程式設計
(Programming)

真理大學 資訊工程系 吳汶涓老師

CH08

字元與字串

本章綱要

8-1 簡介
8-2 字串和字元的基本知識
8-3 字元處理函式庫
8-4 字串轉換函式
8-5 標準輸入/輸出函式庫函式
8-6 字串處理函式庫 - 字串操作函式
8-7 字串處理函式庫 - 比較函式
8-8 字串處理函式庫 - 搜尋函式
8-9 字串處理函式庫 - 記憶體函式
8-10 字串處理函式庫 - 其他函式

8.6 字串處理函式庫 – 字串操作函式

- **字串處理的功用 (#include <string.h>)**
  - 操作字串資料 (複製、比較、連接字串等)
  - 搜尋字串
  - 字串斷字
  - 計算字串長度

函式原型	函式的描述
`char *strcpy( char *s1, const char *s2 )`	將字串 s2 複製至陣列 s1。並傳回 s1。
`char *strncpy( char *s1, const char *s2, size_t n )`	將字串 s2 的最多 n 個字元複製至陣列 s1。並傳回 s1。
`char *strcat( char *s1, const char *s2 )`	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。
`char *strncat( char *s1, const char *s2, size_t n )`	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 8.17　字串處理函式庫的字串操作函式

範例: 字串複製 (strcpy, strncpy)

```
 3  #include <stdio.h>
 4  #include <string.h>
 5
 6  int main( void )
 7  {
 8     char x[] = "Happy Birthday to You";
 9     char y[ 25 ];
10     char z[ 15 ];
11
13     printf( "%s%s\n%s%s\n",
14        "The string in array x is: ", x,
15        "The string in array y is: ", strcpy( y, x ) );
16
19     strncpy( z, x, 14 );
20
21     z[ 14 ] = '\0';
22     printf( "The string in array z is: %s\n", z );
23     return 0;
24  }
```

```
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
```

strcpy將字串x複製到字元陣列y

strncpy複製x中14個字元到陣列z

圖 8.18　strcpy 和 strncpy 的使用方式

課本pp. 8-20

範例: 字串連接 (strcat, strncat)

```
3  #include <stdio.h>
4  #include <string.h>
5
6  int main( void )
7  {
8     char s1[ 20 ] = "Happy ";
9     char s2[] = "New Year ";
10    char s3[ 40 ] = "";
12    printf( "s1 = %s\ns2 = %s\n", s1, s2 );
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
22    printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );
24    return 0;
26 }
```

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

strcat將字串**s2**的字元加到字串**s1**的末端

strncat 將字串**s1**的前6個字元加到字串**s3**的末端，並自動補'\0'

圖 8.19 strcat 和 strncat 的使用方式

課本pp. 8-22

8.7 字串處理函式庫 – 比較函式

- 字串比較 (strcmp, strncmp)
  - 電腦比較的是字串中字元的ASCII數碼值

函式原型	函式的描述
`int strcmp( const char *s1, const char *s2 );`	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。
`int strncmp( const char *s1, const char *s2, size_t n );`	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 8.20　字串處理函式庫的字串比較函式

```
s1 = Happy New Year
s2 = Happy New Year
s3 = Happy Holidays

strcmp(s1, s2) =  0
strcmp(s1, s3) =  1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) =  0
strncmp(s1, s3, 7) =  6
strncmp(s3, s1, 7) = -6
```

```c
#include <stdio.h>
#include <string.h>
int main( void )
{
   const char *s1 = "Happy New Year";
   const char *s2 = "Happy New Year";
   const char *s3 = "Happy Holidays";

   printf("%s%s\n%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n\n",
          "s1 = ", s1, "s2 = ", s2, "s3 = ", s3,
          "strcmp(s1, s2) = ", strcmp( s1, s2 ),
          "strcmp(s1, s3) = ", strcmp( s1, s3 ),
          "strcmp(s3, s1) = ", strcmp( s3, s1 ) );
   printf("%s%2d\n%s%2d\n%s%2d\n",
          "strncmp(s1, s3, 6) = ", strncmp( s1, s3, 6 ),
          "strncmp(s1, s3, 7) = ", strncmp( s1, s3, 7 ),
          "strncmp(s3, s1, 7) = ", strncmp( s3, s1, 7 ) );

   return 0;
}
```

strcmp 比較字串 **s1** 和 **s2**

strncmp 比較 **s1** 的前六個字元和 **s3** 的前六個字元

圖 8.21　strcmp 和 strncmp 的使用方式

課本pp. 8-22

常見的程式設計錯誤 8.6

誤以爲 strcmp 和 strncmp 會在他們的引數相同時傳回 1，這會造成一個邏輯錯誤。這兩個函式都在相等時傳回 0 (C 的僞值)。因此，當你在檢查兩個字串是否相等時，strcmp 或 strncmp 所傳回的結果應與 0 比較，來判斷兩個字串是否相等。

可攜性的小技巧 8.3

用來表示字元的內部數碼可能會隨電腦的不同而有所差異。

練習

- 撰寫程式來判斷使用者輸入的帳號密碼是否與程式設定的帳密相同。

8.8 字串處理函式庫 – 搜尋函式

函式原型	函式的描述
`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。

圖 8.22　字串處理函式庫的字串操作函式

函式原型	函式的描述
`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 中所含的字元爲分隔點。第一次呼叫是以 s1 作爲第一個引數，而接下來的呼叫則以 NULL 作爲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 8.22　字串處理函式庫的字串操作函式

```c
#include <stdio.h>
#include <string.h>
int main( void ) {
   const char *string = "This is a test";
   char character1 = 'a';
   char character2 = 'z';

   if ( strchr( string, character1 ) != NULL ) {
      printf( "\'%c\' was found in \"%s\".\n",
         character1, string );
   }
   else {
      printf( "\'%c\' was not found in \"%s\".\n",
         character1, string );
   }
   if ( strchr( string, character2 ) != NULL ) {
      printf( "\'%c\' was found in \"%s\".\n",
         character2, string );
   }
   else {
      printf( "\'%c\' was not found in \"%s\".\n",
         character2, string );
   }
   return 0;
}
```

strchr尋找**character1**在**string**中第一次出現的位置

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

課本pp. 8-24

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

int main( void )
{
   const char *string1 = "The value is 3.14159";
   const char *string2 = "1234567890";

   printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
      "string1 = ", string1, "string2 = ", string2,
      "The length of the initial segment of string1",
      "containing no characters from string2 = ",
      strcspn( string1, string2 ) );

   return 0;
}
```

strcspn 會計算**string1**中，遇到第一個屬於**string2**的字元時，之前共有幾個字元

```
string1 = The value is 3.14159
string2 = 1234567890

The length of the initial segment of string1
containing no characters from string2 = 13
```

課本pp. 8-25

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

int main( void )
{
   const char *string1 = "This is a test";
   const char *string2 = "beware";

   printf( "%s\"%s\"\n'%c'%s\n\"%s\"\n",
      "Of the characters in ", string2,
      *strpbrk( string1, string2 ),
      " appears earliest in ", string1 );
   return 0;
}
```

strpbrk 會回傳一個指標，指向 **string2**中任何一個字元在字串 **string1**中第一次出現的位置

```
Of the characters in "beware"
'a' appears earliest in
"This is a test"
```

```c
/* Fig. 8.26: fig08_26.c */
#include <stdio.h>
#include <string.h>

int main( void )
{
   const char *string1 = "A zoo has many animals including zebras";
   int c = 'z';
   printf( "%s\n%s'%c'%s\"%s\"\n",
           "The remainder of string1 beginning with the",
           "last occurrence of character ", c,
           " is: ", strrchr( string1, c ) );
   return 0;
}
```

strrchr 會回傳字元 **c** 在字串 **string1** 中最後一次出現的位置

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

課本pp. 8-26

```c
/* Fig. 8.27: fig08_27.c */
#include <stdio.h>
#include <string.h>

int main( void )
{
   const char *string1 = "The value is 3.14159";
   const char *string2 = "aehi lsTuv";

   printf( "%s%s\n%s%s\n\n%s\n%s%u\n",
      "string1 = ", string1, "string2 = ", string2,
      "The length of the initial segment of string1",
      "containing only characters from string2 = ",
      strspn( string1, string2 ) );
   return 0;
}
```

strspn計算從**string1**的開頭共有多少個屬於**string2**的字元

```
string1 = The value is 3.14159
string2 = aehi lsTuv

The length of the initial segment of string1
containing only characters from string2 = 13
```

```c
/* Fig. 8.28: fig08_28.c  */
#include <stdio.h>
#include <string.h>
int main( void )
{
   const char *string1 = "abcdefabcdef";
   const char *string2 = "def";

   printf( "%s%s\n%s%s\n\n%s\n%s%s\n",
      "string1 = ", string1, "string2 = ", string2,
      "The remainder of string1 beginning with the",
      "first occurrence of string2 is: ",
      strstr( string1, string2 ) );
   return 0;
}
```

strstr找出string2第一次出現在string1的位置

```
string1 = abcdefabcdef
string2 = def

The remainder of string1 beginning with the
first occurrence of string2 is: defabcdef
```

```
1  /* Fig. 8.29: fig08_29.c  */
3  #include <stdio.h>
4  #include <string.h>
5
6  int main( void )
7  {
9     char string[] = "This is a sentence with 7 tokens";
10    char *tokenPtr;
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, " " );
17
19    while ( tokenPtr != NULL ) {
20       printf( "%s\n", tokenPtr );
21       tokenPtr = strtok( NULL, " " );
22    }
24    return 0;
26 }
```

```
The string to be tokenized is:
This is a sentence with 7 tokens

The tokens are:
This
is
a
sentence
with
7
tokens
```

strtok會以空白字元為界限，將字串 **string**切割成一個個的字符

以NULL再次呼叫 **strtok** ，表示將繼續對前一個字串進行字符的分割

注意: 分界字元通常為空白或標點符號，或可以是任何指定字元

課本pp. 8-28

練習

- 撰寫程式，將電話號碼以字串形式輸入，使用strtok()取出字符，並用strcpy()或strcat() 取出其中的區碼及電話號碼印出來。(ex. 8.14)

- 輸入一行文字，用strtok()將該字串切成數個字符，然後以相反的順序輸出每個字符。(ex. 8.15)

- 輸入數行文字，用strchr()搜尋出某字元共出現了幾次。(ex. 8.18)

課本pp. 8-45

8.9 字串處理函式庫 – 記憶體函式

- 記憶體函式
  - 需 **#include <string.h>**
  - 有關記憶體區塊的操作、比較及搜尋函式
  - 可以處理任何區塊的資料

- 指標參數為 **void ***
  - 所有指標都可以設定給void *，反之亦然
  - void * 指標不能求值
    - 所以每個函式都會有一個指定大小的引數，用來指出此函式要處理的字元個數

函式原型	函式的描述
`void *memcpy( void *s1, const void *s2, size_t n );`	從指標 s2 所指之物件複製 n 個字元到 s1 所指的物件。會傳回指向 s1 的指標。
`void *memmove( void *s1, const void *s2, size_t n );`	從指標 s2 所指之物件複製 n 個字元到 s1 所指的物件。此函式會先將 s2 所指之物件的 n 個字元複製到一暫時的陣列中，然後再從此暫時陣列複製 n 個字元到 s1 所指的物件。函式會傳回指向 s1 的指標。
`int memcmp( const void *s1, const void *s2, size_t n );`	比較 s1 和 s2 所指之物件的前 n 個字元。如果 s1 等於 s2 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。
`void *memchr( const void *s, int c, size_t n );`	找出字元 c (轉換爲 unsigned char) 在 s 所指之物件的前 n 個字元中，第一次出現的位置。如果找到 c 的話，傳回一個指向 c 的指標。否則傳回 NULL。
`void *memset( void *s, int c, size_t n );`	將 s 所指之物件的前 n 個字元全指定爲 c (轉換爲 unsigned char)。傳回一個指向結果的指標。

圖 8.30　字串處理函式庫的記憶體函式

課本pp. 8-30

```c
/* Fig. 8.31: fig08_31.c */
#include <stdio.h>
#include <string.h>
int main( void )
{
   char s1[ 17 ];
   char s2[] = "Copy this string";

   memcpy( s1, s2, 17 );
   printf( "%s\n%s\"%s\"\n",
           "After s2 is copied into s1 with memcpy,",
           "s1 contains ", s1 );
   return 0;
}
```

```
After s2 is copied into s1 with memcpy,
s1 contains "Copy this string"
```

memcpy複製物件**s2**的前17個字元到物件**s1**

```c
/* Fig. 8.32: fig08_32.c */
#include <stdio.h>
#include <string.h>
int main( void )
{
   char x[] = "Home Sweet Home";
   printf( "%s%s\n", "The string in array x before memmove is: ", x );
   printf( "%s%s\n", "The string in array x after memmove is: ",
           memmove( x, &x[ 5 ], 10 ) );
   return 0;

}
```

```
The string in array x before memmove is: Home Sweet Home
The string in array x after memmove is: Sweet Home Home
```

memmove利用暫時的陣列將**x[5]**的前10個字元複製到物件**x**中

```c
/* Fig. 8.33: fig08_33.c */
#include <stdio.h>
#include <string.h>
int main( void )
{
   char s1[] = "ABCDEFG";
   char s2[] = "ABCDXYZ";
   printf( "%s%s\n%s%s\n\n%s%2d\n%s%2d\n%s%2d\n",
           "s1 = ", s1, "s2 = ", s2,
           "memcmp( s1, s2, 4 ) = ", memcmp( s1, s2, 4 ),
           "memcmp( s1, s2, 7 ) = ", memcmp( s1, s2, 7 ),
           "memcmp( s2, s1, 7 ) = ", memcmp( s2, s1, 7 ) );

   return 0;
}
```

```
s1 = ABCDEFG
s2 = ABCDXYZ

memcmp( s1, s2, 4 ) =  0
memcmp( s1, s2, 7 ) = -1
memcmp( s2, s1, 7 ) =  1
```

memcmp 比較物件**s1**和**s2**的前四個字元

```c
/* Fig. 8.34: fig08_34.c */
#include <stdio.h>
#include <string.h>
int main( void )
{
   const char *s = "This is a string";
   printf( "%s\'%c\'%s\"%s\"\n",
           "The remainder of s after character ", 'r',
           " is found is ", memchr( s, 'r', 16 ) );
   return 0;
}
```

```
The remainder of s after character 'r' is found is "ring"
```

memchr 在物件s的前16個字元中，尋找字元 **r** 第一次出現的位置

8.10 字串處理函式庫 – 其他函式

函式原型	函式的描述
`char *strerror( int errornum );`	將 errornum（錯誤代碼）對應成與系統相關的錯誤訊息字串。並且傳回此字串的指標。
`size_t strlen( const char *s );`	算出字串 s 的長度。回傳此長度（此長度並不包含結束字元）。

```
1  /* Fig. 8.38: fig08_38.c */
3  #include <stdio.h>
4  #include <string.h>
6  int main( void ) {
9     const char *string1 = "abcdefghijklmnopqrstuvwxyz";
10    const char *string2 = "four";
11    const char *string3 = "Boston";
13    printf("%s\"%s\"%s%lu\n%s\"%s\"%s%lu\n%s\"%s\"%s%lu\n",
14       "The length of ", string1, " is ",
15       ( unsigned long ) strlen( string1 ),
16       "The length of ", string2, " is ",
17       ( unsigned long ) strlen( string2 ),
18       "The length of ", string3, " is ",
19       ( unsigned long ) strlen( string3 ) );
21    return 0;
23 }
```

```
The length of "abcdefghijklmnopqrstuvwxyz" is 26
The length of "four" is 4
The length of "Boston" is 6
```

strlen 回傳 string1長度