國立屏東商業技術學院 資訊工程系 物件導向程式設計

6. 格式化輸入與輸出

<本章內容絕大部份與C語言一致>

- format specifiers
- 指定輸出的格式
- 指定輸入的格式

6.1 printf()函式的格式指定子

事實上，格式指定子(format specifier)也是有格式的，請參考figure 1。

Fig. 1: format specifier的格式

所有的format specifier都必須以%開頭，其它欄位說明如下：

6.1.1 Conversion

因為printf()函式的輸出，是依照在格式字串的定義來將特定型態的資料，以format specifier的方式加以輸出；因此我們把下列這些specifier稱為conversion。表示將資料轉換型態後輸出。conversion specifier以%開頭。C語言共有以下的conversion specifier。如table 1。

Format Specifier	**意義**
d, i	十進制的整數
o	八進制的整數
x, X	十六進制的整數
u	unsigned整數
f	float型態
e	以scientific notation表示
g,G	在%f或%e的結果中選擇較短者
c	字元
s	字串(詳細說明請參考第13章)
p	記憶體位址
n	將到目前為止已輸出的字元數，寫入到對應的參數中
%	輸出%

Tab. 1: printf()的Conversion Specifiers

請參考以下的範例，編寫成一個程式並加以編譯執行，想一想n是什麼用途？

```
int x;

printf("This is\n a test for %n.\n", &x);
```

6.1.2 Flags

Flags為選擇性的(optional)，超過一個以上的flag也是允許的，請參考下表table 2。

Flag	意義
-	置左對齊(預設為置右對齊)
+	強制顯示正負號(預設僅於負數時顯示)
#	若為八進制數值，強制其以0開頭
	若為非0的十六進制數值，強制以0x或0X開頭
	若為浮點數且輔以g或G conversions時，其右側(尾端)的0不予移除
0	若指定數值顯示位數時，其位數不足時於左側補0
	若conversion為i,d,o,u,x或X且有指定precision時，則忽略此flag
	- flag優先於 0

Tab. 2: printf()的格式指定子的Flags欄位

6.1.3 Minimum Field Width

此欄位也是可選擇性的，用以定義當數值資料顯示時的最少位數。當位數不足時，預設會在數值的左側以空白補滿(也就是置右對齊)。當然，當位數超過時，數值資料還是會完整的顯示的。若數值為浮點數時，此欄位後面若無指定小數點後的位數時，此欄位則用以定義小數點後的位數，不足處補0。

要特別注意的是，這個欄位除了以整數決定顯示位數外，也可以使用*(星號)。一但使用了星號，minimum field width的值就要由接在格式字串後的的參數來決定，請參考下面的例子：

```
  printf("%*d", 10, 8343); //在格式字串後面有兩個參數, 10會代入前面的*, 為後面的8343限制其minimum field width
```

6.1.4 Precision

此欄位也是可選擇性的，以.開頭後接一個整數，該整數的意義取決於所使用的conversion，請參考table ##:

Conversions	意義
d, i, o, u, x, X	可以minimum field width併用，當位數不足時，precision的部份會在左側補0。

Conversions	意義
e, E, f, F	定義在小數點後的位數，位數不足時則在右側補0
g, G	定義significant部份的位數
s	定義最多可顯示的bytes

Tab. 3: printf()的格式指定子的precision欄位

6.1.5 Length Modifier

此部份亦為選擇性的，用以補充說明欲顯示的資料之型態前是否要加short或long，視conversion的不同，參考table 4。

Length Modifier	Conversions	意義
h	d, i, o, u, x, X	short int, unsigned short int
	n	short int *
l(小寫的L)	d, i, o, u, x, X	long int, unsigned long int
	n	long int *
L	a, A, e, E, f, F, g, G	long double

Tab. 4: printf()的格式指定子的Length Modifier欄位

6.2 scanf()函式的格式指定子

scanf()和printf()函式都有格式字串，但scanf()是用以指定輸入的資料之格式，scanf()函式的格式字串可以包含以下三個部份，如figure 2。

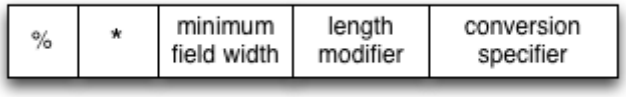

Fig. 2: scanf()的format specifier的格式

- Conversion specifiers
  - 指定要將使用者從標準輸入管道(也就是鍵盤)輸入的資料，轉換成何種型態的資料。
- White-space字元
  - space, tab與enter三者稱為white-space，在格式字串中的一個或一個以上連續white-space，會用以對應在輸入資料中的一個或一個以上的white-space。
- Non-white-space字元
  - 在格式字串中的非空白字元，會對應在輸入資料中的相同字元。

例如：下面的程式碼要求使用者輸入(XXX) XXXX-XXXX格式的電話號碼，不過這個例子並沒能指定各部份的位數。

```
scanf("(%d) %d-%d", &area, &prefix, &postfix);
```

6.2.1 conversion specifier

關於conversion specifier與printf()十分相似，但仍有不同之處。彙整於table 5。

Format Specifier	意義
d	十進制的整數
i	十進制的整數，但若以0或0x或0X開頭則為八進制或十六進制的整數
o	八進制的unsigned int
x, X	十六進制的unsigned int。x與X指定十六進制數字中a-e為大寫或小寫
u	十進制的unsigned int
f, e, E, g, G	取得float型態的浮點數，預設小數點後有六位，若不足則補0
c	字元
s	字串(詳細說明請參考第13章)
[	取回符合scanset條件的字串(詳細說明請參考第13章)
p	記憶體位址
n	將到目前為此已輸入的字元數，寫入到對應的參數中
%	取回%

Tab. 5: scanf()的Conversion Specifiers

6.2.2 Maximum Field Width

此部份為選擇性，指定所取回的資料最大的字元數，但在資料左側的空白不列入計算。

6.2.3 Length Modifier

此部份為選擇性，指定所取回的資料為short或long型態。視conversion的不同，請參考table 6。

Length Modifier	Conversions	意義
h	d, i, o, u, x, X, n	short int, unsigned short int
l	d, i, o, u, x, X, n	long int, unsigned long int
	e, E, f, F, g, G	double
	c, s, [	wchar_t (寬字元，詳見第13章
L	e, E, f, F, g, G	long double

Tab. 6: scanf()的格式指定子的Length Modifier欄位

6.3 printf()與scanf()應用範例

6.3.1 I/O轉向與管線(I/O redirect and pipeline)

考慮下面兩個簡單的程式：

```
using namespace std;
#include <iostream>

int main()
{
  cout << 123 << endl;
}
```

```
using namespace std;
#include <iostream>

int main()
{
  int x;
  cin >> x;
  cout << x*2;
}
```

我們使用以下的命令，來將其compile成r1與r2

```
[12:56 user@ws example] g++ -o r1 r1.cpp
[12:56 user@ws example] g++ -o r2 r2.cpp
```

在Linux/Unix系統上執行程式時，預設會開啟三個通道standard input(標準輸入), standard output(標準輸出)與(standard error)標準錯誤，其中standard input連結到鍵盤，而後兩者都連結到螢幕。透過'<'、'>'與'»'可以將這些預設的通道轉向(redirect)例如：

```
[1:18 user@ws example] ./r1
123
[1:18 user@ws example] ./r1 > result.1
[1:18 user@ws example] cat result.1
123
[1:18 user@ws example] ./r1 >> result.1
[1:18 user@ws example] cat result.1
123
123
[1:18 user@ws example]
```

其中'>'會產生新的檔案，若檔案已存在則會被覆蓋；'»'則是會將內容附加到檔案的後面，若檔案不存在則會建立一個新檔案。我們也可以將文字檔案的內容轉向給r2例如:

```
[1:24 user@ws example] cat result.1
123
```

```
[1:24 user@ws example] ./r2 < result.1
246
```

在Linux/Unix系統中，還有一個有用的工具稱為管線(pipeline)，我們可以將程式間的輸出與輸入串連起來，例如：

```
[1:24 user@ws example] ./r1 | ./r2
246
```

6.3.2 scanf()輸入多筆資料

scanf()可以讓我們一次取得一筆以上的輸入，例如：

```
#include <iostream>

int main()
{
   int x,y;

   scanf("%d%d", &x, &y);

   printf("x=%d, y=%d\n", x, y);
}
```

其中第7行在格式字串中要求取回兩個整數，請執行這個程式，並且輸入3和5。有哪些方法可以輸入這兩個整數？試著輸入以下的組合：

(為便於討論，我們使用S,T,E代表空白，Tab與Enter)

- 3 S 5 E
- E S 3 T 5 E
- S 3 SS 5 E
- 3 TT 5 TS E
- T 3 SSSS 5 SSS E

由結果可得知，在這兩個整數的左側、中間與右側，不論你輸入幾個空白，tab或enter，其結果都相同。如本章前面所說明過的，我們將連續的空白，tab與enter的組合，視為一個white-space，令W={space, tab, enter}*，其中*代表重覆0次或多次，則兩個整數3與5的各種可能的輸入可以歸納如下：

- W 3 W 5 W E

注意，我們如果把第7行的格式字串，改成“%d %d”，“ %d %d “，” %d%d “，其結果仍相同。以下我們先列出格式字串的內容，再說明不同的輸入的結果:

- ”%1d %1d”
  1. W 1 W 1 W E → x=1, y=1
  2. 11 E → x=1, y=1
  3. 123 E → x=1, y=2
- “%2d %2d”

1. W 3 W 5 W E → x=3, y=5
2. W 12 W 34 W E → x=12, y=34
3. W 12345 W E → x=12, y=34

6.3.3 scanf()略過輸入資料

scanf()也可以讓我們略過部份的輸入。請再考慮以下的程式：

```
#include <iostream>

int main()
{
   int x;

   scanf("%*d %2d", &x);

   printf("x=%d\n", x);
}
```

以下為幾個輸入的結果：

1. W 1 W 2 W E → x=2
2. W 1234 W 5 E → x=5

6.3.4 scanf()與\n的問題

考慮以下的程式碼：

```
scanf("%c", &c1);
...
scanf("%c", &c2);
```

這個程式片段取回了兩個字元c1與c2，假設我們想輸入的是'A'與'B'，但是因為輸入'A'時，必須在鍵盤上輸入'A'與Enter，所以當第一個scanf()取得'A'後，下一個scanf()就直接取得了前面所輸入的Enter，也就是說c2的內容成了'\n'，要解決這個問題，只要在scanf()的格式字串中多加一個空白即可，以下是修改過的程式碼：

```
scanf("%c", &c1);
...
scanf("  %c", &c2);
```

另外要注意，如果在格式字串的結尾處多了一個空白，則會使scanf()多進行一次的資料讀取：包含一筆資料與一個Enter□請參考以下的範例：

```
printf("Please input a number: ");
scanf("%d ", &x);
printf("x=%d", x);
```

其執行結果如下：

```
Please input a number: 34   // 資料輸入後(按下Enter後)程式沒有回應，除非在繼續輸入一些資料，再按一次Enter程式才會繼續執行
5
x=34
```

6.3.5 printf()與scanf()的傳回值

雖然我們不常這樣使用，但在使用printf()與scanf()函式時，是可以取得整數的傳回值。以printf()函式為例，其傳回值為其成功輸出的字元數；而scanf()的傳回值則為成功取得的資料個數。請參考下面的程式碼：

```
int x,y;
char c;

y = scanf("%d %c", &x, &c);

x = printf("The number of input data is %d.\n", y);

printf("The above line has %d characters.\n", x);
```

這程式要求使用者輸入一個整數與一個字元，當輸入正確時□scanf()順利取得兩個資料項目，所以其傳回值為2。假設使用者剛好將整數與字元的順序弄反了，則其一個資料項目都讀不到，其傳回值則為0。

```
[12:49 user@ws example] ./a.out
124 d
The number of input data is 2.
The above line has 31 characters.
[12:49 user@ws example] ./a.out
d 124
The number of input data is 0.
The above line has 31 characters.
[12:49 user@ws example] ./a.out
```

適當的利用這個傳回值，可以在未來做使用者輸入資料的檢查。