

## 程式設計第8章



真理大學資工系 洪麗玲  
llhung@mail.au.edu.tw

### 8.4 字串轉換函式



**字串轉換函式 (string conversion function)** 可以由數字所組成的字串轉換成整數或浮點數值。圖8.5列出所有的字串轉換函式。

函式原型	函式的描述
<code>double strtod( const char *nPtr, char **endPtr );</code>	將字串 nPtr 轉換成 double。
<code>long strtol( const char *nPtr, char **endPtr, int base );</code>	將字串 nPtr 轉換成 long。
<code>unsigned long strtoul( const char *nPtr, char **endPtr, int base );</code>	將字串 nPtr 轉換成 unsigned long。

圖8.5 一般公用函式庫當中的字串轉換函式



### ❖ 8.4.1 函式 `strtod`

- 圖8.6的`strtod`函式會將代表浮點數的一連串字元轉換為`double`。

### ❖ 8.4.2 函式 `strtol`

- 圖8.7的`strtol`函式會從一個字串找出代表一個整數的字元，然後將這些字元轉換為`long int`。

### ❖ 8.4.3 函式 `strtoul`

- 圖8.8的`strtoul`函式會從一個字串中找出代表一個`unsigned long`整數的字元，然後將這些字元轉換成`unsigned long`整數值。

### ❖ Fig8-6



```

1 // Fig. 8.6: fig08_06.c
2 // Using function strtod
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int main( void )
7 {
8     // initialize string pointer
9     const char *string = "51.2% are admitted"; // initialize string
10
11     double d; // variable to hold converted sequence
12     char *stringPtr; // create char pointer
13
14     d = strtod( string, &stringPtr );
15
16     printf( "The string \"%s\" is converted to the\n", string );
17     printf( "double value %.2f and the string \"%s\"\\n", d, stringPtr );
18 } // end main

```

The string "51.2% are admitted" is converted to the double value 51.20 and the string "% are admitted"

圖8.6 使用函式`strtod`



## 8.5 標準輸入／輸出函式庫函式



- 圖8.9列出標準輸入／輸出函式庫 (`<stdio.h>`) 中，關於字元和字串輸入／輸出的函式。

函式原型	函式的描述
<code>int getchar( void );</code>	從標準輸入讀進下一個字元，並以整數值傳回。
<code>char *fgets( char *s, int n, FILE *stream);</code>	從指定的串流持續讀進字元到陣列 <code>s</code> 中，直到出現 <code>newline</code> 或 <code>end-of-file</code> 字元，或是直到讀入 <code>n-1</code> 個字元為止。在本章中，我們指定串流為 <code>stdin</code> ——標準輸入串流，通常用來從鍵盤讀入字元。陣列的最後會附加上結束的 <code>null</code> 字元。將讀入 <code>s</code> 的字串傳回。

圖8.9 標準輸入／輸出函式庫的字元和字串函式(1/2)

WATSE

函式原型	函式的描述
<code>int putchar( int c );</code>	印出存放在 <code>c</code> 裡的字元，並將此字元以整數傳回。
<code>int puts( const char *s );</code>	印出字串 <code>s</code> 並且後面跟著一個換行字元。假如成功，則傳回一個非零的整數，假如發生錯誤，則傳回 <code>EOF</code> 。
<code>int sprintf( char *s, const char *format, ... );</code>	和 <code>printf</code> 相同，不過輸出是放到陣列 <code>s</code> 而不是印到螢幕上。傳回寫入 <code>s</code> 中的字元數量，假如錯誤發生，則傳回 <code>EOF</code> 。[注意：在本章安全程式設計章節及附錄F中，我們將討論更多有關 <code>snprintf</code> 及 <code>snprintf_s</code> 的安全相關函數。]
<code>int sscanf( char *s, const char *format, ... );</code>	和 <code>scanf</code> 相同，不過輸入是從陣列 <code>s</code> 讀進而不是鍵盤。傳回函數成功讀入之項目的數量，假如發生錯誤，則傳回 <code>EOF</code> 。

圖8.9 標準輸入／輸出函式庫的字元和字串函式(2/2)

WATSE



### ❖ 8.5.1 函式fgets和putchar

- 圖8.10的程式使用**fgets**和**putchar**函式從標準輸入 (鍵盤) 讀進一行文字，然後以遞迴的方式，反向地輸出這行文字。

### ❖ Fig8.10

WATSE

```

1 // Fig. 8.10: fig08_10.c
2 // Using functions fgets and putchar
3 #include <stdio.h>
4 #define SIZE 80
5
6 void reverse( const char * const sPtr ); // prototype
7
8 int main( void )
9 {
10     char sentence[ SIZE ]; // create char array
11
12     puts( "Enter a line of text:" );
13
14     // use fgets to read line of text
15     fgets( sentence, SIZE, stdin );
16
17     puts( "\nThe line printed backward is:" );
18     reverse( sentence );
19 } // end main
20
21 // recursively outputs characters in string in reverse order
22 void reverse( const char * const sPtr )
23 {

```

圖8.10 函式fgets和putchar的使用方式(1/2)

WATSE

```

24 // if end of the string
25 if ( '\0' == sPtr[ 0 ] ) { // base case
26     return;
27 } // end if
28 else { // if not end of the string
29     reverse( &sPtr[ 1 ] ); // recursion step
30     putchar( sPtr[ 0 ] ); // use putchar to display character
31 } // end else
32 } // end function reverse

```

Enter a line of text:  
 Characters and Strings  
 The line printed backward is:  
 sgnirtS dna sretcarahC

Enter a line of text:  
 able was I ere I saw elba  
 The line printed backward is:  
 able was I ere I saw elba

圖8.10 函式fgets和putchar的使用方式(2/2)



### ❖ 8.5.2 函式getchar

- 圖8.11的程式使用**getchar**和**puts**函式，從標準輸入讀取一些字元到字元陣列**sentence**中，然後將此字元陣列以字串印出。

```

1 // Fig. 8.11: fig08_11.c
2 // Using function getchar.
3 #include <stdio.h>
4 #define SIZE 80
5
6 int main( void )
7 {
8     int c; // variable to hold character input by user
9     char sentence[ SIZE ]; // create char array
10    int i = 0; // initialize counter i
11

```

圖8.11 使用函式getchar (1/2)



```

12 // prompt user to enter line of text
13 puts( "Enter a line of text:" );
14
15 // use getchar to read each character
16 while ( i < SIZE - 1 && ( c = getchar() ) != '\n' ) {
17     sentence[ i++ ] = c;
18 } // end while
19
20 sentence[ i ] = '\0'; // terminate string
21
22 // use puts to display sentence
23 puts( "\nThe line entered was:" );
24 puts( sentence );
25 } // end main

```

```

Enter a line of text:
This is a test.

The line entered was:
This is a test.

```

圖8.11 使用函式getchar (2/2)



## 8.6 字串處理函式庫的字串操作函式



- 字串處理函式庫 (`<string h>`) 提供了許多函式，可以操作字串資料 (**複製字串copy strings**) 以及 **連接字串concatenating string**、**比較兩個字串 (comparing strings)**、搜尋字串當中的某些字元或其他字串、**將字串字符化 (tokenizing strings)** (將字串分割為邏輯片段)、以及**計算字串長度 (determining the length of strings)**。這些函式列在圖8.14中。



函式原型	函式的描述
<code>char *strcpy( char *s1, const char *s2 )</code>	將字串 s2 複製至陣列 s1。並傳回 s1。
<code>char *strncpy( char *s1, const char *s2, size_t n )</code>	將字串 s2 的最多 n 個字元複製至陣列 s1。並傳回 s1。
<code>char *strcat( char *s1, const char *s2 )</code>	將字串 s2 接到陣列 s1 的尾端。s2 的第一個字元會覆寫 s1 的結束字元。並傳回 s1。
<code>char *strncat( char *s1, const char *s2, size_t n )</code>	將字串 s2 的最多 n 個字元接到陣列 s1 的尾端。s2 的第一個字元會覆寫 s1 的結束字元。並傳回 s1。

圖8.14 字串處理函式庫的字串操作函式



### ❖ 8.6.1 函式strcpy和strncpy

- 圖8.15的程式使用了**strcpy**將陣列**x**中整個字串複製給陣列**y**，以及使用**strncpy**將陣列**x**的前14個字元複製到陣列**z**。

```

1 // Fig. 8.15: fig08_15.c
2 // Using functions strcpy and strncpy
3 #include <stdio.h>
4 #include <string.h>
5 #define SIZE1 25
6 #define SIZE2 15
7
8 int main( void )
9 {

```

圖8.15 函式strcpy和strncpy的使用(1/2)



```

10 char x[] = "Happy Birthday to You"; // initialize char array x
11 char y[ SIZE1 ]; // create char array y
12 char z[ SIZE2 ]; // create char array z
13
14 // copy contents of x into y
15 printf( "%s%s\n%s%s\n",
16         "The string in array x is: ", x,
17         "The string in array y is: ", strcpy( y, x ) );
18
19 // copy first 14 characters of x into z. Does not copy null
20 // character
21 strncpy( z, x, SIZE2 - 1 );
22
23 z[ SIZE2 - 1 ] = '\0'; // terminate string in z
24 printf( "The string in array z is: %s\n", z );
25 } // end main

```

```

The string in array x is: Happy Birthday to You
The string in array y is: Happy Birthday to You
The string in array z is: Happy Birthday

```

圖8.15 函式strcpy和strncpy的使用(2/2)

WATSE



### ❖ 8.6.2 函式strcat和strncat

- 函式**strcat**會將它的第二個引數 (字串) 串接到第一個引數 (含有一個字串的字元陣列)。函式**strncat**會將第二個字串中某特定數量的字元串接到第一個字串。圖8.16的程式示範了**strcat**和**strncat**函式的使用方法。

```

1 // Fig. 8.16: fig08_16.c
2 // Using functions strcat and strncat
3 #include <stdio.h>
4 #include <string.h>
5

```

圖8.16 strcat和strncat的使用方式(1/2)

WATSE



```

6  int main( void )
7  {
8      char s1[ 20 ] = "Happy "; // initialize char array s1
9      char s2[] = "New Year "; // initialize char array s2
10     char s3[ 40 ] = ""; // initialize char array s3 to empty
11
12     printf( "s1 = %s\ns2 = %s\n", s1, s2 );
13
14     // concatenate s2 to s1
15     printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );
16
17     // concatenate first 6 characters of s1 to s3. Place '\0'
18     // after last character
19     printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );
20
21     // concatenate s1 to s3
22     printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );
23 } // end main

```

```

s1 = Happy
s2 = New Year
strcat( s1, s2 ) = Happy New Year
strncat( s3, s1, 6 ) = Happy
strcat( s3, s1 ) = Happy Happy New Year

```

圖8.16 strcat和strncat的使用方式(2/2)



## 8.7 字串處理函式庫的比較函式



- 本節介紹字串處理函式庫的**字串比較函式 (string comparison functions)**：**strcmp**和**strncmp**。圖8.17列出這兩個函式的原型和概括性的功能描述。

函式原型	函式的描述
<code>int strcmp( const char *s1, const char *s2 );</code>	比較字串 s1 與 s2。如果 s1 與 s2 相等則傳回 0；如果 s1 小於 s2 則傳回負值；如果 s1 大於 s2 則傳回正值。
<code>int strncmp( const char *s1, const char *s2, size_t n );</code>	比較字串 s1 與 s2 (最多比較 n 個字元)。如果 s1 與 s2 相等則傳回 0；如果 s1 小於 s2 則傳回負值；如果 s1 大於 s2 則傳回正值。

圖8.17 字串處理函式庫的字串比較函式



## 8.8 字串處理函式庫的搜尋函式



- 本節將介紹字串處理函式庫中用來搜尋字串中某些字元或其他字串的函式。這些函式列在圖 8.19 中。

```
char *strchr( const char *s, int c );
```

找出字元 **c** 在字串 **s** 中第一次出現的位置。如果有找到的話，則傳回 **c** 在 **s** 中所在位置的指標。不然則傳回 **NULL** 指標。

```
size_t strcspn( const char *s1, const char *s2 );
```

計算並且傳回字串 **s1** 中，遇到第一個屬於字串 **s2** 中的字元時，共有幾個字元。

```
size_t strspn( const char *s1, const char *s2 );
```

計算並且傳回字串 **s1** 中，遇到第一個不屬於字串 **s2** 中的字元時，共有幾個字元。

```
char *strpbrk( const char *s1, const char *s2 );
```

找出字串 **s2** 中任何字元在字串 **s1** 中第一次出現的位置。如果有找到的話，則傳回此字元在 **s1** 中所在位置的指標。不然則傳回 **NULL** 指標。

### 函式原型與描述

```
char *strrchr( const char *s, int c );
```

找出字元 **c** 在字串 **s** 中最後一次出現的位置。如果找到的話，傳回 **c** 在 **s** 中所在位置的指標。不然則傳回 **NULL** 指標。

```
char *strstr( const char *s1, const char *s2 );
```

找出字串 **s2** 在字串 **s1** 中第一次出現的地方。如果找到的話，傳回此字串在 **s1** 中所在位置的指標。不然則傳回 **NULL** 指標。

```
char *strtok( char *s1, const char *s2 );
```

一連串的 **strtok** 呼叫會將字串 **s1** 切割成一個個的字符 (token)。而這些字符是以字串 **s2** 中所含的字元為分隔點。第一次呼叫是以 **s1** 作為第一個引數，而接下來的呼叫則以 **NULL** 作為第一個引數並持續對同一字串切割字符。每次呼叫都會傳回一個指向目前字符的指標。如果已經沒有字符則會傳回空字元。

圖8.19 字串處理函式庫的搜尋函式(2/2)



### ❖ 8.8.1 函式 strchr

- 函式 **strchr** 尋找字串中某個字元第一次出現的位置。

```

1 // Fig. 8.20: fig08_20.c
2 // Using function strchr
3 #include <stdio.h>
4 #include <string.h>
5
6 int main( void )
7 {
8     const char *string = "This is a test"; // initialize char pointer
9     char character1 = 'a'; // initialize character1
10    char character2 = 'z'; // initialize character2
11

```

圖8.20 使用函式 strchr (1/2)



```

12    // if character1 was found in string
13    if ( strchr( string, character1 ) != NULL ) {
14        printf( "'%c' was found in \"%s\".\n",
15               character1, string );
16    } // end if
17    else { // if character1 was not found
18        printf( "'%c' was not found in \"%s\".\n",
19               character1, string );
20    } // end else
21
22    // if character2 was found in string
23    if ( strchr( string, character2 ) != NULL ) {
24        printf( "'%c' was found in \"%s\".\n",
25               character2, string );
26    } // end if
27    else { // if character2 was not found
28        printf( "'%c' was not found in \"%s\".\n",
29               character2, string );
30    } // end else
31 } // end main

```

```

'a' was found in "This is a test".
'z' was not found in "This is a test".

```

圖8.20 使用函式 strchr (2/2)





### ❖ 8.8.4 使用函式strchr

- 函式**strchr**會尋找字串中某字元最後一次出現的位置。如果找到的話，**strchr**傳回指向此字元的指標，否則便傳回**NULL**。圖8.23的程式在字串"**A zoo has many animals including zebras**"裡找出最後一個'**z**'的位置。

WATSE

```

1 // Fig. 8.23: fig08_23.c
2 // Using function strchr
3 #include <stdio.h>
4 #include <string.h>
5
6 int main( void )
7 {
8     // initialize char pointer
9     const char *string1 = "A zoo has many animals including zebras";
10
11     int c = 'z'; // character to search for
12
13     printf( "%s\n%s%c%s\n%s\n",
14            "The remainder of string1 beginning with the",
15            "last occurrence of character ", c,
16            " is: ", strchr( string1, c ) );
17 } // end main

```

The remainder of string1 beginning with the  
last occurrence of character 'z' is: "zebras"

圖8.23 使用函式strchr

WATSE



### ❖ 8.8.7 函式 strtok

- 函式 **strtok** (圖8.26) 用來將字串切成數個**字符 (token)**。字符是由**分界字元 (delimiter)**，通常為空白或標點符號，但分界字元可以是任何字元) 所分隔出的一連串字元。

WATSE

```

1 // Fig. 8.26: fig08_26.c
2 // Using function strtok
3 #include <stdio.h>
4 #include <string.h>
5
6 int main( void )
7 {
8     // initialize array string
9     char string[] = "This is a sentence with 7 tokens";
10    char *tokenPtr; // create char pointer
11
12    printf( "%s\n%s\n\n%s\n",
13           "The string to be tokenized is:", string,
14           "The tokens are:" );
15
16    tokenPtr = strtok( string, " " ); // begin tokenizing sentence
17
18    // continue tokenizing sentence until tokenPtr becomes NULL
19    while ( tokenPtr != NULL ) {
20        printf( "%s\n", tokenPtr );
21        tokenPtr = strtok( NULL, " " ); // get next token
22    } // end while
23 } // end main

```

圖8.26 使用函式 strtok(1/2)

WATSE

## ❖ 8.10.2 函式strlen

- 函式**strlen**以一個字串做為引數，它會傳回此字串的字元數。圖8.35的程式示範了**strlen**的使用。

```

1 // Fig. 8.35: fig08_35.c
2 // Using function strlen
3 #include <stdio.h>
4 #include <string.h>
5
6 int main( void )
7 {
8     // initialize 3 char pointers
9     const char *string1 = "abcdefghijklmnopqrstuvwxy";
10    const char *string2 = "four";
11    const char *string3 = "Boston";
12
13    printf("%s\\\"%s\\\"%s%u\\n%s\\\"%s\\\"%s%u\\n%s\\\"%s\\\"%s%u\\n",
14          "The length of ", string1, " is ", strlen( string1 ),
15          "The length of ", string2, " is ", strlen( string2 ),
16          "The length of ", string3, " is ", strlen( string3 ) );
17 } // end main

```

```

The length of "abcdefghijklmnopqrstuvwxy" is 26
The length of "four" is 4
The length of "Boston" is 6

```

## 作業



- ❖ 題目: 修改回文程式(文字與數字都能處理)，成為句子變化程式。(注意:不是全部反轉喔!)
- ❖ 輸入: this is a book
- ❖ 輸出: book a is this