第八章講授重點

字元與字串

講授重點

1. 字串與字元之認識
2. 字元處理函式庫
3. 標準輸入/輸出函式庫
4. 字串操作函數

1. 字串與字元之認識

- 字元是構成原始程式的基本元件。
- 字元常數是一個`int`值，以單引號括起來的字元來表示。
- 字元常數的值便是此字元在機器的字元集(character set) 中的整數值。
- 例如 ' z ' =122 (ASCII)代表了z的整數值，
- ' \ n ' =10 (ASCII)代表了換行字元的整數值。

1. 字串與字元之認識(續)

- 字串是視爲單一個體的一連串字元。
- 字串可以含有字母、數字、以及數種特殊的字元 (如 +、-、*、/、$ 等)。
- C裡的字串寫在雙引號裡。
- 以空字元 ('\0') 來做爲結束的字元陣列。
- 經由指向字串第一個字元的指標存取這個字串。
- 字串類似陣列，陣列也是一個指向第一個陣列元素的指標。

1. 字串與字元之認識(續)

- ```
  char color[] = "blue";
  const char *colorPtr = "blue";
  ```
- 第一個宣告建立一個5個元素的陣列color，它的元素分別爲'b'、'1'、'u'、'e'和'\0'。
- 第二個宣告建立了一個指標變數colorPtr，它指向存放在記憶體中某個位置的字串"blue"。
- 上述陣列定義也可以寫成

```
char color[] = { 'b', 'l', 'u', 'e', '\0' };
```
```

1. 字串與字元之認識(續)

- 例如，以下的敘述式將會指定一個字串給字元陣列**word[20]**。
  - `scanf( "%s", word );`
- 使用者輸入的字串存放在**word**。
- 注意到word是陣列，所以使用函式**scanf**時word之前不需加&。
- 函式**scanf**一直讀入字元，直到遇上了空白、tab、新行、或end-of-file。
- 請注意，如果輸入字串的長度超過了19個字元，你的程式可能會當掉

2. 字元處理函式庫

- 字元處理函式庫 (character-handling library) <ctype.h> 包括了數個執行字元資料測試和操作的函式。
- 每個函式使用一個字元 (以int或EOF表示) 做爲引數。
- EOF的值通常爲-1，而有些硬體架構不允許將負的值存放在char變數。因此，字元處理函式便會將字元當成整數來進行操作。
- 圖8.1列出字元處理函式庫的所有函式。

原型	函式的描述
`int isdigit( int c );`	如果 c 爲一數字則傳回眞，否則傳回 0 (僞)。
`int isalpha( int c );`	如果 c 爲一字母則傳回眞，否則傳回 0。
`int isalnum( int c );`	如果 c 爲一數字或字母則會傳回眞，否則傳回 0。
`int isxdigit( int c );`	如果 c 爲一 16 進位的數字則會傳回眞，否則傳回 0。(請參考附錄 C “數字系統”對二進制，八進制，十進制和十六進制數字的進一步描述)
`int islower( int c );`	如果 c 爲一小寫字母則傳回眞，否則傳回 0。
`int isupper( int c );`	如果 c 爲一大寫字母則傳回眞，否則傳回 0。
`int tolower( int c );`	如果 c 是一個大寫字母，tolower 函式就傳回小寫的 c。如果不是，tolower 函式就傳回原來的引數。
`int toupper( int c );`	如果 c 是一個小寫字母，toupper 函式就會傳回大寫的 c。如果不是，toupper 函式就傳回原來的引數。

圖 8.1　字元處理函式庫 <ctype.h> 的函式

`int isspace( int c );`	如果 c 爲一空白字元則傳回眞。空白字元包括：換行 ('\n')，空白 (' ')，跳頁 ('\f')，回車 ('\r')，水平跳格 ('\t') 及垂直跳格 ('\v')。否則傳回 0。
`int iscntrl( int c );`	如果 c 爲一控制字元則傳回眞，否則傳回 0。
`int ispunct( int c );`	如果 c 是空格、數字以及字母以外的可列印字元，函式就會傳回眞；不然傳回就是 0。
`int isprint( int c );`	如果 c 是包含空格 (' ') 的可列印字元，函式就會傳回眞；不然就傳回零。
`int isgraph( int c );`	如果 c 是空格 (' ') 以外的可列印字元，函式就會傳回眞；不然就傳回零。

圖 8.1　字元處理函式庫 <ctype.h> 的函式

例題一

```
1  /* Fig. 8.2: fig08_02.c
2     Using functions isdigit, isalpha, isalnum, and isxdigit */
3  #include <stdio.h>
4  #include <ctype.h>
5
6  int main( void )
7  {
8     printf( "%s\n%s%s\n%s%s\n\n", "According to isdigit: ",
9        isdigit( '8' ) ? "8 is a " : "8 is not a ", "digit",
10       isdigit( '#' ) ? "# is a " : "# is not a ", "digit" );
11
12    printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n\n",
13       "According to isalpha:",
14       isalpha( 'A' ) ? "A is a " : "A is not a ", "letter",
15       isalpha( 'b' ) ? "b is a " : "b is not a ", "letter",
16       isalpha( '&' ) ? "& is a " : "& is not a ", "letter",
17       isalpha( '4' ) ? "4 is a " : "4 is not a ", "letter" );
18
```

例題一(續)

```
19      printf( "%s\n%s%s\n%s%s\n%s%s\n\n",
20         "According to isalnum:",
21         isalnum( 'A' ) ? "A is a " : "A is not a ",
22         "digit or a letter",
23         isalnum( '8' ) ? "8 is a " : "8 is not a ",
24         "digit or a letter",
25         isalnum( '#' ) ? "# is a " : "# is not a ",
26         "digit or a letter" );
27
28      printf( "%s\n%s%s\n%s%s\n%s%s\n%s%s\n%s%s\n",
29         "According to isxdigit:",
30         isxdigit( 'F' ) ? "F is a " : "F is not a ",
31         "hexadecimal digit",
32         isxdigit( 'J' ) ? "J is a " : "J is not a ",
33         "hexadecimal digit",
34         isxdigit( '7' ) ? "7 is a " : "7 is not a ",
35         "hexadecimal digit",
36         isxdigit( '$' ) ? "$ is a " : "$ is not a ",
37         "hexadecimal digit",
38         isxdigit( 'f' ) ? "f is a " : "f is not a ",
39         "hexadecimal digit" );
40      return 0; /* indicates successful termination */
41   } /* end main */
```

例題一(續)

```
According to isdigit:
8 is a digit
# is not a digit

According to isalpha:
A is a letter
b is a letter
& is not a letter
4 is not a letter

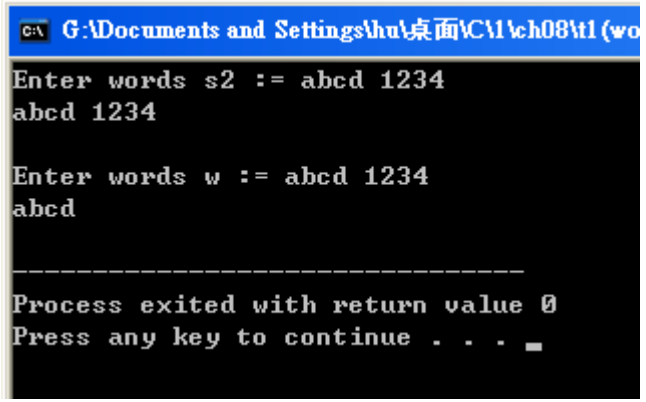
According to isalnum:
A is a digit or a letter
8 is a digit or a letter
# is not a digit or a letter

According to isxdigit:
F is a hexadecimal digit
J is not a hexadecimal digit
7 is a hexadecimal digit
$ is not a hexadecimal digit
f is a hexadecimal digit
```

例題二

t1(word input).c

```c
// to input words
#include <stdio.h>

int main( void )
{
    float a,b,c,z;
    char w[40],s2[40];

    printf( "Enter words s2 := " );
    fgets(s2,40,stdin);
    printf("%s\n", s2);

    printf( "Enter words w := " );
    scanf("%s",w);
    printf("%s\n", w);
    return 0;

}
```



2. 字元處理函式庫(續)

- 本節介紹一般公用函式庫 (general utilities library <stdlib.h>) 裡的字串轉換函式 (string conversion functions)。
- 這些函式由數字組成字串轉換成整數或浮點數值。
- 圖8.2列出字串轉換函式。
- 注意，函式的標頭都使用const來宣告變數nPtr (由右至左讀成「nPtr是個指向字元常數的指標」)。const將引數宣告爲不可更改的。

2. 字元處理函式庫(續)

函式原型	函式的描述
`double atof( const char *nPtr );`	將字串 nPtr 轉換成 double。
`int atoi( const char *nPtr );`	將字串 nPtr 轉換成 int。
`long atol( const char *nPtr );`	將字串 nPtr 轉換成 long int。
`double strtod( const char *nPtr, char **endPtr );`	將字串 nPtr 轉換成 double。
`long strtol( const char *nPtr, char **endPtr, int base );`	將字串 nPtr 轉換成 long。
`unsigned long strtoul( const char *nPtr, char **endPtr, int base );`	將字串 nPtr 轉換成 unsigned long。

圖8.2 一般公用函式庫當中的字串轉換函式

例題三

```c
/* Fig. 8.9: fig08_09.c
   Using strtod */
#include <stdio.h>
#include <stdlib.h>

int main( void )
{
   /* initialize string pointer */
   const char *string = "51.2% are admitted"; /* initialize string */

   double d; /* variable to hold converted sequence */
   char *stringPtr; /* create char pointer */

   d = strtod( string, &stringPtr );

   printf( "The string \"%s\" is converted to the\n", string );
   printf( "double value %.2f and the string \"%s\"\n", d, stringPtr );
   return 0; /* indicates successful termination */
} /* end main */
```

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

例題四

```c
/* Fig. 8.10: fig08_10.c
   Using strtol */
#include <stdio.h>
#include <stdlib.h>

int main( void )
{
   const char *string = "-1234567abc"; /* initialize string pointer */

   char *remainderPtr; /* create char pointer */
   long x; /* variable to hold converted sequence */

   x = strtol( string, &remainderPtr, 0 );

   printf( "%s\"%s\"\n%s%ld\n%s\"%s\"\n%s%ld\n",
      "The original string is ", string,
      "The converted value is ", x,
      "The remainder of the original string is ",
      remainderPtr,
      "The converted value plus 567 is ", x + 567 );
   return 0; /* indicates successful termination */
} /* end main */
```

例題四(續)

```
The original string is "-1234567abc"
The converted value is -1234567
The remainder of the original string is "abc"
The converted value plus 567 is -1234000
```

例題五

t4.c

```c
/* t4.c : Using strtod */
#include <stdio.h>
#include <stdlib.h>

int main( void )
{
   /* initialize string pointer */
   const char *string = "30.2+27.5"; /* initialize string */

   double d; /* variable to hold converted sequence */
   char *stringPtr; /* create char pointer */

   d = strtod( string, &stringPtr );
   printf( "The string \"%s\" is converted to the\n", string );
   printf( "d = %.2f and the char \"%c\"\n\n", d, *stringPtr );

   string = stringPtr;
   d = strtod( string, &stringPtr );
   printf( "The string \"%s\" is converted to the\n", string );
   printf( "d = %.2f and the char \"%c\"\n", d, *stringPtr );

   return 0;
}
```

```
G:\Documents and Settings\hu\桌面\C\1\ch08\t4.exe
The string "30.2+27.5" is converted to the
d = 30.20 and the char "+"

The string "+27.5" is converted to the
d = 27.50 and the char " "

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

作業

- 請做作業 8_3
- 步驟：

1. 輸入一組字串，如 “34.2-67.5” 或 “3.2*10”等。
2. 請使用函式fgets()與 strtod()，計算其值。
3. 結果印在螢幕上
4. 請參考例題 t1(word input).c, t4(strtod).c

3. 標準輸入/輸出函式庫

- 本節介紹標準輸入/輸出函式庫 (<stdio.h>) 裡，處理字元和字串資料的函式。
- 圖8.3列出標準輸入/輸出函式庫中，關於字元和字串輸入/輸出的函式。

3. 標準輸入/輸出函式庫(續)

函式原型	函式的描述
`int getchar( void );`	從標準輸入讀進下一個字元，並以整數值傳回。
`char *fgets( char *s, int n, FILE *stream);`	從指定的串流持續讀進字元到陣列 s 中，直到出現 newline 或 end-of-file 字元，或是直到讀入n-1個字元爲止。 在本章中，我們指定串流爲 stdin——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 null 字元。將讀入 s 的字串傳回。
`int putchar( int c );`	印出存放在 c 裡的字元，並將此字元以整數傳回。
`int puts( const char *s );`	印出字串 s 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 EOF。

圖8.3 標準輸入／輸出函式庫的字元和字串函式

3. 標準輸入/輸出函式庫(續)

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

和 printf 相同，不過輸出是放到陣列 s 而不是印到螢幕上。傳回寫入 s 中的字元數量，假如錯誤發生，則傳回 EOF。

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

和 scanf 相同，不過輸入是從陣列 s 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 EOF。

圖8.3 標準輸入／輸出函式庫的字元和字串函式

3. 標準輸入/輸出函式庫(續)

- gets( )函數用於從緩衝區中讀取字串，其原型如下：char *gets(char *string);
- gets( )函數從流中讀取字串，直到出現分行符號或讀到檔案結尾爲止，最後加上‘\0’作爲字串結束。所讀取的字串暫存在給定的參數string中。
- 注意：由於gets( )不檢查字串string的大小，必須遇到分行符號或檔結尾才會結束輸入，因此容易造成緩存溢出的安全性問題，導致程式崩潰，可以使用fgets( )代替。

例題六

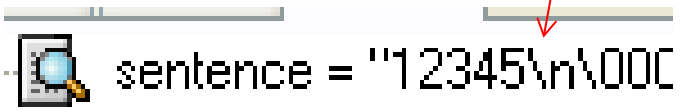
```c
/* Fig. 8.16: fig08_16.c
   Using sscanf */
#include <stdio.h>

int main( void )
{
   char s[] = "31298 87.375"; /* initialize array s */
   int x; /* x value to be input */
   double y; /* y value to be input */

   sscanf( s, "%d%lf", &x, &y );
   printf( "%s\n%s%6d\n%s%8.3f\n",
           "The values stored in character array s are:",
           "integer:", x, "double:", y );
   return 0; /* indicates successful termination */
} /* end main */
```

```
The values stored in character array s are:
integer: 31298
double:  87.375
```

例題七

```
1  /* Fig. 8.13: fig08_13.c
2     Using gets and putchar */
3  #include <stdio.h>
4
5  void reverse( const char * const sPtr ); /* prototype */
6
7  int main( void )
8  {
9     char sentence[ 80 ]; /* create char array */
10
11    printf( "Enter a line of text:\n" );
12
13    /* use fgets to read line of text */
14    fgets( sentence, 80, stdin );
15
16    printf( "\nThe line printed backward is:\n" );
17    reverse( sentence );
18    return 0; /* indicates successful termination */
19 } /* end main */
20
```

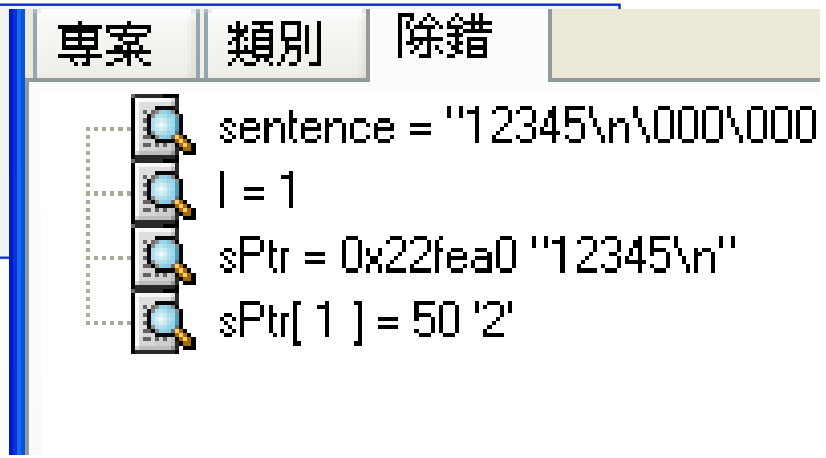
鍵入資料後存放 '\n' 與'\0'



例題七(續)

```
21  /* recursively outputs characters in string in reverse order */
22  void reverse( const char * const sPtr )
23  {
24     /* if end of the string */
25     if ( sPtr[ 0 ] == '\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:
12345
12345

The line printed backward is:

54321
```

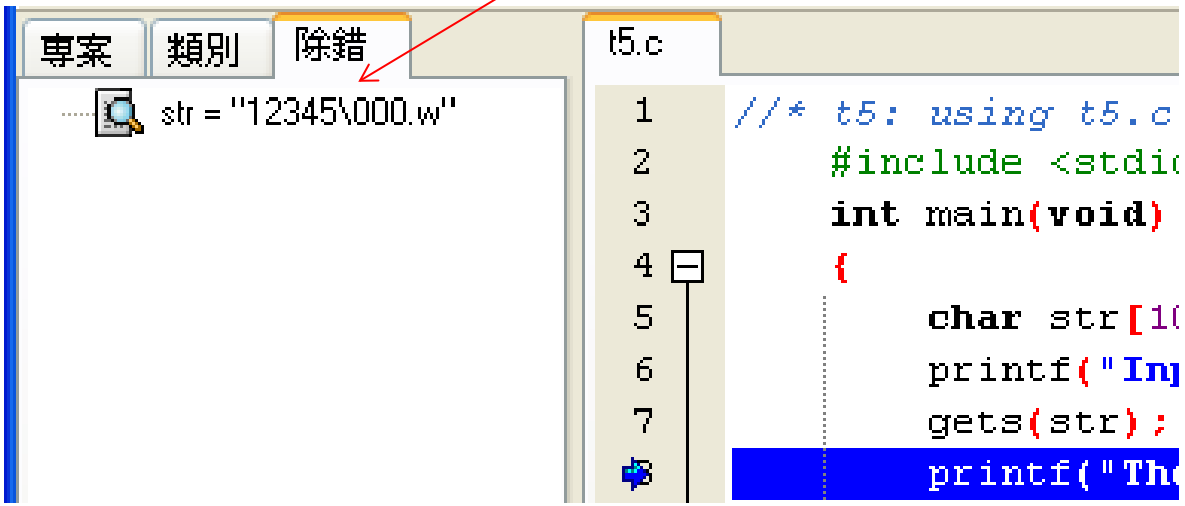


例題七-1

[*] t5.c

```
1  //* t5: using t5.c
2      #include <stdio.h>
3      int main(void)
4      {
5          char str[10];
6          printf("Input a string.\n");
7          gets(str);
8          printf("The string you input is: %s",str);     //?出所有的值，注意a
9      }
```

鍵入資料後存放 '\0'



例題八

```c
/* Fig. 8.14: fig08_14.c
   Using getchar and puts */
#include <stdio.h>

int main( void )
{
   char c; /* variable to hold character input by user */
   char sentence[ 80 ]; /* create char array */
   int i = 0; /* initialize counter i */

   /* prompt user to enter line of text */
   puts( "Enter a line of text:" );

   /* use getchar to read each character */
   while ( ( c = getchar() ) != '\n') {
      sentence[ i++ ] = c;
   } /* end while */

   sentence[ i ] = '\0'; /* terminate string */

   /* use puts to display sentence */
   puts( "\nThe line entered was:" );
   puts( sentence );
   return 0; /* indicates successful termination */
} /* end main */
```

```
Enter a line of text:
This is a test.

The line entered was:
This is a test.
```

4. 字串操作函式

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8.4 字串處理函式庫的字串操作函式

4. 字串操作函式(續)

- 請注意函式strncpy和strncat均有一個型別爲size_t的參數。在C的標準裡，size_t定義成sizeof運算子的回傳型別。

例題九

```c
/* Fig. 8.19: fig08_19.c
   Using strcat and strncat */
#include <stdio.h>
#include <string.h>

int main( void )
{
   char s1[ 20 ] = "Happy "; /* initialize char array s1 */
   char s2[] = "New Year "; /* initialize char array s2 */
   char s3[ 40 ] = ""; /* initialize char array s3 to empty */

   printf( "s1 = %s\ns2 = %s\n", s1, s2 );

   /* concatenate s2 to s1 */
   printf( "strcat( s1, s2 ) = %s\n", strcat( s1, s2 ) );

   /* concatenate first 6 characters of s1 to s3. Place '\0'
      after last character */
   printf( "strncat( s3, s1, 6 ) = %s\n", strncat( s3, s1, 6 ) );

   /* concatenate s1 to s3 */
   printf( "strcat( s3, s1 ) = %s\n", strcat( s3, s1 ) );
   return 0; /* indicates successful termination */
} /* end main */
```

4. 字串操作函式(續)

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

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8.5　字串處理函式庫的字串比較函式

4. 字串操作函式(續)

- 字串比較函式 (string comparison functions)：`strcmp` 和`strncmp`。
- 圖8.5列出這兩個函式的原型和概括性的功能描述。

例題十

```c
/* Fig. 8.21: fig08_21.c
   Using strcmp and strncmp */
#include <stdio.h>
#include <string.h>

int main( void )
{
   const char *s1 = "Happy New Year"; /* initialize char pointer */
   const char *s2 = "Happy New Year"; /* initialize char pointer */
   const char *s3 = "Happy Holidays"; /* initialize char pointer */

   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; /* indicates successful termination */
} /* end main */
```

例題十(續)

```
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
```

4. 字串操作函式(續)

- 例題十使用strcmp和strncmp比較三個字串。
- strcmp函式會一個字元一個字元地比較它的第一個字串引數和第二個字串引數。
- 如果兩個字串相等，則此函式會傳回0；如果第一個字串小於第二個字串，則此函式會傳回負的值；如果第一個字串大於第二個字串，則此函式會傳回正的值。
- strncmp函式的運作和strcmp十分類似，不過strncmp可以指定要進行比較的字元個數。
- 此外，strncmp不會對空字元之後的字元進行比較。
- 此程式印出每個函式呼叫所傳回的整數值。

4. 字串操作函式(續)

- 英文字母順序來解釋字串「大於」或「小於」另一字串的函義爲何。
- "Jones"排在"Smith"之前，因爲"Jones"的第一個英文字母會排在"Smith"的第一個字母之前。

作業

- 請做作業8_2:分別輸入兩組字串，結合成一組字串
- 參考fig08_13.c, fig08_14.c ，輸入兩組字串。
- 參考fig08_18.c, fig08_19.c，使用strcat及strcpy。
- 以c程式函式方式撰寫
- 其結果印在螢幕上
- 請參考指令如 fgets, getchar, strcat, strcpy.

4. 字串操作函式(續)

- 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8.6中。
- 請注意，函式`strcspn`和`strspn`的傳回型別爲`size_t`。

4. 字串操作函式(續)

函式原型與描述

```
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8.6 字串處理函式庫的字串操作函式

4. 字串操作函式(續)

```
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8.6 字串處理函式庫的字串操作函式

4. 字串操作函式(續)

- 函式strchr尋找字串中某個字元第一次出現的位置。
- 如果找到的話，strchr會傳回一個指向此字元的指標，否則就傳回空字元。
- 例題十一程式使用strchr來尋找字串"This is "中第一次出現'a'和'z'的地方。

例題十一

```c
/* Fig. 8.23: fig08_23.c
   Using strchr */
#include <stdio.h>
#include <string.h>

int main( void )
{
   const char *string = "This is a test"; /* initialize char pointer */
   char character1 = 'a'; /* initialize character1 */
   char character2 = 'z'; /* initialize character2 */

   /* if character1 was found in string */
   if ( strchr( string, character1 ) != NULL ) {
      printf( "\'%c\' was found in \"%s\".\n",
         character1, string );
   } /* end if */
   else { /* if character1 was not found */
      printf( "\'%c\' was not found in \"%s\".\n",
         character1, string );
   } /* end else */

```

例題十一(續)

```
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
32      return 0; /* indicates successful termination */
33   } /* end main */
```

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

4. 字串操作函式(續)

- 函式strstr會搜尋它的第一個字串引數，找出第二個字串引數第一次出現的位置。
- 如果找到的話，便傳回指向第二個引數出現在第一個引數內之位置的指標。
- 例題十二程式使用`strstr`來找出字串`"abcdefabcdef"`內的`"def"`字串。

例題十二

```c
/* Fig. 8.28: fig08_28.c
   Using strstr */
#include <stdio.h>
#include <string.h>

int main( void )
{
   const char *string1 = "abcdefabcdef"; /* string to search */
   const char *string2 = "def"; /* string to search for */

   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; /* indicates successful termination */
} /* end main */
```

```
string1 = abcdefabcdef
string2 = def

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

4. 字串操作函式(續)

- 函式strtok (例題十三)用來將字串切成數個字符 **(token)**。
- 字符是由分界字元 (delimiters，通常爲空白或標點符號，但分界字元可以是任何字元)所分隔出的一連串字元。
- 例如在一行文字中，每個字可視爲一個字符，而分隔這些字的空白則可視爲分界字元。

例題十三

```c
/* Fig. 8.29: fig08_29.c
   Using strtok */
#include <stdio.h>
#include <string.h>

int main( void )
{
   /* initialize array string */
   char string[] = "This is a sentence with 7 tokens";
   char *tokenPtr; /* create char pointer */

   printf( "%s\n%s\n\n%s\n",
      "The string to be tokenized is:", string,
      "The tokens are:" );

   tokenPtr = strtok( string, " " ); /* begin tokenizing sentence */

   /* continue tokenizing sentence until tokenPtr becomes NULL */
   while ( tokenPtr != NULL ) {
      printf( "%s\n", tokenPtr );
      tokenPtr = strtok( NULL, " " ); /* get next token */
   } /* end while */

   return 0; /* indicates successful termination */
} /* end main */
```

例題十三(續)

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

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

作業

請做作業8_1: 計算一組字串裡單字(含數字)的數目
步驟：

1. 參考**fig08_13.c, fig08_14.c .6由鍵盤**輸入一組字串。
2. 參考**fig08_29.c**計算一組字串中單字的數目，如**“I am a young man of 15 years”**，共有**8**個單字。
3. 以**c**程式函式方式撰寫
4. **其結果印在螢幕上**
5. 請參考指令如**fgets, getchar, strtok.**