

程式設計 (一)

CH9. 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, 2021

本章目錄

1. 字元與字串
2. 字串宣告
3. `<stdio.h>` 標準輸入/輸出函式庫
4. `<ctype.h>` 字元處理函式庫
5. `<stdlib.h>` 公用函式庫 - 字串轉換
6. `<string.h>` 字串處理函式庫

字元與字串

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。

我們以單引號括起來的字元來表示常數 (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。

(補充) 字元常數

單引號內可放入 1~4 個字元，一個字元會代表 int 中的 1 個 byte。
因為 int 大小為 4bytes，故不可放入 5 個以上的字元於單引號中。

```
1 // character constant
2 #include <stdio.h>
3
4 int main() {
5     printf("val of 'a'      : (dec) %10d (hex) %8x\n", 'a', 'a');
6     printf("val of 'abc'    : (dec) %10d (hex) %8x\n", 'abc', 'abc');
7     printf("val of 'abcd'   : (dec) %10d (hex) %8x\n", 'abcd', 'abcd');
8 }
```

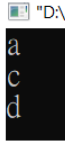
9 "D:\C program\character constant.exe"

```
val of 'a'      : (dec)          97 (hex)       61
val of 'abc'    : (dec)       6382179 (hex)    616263
val of 'abcd'   : (dec) 1633837924 (hex) 61626364
```

(補充)

單引號內放入 2~4 個字元，使用 %c 印出，會因為數值大於 char 範圍導致溢位，只印出最後一個字元。

```
1 // character constant
2 #include <stdio.h>
3
4 int main() {
5     printf("%c\n", 'a');
6     printf("%c\n", 'abc');
7     printf("%c\n", 'abcd');
8 }
```



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 string constant) 會寫在雙引號裡。

我們平常放在 `printf` 或 `scanf` 的第一個引數的，就是字串常數。

我們可以使用%s 將字串印出

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     int years = 22;
6     printf("%s is %d years old.\n", "Tom", years);
7     printf("He is %s.\n", years < 18 ? "a minor" : "an adult");
8 }
9
```

"D:\C program\string.exe"

```
Tom is 22 years old.
He is an adult.
```


字符串宣告

字串宣告

C 語言的字串分為 2 種：

- 宣告於字元陣列中：
在宣告之後，字串可做修改
- 指定給字串指標變數：
在宣告之後不可做修改 (字串常數)

指標變數是一種可以儲存記憶體位址的資料型態，於〈程式設計 III〉會詳細介紹，故本章節只討論「字元陣列中的字串」。

將字串指定在字元陣列中，並將其印出

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     char name[] = "Tom";
6     char phone[] = "(201) 555-1212";
7     int years = 22;
8     printf("%s is %d years old.\n", name, years);
9     printf("The phone number is %s.\n", phone);
10 }
```

11 "D:\C program\string.exe"

```
Tom is 22 years old.
The phone number is (201) 555-1212.
```

字串宣告

宣告字串於字元陣列中的格式有 2 種，
一種是使用雙引號代表字串，
一種是使用一般陣列宣告時使用的大括號，並注意
在最後要有一個結束字元 ('\0'，ASCII 碼為 0)

```
char name1[] = "Tom";  
char name2[] = {'T', 'o', 'm', '\0'};
```

字串宣告

當我們宣告一個字元陣列來存放字串時，這個陣列必須能夠放進整個字串和字串的結束字元。

```
char name1_1[3] = "Tom"; //不行，太小'\0'放不進去
char name1_2[4] = "Tom"; //可以，大小剛好
char name1_3[5] = "Tom"; //可以，多出最後一個元素，初始化為0

char name2_1[] = {'T', 'o', 'm', '\0'}; //可以
char name2_2[] = {'T', 'o', 'm'}; //不行，沒有結束字元
char name2_3[4] = {'T', 'o', 'm'}; /*可以，index 3的元素會初始化為0，
                                     ASCII碼0正好是結束字元*/
```

字串宣告

字元陣列中的字串在宣告之後，不能使用指派雙引號的字串做修改。

```
char name1[10] = "Tom";
```

```
char name2[10];
```

```
name2 = "Tom"; //ERROR!
```

//陣列宣告之後不能再如同宣告時使用雙引號指派

二維字元陣列 (字串陣列)

我們可以使用二維字元陣列來儲存多個字串

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     char name[3][10] = { "Tom", "Jason", "Anna"};
6     for (int i = 0; i < 3; i++) {
7         printf("%s\n", name[i]);
8     }
9 }
10
```

"D:\C program\string.exe"

Tom
Jason
Anna

<stdio.h> 標準輸入/輸出函式庫

字元與字串的輸入輸出

本小節會介紹的函式：

- getchar、 putchar
- gets、 puts
- scanf、 printf
- sscanf、 sprintf

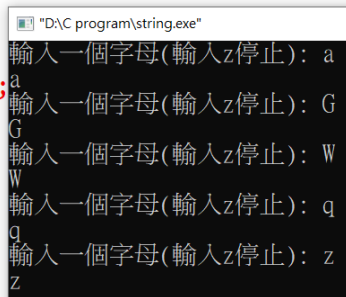
getchar、 putchar

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

getchar 會從標準輸入讀入一個字元，
並回傳該字元的 ASCII 碼。
putchar 會輸出一個字元到標準輸出，
並回傳該字元的 ASCII 碼。

getchar、 putchar 使用範例

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     char c;
6     do {
7         printf("輸入一個字母(輸入z停止): ");
8         c = getchar(); //讀取字母
9         getchar(); //讀取'\n'
10        putchar(c);
11        printf("\n");
12    }while (c != 'z');
13 }
```



"D:\C program\string.exe"

輸入一個字母(輸入z停止): a
a
輸入一個字母(輸入z停止): G
G
輸入一個字母(輸入z停止): W
W
輸入一個字母(輸入z停止): q
q
輸入一個字母(輸入z停止): z
z

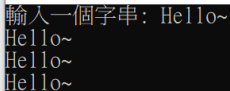
gets、 puts

```
//型態char *代表字串  
char * gets ( char * str );  
int puts ( const char * str );
```

gets 會從標準輸入讀入一行字 (直到換行),
並存入字串 (字元陣列)str 中。
puts 會將字串印出, 並在結尾換行。

gets、puts 使用範例

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     char s[100];
6     int n;
7     printf("輸入一個字串: ");
8     gets(s);
9     for (int i = 0; i < 3; i++)
10         puts(s);
11 }
12
```



"D:\C program\string.exe"
輸入一個字串: Hello~
Hello~
Hello~
Hello~

scanf、printf

```
int scanf ( const char * format, ... );  
int printf ( const char * format, ... );
```

scanf 與 printf 可使用%c 來輸入輸出字元，也可使用%s 來輸入輸出字串。另外，scanf 也可使用掃描集 (scan set) 來輸入字串。

使用 scanf 輸入字串時，字串的名稱前面不要加上取址運算子 (&)，因為該名稱本身就代表字串開始位址。

<stdio.h> 標準輸入/輸出函式庫

使用%s 輸入字串時，會從標準輸入讀取字元直到遇到空白字元，或是指定讀取的最大寬度。

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     char w[10];
6     while (scanf("%9s", w) != EOF) {
7         printf("> ");
8         puts(w);
9     }
10 }
```

"D:\C program\string.exe"

```
The cooooooooookies tastes great.
> The
> coooooooooo
> okies
> tastes
> great.
^Z
```

掃描集 (scan set)

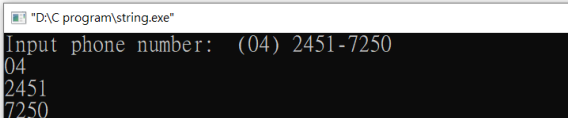
掃描集可以指定讀取的字串可包含的字元，scanf 會從標準輸入讀取字元直到遇到非指定的字元。

掃描集由中括號 [] 包住，並由「-」代表連續：

- %[aeiou] 代表指定'a'、'e'、'i'、'o'、'u'
 - %[0-9A-Za-z] 代表指定全部英文字母與數字
 - %[-+*/%] 代表指定'-'、'+'、'*'、'/'、'%'
- 若掃描集內需指定字元'-', 則須放在最前面。

掃描集使用範例

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     char phone[4][10] = {0}, tmp = 0, n;
6     printf("Input phone number: ");
7     for (n = 0; n < 4 && tmp != '\n'; n++) {
8         scanf("%*[-() ]"); //星號(*)為設定禁止字元，讀取但不存入變數
9                             //連續讀取字元，直到遇到非'-'、'(',')'、' '的字元
10        scanf("%[0-9]", phone[n]); //連續讀取字元，直到遇到非'0'~'9'的字元
11        tmp = getchar();
12    }
13    for (int i = 0; i < n; i++)
14        puts(phone[i]);
15 }
```



反掃描集 (inverted scan set)

反掃描集可以指定讀取的字串不可包含的字元，scanf 會從標準輸入讀取字元直到遇到指定的字元。在掃描集的掃描字元前面加上一個脫字符號 \wedge (caret)，就可建立反掃描集。

掃描集與反掃描集使用範例

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     char number[100][10] = {0}, n = 0;
6     do {
7         scanf("%*[^0-9\n]");
8         scanf("%[0-9]", number[n++]);
9     } while (getchar() != '\n');
10    puts("-----");
11    for (int i = 0; i < n; i++)
12        puts(number[i]);
13 }
```

"D:\C program\string.exe"

小明去買水果，總共買了10顆蘋果，8顆橘子，1顆西瓜

10

8

1

sscanf、sprintf

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

sscanf 和 sprintf 與 scanf 和 printf 的差別在於，sscanf 和 sprintf 的輸入輸出目標是字元陣列 s，而非標準輸入 (鍵盤) 或標準輸出 (螢幕)。

要注意，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 也一樣，每一次輸出都會覆蓋原本的字串。

sscanf、sprintf 使用範例

```
1 // string
2 #include <stdio.h>
3
4 int main() {
5     char sentence[] = "This pen cost me 30 dollars.";
6     char number[10] = "", result[100] = "";
7     sscanf(sentence, "%*[^0-9]%[0-9]", number);
8     sprintf(result, "The first number appears in the sentence is %s.", number);
9     puts(result);
10 }
```

選取 "D:\C program\string.exe"

The first number appears in the sentence is 30.

<ctype.h> 字元處理函式庫

<ctype.h> 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。每個函式都接收了一個無號字元 (表示為 int) 或 EOF 做為引數。

<ctype.h> 字元處理函式庫

本小節會介紹的函式：

- isdigit、isalpha、isalnum、isxdigit
- islower、isupper、tolower、toupper
- isblank、isspace、iscntrl
- ispunct、isprint、isgraph

<ctype.h> 中，is 開頭的函式會檢查引數的值 (字元的 ASCII 碼) 是否符合該函式欲檢查的分類 (數字、英文字母、標點符號...)，並回傳 0(false) 或 非 0(true) 的值。

isdigit、isalpha、isalnum、isxdigit


```
int isdigit ( int c );  
int isalpha ( int c );  
int isalnum ( int c );  
int isxdigit ( int c );
```

| 函式 | 檢查分類 | 符合的字元 |
|----------|---------|---------------------------|
| isdigit | 阿拉伯數字 | '0'~'9' |
| isalpha | 英文字母 | 'A'~'Z', 'a'~'z' |
| isalnum | 英數 | '0'~'9', 'A'~'Z', 'a'~'z' |
| isxdigit | 16 進位數字 | '0'~'9', 'A'~'F', 'a'~'f' |

<ctype.h> 字元處理函式庫

<ctype.h> 判斷字元的函式常搭配 if 判斷句或三元運算子來做使用。isdigit、isalpha 使用範例：

```
1 // ctype
2 #include <stdio.h>
3 #include <ctype.h>
4
5 int main() {
6     char input;
7     while(1) {
8         printf("輸入一個字元(輸入x結束) : ");
9         input = getchar();
10        if (input == 'x')
11            break;
12        getchar();
13        puts(isalpha(input) ? "letter" : "not letter");
14        puts(isdigit(input) ? "number" : "not number");
15    }
16 }
```



```
D:\codeblocks\ctype.exe
輸入一個字元(輸入x結束) : a
letter
not number
輸入一個字元(輸入x結束) : 5
not letter
number
輸入一個字元(輸入x結束) : &
not letter
not number
輸入一個字元(輸入x結束) : x
```

islower、isupper、tolower、toupper

```
int islower ( int c );  
int isupper ( int c );  
int tolower ( int c );  
int toupper ( int c );
```

- islower 會判斷該字元是否為小寫英文字母 ('a'~'z')。
- isupper 會判斷該字元是否為大寫英文字母 ('A'~'Z')。
- tolower 會將大寫字母轉換成小寫字母，並傳回轉換過的小寫字母。
- toupper 會將小寫字母轉換成大寫字母，並傳回轉換過的大寫字母。

toupper 使用範例：

```
1 // ctype
2 #include <stdio.h>
3 #include <ctype.h>
4
5 int main() {
6     char input[100] = {0}, i = 0;
7     printf("輸入一行英文 : ");
8     gets(input);
9     while (input[i] != '\0') {
10         input[i] = toupper(input[i]);
11         i++;
12     }
13     printf("轉換成大寫 : %s\n", input);
14 }
15
```

D:\codeblocks\ctype.exe
輸入一行英文 : I got 100 points in this mid-term exam
轉換成大寫 : I GOT 100 POINTS IN THIS MID-TERM EXAM

isblank、isspace、iscntrl

```
int isblank ( int c );  
int isspace ( int c );  
int iscntrl ( int c );
```

| 函式 | 檢查分類 | 符合的字元 |
|---------|------|-----------------------------------|
| isblank | 空格 | '\t', ' ' |
| isspace | 空白字元 | '\t', '\n', '\v', '\f', '\r', ' ' |
| iscntrl | 控制字元 | ASCII 碼 0~31, 127 |

※ '\t', '\n', '\v', '\f', '\r' 的 ASCII 碼分別是 9~13

ispunct、isprint、isgraph

```
int ispunct ( int c );  
int isprint ( int c );  
int isgraph ( int c );
```

| 函式 | 檢查分類 | 符合的字元 |
|---------|--------------|---------------------------------|
| ispunct | 英式符號 (不含空白) | !"#\$%&'()*+,-./:;<=>?@[\\]^_`{ |
| isprint | 可顯示字元 | ASCII 碼 32~126 |
| isgraph | 可顯示字元 (不含空白) | ASCII 碼 33~126 |

<stdlib.h> 公用函式庫 - 字串轉換

<stdlib.h> 公用函式庫 - 字串轉換

公用函式庫 (general utilities library) <stdlib.h> 定義了許多常用的函式，包含字串轉換、隨機、記憶體分配、排序搜尋、整數取絕對值與除法運算等函式。詳細可參考 **cplusplus** 或**維基百科**等網站。本小節介紹的字串轉換函式 (string conversion function) 可以將由數字所組成的字串轉換成整數或浮點數值。

本小節會介紹的函式：

- atof (ascii to float)
- atoi (ascii to int)
- atoll (ascii to long long)

另外，<stdlib.h> 的字串轉換函式還提供了 strtod、strtoi、strtoll... 等函式，因為需要使用到指標觀念，故留到第 13 章。

atof、atoi、atoll

```
double atof (const char* str);  
int atoi (const char * str);  
long long int atoll ( const char * str );
```

atof 是 ascii to float 的縮寫，會將數字組成的字串轉換成 double 並將該值回傳。

atoi 是 ascii to int 的縮寫，會將數字組成的字串轉換成 int 並將該值回傳。

atoll 是 ascii to long long 的縮寫，會將數字組成的字串轉換成 long long int 並將該值回傳。

<stdlib.h> 公用函式庫 - 字串轉換

atof 雖然字面上是寫著「轉換成 float」，但實際上卻是轉換成 double。以下為 atof 的轉換測試：

```
1 // string conversion function
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 int main() {
6     char str1[] = "360";
7     char str2[] = "3.14159265358979";
8     char str3[] = "3.141 592 653 589 79"; //有非數字的字元間隔
9     char str4[] = "PI = 3.14159265358979"; //開頭非數字
10    printf("%.14f\n", atof(str1));
11    printf("%.14f\n", atof(str2));
12    printf("%.14f\n", atof(str3));
13    printf("%.14f\n", atof(str4));
14 }
```

"D:\codeblocks\string conversion function.exe"

```
360.00000000000000
3.14159265358979
3.14100000000000
0.00000000000000
```

atof、atoi 使用範例

```
1 // string conversion function
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 int main() {
6     char str[] = "PI = 3.14159";
7     char numberInStr[10] = {0};
8     double dNumber;
9     int iNumber;
10    sscanf(str, "%[^0-9.%][0-9.]", numberInStr); //取出字串中首次出現的數字
11    dNumber = atof(numberInStr);
12    iNumber = atoi(numberInStr);
13    printf("converted to double: %f\nconverted to int: %d\n", dNumber, iNumber);
14 }
15
```

"D:\codeblocks\string conversion function.exe"

```
converted to double: 3.141590
converted to int: 3
```

<string.h> 字串處理函式庫

計算字串長度、操作字串資料、比較字串

<string.h> 字串處理函式庫

字串處理函式庫 <string.h> 提供了許多函式，包括操作字串資料 (複製 copy、連接 concatenate)、比較兩個字串 (comparing strings)、搜尋字串當中的某些字元或其他字串、計算字串長度... 等函式。詳細可參考 **cplusplus** 或**維基百科**等網站。

本小節將介紹計算字串長度、操作字串資料、比較字串的相關函式，其他函式將留到第 13 章再做介紹。

<string.h> 字串處理函式庫

本小節會介紹的函式：

- 計算字串長度：strlen
- 複製字串：strcpy、strncpy
- 連接字串：strcat、strncat
- 比較字串：strcmp、strncmp

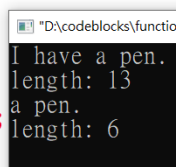
計算字串長度：strlen

```
size_t strlen ( const char * str );  
//size_t是一種無號整數
```

strlen 會計算字串 str 的長度 (記憶體位址 str 到第一次遇到的'\0' 之間的長度，並非陣列大小。)

strlen 使用範例

```
1 // function in string.h
2 #include <stdio.h>
3 #include <string.h>
4
5 int main() {
6     char sentence[50] = "I have a pen.";
7     printf("%s\n", sentence);
8     printf("length: %d\n", strlen(sentence));
9     printf("%s\n", sentence + 7);
10    printf("length: %d\n", strlen(sentence + 7));
11 }
```



```
"D:\codeblocks\functio
I have a pen.
length: 13
a pen.
length: 6
```

關於此程式碼第 9、10 行的 `sentence + 7`，此為表示 `sentence[7]` 的記憶體位址，也可以說是由 `sentence[7]` 當作字串起點的字串。

複製字串：strcpy、strncpy

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

函式 strcpy 會將陣列 source 中整個字串複製 (覆蓋) 給陣列 destination，並在最後補上'\0'。

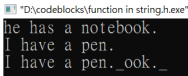
函式 strncpy 會將陣列 source 中的前 num 個字元複製 (覆蓋) 給陣列 destination，但不會在最後補上'\0'。

須注意，destination 的陣列大小需大於欲複製的字串長度。

<string.h> 字串處理函式庫

strcpy 會將第二個引數的字串覆蓋到第一個引數，並在結尾補上一個'\0'。在以下範例輸出的最後一行實測 strcpy 只會複製並覆蓋「來源字串」的長度大小。

```
1 // function in string.h
2 #include <stdio.h>
3 #include <string.h>
4
5 int main() {
6     char sentence1[50] = "I have a pen.";
7     char sentence2[50] = "he has a notebook.";
8     char result[50] = {0};
9     strcpy(result, sentence2);
10    puts(result);
11    strcpy(result, sentence1);
12    puts(result);
13    for (int i = 0; i <= strlen(sentence2); i++) //將result逐字印出
14        putchar(result[i] != '\0' ? result[i] : '_'); //遇到'\0'時，由 '_' 表示
15    puts("");
16 }
```



The terminal output shows three lines: "he has a notebook.", "I have a pen.", and "I have a pen._ook._". The first two lines are the result of strcpy, and the third line is the result of the custom loop that prints characters until the null terminator, which is replaced by an underscore.

<string.h> 字串處理函式庫

strncpy 需指定複製的長度，並在覆蓋到第一個引數的字串之後不會在結尾補上'\0'。

```
1 // function in string.h
2 #include <stdio.h>
3 #include <string.h>
4
5 int main() {
6     char sentence1[50] = "I have a pen.";
7     char sentence2[50] = "he has a notebook.";
8     strncpy(sentence2, sentence1, 6);
9     puts(sentence2);
10 }
11
```

選取 "D:\codeblocks\function in string.h.exe"

I have a notebook.

連接字串：strcat、strncat

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

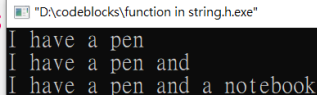
函式 `strcat` 會將字串 `source` 複製字串 `destination` 的尾端，
並在最後補上 `'\0'`。

函式 `strncat` 會將字串 `source` 的前 `num` 個字元複製到字串
`destination` 的尾端，並在最後補上 `'\0'`。

須注意，`destination` 的陣列大小需大於串接後的字串長度。

strcat、strncat 使用範例

```
1 // function in string.h
2 #include <stdio.h>
3 #include <string.h>
4
5 int main() {
6     char sentence1[50] = "I have a pen";
7     char conjunction[50] = " and";
8     char sentence2[50] = "He has a notebook and a pencil";
9     puts(sentence1);
10    strcat(sentence1, conjunction);
11    puts(sentence1);
12    strncat(sentence1, sentence2 + 6, 11);
13    puts(sentence1);
14 }
```



"D:\codeblocks\function in string.h.exe"

```
I have a pen
I have a pen and
I have a pen and a notebook
```

比較字串：strcmp、strncmp

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

strcmp 和 strncmp 會比較字串 str1 與 str2，由字串的第 0 個元素開始比較兩者的 ASCII 碼，若相等則比較下一個元素。strcmp 最多會比較兩字串直到其中一方的字串結尾，strncmp 則會比較最多 num 個字元。

若 str1 小於 str2 則回傳負值，若 str1 大於 str2 則回傳正值，若 str1 與 str2 相等則回傳 0。

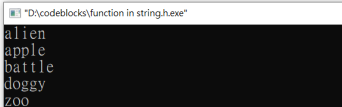
strcmp、strncmp 使用範例

```
1 // function in string.h
2 #include <stdio.h>
3 #include <string.h>
4
5 int main() {
6     char str1[] = "Happy New Year";
7     char str2[] = "Happy New Year";
8     char str3[] = "Happy Halloween";
9     printf("\'%s\' and \''s\' is %s.\n",
10         str1, str2, strcmp(str1, str2) ? "NOT same" : "same");
11     printf("\'%s\' and \''s\' is %s.\n",
12         str1, str3, strcmp(str1, str3) ? "NOT same" : "same");
13     printf("First 5 characters of \''s\' and \''s\' is %s.\n",
14         str1, str3, strncmp(str1, str3, 5) ? "NOT same" : "same");
15 }
16
```

"Happy New Year" and "Happy New Year" is same.
"Happy New Year" and "Happy Halloween" is NOT same.
First 5 characters of "Happy New Year" and "Happy Halloween" is same.

使用 strcmp 與 strcpy 排序字串陣列

```
1 // function in string.h
2 #include <stdio.h>
3 #include <string.h>
4 #define LENGTH 5
5
6 int main() {
7     char words[LENGTH][10] = {"zoo", "apple", "battle", "alien", "doggy"};
8     char tmp[10] = {0};
9     for (int i = 0; i < LENGTH - 1; i++) { //bubble sort
10         for (int j = 0; j < LENGTH - i - 1; j++){
11             if(strcmp(words[j], words[j + 1]) > 0) { //swap
12                 strcpy(tmp, words[j]);
13                 strcpy(words[j], words[j + 1]);
14                 strcpy(words[j + 1], tmp);
15             }
16         }
17     }
18     for (int i = 0; i < LENGTH; i++)
19         puts(words[i]);
20 }
```



參考資料： Deitel, H. M., & Deitel, P. J. (2015). C: How to program.
Upper Saddle River, N.J: Prentice Hall.