C 格式化輸入／輸出

9

學習目標

在本章中，你將學到：

- 使用輸入和輸出資料流。
- 使用所有的顯示格式化功能。
- 使用所有的輸入格式化功能。
- 使用具有欄位寬度和精確度的顯示功能。
- 使用 `printf` 格式控制字串中的格式化旗標。
- 輸出字元常數和跳脫序列。
- 使用 `scanf` 將輸入格式化。

本章綱要

9.1 簡介

9.2 資料流

9.3 使用 printf 的格式化輸出。

9.4 顯示整數

9.5 顯示浮點數

9.6 顯示字串和字元

9.7 其他轉換指定詞

9.8 使用欄位寬度和精準度的顯示方式

9.9 在 printf 格式控制字串中使用旗標

9.10 字元常數和跳脫序列的顯示

9.11 使用 scanf 的格式化輸入。

9.1 簡介

- 表示結果是解決任何問題中很重要的一部分。
- 本章將會深入介紹scanf和printf的格式化功能。
- 這兩個函式分別可以用來從標準輸入資料流 (standard input stream) 輸入資料，以及從標準輸出資料流 (standard output stream) 輸出資料。
- 第8章曾討論過其他四個使用標準輸入和標準輸出的函式－gets、puts、getchar和putchar。
- 呼叫上述的函式時，必須含入《stdio.h》標頭。

9.2 資料流

- 所有的輸入和輸出都是以資料流 (streams) 來完成，資料流也就是位元組所形成的序列。
- 當輸入操作時，位元組就會從裝置 (例如，鍵盤、磁碟機或者網路連線) 流向主記憶體。
- 在輸出操作當中，就是從主記憶體流向一個裝置 (例如顯示器、印表機、磁碟機、網路連線) 的位元組串流。
- 當程式開始執行時，三種資料流會自動連到程式。

9.2 資料流

- 通常標準輸入資料流會連到鍵盤，而標準輸出資料流則會連到螢幕。
- 作業系統通常允許將這些資料流重新導向 (redirected) 到其他的裝置。
- 第三個資料流，標準錯誤資料流 (standard error stream) 也是連接到顯示器。
- 錯誤訊息會輸出到標準錯誤資料流。

9.3 使用 printf 的格式化輸出

- 精確的格式化輸出是以printf來完成的。
- 每一個printf呼叫都包含一個格式控制字串 (format control string)，它會用來描述輸出的格式。
- 格式控制字串也包含轉換指定詞、旗標、欄位寬度、精確度和字元常數。
- 再加上使用(%)，就構成轉換指定 (conversion specifications)。

9.3 使用 printf 的格式化輸出

- printf函式可以執行以下列出的格式化功能，本章將討論所列出的每一種功能。
  - 捨入 (Rounding) 浮點數值到指定的小數位數。
  - 對齊 (Aligning) 有多行數字的小數欄位。
  - 將輸出的結果向右對齊 (Right-justification) 或向左對齊 (left-justification)。
  - 在一行輸出當中精確的位置插入字元常數 (Inserting literal characters)。
  - 以指數格式表示浮點數(floating-point numbers)。
  - 以八進位制和十六進位制的格式來表示無號整數。(請參閱附錄C數字系統，以便獲得更多關於八進位和十六進位的資訊。)
  - 用固定的欄位寬度和精確度來顯示所有型別的資料。

9.3 使用 printf 的格式化輸出

- printf函式有以下格式：
  - printf(*format-control-string*, other-arguments);

  其中format-control-string描述輸出格式，而other-arguments（不一定要有）則對應到format-control-string中的每一種轉換指定。
- 每個轉換指定都是以%開始，並且用轉換指定詞當做結束。
- 同一個格式控制字串中可能具有許多個轉換指定。

常見的程式設計錯誤 9.1

忘記用雙引號將格式控制字串括起來會造成語法錯誤。

良好的程式設計習慣 9.1

盡量能讓輸出表示法簡潔，這可以讓程式的輸出更容易閱讀，並且減少使用者的錯誤。

9.4 顯示整數

- 整數是不含小數的數字，例如776、0或-52。
- 整數值可以使用幾種格式中的一種來加以顯示。
- 圖9.1描述整數轉換 (integer conversion) 指定詞。

轉換指定詞	說明
d	顯示有號十進位整數。
i	顯示有號十進位整數。[注意：用在 scanf 的時候，i 和 d 指定詞是不一樣的。]
o	顯示無號八進位整數。
u	顯示無號十進位整數。
x or X	顯示無號十六進位整數。X 會顯示數字 0-9 以及字母 A-F。而 x 則顯示數字 0-9 及字母 a-f。
h or l (letter l)	放在任何整數轉換指定詞之前，分別用來顯示 short 或 long 整數。更嚴謹地說，h 和 l 應該稱爲長度修飾詞 (length modifiers)。

圖 9.1　整數轉換指定詞

9.4 顯示整數

- 圖9.2的程式使用整數轉換指定詞來顯示整數。
- 請注意它只有顯示出負號，而不會顯示正號。
- 還有使用%u讀取-455時 (第15行)，它會轉換成無號值4294966841。

```c
/* Fig 9.2: fig09_02.c */
/* Using the integer conversion specifiers */
#include <stdio.h>

int main( void )
{
   printf( "%d\n", 455 );
   printf( "%i\n", 455 ); /* i same as d in printf */
   printf( "%d\n", +455 );
   printf( "%d\n", -455 );
   printf( "%hd\n", 32000 );
   printf( "%ld\n", 2000000000L ); /* L suffix makes literal a long */
   printf( "%o\n", 455 );
   printf( "%u\n", 455 );
   printf( "%u\n", -455 );
   printf( "%x\n", 455 );
   printf( "%X\n", 455 );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.2　整數轉換指定詞的使用方式

```
455
455
455
-455
32000
2000000000
707
455
4294966841
1c7
1C7
```

圖 9.2　整數轉換指定詞的使用方式

常見的程式設計錯誤 9.2

使用預期是 unsigned 值的轉換指定詞來顯示負的數值。

9.5 顯示浮點數

- 浮點數是指包含小數的數值，例如33.5、0.0或-657.983。
- 浮點數值可以使用幾種格式當中的一種來加以顯示。
- 圖9.3描述浮點數轉換指定詞。
- 轉換指定詞 (conversion specifiers) e和E會將浮點數以指數記號 (exponential notation) 顯示。指數表示法在電腦中和數學的科學記號 (scientific notation) 是一樣的。

9.5 顯示浮點數

- 例如，150.4582可以用科學記號表示如下：
  - $1.504582 x 10^2$
- 也可以用電腦將指數記號表示成
  - 1.504582E+02
- 這個記號代表1.504582乘上10的二次方(E+02)。
- 其中E代表「指數」。

9.5 顯示浮點數

- 用轉換指定詞e、E和f顯示出來的值，預設小數點右邊會有6位數的精確度 (例如1.04592)；但我們也可明確地指定不一樣的精確度。
- 轉換指定詞 f 總是至少會在小數點前面顯示一位。
- 轉換指定詞e和E會分別在指數前面顯示小寫的e和大寫的E，它們都只在小數點前面顯示一位。
- 轉換指定詞 g (或 G) 會顯示沒有補上0的 e(E) 或 f 格式 (例如 1.234000 會顯示成 1.234)。

9.5 顯示浮點數

- 如果數值轉換成指數記號之後，該數值的指數小於 -4 或是大於等於指定的精準度 (也就是 g 和 G 預定的六位有效位數)，則這個數值就會使用e (E)顯示；
- 不然就會使用轉換指定詞f顯示這個數值。
- 使用g或G輸出數值的小數部分補上的0不會顯示。
- 但至少會輸出一個小數位數。

9.5 顯示浮點數

- 使用轉換指定詞g，數值0.0000875、8750000.0、8.75、87.50和875將分別會顯示成8.75e-05、8.75e+06、8.75、87.5和875。
- 數值0.0000875使用了e的表示法，因為當它轉換成指數記號時，它的指數會小於-4 (也就是-5) 。
- 數值8750000.0也使用e的表示法，因為它的指數(６)等於預設的精準度。

轉換指定詞	說明
e or E	以指數記號表示一個浮點數值。
f	以固定點表示法顯示浮點數值。[請注意：在C99中也可以使用F。]
g or G	以浮點數形式 f 或指數形式 e (或 E) 顯示浮點數值的大小。
L	放在任何浮點轉換指定詞之前，表示顯示的數值是 long double 浮點數。

圖 9.3　浮點數轉換指定詞的使用

9.5 顯示浮點數

- 轉換指定詞g和G指出顯示的最大有效數字，這包括小數點左邊的有效數字。
- 如果使用%g，數值1234567.0將會顯示為1.23457e+06 (請記住，所有的浮點數轉換指定詞的預設精確度都是6)。
- 請注意，這個結果中共有6位有效數字。
- 當數值以指數記號顯示時，g和G之間的差別與e和E之間的差別是相同的－小寫的g會輸出小寫的e，大寫的G會輸出大寫的E。

測試和除錯的小技巧 9.1

當輸出資料時，請確定使用者察覺到資料會因爲格式化而不是很精確 (也就是指定精確度會產生捨去誤差)。

9.5 顯示浮點數

- 圖9.4示範每個浮點數轉換指定詞的用法。
- 請注意，%E、%e以及%g轉換指定詞會使輸出的數值四捨五入，而轉換指定詞%f則不會。
- 某些編譯器的輸出中，指數會以加號再加兩位數字的形式出現。

```c
/* Fig 9.4: fig09_04.c */
/* Printing floating-point numbers with
   floating-point conversion specifiers */

#include <stdio.h>

int main( void )
{
   printf( "%e\n", 1234567.89 );
   printf( "%e\n", +1234567.89 );
   printf( "%e\n", -1234567.89 );
   printf( "%E\n", 1234567.89 );
   printf( "%f\n", 1234567.89 );
   printf( "%g\n", 1234567.89 );
   printf( "%G\n", 1234567.89 );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.4　浮點數轉換指定詞的使用

```
1.234568e+006
1.234568e+006
-1.234568e+006
1.234568E+006
1234567.890000
1.23457e+006
1.23457E+006
```

圖 9.4　浮點數轉換指定詞的使用

9.6 顯示字串和字元

- 轉換指定詞c和s分別是用來顯示個別的字元和字串。
- 轉換指定詞 **c** 必須使用char引數，
- 轉換指定詞 **s** 的引數是指向char的指標。
- 轉換指定詞s會不斷顯示字元，直到遇到結束的空('\0')字元為止。
- 圖9.5中的程式說明使用指定詞c和s的字元和字串。

常見的程式設計錯誤 9.3

使用 %c 顯示字串是一個錯誤。轉換指定詞 %c 希望收到 char 引數。但是字串是指向 char 的指標 (也就是 char *)。

常見的程式設計錯誤 9.4

使用 %s 來顯示 char 引數會導致致命的執行期錯誤，也稱爲記憶體逾越存取的錯誤。轉換指定詞 %s 希望收到的引數是指向 char 的指標。

常見的程式設計錯誤 9.5

使用單引號括住字元字串會導致語法錯誤。字串必須用雙引號括住。

常見的程式設計錯誤 9.6

用雙引號括住字元常數會產生一個含有兩個字元的字串，其中第二個字元就是結束的空字元。

```c
/* Fig 9.5: fig09_05c */
/* Printing strings and characters */
#include <stdio.h>

int main( void )
{
   char character = 'A'; /* initialize char */
   char string[] = "This is a string"; /* initialize char array */
   const char *stringPtr = "This is also a string"; /* char pointer */

   printf( "%c\n", character );
   printf( "%s\n", "This is a string" );
   printf( "%s\n", string );
   printf( "%s\n", stringPtr );
   return 0; /* indicates successful termination */
} /* end main */
```

```
A
This is a string
This is a string
This is also a string
```

圖 9.5　字元和字串轉換指定詞的使用方法

9.7 其他轉換指定詞

- 剩下的三種轉換指定詞是p、n和% (圖9.6)。
- 轉換指定詞%n會儲存目前printf敘述式已輸出的字元個數－相對應的引數是一個指向整數變數的指標，而字元個數儲存在該變數中。使用%n時不會顯示任何東西。
- 轉換指定詞%則使百分比符號用來輸出。

轉換指定詞	說明
p	使用實作環境所定義的方法顯示一個指標值。
n	儲存目前 printf 敘述式已經輸出的字元個數。指向整數的指標會用來當作對應的引數。不會顯示任何資料。
%	顯示百分比字元。

圖 9.6　其他轉換指定詞

9.7 其他轉換指定詞

- 圖9.7的程式當中，%p顯示了ptr的值以及x的位址；因為ptr的值設定為x的位址，所以這兩個值是相同的。
- 接著%n將第三個printf敘述式 (第15列) 顯示過的字元個數存放到整數變數y，然後顯示y的值。
- 最後一個printf敘述式 (第21行) 則使用了%%來顯示字元字串當中的%字元。

9.7 其他轉換指定詞

- 要注意每個printf呼叫都會傳回一個值－如果輸出發生錯誤，就會傳回一個負值；不然就會傳回輸出的字元個數。
- [請注意：本範例無法在微軟的Visual C++環境中執行，微軟由於「安全因素」限制了%n的使用。]

可攜性的小技巧 9.1

轉換指定詞 p 會以實作環境定義的方式來顯示位址 (大部分的系統都使用十六進位制表示法，而不使用十進位制表示法)。

常見的程式設計錯誤 9.7

嚐試顯示百分比字元，但是在格式控制字串當中使用 % 而不是使用 %%。當 % 出現在格式控制字串中時，它的後面必須接著一個轉換指定詞。

```c
/* Fig 9.7: fig09_07.c */
/* Using the p, n, and % conversion specifiers */
#include <stdio.h>

int main( void )
{
   int *ptr; /* define pointer to int */
   int x = 12345; /* initialize int x */
   int y; /* define int y */

   ptr = &x; /* assign address of x to ptr */
   printf( "The value of ptr is %p\n", ptr );
   printf( "The address of x is %p\n\n", &x );

   printf( "Total characters printed on this line:%n", &y );
   printf( " %d\n\n", y );

   y = printf( "This line has 28 characters\n" );
   printf( "%d characters were printed\n\n", y );

   printf( "Printing a %% in a format control string\n" );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.7　p、n 和 % 轉換指定詞的使用方式

```
The value of ptr is 0012FF78
The address of x is 0012FF78

Total characters printed on this line: 38

This line has 28 characters
28 characters were printed

Printing a % in a format control string
```

圖 9.7　p、n 和 % 轉換指定詞的使用方式

9.8 使用欄位寬度和精準度的顯示方式

- 顯示在欄位當中資料的位數是由欄位寬度 (field width) 來指定。
- 如果欄位寬度大於要顯示的資料，則程式的資料在欄位中會靠右對齊。
- 代表欄位寬度的整數，可以加入轉換指定詞中的百分比符號（％）與轉換指定詞之間 (例如：％4d)。
- 圖9.8中的程式顯示兩組整數，每一組有5個整數。若要顯示的數值位數小於欄位寬度，則此數字會靠右對齊。
- 若要顯示的數值的位數大於欄位寬度，則程式會自動增加欄位寬度。負數的負號則會佔欄寬的一個位置。
- 欄位寬度可以與各種轉換指定詞一起使用。

常見的程式設計錯誤 9.8

顯示數值時沒有提供足夠的欄位寬度。可能會造成其他資料顯示的移位，也可能會造成令人困擾的輸出。請確定你的資料大小！

```c
/* Fig 9.8: fig09_08.c */
/* Printing integers right-justified */
#include <stdio.h>

int main( void )
{
   printf( "%4d\n", 1 );
   printf( "%4d\n", 12 );
   printf( "%4d\n", 123 );
   printf( "%4d\n", 1234 );
   printf( "%4d\n\n", 12345 );

   printf( "%4d\n", -1 );
   printf( "%4d\n", -12 );
   printf( "%4d\n", -123 );
   printf( "%4d\n", -1234 );
   printf( "%4d\n", -12345 );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.8　整數在欄位當中靠右對齊

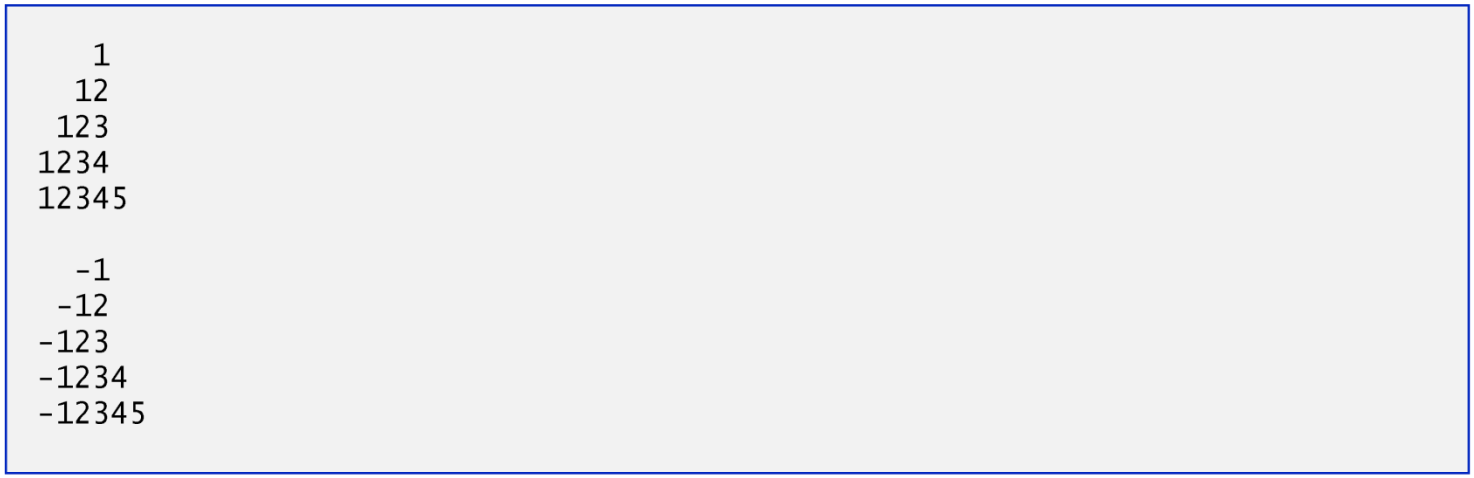

圖 9.8　整數在欄位當中靠右對齊

9.8 使用欄位寬度和精準度的顯示方式

- printf函式也提供指定顯示資料精確度的功能。
- 精確度對於不同的資料型別有不同的意義。
- 和整數轉換指定詞一起使用時，精確度指出程式至少必須顯示多少個位數。
- 如果顯示數值的位數小於指定的精確度，則多出來的位數就會補上0。
- 假如精準度並非以零或是小數點來表示，則會補上空白。

9.8 使用欄位寬度和精準度的顯示方式

- 整數預設的精確度是1。
- 當和浮點數轉換指定詞e、E和f一起使用的時候，精確度代表小數點後面應該有幾個位數。
- 和轉換指定詞g和G一起使用的時候，精準度代表的是顯示的最大有效數字位數。
- 和轉換指定詞s一起使用的時候，精確度代表的是從這個字串顯示的字元個數的最大值。
- 當使用精確度時，放一個小數點(.)，接著，在轉換指定詞和百分比符號之間放置一個表示精確度的整數。

9.8 使用欄位寬度和精準度的顯示方式

- 圖9.9中的程式示範如何在格式控制字串中使用精確度。
- 要注意，顯示浮點數值時，假如所指定的精確度小於浮點數的小數位數，則此浮點數就會四捨五入。

```c
/* Fig 9.9: fig09_09.c */
/* Using precision while printing integers,
   floating-point numbers, and strings */
#include <stdio.h>

int main( void )
{
   int i = 873; /* initialize int i */
   double f = 123.94536; /* initialize double f */
   char s[] = "Happy Birthday"; /* initialize char array s */

   printf( "Using precision for integers\n" );
   printf( "\t%.4d\n\t%.9d\n\n", i, i );

   printf( "Using precision for floating-point numbers\n" );
   printf( "\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f );

   printf( "Using precision for strings\n" );
   printf( "\t%.11s\n", s );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.9　使用精準度顯示幾種型別的資訊

```
Using precision for integers
        0873
        000000873

Using precision for floating-point numbers
        123.945
        1.239e+002
        124

Using precision for strings
        Happy Birth
```

圖 9.9　使用精準度顯示幾種型別的資訊

9.8 使用欄位寬度和精準度的顯示方式

- 欄位寬度和精確度可以一起使用，方法是先寫欄位寬度，再寫一個小數點，最後才寫精確度。例如以下敘述式
  - `printf( "%9.3f", 123.456789 );`

  會顯示123.457，小數點右邊有三位，並且在寬度為9的欄位當中向右對齊。
- 程式也可以在格式控制字串後面的引數列中，使用整數運算式來指定欄位寬度和精確度。

9.8 使用欄位寬度和精準度的顯示方式

- 要使用這項功能，你必須加入星號(*)取代欄位寬度或精確度 (或是二者)。
- 引數列中和int相對的引數會先計算出來，並且放置在星號的位置。
- 欄位寬度的數值可以是正的或負的 (負的欄位寬度會讓顯示的資料靠左對齊，我們會在下一節討論它。)。
- 敘述式
  - `printf( "%*.*f", 7, 2, 98.736 );`

  用7代表欄位寬度、2代表精準度，並且將輸出的數值98.74靠右對齊。

9.9 在 printf 格式控制字串中使用旗標

- printf函式還提供了旗標來輔助它的格式化輸出功能。
- 共有5種旗標可以用於格式控制字串中 (圖9.10)。
- 要在格式化控制字串中使用旗標，請直接在百分記號右邊放置旗標。
- 同一個轉換指定詞中可以同時使用數個旗標。

旗標	說明
- (負號)	在指定的欄位中將輸出靠左對齊。
+ (正號)	在正值之前顯示正號，在負值之前顯示負號。
space	在正數之前顯示一空格，但不印 + 號。
#	和八進位轉換指定詞 o 一起使用的時候，輸出之前會加上 0。
	和十六進位轉換指定詞 x 或 X 一起使用的時候，在輸出之前會加上 0x 或 0X。
	沒有小數部份但是會以 e,E,f,g 或 G 顯示的浮點數來顯示小數點。(一般只在含有小數部分時才會顯示小數點。) 對於 g 和 G 指定詞而言，小數點後面多餘的零不會消除。
0 (零)	使用前導的零填補欄位。

圖 9.10　格式化字串控制旗標

9.9 在 printf 格式控制字串中使用旗標

- 圖9.11的程式示範字串、整數、字元以及浮點數的靠右對齊和靠左對齊。

```c
/* Fig 9.11: fig09_11.c */
/* Right justifying and left justifying values */
#include <stdio.h>

int main( void )
{
   printf( "%10s%10d%10c%10f\n\n", "hello", 7, 'a', 1.23 );
   printf( "%-10s%-10d%-10c%-10f\n", "hello", 7, 'a', 1.23 );
   return 0; /* indicates successful termination */
} /* end main */
```

```
     hello         7         a  1.230000

hello     7         a         1.230000
```

圖 9.11　字串在欄位當中靠左對齊

9.9 在 printf 格式控制字串中使用旗標

- 圖9.12的程式分別使用有+旗標 (flag) 和沒有+旗標的方式，顯示一個正數和一個負數。
- 在兩種情況中都會顯示負號，但是正號只有在使用+旗標的時候才會顯示出來。

```c
/* Fig 9.12: fig09_12.c */
/* Printing numbers with and without the + flag */
#include <stdio.h>

int main( void )
{
   printf( "%d\n%d\n", 786, -786 );
   printf( "%+d\n%+d\n", 786, -786 );
   return 0; /* indicates successful termination */
} /* end main */
```

```
786
-786
+786
-786
```

圖 9.12　使用或沒有使用 + 旗標來顯示正數和負數

9.9 在 printf 格式控制字串中使用旗標

- 圖9.13的程式使用空白旗標 (space flag) 在一個正數前面增加一格空白。
- 這對於具有同樣位數的正數和負數的對齊很有幫助。
- 請注意，因為負號的關係，輸出值-547之前不會放置一個空白

```c
/* Fig 9.13: fig09_13.c */
/* Printing a space before signed values
   not preceded by + or - */
#include <stdio.h>

int main( void )
{
   printf( "% d\n% d\n", 547, -547 );
   return 0; /* indicates successful termination */
} /* end main */
```

```
 547
-547
```

圖 9.13　使用空白旗標

9.9 在 printf 格式控制字串中使用旗標

- 圖9.14的程式使用#旗標 (flag) 在八進位制數值的前面加上一個0，在十六進位制數值前面加上0x和0X，並強迫以g顯示的數值顯示小數點。

```c
/* Fig 9.14: fig09_14.c */
/* Using the # flag with conversion specifiers
   o, x, X and any floating-point specifier */
#include <stdio.h>

int main( void )
{
   int c = 1427; /* initialize c */
   double p = 1427.0; /* initialize p */

   printf( "%#o\n", c );
   printf( "%#x\n", c );
   printf( "%#X\n", c );
   printf( "\n%g\n", p );
   printf( "%#g\n", p );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.14　使用#旗標

```
02623
0x593
0X593

1427
1427.00
```

圖 9.14　使用#旗標

9.9 在 printf 格式控制字串中使用旗標

- 圖9.15的程式結合了+旗標和0 (零) 旗標，在9位的欄位當中將452顯示成具有+號和前導零，然後只使用0旗標和9個欄位將452再顯示一次。

```c
/* Fig 9.15: fig09_15.c */
/* Printing with the 0( zero ) flag fills in leading zeros */
#include <stdio.h>

int main( void )
{
   printf( "%+09d\n", 452 );
   printf( "%09d\n", 452 );
   return 0; /* indicates successful termination */
} /* end main */
```

```
+00000452
000000452
```

圖 9.15　使用 0 (零) 旗標

9.10 字元常數和跳脫序列的顯示

- 大多數想用printf敘述式顯示的字元常數都可以放在格式控制字串中。
- 但是有許多有「問題」的字元，像是用來隔開格式控制字串的雙引號(")。
- 新行和水平跳格都必須用跳脫序列(escape sequences)來表示。
- 跳脫序列會用反斜線(\ )加上一個特別的跳脫字元(escape character)來加以表示。
- 圖9.16列出所有的跳脫序列以及它們執行的動作。

常見的程式設計錯誤 9.9

嘗試使用 `printf` 敘述式顯示單引號、雙引號、問號或反斜線等字元常數，但是卻沒有在這些字元的前面加上反斜線來構成正確的跳脫字元，會造成錯誤。

跳脫序列	說明
\' (單引號)	輸出單引號 (') 字元。
\" (雙引號)	輸出雙引號 (") 字元。
\? (問號)	輸出問號 (?) 字元。
\\ (反斜線)	輸出反斜線 (\ ) 字元。
\a (警告音或鈴聲)	輸出可聽見的聲音 (鈴聲) 或聽的到的警告聲。
\b (退格)	將游標在目前所在的行當中往後移一位。
\f (跳頁)	將游標移到下一個邏輯頁的起始位置。
\n (新行)	將游標移到下一行的起始位置。
\r (正向換行符號)	將游標移至目前所在行的起始位置。
\t (水平跳格)	將游標移至下一個水平跳格的位置。
\v (垂直跳格)	將游標移至下一個垂直跳格的位置。

圖 9.16　跳脫序列

9.11 使用 scanf 的格式化輸入

- 精確的輸入格式化會使用scanf來加以完成。
- 每個scanf敘述式都包含格式控制字串，它會用來描述要輸入資料的格式。
- 格式控制字串包含轉換指定詞和字元常數。
- scanf函式具有以下的輸入格式化功能：
  - 輸入所有型別的資料。
  - 從輸入資料流當中輸入指定的字元。
  - 跳過輸入資料流中的某些指定字元。

9.11 使用 scanf 的格式化輸入

- scanf函式會寫成以下的形式：
  - scanf(*format-control-string*, *other-arguments*);

其中format-control-string描述輸入的格式，other-arguments則是一些指向變數的指標，這些變數會用來存放輸入的資料。

良好的程式設計習慣 9.2

當輸入資料時，每次只提示使用者輸入一個或少數資料即可。不要要求使用者在一個提示中輸入太多資料。

良好的程式設計習慣 9.3

請考慮當輸入錯誤的資料時，使用者和你的程式應該做些什麼，例如：輸入了對程式來說無意義的整數值，或是字串遺漏了標點和空白。

9.11 使用 scanf 的格式化輸入

- 圖9.17整理用來輸入所有型別的資料的轉換指定詞。
- 本節剩餘的部分會提供程式來示範如何使用各scanf轉換指定詞來讀取資料。

轉換指定詞	說明
整數	
d	讀進一個有正負號的十進位整數。對應的引數是指向整數的指標。
i	讀進一個有正負號的十進位，八進位或十六進位整數。對應的引數是指向整數的指標。
o	讀進一個八進位整數。對應的引數是指向無號整數的指標。
u	讀進一個無號的十進位整數。對應的引數是指向無號整數的指標。
x or X	讀進一個十六進位的整數。對應的引數是指向無號整數的指標。
h 或 l	用於任何整數轉換指定之前，表示希望輸入 short 或 long 的整數。

圖 9.17　scanf 的轉換指定詞

浮點數	
e, E, f, g or G	讀進一個浮點數。對應的引數是指向浮點變數的指標。
l or L	用於任何浮點數轉換指定之前，表示希望輸入 double 或 long double 的浮點數。對應的引數是指向 double 或 long double 變數的指標。
字元和字串	
c	讀進一個字元。對應的引數是一個指向 char 的指標；不會加上空 ('\0') 字元。
s	讀進一個字串。對應的引數是一個指向型別爲 char 的陣列的指標，並且該陣列足以儲存字串以及結尾的空 ('\0') 字元—這是自動增加的。

圖 9.17　scanf 的轉換指定詞

掃瞄集	
[scan characters]	在一字串當中尋找存放在某陣列中的字元。
其它	
p	讀取一個在 printf 敘述式中和 %p 的輸出具有相同形式的位址。
n	儲存目前爲止 scanf 函式讀入的字元個數。對應的引數是指向整數的指標。
%	跳過輸入時的百分比符號 (%)。

圖 9.17　scanf 的轉換指定詞

9.11 使用 scanf 的格式化輸入

- 圖9.18的程式使用不同的引數轉換指定詞來讀取數個引數，然後以十進位制將這些引數顯示出來。
- 注意，其中的%i可以用來輸入十進位制、八進位制、以及十六進位制的整數。

```c
/* Fig 9.18: fig09_18.c */
/* Reading integers */
#include <stdio.h>

int main( void )
{
   int a;
   int b;
   int c;
   int d;
   int e;
   int f;
   int g;

   printf( "Enter seven integers: " );
   scanf( "%d%i%i%i%o%u%x", &a, &b, &c, &d, &e, &f, &g );

   printf( "The input displayed as decimal integers is:\n" );
   printf( "%d %d %d %d %d %d %d\n", a, b, c, d, e, f, g );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.18　使用整數轉換指定詞來讀取輸入

```
Enter seven integers: -70 -70 070 0x70 70 70 70
The input displayed as decimal integers is:
-70 -70 56 112 56 70 112
```

圖 9.18　使用整數轉換指定詞來讀取輸入

9.11 使用 scanf 的格式化輸入

- 要輸入浮點數的時候，程式可以使用浮點轉換指定詞e、E、f、g或G。
- 圖9.19的程式使用三種浮點轉換指定詞讀進三個浮點數。並且將這三個數字以轉換指定詞f顯示出來。
- 程式的輸出說明浮點數值的不精確性，你可以從第三個顯示的數值察覺這項事實。

```c
/* Fig 9.19: fig09_19.c */
/* Reading floating-point numbers */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   double a;
   double b;
   double c;

   printf( "Enter three floating-point numbers: \n" );
   scanf( "%le%lf%lg", &a, &b, &c );

   printf( "Here are the numbers entered in plain\n" );
   printf( "floating-point notation:\n" );
   printf( "%f\n%f\n%f\n", a, b, c );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.19　使用浮點數轉換指定詞來讀取輸入

```
Enter three floating-point numbers:
1.27987 1.27987e+03 3.38476e-06
Here are the numbers entered in plain
floating-point notation:
1.279870
1279.870000
0.000003
```

圖 9.19　使用浮點數轉換指定詞來讀取輸入

9.11 使用 scanf 的格式化輸入

- 字元和字串分別用轉換指定詞c和s來進行輸入。
- 圖9.20的程式會提示使用者輸入一個字串。
- 這個程式使用%c來讀取字串中的第一個字元，將它存放到字元變數x，然後用%s讀入字串的剩餘部分，並且將它存到字元陣列y。

```c
/* Fig 9.20: fig09_20.c */
/* Reading characters and strings */
#include <stdio.h>

int main( void )
{
   char x;
   char y[ 9 ];

   printf( "Enter a string: " );
   scanf( "%c%s", &x, y );

   printf( "The input was:\n" );
   printf( "the character \"%c\" ", x );
   printf( "and the string \"%s\"\n", y );
   return 0; /* indicates successful termination */
} /* end main */
```

```
Enter a string: Sunday
The input was:
the character "S" and the string "unday"
```

圖 9.20　輸入字元和字串

9.11 使用 scanf 的格式化輸入

- 一連串的字元可以用掃瞄集 (scan set) 來進行輸入。
- 掃瞄集是格式控制字串中，一些由中括號 [ ] 括起來，並且前面加上百分比符號的字元。
- 掃瞄集會掃瞄輸入資料流中的字元，找出屬於此掃瞄集的字元。
- 每次找到一個字元時，程式就會存放到掃瞄集對應的引數中－也就是指向字元陣列的指標。
- 當遇到一個不包含在掃描集中的字元時，掃描集就停止輸入字元。

9.11 使用 scanf 的格式化輸入

- 如果輸入資料流的第一個字元不在掃瞄集中，則只有**null**字元會存放到陣列中。
- 圖9.21的程式會使用掃瞄集[aeiou]來掃瞄輸入資料流中的母音字母。
- 注意，程式會讀入前7個字母。
- 第8個字母(h)不屬於掃瞄集，所以掃瞄的動作到此結束。

```c
/* Fig 9.21: fig09_21.c */
/* Using a scan set */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   char z[ 9 ]; /* define array z */

   printf( "Enter string: " );
   scanf( "%[aeiou]", z ); /* search for set of characters */

   printf( "The input was \"%s\"\n", z );
   return 0; /* indicates successful termination */
} /* end main */
```

```
Enter string: ooeeooahah
The input was "ooeeooa"
```

圖 9.21　使用掃描集

9.11 使用 scanf 的格式化輸入

- 我們也可以使用反掃瞄集 (inverted scan set) 來掃瞄不在掃瞄集中的字元。
- 只要在掃瞄集中的掃瞄字元的前面加上一個脫字符號 (^) 就可建立反掃瞄集。
- 這會使scanf儲存沒有出現在掃描集輸入的字元。
- 當遇到第一個屬於反掃瞄集中的字元時，這個輸入動作就會結束。
- 圖9.22的程式使用反掃瞄集[^aeiou]來找尋子音字母－更適當的說法是尋找「非母音」字母。

```c
/* Fig 9.22: fig09_22.c */
/* Using an inverted scan set */
#include <stdio.h>

int main( void )
{
   char z[ 9 ];

   printf( "Enter a string: " );
   scanf( "%[^aeiou]", z ); /* inverted scan set */

   printf( "The input was \"%s\"\n", z );
   return 0; /* indicates successful termination */
} /* end main */
```

```
Enter a string: String
The input was "Str"
```

圖 9.22　使用反掃描集

9.11 使用 scanf 的格式化輸入

- 欄位寬度也可以用於scanf的轉換指定詞中，用來從輸入資料流讀取指定個數的字元。
- 圖9.23的程式會輸入一連串的數字字元，其中前兩個字元讀取成兩位數的整數，其他的字元則讀取成另一個整數。

```c
/* Fig 9.23: fig09_23.c */
/* inputting data with a field width */
#include <stdio.h>

int main( void )
{
   int x;
   int y;

   printf( "Enter a six digit integer: " );
   scanf( "%2d%d", &x, &y );

   printf( "The integers input were %d and %d\n", x, y );
   return 0; /* indicates successful termination */
} /* end main */
```

```
Enter a six digit integer: 123456
The integers input were 12 and 3456
```

圖 9.23　使用欄位寬度來輸入資料

9.11 使用 scanf 的格式化輸入

- 我們常需要跳過輸入資料流中的某些字元。
- 例如日期可能會輸入為
  - 11-10-1999
- 日期中的每一個數字都需要儲存起來，但分隔數字的一劃則可以丟棄。
- 為了消去不需要的字元，我們可以將這些字元放到scanf格式控制字串中 (空白字元如空格、新行和跳格－這些會跳過所有前導的空白)。

9.11 使用 scanf 的格式化輸入

- 例如，要在輸入中跳過橫線，你可以使用敘述式
  - `scanf( "%d-%d-%d", &month, &day, &year );`
- 雖然這個scanf可以消去上述輸入的橫線，但是日期也可能會以下面的方式進行輸入
  - 10/11/1999
- 在這種情形，前一個scanf將無法去除不需要的字元。
- 基於這個原因，scanf提供設定禁止字元(assignment suppression character)*。

9.11 使用 scanf 的格式化輸入

- 這個字元可使scanf從輸入的資料流中讀進任何型別的資料，並將它丟棄而不設定給變數。
- 圖9.24的程式在%c轉換指定中使用了設定禁止字元，表示應該從輸入資料流中讀入一個字元並且將它丟棄。
- 所以程式只會儲存月、日、年。
- 將這些變數的值顯示出來以證明它們都是正確輸入的。
- 每個scanf呼叫的引數列上沒有變數會對應到使用設定禁止字元的轉換指定詞，因為程式並不會對這種轉換指定執行任何的指定動作。
- 對應的字元只會丟棄。

```c
/* Fig 9.24: fig09_24.c */
/* Reading and discarding characters from the input stream */
#include <stdio.h>

int main( void )
{
   int month1;
   int day1;
   int year1;
   int month2;
   int day2;
   int year2;

   printf( "Enter a date in the form mm-dd-yyyy: " );
   scanf( "%d%*c%d%*c%d", &month1, &day1, &year1 );

   printf( "month = %d  day = %d  year = %d\n\n", month1, day1, year1 );

   printf( "Enter a date in the form mm/dd/yyyy: " );
   scanf( "%d%*c%d%*c%d", &month2, &day2, &year2 );

   printf( "month = %d  day = %d  year = %d\n", month2, day2, year2 );
   return 0; /* indicates successful termination */
} /* end main */
```

圖 9.24　從輸入資料流中讀進並且捨棄字元

```
Enter a date in the form mm-dd-yyyy: 11-18-2003
month = 11  day = 18  year = 2003

Enter a date in the form mm/dd/yyyy: 11/18/2003
month = 11  day = 18  year = 2003
```

圖 9.24　從輸入資料流中讀進並且捨棄字元