

# 程式設計 (一)

## CH8. 字元與字串

Ming-Hung Wang 王銘宏

tonymhwang@cs.ccu.edu.tw

Department of Computer Science and Information Engineering  
National Chung Cheng University

Fall Semester, 2022

# 目錄

1. 字元與字串
2. 字元處理函式庫
3. 字串轉換函式
4. 字串處理函式庫
5. 標準輸入輸出函式庫
6. 格式化字串
  - 格式化輸出
  - 格式化輸入

# 字元與字串

## Strings and Characters

## 字元

字元包含控制字元與可顯示字元，可顯示字元又包含數字、英文字母與符號。

以單引號括起來的字元來表示字元常數 (character constant)。字元常數是一個 int 值，例如 'z' 代表了 z 的整數值 122，'\n' 代表了換行字元的整數值 10。

## 字串

字串 (string) 是視為單一個體的一連串字元。C 裡的字串常數 (string literal 或稱 string constant) 會寫在雙引號裡。

我們平常放在 printf 或 scanf 的第一個引數，以雙引號括住的，就是字串常數。

# 字元與字串

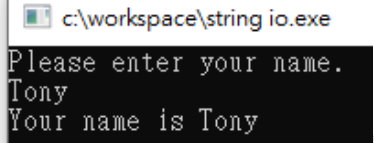
在 C 語言裡，可變的字串儲存在 char 陣列裡，而字串常數可使用 char\* 變數儲存其位址。並且，字串的最後會有一個**字串結尾符號'\0'**代表字串的尾端。

```
char s1[] = "Hello";  
// same as: char s1[] = {'H', 'e', 'l', 'l', 'o', '\0'};  
  
char *s2 = "World";  
// Constant string, use "const char *s2" is better
```

# 字元與字串

使用%s 格式指定字輸出字串，或讀入字串直到空白字元。scanf 時，字串不需要使用 & 取址，因為字串名稱本身就代表該字串的位址。

```
1  ✓ #include <stdio.h>
2  #include <stdlib.h>
3
4  ✓ int main()
5  {
6      const char *text = "Please enter your name.\n";
7      char name[100];
8      printf(text);
9      scanf("%s", name);
10     printf("Your name is %s\n", name);
11 }
```



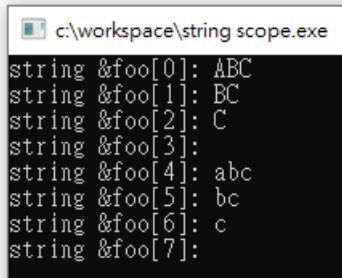
c:\workspace\string io.exe

Please enter your name.  
Tony  
Your name is Tony

# 字元與字串

C 語言中，字串是由 char 的位址表示，解讀範圍是由該位址直到第一個'\0' 為止。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char foo[] = {'A', 'B', 'C', '\0', 'a', 'b', 'c', '\0'};
7      char *p = foo, i;
8      for (i = 0; i < 8; i++)
9          printf("string &foo[%d]: %s\n", i, p + i);
10 }
```




```
c:\workspace\string scope.exe
string &foo[0]: ABC
string &foo[1]: BC
string &foo[2]: C
string &foo[3]:
string &foo[4]: abc
string &foo[5]: bc
string &foo[6]: c
string &foo[7]:
```



# 字元與字串

使用二維字元陣列作為字串陣列。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char classMember[][20] = {"Tony", "Kevin", "Jimmy"};
7      int i;
8      for (i = 0; i < 3; i++)
9          printf("%d. %s\n", i + 1, classMember[i]);
10 }
```


 c:\workspace\string array.exe

```
1. Tony
2. Kevin
3. Jimmy
```

# 字元與字串

使用字元指標陣列作為字串常數陣列。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      const char *classMember[] = {"Tony", "Kevin", "Jimmy"};
7      int i;
8      for (i = 0; i < 3; i++)
9          printf("%d. %s\n", i + 1, classMember[i]);
10 }
```

 c:\workspace\string array.exe

```
1. Tony
2. Kevin
3. Jimmy
```

# 字元處理函式庫

## Character Handling Library

# 字元處理函式庫

字元處理函式庫 `<ctype.h>` 提供了判斷字元類別與大小寫轉換的函式。每個函式都接收了一個無號字元 (表示為 `int`) 做為引數，並會回傳一個 `int` 值。

判斷字元類別的函式以 `is` 開頭，  
例如 `int isdigit(int c);`、`int isgraph(int c);`、`int isupper(int c);`

大小寫轉換的函式以 `to` 開頭，  
包含 `int toupper(int c);` 與 `int tolower(int c);`

# 字元處理函式庫

判斷字元類別的函式會回傳非 0 或 0 的值代表符合或不符合該類別。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4
5  int main()
6  {
7      char t;
8      printf("Input an alphabat: ");
9      scanf("%c", &t); // an alphabat
10     if (isalpha(t))
11     |   printf("Yes, you input an alphabat\n");
12     scanf("%c", &t); // new line (enter)
13     if (isspace(t))
14     |   printf("* Notice: %%c can read space character\n");
15 }
```

c:\workspace\char class.exe

```
Input an alphabat: A
Yes, you input an alphabat
* Notice: %%c can read space character
```

# 字元處理函式庫

判斷參數是否為控制字元、空白字元、可印字元、圖形字元：

ASCII	chars	isctrl	isblank	isspace	isgraph	isprint
0x00-0x08、 0x0E-0x1F、 0x7F	控制字元	V				
0x09	'\t' (tab)	V	V	V		
0x0A-0x0D	'\f' '\v' '\n' '\r'	V		V		
0x20	' ' (space)		V	V		V
0x21-0x7E	可視字元				V	V

# 字元處理函式庫

可視字元中，判斷符號、英數文字：

ASCII	chars	isgraph	ispunct	isalnum	isdigit	isalpha
0x21-0x2F、 0x3A-0x40、 0x5B-0x60、 0x7B-0x7E	!"#\$%&'()*+,-./ :;<=>?@ [\]^_` { }~	V	V			
0x30-0x39	0123456789	V		V	V	
0x41-0x5A 0x61-0x7A	ABC...Z abc...z	V		V		V
others						

# 字元處理函式庫

英數文字中，判斷 10 進位、16 進位數字、英文大小寫：


ASCII	chars	isdigit	isxdigit	isalpha	isupper	islower
0x30-0x39	0123456789	V	V			
0x41-0x46	ABCDEF		V	V	V	
0x47-0x5A	GHI...Z			V	V	
0x61-0x66	abcdef		V	V		V
0x67-0x7A	ghi...z			V		V
others						



# 字元處理函式庫

大小寫轉換的函式，當引數為英文字母時，會回傳該英文字母的大寫 (toupper) 或小寫 (tolower)，否則回傳與引數相同的數值。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4
5  int main()
6  {
7      int i;
8      char s[] = "I'm fine, thank you.";
9      for (i = 0; s[i] != '\0'; i++)
10         s[i] = toupper(s[i]);
11     printf("%s\n", s);
12 }
```

 c:\workspace\to upper.exe

I'M FINE, THANK YOU.

# 字串轉換函式

## String Conversion Functions

# 字串轉換函式

公用函式庫 `<stdlib.h>` 定義了字串轉數值的函式 `strtof`、`strtod`、`strtol`、`strtoll`、`strtoul`、`strtoull`，分別將字串轉換成不同數值型態。

```
double strtod (const char* str, char** endptr);
```

```
long int strtol (const char* str, char** endptr, int base);
```

函式的第一個參數 `str` 為欲轉換的字串，第二個參數 `endptr` 是字串指標，函式會以 `call by address` 的方式將 `endptr` 指向 `str` 中轉換完成的後一個字元。轉整數的函式有第三個參數 `base`，代表進位制。

# 字串轉換函式

strtod (string to double) 使用範例：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4  int main()
5  {
6      char text[] = "3 students get 90.3 on the test.";
7      char *ptr = text;
8      while (*ptr != '\0')
9      {
10         if (isdigit(*ptr))
11             printf("%f\n", strtod(ptr, &ptr));
12         else
13             ptr++;
14     }
15 }
```

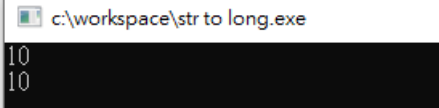
c:\workspace\str to double.exe

3.000000  
90.300000

# 字串轉換函式

字串轉整數的函式中，表進位制的引數可填入 0 或 2~36，填入 0 時會以前綴判斷 8、10、16 進位。strtol (string to long) 使用範例：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4  int main()
5  {
6      char text[] = "0xA is hex of 10.";
7      char *ptr = text;
8      while (*ptr != '\0')
9      {
10         if (isdigit(*ptr))
11             printf("%ld\n", strtol(ptr, &ptr, 0));
12         else
13             ptr++;
14     }
15 }
```



c:\workspace\str to long.exe

10  
10

# 字串轉換函式

若不需要取得轉換完成的後一個字元位址，且字串中欲轉換的是 10 進位的數字，可使用 `atof`、`atoi`、`atol`、`atoll`，分別將字串轉換成不同數值型態。

函式原型	呼叫	相當於
<code>double atof (const char* str);</code>	<code>atof(s)</code>	<code>strtod(s, NULL)</code>
<code>int atoi (const char* str);</code>	<code>atoi(s)</code>	<code>(int)strtol(s, NULL, 10)</code>
<code>long int atol ( const char* str);</code>	<code>atol(s)</code>	<code>strtol(s, NULL, 10)</code>
<code>long long int atoll ( const char* str);</code>	<code>atoll(s)</code>	<code>strtoll(s, NULL, 10)</code>

# 字串處理函式庫

## String Handling Library

# 字串處理函式庫

字串處理函式庫 `<string.h>` 包含字串複製、比較、搜尋等函式。以下為本節會介紹的函式：

字串長度	<code>strlen</code>	搜尋字元	<code>strchr</code> 、 <code>str<sup>r</sup>chr</code>
字串複製	<code>strcpy</code> 、 <code>str<sup>n</sup>cpy</code>	搜尋子字串	<code>strstr</code>
字串串接	<code>strcat</code> 、 <code>str<sup>n</sup>cat</code>	搜尋字元集	<code>strpbrk</code> 、 <code>strspn</code> 、 <code>str<sup>c</sup>spn</code>
字串比較	<code>strcmp</code> 、 <code>str<sup>n</sup>cmp</code>	字串分割	<code>strtok</code>

另外，`<string.h>` 也包含記憶體 (位元組陣列) 相關的操作，本章會介紹 `memset`。




# 字串處理函式庫

```
size_t strlen(const char* str);
```

strlen 會回傳 str 的字串長度 (不包含'\0')。型態 size\_t 是一種無號整數型態，於 64 位元電腦中大小為 8bytes，32 位元電腦中為 4bytes。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  int main()
5  {
6      char text[100] = "A have a pen.";
7      printf("'s' has %d words\n", text, strlen(text));
8  }
```

 c:\workspace\string length.exe

'A have a pen.' has 13 words

# 字串處理函式庫

```
char* strcpy(char* dest, const char* source);  
char* strncpy(char* dest, const char* source, size_t num);
```

將 source 的字串複製到 dest。使用 strncpy 時最多複製 num 個字元，如果 source 的長度大於 num，則不會在 dest 最後加上'\0'。

```
1  #include <stdio.h>  
2  #include <string.h>  
3  int main()  
4  {  
5      char foo[] = "Mississippi River";  
6      char bar[100] = "xxxxxxxxxxxxxxxxxxxxxx";  
7      strncpy(bar, foo, 5);  
8      puts(bar);  
9      strcpy(bar, foo);  
10     puts(bar);  
11 }
```

c:\workspace\string copy.exe

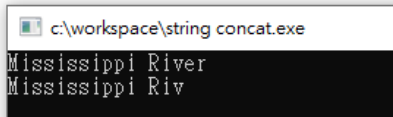
Missixxxxxxxxxxxxxxxxxx  
Mississippi River

# 字串處理函式庫

```
char* strcat(char* dest, const char* source);  
char* strncat(char* dest, const char* source, size_t num);
```

將 source 的字串複製到 dest 的字串尾端，並在尾端加上'\0'。使用 strncat 時最多複製 num 個字元。

```
1  #include <stdio.h>  
2  #include <string.h>  
3  int main()  
4  {  
5      char foo[100] = "Mississippi ";  
6      char bar[100] = "Mississippi ";  
7      char cat[] = "River";  
8      strcat(foo, cat);  
9      strncat(bar, cat, 3);  
10     puts(foo);  
11     puts(bar);  
12 }
```



c:\workspace\string\_concat.exe

Mississippi River  
Mississippi Riv

# 字串處理函式庫

```
int strcmp(const char* str1, const char* str2 );  
int strncmp(const char* str1, const char* str2, size_t num );
```

由 index 小到大依序比對 str1 與 str2 的每個字元，出現第一個不相同的字元時：

- 若 str1 的字元較小則回傳負數
- 若 str1 的字元較大則回傳正數

若 str1 與 str2 相同則回傳 0。使用 strncmp 時最多比較 num 個字元。

# 字串處理函式庫

## strcmp 使用範例：

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  char cmp(char *s1, char *s2)
6  {
7      int result = strcmp(s1, s2);
8      if (result > 0) return '>';
9      else if (result < 0) return '<';
10     return '=';
11 }
12
```

```
13 int main()
14 {
15     char a[] = "ABCABC123";
16     char b[] = "ABCAbc123";
17     char c[] = "ABCABC";
18     printf("%s %c %s\n", a, cmp(a, b), b);
19     printf("%s %c %s\n", a, cmp(a, c), c);
20 }
```

c:\workspace\string compare.exe

ABCABC123 < ABCAbc123

ABCABC123 > ABCABC

# 字串處理函式庫

```
char* strchr(const char *str, int character);  
char* strrchr(const char *str, int character);  
char* strstr(const char *str1, const char* str2);
```

strchr 在 str 中尋找第一次出現的 character 字元。

str<sup>r</sup>chr 在 str 中尋找最後一次出現的 character 字元。


strstr 在 str1 中尋找第一次出現的 str2 子字串。

若有搜尋成功，則會回傳找到的字元的指標 (strstr 會回傳子字串開頭的指標)。若沒有搜尋到目標，則回傳 NULL。

# 字串處理函式庫

strchr、strrchr、strstr 會回傳符合的字元的指標，減去字串開頭的指標可得到符合的字元的 index。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int main()
6  {
7      char foo[] = "Mississippi River";
8      char search = 'i', subStr[] = "ppi";
9      printf("The first '%c' in '%s' is at index %d.\n", search, foo, strchr(foo, search) - foo);
10     printf("The last '%c' in '%s' is at index %d.\n", search, foo, strrchr(foo, search) - foo);
11     printf("The first '%s' in '%s' is start at index %d.\n", subStr, foo, strstr(foo, subStr) - foo);
12 }
```

 c:\workspace\string search.exe

```
The first 'i' in 'Mississippi River' is at index 1.
The last 'i' in 'Mississippi River' is at index 13.
The first 'ppi' in 'Mississippi River' is start at index 8.
```

```
char* strpbrk(const char *str1, const char* str2);
```

strpbrk 是 string pointer break 的縮寫，會在 str1 中尋找第一次出現屬於 str2 字串中的字元 (此時 str2 可稱為字元集)。

若有搜尋成功，則會回傳找到的字元的指標。若沒有搜尋到目標，則回傳 NULL。



# 字串處理函式庫

strpbrk 會回傳符合的字元的指標，減去字串開頭的指標可得到符合的字元的 index。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int main()
6  {
7      char foo[] = "Mississippi River", *ptr = foo - 1;
8      printf("Index of vowels in '%s': ", foo);
9      while (ptr = strpbrk(ptr + 1, "AEIOUaeiou"))
10         printf("%d ", ptr - foo);
11         puts(""); // print new line
12 }
```

c:\workspace\string ptr break.exe

Index of vowels in 'Mississippi River': 1 4 7 10 13 15

# 字串處理函式庫

```
size_t strspn(const char* str1, const char* str2);  
size_t strcspn(const char* str1, const char* str2);
```

strspn 是 string span 的縮寫，span 為持續範圍的意思。

strspn 會計算從 str1 開頭，屬於 str2 字元集的連續長度。

strcspn 會計算從 str1 開頭，**不**屬於 str2 字元集的連續長度。

# 字串處理函式庫

## strspn 與 strcspn 使用範例：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int main()
6  {
7      char foo[] = "Phone number 0932123456. Don't forget it.";
8      char tmp[100] = "";
9      int start = strcspn(foo, "0123456789"); // count chars not belong to 0~9
10     int len = strspn(foo + start, "0123456789"); // count chars belong to 0~9
11     strncpy(tmp, foo + start, len);
12     printf("Start index: %d\nLength: %d\nNumber: %s\n",
13           start, len, tmp);
14 }
```

c:\workspace\string span.exe

Start index: 13  
Length: 10  
Number: 0932123456

# 字串處理函式庫

```
char* strtok(char* str, const char* delimiters);
```

strtok 是 string token 的縮寫，將 str 中屬於 delimiters 字元集的字元切割、拆分為 tokens。

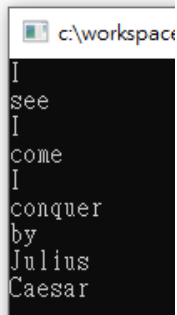
在此函式第一次呼叫時，參數 str 作為切割字串的起始位置。隨後的呼叫時，str 為 NULL 表示從上一次切割的結尾繼續切割。strtok 的回傳值是每個拆分後的 token 的第一個字元位址。

注意此函式對於 str 是破壞性的，會把 str 中屬於 delimiters 的元素設為 '\0'。

# 字串處理函式庫

## strtok 使用範例：

```
1  √ #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  √ int main()
6  {
7      char text[] = "\"I see, I come, I conquer.\" by Julius Caesar.";
8      char *token[100] = {NULL};
9      int i, tokenCount = 0;
10     token[0] = strtok(text, " ,.?!\"'");
11     while (token[++tokenCount] = strtok(NULL, " ,.?!\"'"));
12     for (i = 0; i < tokenCount; i++)
13         puts(token[i]);
14 }
```



```
c:\workspace
I
see
I
come
I
conquer
by
Julius
Caesar
```

```
void* memset(void* ptr, int value, size_t num);
```

memset 是記憶體操作的函式，會將任意型態的指標 ptr 往後 num 個 bytes，每個 bytes 的值設為 value。並回傳 ptr。

此函式可傳入所有型態的陣列，且不會以陣列元素大小為單位設定值，而是一律以 byte 為單位。此函式常拿來將陣列歸零。

# 字串處理函式庫

使用 `memset` 將陣列所佔的記憶體歸零。

```
1  #include <string.h>
2
3  int main()
4  {
5      int foo[100];
6      char bar[200];
7      memset(foo, 0, sizeof(int) * 100); // Zero the array foo
8      memset(bar, 0, sizeof(char) * 200); // Zero the array bar
9  }
```

# 標準輸入輸出函式庫

## Standard I/O Library



# 標準輸入輸出函式庫

標準輸入輸出函式庫 `<stdio.h>` 提供了標準串流、字串、檔案的讀寫。本章著重於標準串流與字串作為目標的字元字串讀寫，將介紹以下函式：

- 字串輸入輸出： `gets`、 `puts`
- 字元輸入輸出： `getchar`、 `putchar`
- 格式化輸入輸出至字串： `sscanf`、 `sprintf`

# 標準輸入輸出函式庫

```
char* gets(char* str);  
int puts(const char* str);
```

`gets` 會由標準輸入流 (`stdin`) 讀入字串到位址 `str`，直到 `'\n'` 或直到 EOF。若讀取成功則回傳 `str`，若讀取開始時及遇到 EOF，並無讀取到任何文字則回傳 `NULL` 且不更動位址 `str` 的內容。

`puts` 會輸出字串 `str` 至標準輸出流 (`stdout`)，並在最後輸出一個換行。

# 標準輸入輸出函式庫

因為 `gets` 並無法預測讀取的長度，具有安全上的危險性，在 C11 與 C++14 版本之後將被移除，可使用 `fgets` 代替。


```
char* fgets(char* str, int num, FILE* stream);
```

`gets` 會讀入字串到位址 `str`，直到 `'\n'`、EOF 或直到讀取了 `num-1` 個字（-1 是為了保留 `'\0'`）。`stream` 為讀取的資料來源，標準輸入流為 `stdin`。

# 標準輸入輸出函式庫

gets、puts 使用範例：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char input[100];
7      printf("Input a sentence: ");
8      gets(input); // better : fgets(input, 100, stdin);
9      puts(input);
10 }
```

 c:\workspace\gets puts.exe

```
Input a sentence: I think, therefore I am
I think, therefore I am
```

# 標準輸入輸出函式庫

```
int getchar();  
int putchar(int character);
```

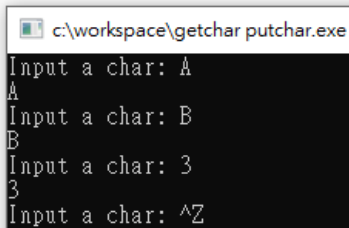
`getchar` 會從 `stdin` 讀取一個字元，並回傳該字元的 ASCII 碼。若讀取至 EOF，則會回傳 EOF(值為-1)。

`putchar` 會輸出一個字元 `character` 至 `stdout`，並回傳輸出的字元。

# 標準輸入輸出函式庫

## getchar、putchar 使用範例：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char input;
7      do
8      {
9          printf("Input a char: ");
10         input = getchar(); // read a char
11         getchar(); // read '\n'
12         if (input != EOF)
13             putchar(input);
14         puts(""); // new line
15     } while (input != EOF);
16 }
```



The screenshot shows a command prompt window with the title "c:\workspace\getchar putchar.exe". The output of the program is as follows:

```
Input a char: A
A
Input a char: B
B
Input a char: 3
3
Input a char: ^Z
```

# 標準輸入輸出函式庫

```
int sscanf(const char* str, const char* format, ...);  
int sprintf(char* str, const char* format, ... );
```


sscanf、sprintf 是格式化輸入輸出函式，與 scanf、printf 的差別在於，sscanf、sprintf 的輸入輸出目標是字串 str，而非標準串流 (stdin、stdout)。

若使用 sscanf 重複讀取同一個字元陣列，每一次讀取都會從頭開始讀，並不會繼承上一次的結束位置。sprintf 每一次輸出會覆蓋原本的字串。

# 標準輸入輸出函式庫

sscanf、sprintf 使用範例：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char text[100];
7      int num;
8      sprintf(text, "%#X is a lucky number.", 777);
9      puts(text);
10     sscanf(text, "%i", &num);
11     printf("Read a number %d\n", num);
12     puts("");system("PAUSE");
13 }
```

 c:\workspace\str scan print.exe

```
0X309 is a lucky number.
Read a number 777
```



# 格式化字符串

## Format String

# 格式化字串

格式化字串 (format string) 是 scanf、printf、sscanf、ssprintf 等格式化輸入輸出函式的引數。格式化字串中，以 % 百分比符號開頭的字串稱為格式指定詞 (format specifier)。

格式指定詞於 printf、ssprintf 等函式的格式化字串引數中，指定數值的資料型態以及輸出時的格式。於 scanf、sscanf 的格式化字串引數中，指定輸入的格式以及儲存至記憶體體的資料型態 (攸關記憶體大小)。

## 格式化輸出


格式化輸出的格式指定詞結構如下：

```
%[flags][width][.precision][length]specifier
```

包含最一開始的%、旗標 (flags)、欄位寬度 (field width)、精確度 (precision)、長度符號 (length specifier) 和轉換符號 (conversion specifier)，其中除了% 與轉換符號以外都是可選擇省略的。

# 格式化字串：格式化輸出

以下為格式化輸出中的轉換符號 (conversion specifier)

轉換符號	意義	範例
d	有號整數，10 進位	110
u	無號整數，10 進位	110
o	無號整數，8 進位	156
x、X	無號整數，16 進位 (大、小寫)	6e、6E
i	有號整數，同 d	
f	浮點數，小數格式	0.000314
e、E	浮點數，科學記號格式 (大、小寫)	3.141592e-004
g、G	浮點數，由數值的科學記號指數部分大小決定以 f 或 e、E 表示	
c	字元	n
s	字串	Hello
p	記憶體位址，16 進位	
%	百分比符號	%

# 格式化字串：格式化輸出

長度符號 (length specifier) 指的是轉換符號前的 hh、h、l、ll，搭配轉換符號決定引數的型態。

	<b>d, i</b>	<b>u, o, x, X</b>	<b>f, e, E, g, G</b>	<b>c</b>	<b>s</b>	<b>p</b>
<b>hh</b>	char	unsigned char				
<b>h</b>	short	unsigned short				
	int	unsigned int	double	char	char*	void*
<b>l</b>	long	unsigned long				
<b>ll</b>	long long	unsigned long long				

# 格式化字串：格式化輸出

`%[flags][width][.precision][length]specifier`

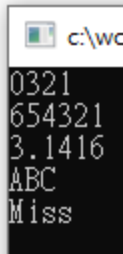
精確度 (precision) 以點 (.) 後帶有一個非負整數組成，於不同型態有不同效果：

- 整數 (d, i, u, o, x, X)：指定數值的最少位數，不足則在前方補 0，預設為 0。
- 浮點數 (f, e, E)：指定小數點後的顯示位數，預設為 6。
- 字串 (s)：指定要印出的最大字數，預設為全部印出。

# 格式化字串：格式化輸出

精確度為 4 時，輸出整數、浮點數、字串的示範：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("%.4d\n", 321); // shorter than 4
7      printf("%.4d\n", 654321); // longer than 4
8      printf("%.4f\n", 3.14159365);
9      printf("%.4s\n", "ABC"); // shorter than 4
10     printf("%.4s\n", "Mississippi"); // longer than 4
11 }
```



```
c:\wc
0321
654321
3.1416
ABC
Miss
```

# 格式化字串：格式化輸出

`%[flags][width][.precision][length]specifier`

欄位寬度 (field width) 代表要印出的最小字元數。如果要印出的內容比這個數字短，則會使用空格在左方填充。即使結果較大，該值也不會被截斷。

欄位寬度可搭配旗標 (flags) 變化：


- 旗標 `-`：如果要印出的內容比欄位寬度短，則會使用空格在右方填充。
- 旗標 `0`：如果要印出的內容比欄位寬度短，則會在左方補 `0`。當整數格式使用精確度時，旗標 `0` 是無效的。



# 格式化字串：格式化輸出

欄位寬度為 6 時，輸出整數、浮點數、字串的示範：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("|%6d|%6.2f|%6s|\n", -123, 3.14, "ABC");
7      printf("|%-6d|%-6.2f|%-6s|\n", -123, 3.14, "ABC");
8      printf("|%06d|%06.2f|%06s|\n", -123, 3.14, "ABC");
9  }
```

 c:\workspace\format field width.exe

```
| -123|  3.14|   ABC|
|-123 | 3.14 |ABC  |
|-00123|003.14|000ABC|
```

# 格式化字串：格式化輸出

`%[flags][width][.precision][length]specifier`

旗標 (flags) 除了影響欄位寬度效果外，還可決定整負號的顯示以及顯示數值類別的特徵。

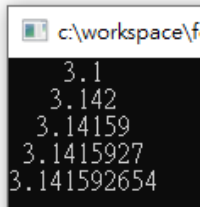
旗標	說明
—	使用空格在右方填充直到長度到達欄位寬度。
0	在左方補 0 直到長度到達欄位寬度。
+	在數值不為負數時，在值之前顯示 + 號。
(空格)	在數值不為負數時，在值之前插入空格。
#	8、16 進位整數時顯示前綴 0、0x、0X， 浮點數格式強制顯示小數點。

# 格式化字串：格式化輸出

## 星號

欄位寬度與欄位寬度的數值可使用星號 (\*) 代替，由引數提供數值。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int i;
7      double foo = 3.14159265358979;
8      for(i = 1; i <= 5; i++)
9          printf("%*.*f\n", 6 + i, i * 2 - 1, foo);
10 }
```



```
c:\workspace\f
3.1
3.142
3.14159
3.1415927
3.141592654
```

# 格式化字串：格式化輸出

## 思考

不依靠電腦寫出下列敘述句的輸出：

1. `printf("%+—6.3f", 3.14159);`
2. `printf("%#08x", 32);`

## 格式化輸入

格式化輸入的格式指定詞結構如下：

```
%[*][width][length]specifier
```

包含最一開始的%、星號 (\*)、欄位寬度 (field width)、長度符號 (length specifier) 和轉換符號 (conversion specifier)，其中除了% 與轉換符號以外都是可選擇省略的。

# 格式化字串：格式化輸入

以下為格式化輸入中的轉換符號 (conversion specifier)

轉換符號	意義	接受字元
d	10 進位整數	可選以 +、- 開頭，接受 0-9
u	10 進位無號整數	可選以 +、- 開頭，接受 0-9
o	8 進位無號整數	可選以 +、- 開頭，接受 0-7
x、X	16 進位無號整數	可選以 +、- 開頭，接受 0-9、A-F、a-f
i	整數	可選以 +、- 開頭，預設接受 0 到 9，前綴 0 則接受 0-7，前綴 0x 或 0X 則接受 0-9、A-F、a-f
f、e、E g、G	浮點數	可選以 +、- 開頭，接受 0-9 並可選包含 1 個小數點，在其後方可選加入 E 或 e 與十進位整數
c	字元	任意字元
s	字串	任意非空白的連續字元
[ ]、[ ^ ]	掃描集、反掃描集	屬於/不屬於字元集中的連續字元
%	百分比符號	%

# 格式化字串：格式化輸入

長度符號 (length specifier) hh、h、l、ll，搭配轉換符號決定引數的型態。

	<b>d, i</b>	<b>u, o, x, X</b>	<b>f, e, E, g, G</b>	<b>c, s, [ ], [^]</b>
<b>hh</b>	char*	unsigned char*		
<b>h</b>	short*	unsigned short*		
	int*	unsigned int*	float*	char*
<b>l</b>	long*	unsigned long*	double*	
<b>ll</b>	long long*	unsigned long long*		

## 掃描集、反掃描集

掃描集 [ ] 可以指定讀取的字串可包含的字元，將讀取字元直到遇到非指定的字元。反掃描集 [^] 可以指定讀取的字串不可包含的字元。

指定的字元由 [ ] 或 [^] 包住，並由 “-” 代表連續：


- %[aeiou] 讀取由 'a'、'e'、'i'、'o'、'u' 組成的字串。
- %[ ^aeiou] 讀取不包含 'a'、'e'、'i'、'o'、'u' 的字串。
- %[0-9A-Za-z] 讀取由英文字母與數字組成的字串。
- %[ ^-+\*/% ] 讀取不可包含 '-'、'+'、'\*'、'/'、'%' 的字串。  
(若掃描集內需指定字元 '-', 須放在最前面。)



# 格式化字串：格式化輸入

## 掃描集、反掃描集範例：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char url[] = "youtu.be/mSCeOuyQEQk?t=20";
7      char domain[100], path[100];
8      sscanf(url, "%[^/]/%[0-9A-Za-z]", domain, path);
9      printf("%s\ndomain: %s\npath  : %s\n", url, domain, path);
10 }
```

 c:\workspace\format\_scanset.exe

```
youtu.be/mSCeOuyQEQk?t=20
domain: youtu.be
path  : mSCeOuyQEQk
```

# 格式化字串：格式化輸入

`%[*][width][length]specifier`

格式化輸入的欄位寬度表示讀入的最大字元數。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      int foo, bar;
7      scanf("%1d%1d", &foo, &bar);
8      printf("foo = %d\nbar = %d\n", foo, bar);
9  }
```

c:\workspace\format input width.exe

```
34567
foo = 3
bar = 4
```

# 格式化字串：格式化輸入

如果將%c 指定欄位寬度則會讀入指定長度的字元陣列。

%c 與%s 的差別在於，%c 預設讀入 1 個字元且不會在結尾加上'\0'，  
%s 會讀入字元直到空白字元就不會繼續讀取，且會在結尾加上'\0'。

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      char text[] = "Hello, how are you?";
7      char foo[] = "XXXXXXXXXXXXXXXXXXXXXXXXX";
8      char bar[] = "XXXXXXXXXXXXXXXXXXXXXXXXX";
9      sscanf(text, "%10s", foo);
10     sscanf(text, "%10c", bar);
11     puts(foo);
12     puts(bar);
13 }
```

c:\workspace\format input char array.exe

Hello,  
Hello, howXXXXXXXXXXXXXXXXX

# 格式化字串：格式化輸入

`%[*][width][length]specifier`

格式化輸入的星號 (\*) 代表讀取但是不儲存。

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char foo[] = "Phone number 0932123456. Don't forget it.";
6      char num[100] = "";
7      sscanf(foo, "%*[^0-9]%[0-9]", num);
8      printf("Number: %s\n", num);
9  }
```

c:\workspace\format input star.exe

Number: 0932123456