

CHAPTER 9 - FORMATTED INPUT/OUTPUT

9.1 Introduction

- 這章主要說明兩個要點
 - 輸出/入結果的呈現
 - `scanf` and `printf`
 - Streams (input and output)
 - `gets`, `puts`, `getchar`, `putchar` (in `<stdio.h>`)

9.2 Streams

- Streams

- 可以想像為組織成一行一行的一連串的字元
 - 每一行包含零個或多個字元，每行結尾為'\n'
 - ANSI C必須支援一行至少254字元
- 可以重新導向
 - Standard input – keyboard (stdin)
 - Standard output – screen (stdout)
 - Standard error – screen (stderr)

J:\Course\C programming\fall-2006\examples\chapter09\stdstream.c

Stdstream.c

```
#include<stdio.h>

main()
{
    int x,y;
    while(scanf("%d %d",&x,&y)==2) {
        printf("the max of %d and %d is %d\n",x,y,(x>y)?x:y);
    }
}
```

J:\Course\C programming\fall-2006\examples\chapter09\pipestream.c

Pipestream.c

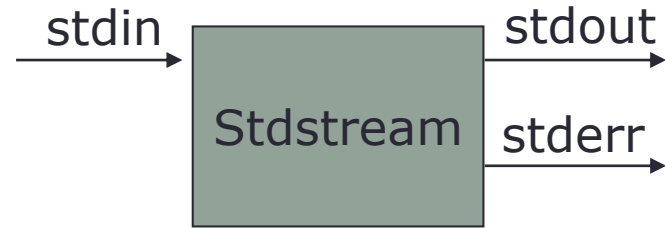
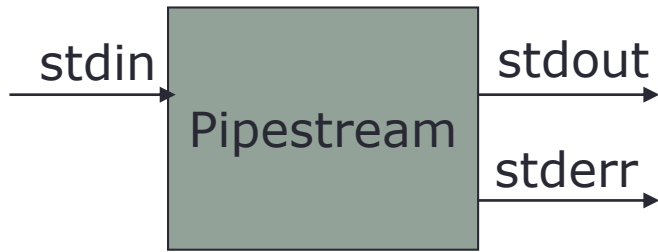
```
#include<stdio.h>
#include<stdlib.h>

main()
{
    int i;
    srand(time(NULL));
    for(i = 0; i < 10; i++) {
        printf("%d %d\n",rand(),rand());
    }
}
```

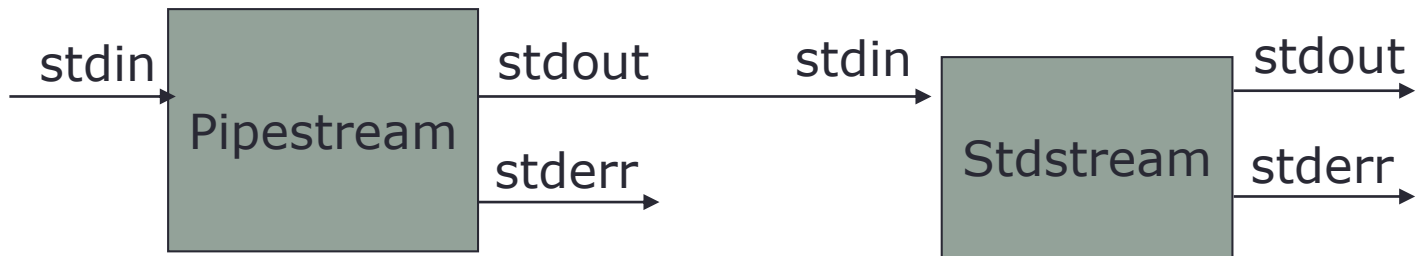
C:\E:\WINDOWS\system32\cmd.exe

```
J:\Course\C programming\fall-2006\examples\chapter09>pipestream | stdstream
the max of 23589 and 24967 is 24967
the max of 18735 and 2615 is 18735
the max of 4883 and 2035 is 4883
the max of 16300 and 11294 is 16300
the max of 20425 and 32548 is 32548
the max of 6143 and 30814 is 30814
the max of 26846 and 29230 is 29230
the max of 18949 and 9160 is 18949
the max of 26286 and 30348 is 30348
the max of 32668 and 31448 is 32668
```

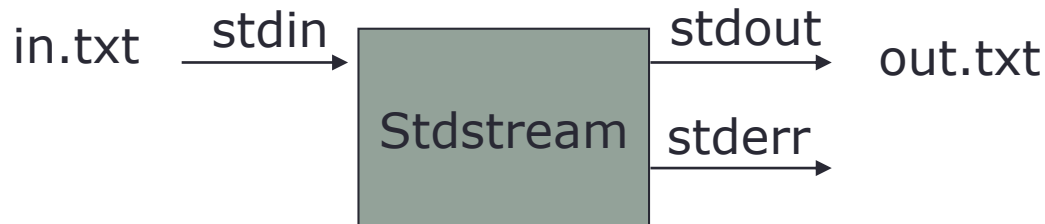
J:\Course\C programming\fall-2006\examples\chapter09>



Command line> Pipestream | Stdstream



Command line> Stdstream < in.txt > out.txt



printf函數

- printf是最常用來輸出資料的函數。

範例

```
int i, j;
```

```
float x,y;
```

```
printf("i=%d, j=%d, x=%f, y=%f\n", i, j, x, y);
```

A diagram with red arrows and circles illustrating argument passing in the printf function call. Red circles are drawn around the format string "%d", "%d", "%f", and "%f" in the format string, and around the variables i, j, x, and y in the argument list. Red arrows point from each variable to its corresponding conversion specifier in the format string: from i to the first %d, from j to the second %d, from x to the first %f, and from y to the second %f.

- printf會輸出字串 "i=%d, j=%d, x=%f, y=%f\n", 並且根據字串內的轉換指引(conversion specification)插入對資料轉換後的字串。
- 轉換指引是由%開始，緊接著符號的是指引printf如何將資料由內部格式轉換成字元形式。

printf說明

- **printf**
 - 指令格式
 - `printf(format_string, other_arguments) ;`
 - *format_string*: 描述輸出字串與資料格式
 - *other_arguments*: 相對於*format_string*裡每個轉換指引的資料
 - 恰當設定輸出格式後可以辦到
 - 顯示數字位數,
 - 行對齊,
 - 靠左或右對齊,
 - 顯示一般字元,
 - 指數的格式,
 - 16進位數,
 - 固定欄寬與精確度

printf注意事項

- 對於printf參數，編譯器只能檢查第一個參數是否為字串。至於要輸出的資料與轉換指引間是否個數與型態相符，就不檢查了。

- `int printf(const char*, ...);`

範例

```
int i, j;
```

```
float x,y;
```

```
printf("%d %d\n", i);
```

```
printf("%f %d\n",i, x);
```

皆會輸出錯誤的資訊

```
printf("%d\n", i, j);
```

則只輸出i，j會被省略(why ??)

常用於printf的轉換指引

- 整數(integer)的轉換指引
 - %d
- 浮點數(floating-point number)的轉換指引
 - %e or %E指數形式表示
 - 150.3 is 1.503E+02 in exponential (E stands for exponent)
 - %f – 印出浮點數，小數點左邊至少印一位
 - %g (or G) – 類似f但是尾部的零不會印
 - 1.2300 印出 1.23
 - 當exponent<-4或>=精確度(內定為6位數)使用指數形式表示
- 字元轉換指引
 - %c
- 字串轉換指引
 - %s
- %%印出%符號

Data type	printf conversion specification	scanf conversion specification
<i>Integer types</i>		
unsigned long long int	%llu	%llu
long long int	%lld	%lld
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
unsigned short	%hu	%hu
short	%hd	%hd
char	%c	%c

Data type	printf conversion specification	scanf conversion specification
<i>Floating-point types</i>		
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f

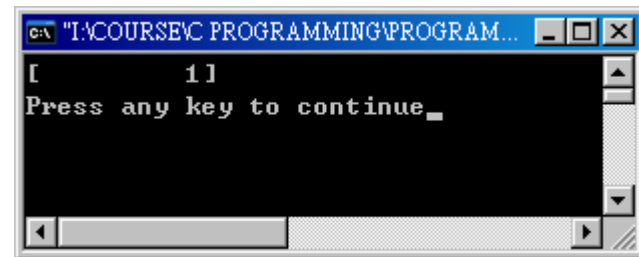
欄位寬度 (field width)

- 寫在%與轉換指引間設定資料印出的欄位寬

例如%**4d** – 此整數佔**4**個字元寬

- 欄位寬度若大於資料寬度則內定向右對齊
 - 欄位寬度若小於資料寬度則會以印出資料為原則，不受限欄寬
- (minus sign)**佔1個字元

```
#include<stdio.h>
int main(void)
{
    printf("[%10d]\n",1);
}
```

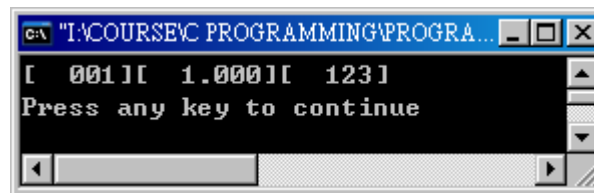


精確度(precision):輸出位數

精確度寫在這裡

- 格式: `% .3f`
- 這個欄位的意義隨資料型態而異
 - 整數 (default 1)
 - 最少印幾個數字，若不足前面位數補零
 - 浮點數
 - 小數點後印幾個數字
 - For `g` – maximum number of significant digits
 - 字串
 - 最多印幾個字元

```
#include <stdio.h>
int main(void)
{
    printf("[%5.3d][%7.3f][%5.3s]\n", 1, 1.0f,
        "123456");
}
```



欄位寬度與精確度同時寫

- 欄位寬度+精確度
 - %width.precision**
 - %5.3f**
 - %-5.3f (向左對齊)
 - %5.3f (向右對齊)
 - 精確度必須是正數
 - 可用變數來設定欄寬與精確度。放*在欄位寬度或精確度的地方
 - Example:

```
printf( "%*.*f", 7, 2, 98.736 );
```

欄位寬度

精確度

範例

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int i;
```

```
    float x;
```

```
    i = 40;
```

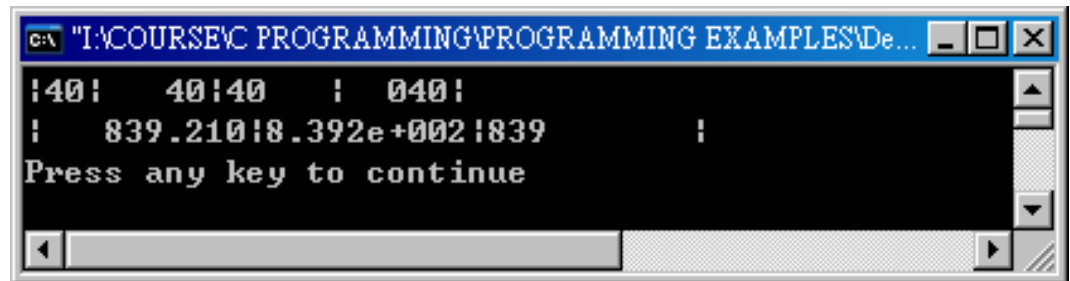
```
    x = 839.21f;
```

```
    printf("|%d|%5d|%-5d|%5.3d|\n", i, i, i, i);
```

```
    printf("|%10.3f|%10.3e|%-10.3g|\n", x, x, x);
```

```
    return 0;
```

```
}
```



```
C:\I\COURSE\C PROGRAMMING\PROGRAMMING EXAMPLES\De...  
|40| 40|40 | 040|  
| 839.210|8.392e+002|839 |  
Press any key to continue
```

Escape sequence

- 字串裡我們需要輸入一些字元，但是它可能是打不出來的(如換行)，或是會造成編譯器混淆(如 ")。這時就要用escape sequence。

例如要輸出"Hello"，指令是printf("\\"Hello\\")

- 常用escape sequence

Alert	\a
Backspace	\b
New line	\n
Horizontal tab	\t
"	\"
\	\\
'	\'

安全的使用printf

- 針對安全性考量，使用printf時要避免單一引數。
 - 例子一
 - `printf("Hello world\n");` //單一引數，改寫之。
 - `puts("Hello world");`
 - 例子二
 - `printf("Hello world");` //單一引數，改寫之。
 - `printf("%s","Hello world");`

scanf函數

- **scanf**是最常用來輸入資料的函數。

範例

```
int i, j;  
float x, y;  
scanf("%d%d%f%f", &i, &j, &x, &y);
```

- **scanf**
 - 格式化輸入
 - 能力
 - 輸入各型態資料
 - 輸入特定字元
 - 略過特定字元
- 指令格式
 - **scanf(format_string, other_arguments);**
 - **format_string**: 描述輸入資料格式
 - **other_arguments**: 欲存資料的位置
 - 可以用欄位寬度來指明輸入多少字元

常用於scanf的轉換指引

- 整數 %d
- 浮點數 %f (%e %g效果一樣)
- 字元 %c
- 字串 %s

Data type	printf conversion specification	scanf conversion specification
<i>Integer types</i>		
unsigned long long int	%llu	%llu
long long int	%lld	%lld
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
unsigned short	%hu	%hu
short	%hd	%hd
char	%c	%c

Data type	printf conversion specification	scanf conversion specification
<i>Floating-point types</i>		
long double	%Lf	%Lf
double	%f	%lf
float	%f	%f

scanf運作原理 (1)

輸入: 1-20.3-4.0e3

```
#include <stdio.h>
int main(void)
{
    int i,j;
    float x,y;
    scanf("%d%d%f%f",&i,&j,&x,&y);
    printf("%d %d %f %f\n",i,j,x,y);
    return 0;
}
```



```
C:\I:\COURSE\VC PROGRAMMING\PROGRAMMING EXAMPLE...
1-20.3-4.0e3
1 -20 0.300000 -4000.000000
Press any key to continue
```

- (1) 忽略空白, \t, 或換行符號至第一個符合%d的字元, 然後一直讀字元直到讀到違反%d的字元。放回此字元。將目前讀到的轉成整數存到i。
- (2) 忽略空白, \t, 或換行符號至第一個符合%d的字元, 然後一直讀字元直到讀到違反%d的字元。放回此字元。將目前讀到的轉成整數存到j。
- (3) 忽略空白, \t, 或換行符號至第一個符合%f的字元, 然後一直讀字元直到讀到違反%f的字元。放回此字元。將目前讀到的轉成浮點數存到x。
- (4) 忽略空白, \t, 或換行符號至第一個符合%f的字元, 然後一直讀字元直到讀到違反%f的字元。放回此字元。將目前讀到的轉成浮點數存到y。

輸入: ■1-20.3-4.0e3

```
#include <stdio.h>
int main(void)
{
    int i,j;
    float x,y;
    scanf("%d%d%f%f",&i,&j,&x,&y);
    printf("%d %d %f %f\n",i,j,x,y);
    return 0;
}
```

■表示空白

i: 1

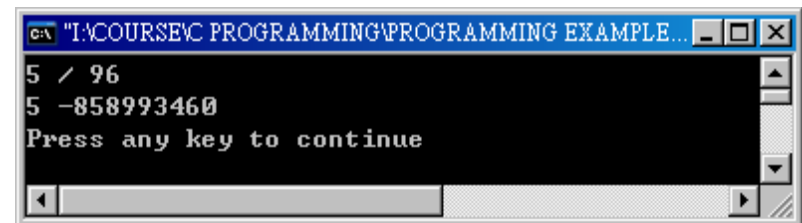
j: -20

x: .3

y: -4.0e3

The pattern-matching concept of scanf

- `scanf("%d/%d", &i, &j);`
 - 輸入: **5**/**96**
 - 表示空白
 - (1)忽略空白, `\t`, 或換行符號至第一個符合`%d`的字元, 然後一直讀字元直到讀到違反`%d`的字元`/`。放回此字元`/`。將目前讀到的轉成整數存到`i`。
 - (2)讀入字元`/`, 其符合`/`, `scanf`忽略它。
 - (3)忽略空白, `\t`, 或換行符號至第一個符合`%d`的字元, 然後一直讀字元直到讀到違反`%d`的字元。放回此字元。將目前讀到的轉成整數存到`j`。
 - 輸入: **5**/**96**
 - (1)忽略空白, `\t`, 或換行符號至第一個符合`%d`的字元, 然後一直讀字元直到讀到違反`%d`的字元。放回此字元。將目前讀到的轉成整數存到`i`。
 - (2)讀入字元, 不符合`/`, 放回此字元, `scanf`停止。下一個`scanf`會遇到/`96`
- `scanf("%d /%d", &i, &j);`
 - 輸入: **5**/**96**



```

C:\T\COURSE\C PROGRAMMING\PROGRAMMING EXAMPLE...
5 / 96
5 -858993460
Press any key to continue
  
```

scanf注意事項

- **format_string**裡除了轉換指引外，不要亂加字元，裡面的字元只與控制輸入格式有關，與輸出完全無關。
- 使用scanf的回傳值來判斷scanf是否成功讀入資料
 - `int result = scanf("%d%d",&num1,&num2);`
 - 只要成功讀入num1, num2，result值為2(成功讀入2項資料)。

Program: 兩分數相加

$$\frac{num1}{denom1} + \frac{num2}{denom2} = \frac{num1 * denom2 + num2 * denom1}{denom1 * denom2}$$

```
#include <stdio.h>

int main(void)
{
    int num1, denom1, num2, denom2, result_num, result_denom;

    printf("Enter first fraction: ");
    scanf("%d/%d", &num1, &denom1);

    printf("Enter second fraction: ");
    scanf("%d/%d", &num2, &denom2);

    result_num = num1 * denom2 + num2 * denom1;
    result_denom = denom1 * denom2;
    printf("The sum is %d/%d\n", result_num, result_denom);

    return 0;
}
```