將把鍵盤當作輸入字串的來源。
    - 而輸入字串的長度不可以超過80個字元（80記載於fget的引數中）。
  - (2)第16行的輸出與第17的輸出產生了換行現象，這是因爲使用fgets()輸入字串時，它會將最後由【Enter】鍵所造成的\n換行字元也存入了str1字串中。所以在上述的執行結果中，事實上str1的內容爲『I love C language\n』。

4.3.3 fgets()與fputs()

- fputs( )－寫入字串到檔案串流內
  - 語法：

```
標頭檔：#include <stdio.h>
語法：int fputs(const char *s, FILE *stream);
功能：寫入字串到檔案中。
```

  - 語法說明：
    - (1)回傳值：當回傳值等於EOF，代表寫入出現錯誤；當回傳值不等於EOF，代表表示寫入成功。
    - (2)s：欲寫入的字串。
    - (3)stream：一個已開啓的檔案指標，代表要寫入的檔案串流。若stream指定爲stdout，則目標會改爲螢幕，也就是將字串輸出到螢幕中。

4.3.3 fgets()與fputs()

- 範例4-18：ch4_18.c

```
1  /*************************
2      檔名:ch4_18.c
3      功能:fputs函式
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   char *str1="Hello";
10  char *str2="Welcome!";
11  fputs(str1,stdout);
12  fputs("\n",stdout);
13  fputs(str2,stdout);
14  /*  system("pause");  */
15 }
```

4.3.3 fgets()與fputs()

- 執行結果：

```
Hello
Welcome!
```

- 範例說明：
  - (1)本範例改寫自範例4-15，這一次我們使用fputs()來取代puts()函式，並且將目標檔案設爲stdout，也就是標準輸出裝置『螢幕』。
  - (2)使用fputs()時，只會將字串輸出，而不會自動換行，因此我們必須在第15行中加入換行字串的輸出，才能夠完成範例4-15的效果。

4.4 非ANSI C所提供的輸出入函式

- 除了上述的輸出入函式外，在某些編譯器中，還提供了getche()與getch()等的輸出入函式，這些函式由於不是標準ANSI C定義的函式，所以您必須確認您的編譯器是否提供了該函式庫及函式，若讀者使用提供<conio.h>函式庫的編譯器則可以使用下表所列的各個函式：

函式	引入標頭檔	說明
getche()	<conio.h>	由標準輸入裝置（鍵盤）取得單一字元，並顯示於螢幕上。（輸入完畢後，不需要按【Enter】鍵）
getch()	<conio.h>	由標準輸入裝置（鍵盤）取得單一字元，但不顯示於螢幕上。（輸入完畢後，不需要按【Enter】鍵）

<conio.h>所提供的輸入函式（非ANCI C標準）

4.4 非ANSI C所提供的輸出入函式

- 範例4-19：ch4_19.c

```
1  /*****************************
2      檔名:ch4_19.c
3      功能:getche()函式
4   *****************************/
5  #include <stdio.h>
6  #include <conio.h>  /* 必須引入conio.h */
7  #include <stdlib.h>
8  void main(void)
9  {
10  char ch1;
11   printf("請輸入一個字元:");
12  ch1= getche();
13  printf("\n");
14  printf("您所輸入的是字元是%c",ch1);
15  /*  system("pause");  */
16 }
```

4.4 非ANSI C所提供的輸出入函式

- 執行結果：

```
請輸入一個字元:p
您所輸入的是字元是p
```

- 範例說明：
  - 本範例必須引入<conio.h>才能夠使用getche()函式。
  - 第15行的getche()函式會將輸入的字元『p』存入ch1。
  - 輸入字元『p』之後，不必按下【Enter】鍵，就會自動執行儲存字元的動作。
  - 【本範例測試時所使用的平台爲Windows xp，編譯器爲Dev-C++。】

4.5 深入探討C語言的輸出入

- 在學會了C語言基本的輸出與輸入後，我們必須深入探討C語言有關於輸出入的相關問題，以便建立重要正確觀念。
  - 在第一章中，我們曾經介紹過電腦組成的5大單元，其中兩個單元爲輸入及輸出單元（簡稱I/O；Input/Output）通常是螢幕、鍵盤等等。
  - C語言其實並不具備I/O的處理能力，更明確的說，C語言本身並不提供此類的指令，因此，當我們需要在程式中處理有關I/O的動作時，必須藉助C函式庫的函式來加以完成。
  - 舉例來說，我們常常會使用printf()函式來輸出文字到螢幕I/O裝置，事實上，當編譯器編譯含有printf()函式的C語言程式時，並不會直接編譯printf()函式，而是將之留給連結器來做連結（Link），這是因爲I/O與硬體結構息息相關。
  - 因此，爲了提高C語言的可攜性，若將I/O功能從編譯器分離成爲獨立的函式，將可以在不同機器上，使用不同的函式庫（名稱相同但實作不同），就可以達到不修改原始程式，而能夠重新編譯連結成可用於該機器的執行檔。

4.5.2 I/O的轉向

- 設計程式時，我們會指定資料的輸入來源（例如：鍵盤）及輸出裝置（例如：螢幕），當程式完成後，若希望更改輸出入裝置，則必須修改程式內容，並且再度編譯、連結後才能改變這個現象，若在Linux／Unix／Dos等環境下，則可以透過I/O的轉向來改善這種狀況。
  - 一般所謂的標準輸出入裝置（standard input/output；簡稱stdio）指的是『鍵盤』與『螢幕』。
  - 許多作業系統提供的指令都是使用標準輸出裝置『螢幕』來做爲指令的輸出。
    - 例如在Dos中執行『dir』或在Linux/Unix中執行『ls』，則會將該目錄下的所有檔案資訊輸出到螢幕上，但其實，我們可以改變這個I/O的輸出裝置，例如執行『dir > test.txt』及『ls > test.txt』會將目錄下的檔案資訊輸出到text.txt檔案中，而非輸出到螢幕上。

4.5.2 I/O的轉向

- Linux/Unix/Dos所提供的轉向符號是『>』、『>>』、『<』，格式如下：
  - (1)執行檔名稱 > 檔案F1：
    - 將執行檔的輸出轉向送往檔案F1，同時檔案F1內的原始資料會被清除；換句話說，原本應該輸出在螢幕上的文字將不會顯示在螢幕上，而是存入了檔案F1之中，並將檔案F1的原始資料覆蓋。
  - (2)執行檔名稱 >> 檔案F2：
    - 將執行檔的輸出轉向送往檔案F2，同時檔案F內的原始資料不會被清除；換句話說，原本應該輸出在螢幕上的文字將不會顯示在螢幕上，而是存入了檔案F2的末端（檔案F2的原始資料仍舊保存著）。
  - (3)執行檔名稱 < 檔案F3：
    - 將檔案F3送往執行檔處理。

4.6 本章回顧

- 在本章中，我們學習了C語言的輸出入函式。printf()是C語言<stdio.h>函式庫提供最常見的輸出函式，scanf()則是<stdio.h>函式庫提供最常見的輸入函式，並且scanf()必須提供變數位址以供存放輸入值。
- 除此之外，我們還介紹了getchar()、putchar()、puts()等C語言提供的輸出入函式。同時，我們建議使用fgets()來取代gets()，以避免無法預期的錯誤。

本章習題