

# 計算機程式設計

## C語言 String

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 字串宣告與初值設定

❑ 字元以單引號包圍，而字串則是以雙引號包圍：

- 'a'                      /\* 這是字元常數 a \*/
- "a"                     /\* 這是字串常數 a \*/
- "Sweet home"        /\* 這是字串常數 Sweet home \*/

```
char 字元陣列名稱[陣列大小] = 字串常數;
```

```
char str[] = "Sweet home";
```

0	1	2	3	4	5	6	7	8	9	10
S	w	e	e	t		h	o	m	e	\0

# 字元與字串之比較

- ❑ 字元以單引號包圍，而字串則是以雙引號包圍：

```
01 #include <stdio.h>
02 int main() {
03     char ch = 'a';
04     char str1[] = "a";
05     char str2[] = "Sweet home";
06     printf("ch size=%d\n", sizeof(ch));
07     printf("str1 size=%d\n", sizeof(str1));
08     printf("str2 size=%d\n", sizeof(str2));
09 }
```

```
ch size=1
str1 size=2
str2 size=11
```

# 字串輸入與輸出函數

## □ fgets() 與 puts()

- fgets(字元陣列名稱, 輸入字元數, stdin);
- puts(字元陣列名稱); 或 puts(字串常數);

```
01 #include <stdio.h>
02 int main() {
03     char name[5];
04     puts("What's your name?");
05     fgets(name, 5, stdin);
06     puts("Hi!");
07     puts(name);
08 }
```

```
What's your
name?
JJ
Hi!
JJ
```

# 字串輸入與輸出函數

## □ 大小寫的轉換

```
01 #include <stdio.h>
02 void toUpper(char s[]){
03     int i = 0;
04     while(s[i] != '\0'){
05         if(s[i]>=97 && s[i]<=122){
06             s[i] -= 32;
07             i++;
08         }
09     }
10 }
11 int main() {
12     char str[15];
13     puts("請輸入一個字串");
14     fgets(str, 15, stdin);
15     toUpper(str);
16     printf("轉成大寫後: %s", str);
17     return 0;
18 }
```

第4~10行，若字元陣列無'\0'，將變成，無窮迴圈。為設計安全程式，需設定i大於某數值時跳出迴圈。

請輸入一個字串  
abcdefg  
轉成大寫後:  
ABCDEFG

# 問題

```
01 #include <stdio.h>
02 void com(char *a, char **aa) {
03     aa=a;           // 資料型別錯誤，編譯器警告
04     printf("%s\n", aa);
05 }
06 int main() {
07
08     char *a="hello";
09     char *aa;
10     com(a, aa);
11     return 0;
12 }
```

# 字元陣列與字串常數

```
#include <stdio.h>
int main() {
    char ch = 'a';
    char str1[] = "a";
    //str2是陣列，有自己獨立的連續空間
    // 將11個字元複製到此空間
    char str2[] = "Sweet home";
    //編譯器創造11個字元空間，常數
    // str3, str4 指向同一個空間
    char *str3 = "Sweet home";
    char *str4 = "Sweet home";
    printf("ch size=%d\n", sizeof(ch));
    printf("str1 size=%d\n", sizeof(str1));
    printf("str2 size=%d\n", sizeof(str2));
    printf("str3 size=%d\n", sizeof(str3));
    printf("str4 size=%d\n", sizeof(str4));
    printf("str2 =%x\n", str2);
    printf("str3 =%x\n", str3);
    printf("str4 =%x\n", str4);
}
```

```
ch size=1
str1 size=2
str2 size=11
str3 size=8
str4 size=8
str2 =61fdfb
str3 =404000
str4 =404000
```

str2是陣列，有11個空間

str3,4是指標，有8個空間

// str3, str4 指向同一個空間

# 字元陣列與字串常數

```
#include <stdio.h>
void print(char *s2, char *s3, char *s4) {
    printf("s2 =%s\n", s2);
    printf("s3 =%s\n", s3);
    printf("s4 =%s\n", s4);
}
int main() {
    char str2[] = "Sweet home";
    //編譯器創造11個字元空間，常數
    // str3, str4 指向同一個空間
    char *str3 = "Sweet home";
    char *str4 = "Sweet home";
    print(str2, str3, str4);
    str2[0]='X';
    print(str2, str3, str4);
    // 修改到str3, str4共同指向的空間
    // 空間內為常數，故產生錯誤
    str3[0]='P';
    print(str2, str3, str4);
}
```

// 修改str2空間的值，不會影響到str3, str4

```
s2 =Sweet home
s3 =Sweet home
s4 =Sweet home
s2 =Xweet home
s3 =Sweet home
s4 =Sweet home
```

Process returned -1073741819 (0xC0000005)



# 字串陣列

## ❑ 字串陣列的宣告

○char 字元陣列名稱[字串的個數][字串長度];

## ❑ 字串陣列的宣告與初值設定

○char 字元陣列名稱[字串的個數][字串長度]= {"字串常數1", "字串常數2", ..., "字串常數n"};

```
char customer[6][15];  
char S[3][10]={"Tom","Lily","James Lee"};
```

# 字串陣列元素的存取

## □ 字串陣列

S[0]	0022FE20
S[1]	0022FE2A
S[2]	0022FE34

0	1	2	3	4	5	6	7	8	9	
T	o	m	\0							
L	i	l	y	\0						
J	a	m	e	s	\0					

```
01 #include <stdio.h>
02 int main() {
03     char S[3][10] = {"Tom", "Lily", "James"};
04     int i;
05     for(i=0; i<3; i++)
06         printf("S[%d]=%s\n", i, S[i]);
07     printf("\n");
08     for(i=0; i<3; i++){
09         printf("S[%d]=%p\n", i, S[i]);
10         printf("S[%d][0]=%p\n", i, &S[i][0]);
11     }
12 }
```

S[0]=Tom  
S[1]=Lily  
S[2]=James

S[0]=0022FE20  
S[0][0]=0022FE20  
S[1]=0022FE2A  
S[1][0]=0022FE2A  
S[2]=0022FE34  
S[2][0]=0022FE34

# 複製字串陣列

## ❑ 字串陣列

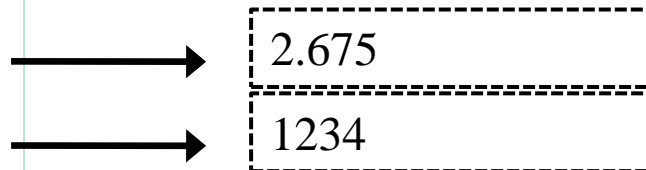
```
01 #include <stdio.h>
02 #define MAX 3
03 #define LENGTH 10
04 int main() {
05     char S1[MAX][LENGTH] = {"Tom", "Lily", "James"};
06     char S2[MAX][LENGTH];
07     int i, j;
08     for(i=0; i<MAX; i++){
09         for(j=0; j<LENGTH; j++){
10             if(S1[i][j] == '\0') break;
11             else S2[i][j] = S1[i][j];
12         }
13         S2[i][j]='\0';
14     }
15     for(i=0; i<MAX; i++)
16         printf("S2[%d]=%s\n", i, S2[i]);
17 }
```

S2[0]=Tom  
S2[1]=Lily  
S2[2]=James

# 字串轉換函數

函數	說明
double atof(char *)	將參數字串轉換成浮點數，如果字串不能轉換傳回0.0
int atoi(char *)	將參數字串轉換成整數，如果字串不能轉換傳回0
long atol(char *)	將參數字串轉換成長整數，如果字串不能轉換傳回0

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 int main() {
04     char buffer1[] = "2.675";
05     char buffer2[] = "1234";
06     double f = atof(buffer1);
07     int n = atoi(buffer);
08     printf("%0.3f\n", f);
09     printf("%d\n", n);
10     return 0;
11 }
```

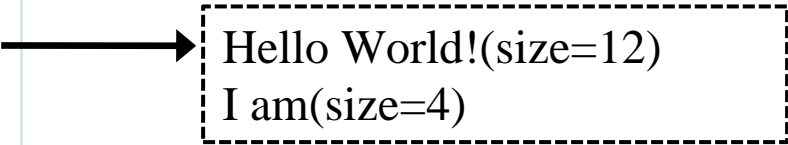


# 字串處理函數

函數	說明
size_t strlen(char *)	傳回參數字串的長度
char *strcpy(char *d, char *s)	將參數字串s複製到字串d，傳回字串d的指標
char *strncpy(char *d, char *s, size_t n)	將參數字串s複製最多n個字元到字串d，傳回字串d的指標，需要自行加上字串結束字元'\0'
char *strcat(char *d, char *s)	將參數字串s連接到字串d之後，傳回字串d的指標
char *strncat(char *d, char *s, size_t n)	將參數字串s連接最多n個字元到字串d之後，傳回字串d的指標
int strcmp(char *d, char *s)	比較參數字串s與字串d， $d < s$ 傳回負值； $d = s$ 傳回0； $d > s$ 傳回正值
int strncmp(char *d, char *s, size_t n)	比較參數字串s與字串d的前n個字元， $d < s$ 傳回負值； $d = s$ 傳回0； $d > s$ 傳回正值
char *strchr(char *d, char c)	傳回指標指向在參數字串d中第1次出現字元c的位置，如果沒有傳回NULL
char *strrchr(char *d, char c)	傳回指標指向在參數字串d中最後1次出現字元c的位置，沒有傳回NULL
char *strstr(char *d, char *s)	傳回指標指向在參數字串d中第1次出現字串s的位置，沒有傳回NULL

# 字串處理函數－複製

```
01 #include <stdio.h>
02 #include <string.h>
03 int main() {
04     char str[50];
05     char str2[50] = "I am John!";
06     int len;
07     strcpy(str, "Hello World!");
08     len = strlen(str);
09     printf("%s(size=%d)", str, len);
10     memset(str, '\0', sizeof(str));
11     strncpy(str, str2, 4);
12     len = strlen(str);
13     printf("%s(size=%d)", str, len);
14 }
```



Hello World!(size=12)  
I am(size=4)

# 字串處理函數－複製

## □ 自訂複製函式

```
#include <stdio.h>
#include <string.h>
void myStrcpy(char * s1, char *s2) {
    for (;(*s1=*s2)!='\0';s1++, s2++);
}
int main() {
    char str[50];
    char str2[50] = "I am John!";
    myStrcpy(str, "Hello World!");
    printf("%s\n", str);
    myStrcpy(str, str2);
    printf("%s\n", str);
    return 0;
}
```



Hello World!  
I am John!

# Exercise

## ❑ 自訂strlen計算字串長度函式

```
#include <stdio.h>
#include <string.h>
int myStrlen(char * s) {
    int r=0;
    for (;*s!='\0';s++) {

    }
    return r;
}
int main() {
    char str[50] = "I am John!";
    int len = myStrlen("Hello World!");
    printf("%d\n", len);
    len = myStrlen(str);
    printf("%d\n", len);
    return 0;
}
```



# Exercise


- ❑ 自訂strlen計算字串長度函式，遞迴

```
#include <stdio.h>
#include <string.h>
int myStrlen(char * s) {
    int r=0;

    return r;
}
int main() {
    char str[50] = "I am John!";
    int len = myStrlen("Hello World!");
    printf("%d\n", len);
    len = myStrlen(str);
    printf("%d\n", len);
    return 0;
}
```

# 字串處理函數－比較

```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[10]="1234";
    char str2[10]="123456";
    int r = strcmp(str1,str2);
    printf("result=%d\n", r);
    r = strncmp(str1,str2,4);
    printf("result=%d\n", r);
}
```



result=-1  
result=0

# 字串處理函數－比較

## □ 自訂比較函式

```
#include <stdio.h>
#include <string.h>
int myStrcmp(char *s1, char *s2) {
    for (;(*s1!='\0')&&(*s2!='\0');s1++, s2++) {
        if (*s1>*s2) return 1;
        else if (*s1<*s2) return -1;
    }
    if (*s1!='\0') return 1;
    else if (*s2!='\0') return -1;
    return 0;
}
int main() {
    char str1[10]="1234";
    char str2[10]="123";
    int r = myStrcmp(str1,str2);
    printf("result=%d\n", r);
}
```

# Exercise

## □ 自訂比較函式-遞迴

```
#include <stdio.h>
#include <string.h>
int myStrcmp(char *s1, char *s2) {

    return 0;
}
int main() {
    char str1[10]="1234";
    char str2[10]="123";
    int r = myStrcmp(str1,str2);
    printf("result=%d\n", r);
}
```

# 字串處理函數—切割

00  
22  
33  
4B  
55  
5A

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

```
int main() {
    char str[]="00:22:33:4B:55:5A";
    char *delim = ":";
    char * pch;
    pch = strtok(str, delim);
    while (pch != NULL) {
        printf ("%s\n",pch);
        pch = strtok (NULL, delim);
    }
}
```

		\0			\0			\0			\0			\0				
0	0	:	2	2	:	3	3	:	4	B	:	5	5	:	5	A	\0	
pch			pch															

# Exercise

## □ 自訂切割函式

```
#include <stdio.h>
#include <string.h>
char* myStrtok(char *s1, char *s2) {
    static char *p;
    if (s1==NULL) s1 = p;
    else p = s1;
    if (*p=='\0') return NULL;
    while ((*p)!=(*s2)) {
        p++;
    }
    *p=='\0';
    p++;
    return s1;
}
```

```
int main() {
    char str[]="00:22:33:4B:55:5A";
    char *delim = ":";
    char * pch;
    pch = myStrtok(str, delim);
    while (pch != NULL) {
        printf ("%s\n",pch);
        pch = myStrtok (NULL, delim);
    }
}
```

S1																	
0	0	:	2	2	:	3	3	:	4	B	:	5	5	:	5	A	\0
P																	

S1																	
0	0	\0	2	2	:	3	3	:	4	B	:	5	5	:	5	A	\0
			P														

# 字串交換

- ❑ 使用區域變數指標，無法達到交換字串目的。
  - 必須使用指標的指標

```
#include<stdio.h>
void swap(char *str1, char *str2) {
    char *temp = str1;
    str1 = str2;
    str2 = temp;
}
int main() {
    char *str1 = "geeks";
    char *str2 = "forgeeks";
    swap(str1, str2);
    printf("str1 is %s, str2 is %s", str1, str2);
    return 0;
}
```

```
void swap1(char **str1_ptr, char **str2_ptr) {
    char *temp = *str1_ptr;
    *str1_ptr = *str2_ptr;
    *str2_ptr = temp;
}
```

# Homework II

## □ 英文字分析、取代、插入、刪除

- 輸入一篇英文文章 A，文章中英文字以一個空白間隔。另外輸入 2 個英文字(word) P、Q。
  - (1) 將文章 A 中 P 字串以 Q 字串取代，並輸出。
  - (2) 在文章 A 中 P 字串前插入 Q 字串，並輸出。
  - (3) 將文章 A 中 P 字串刪除，並輸出。
  - (4) 分析文章 A 所有英文字 (word) 的頻率，依頻率由大自小排序，頻率相同則以 word 由小自大排序(That > This....)輸出。
- 輸入範例說明：
  - 第一行，文章 A
  - 第二行，英文字 P
  - 第三行，英文字 Q



# Homework II

## ○輸出範例說明:

- 第一行，文章 A 將 P 替換成 Q。
- 第二行，文章 A 將 Q 插入 P 前面。
- 第三行，文章 A 將 P 刪除。
- 第四行之後，每一行依序為英文字、出現頻率次數，中間以逗號間隔。

### Sample Input

This is a book That is a cook  
is  
was

### Sample Output

This was a book That was a cook  
This was is a book That was is a cook  
This a book That a cook  
a 2  
is 2  
That 1  
This 1  
book 1  
cook 1

# 範例一 互補字串I

## □ 互補字串

○ 互補字串S1, S2的定義是

- 字串S1, S2沒有重複出現的字元，
- 且S1和S2內的字元包含所有前M個字元，字元由'A'開始。

○ 字串視為一個集合，元素有重複只算一個，且不管排列情況。

- 例如AABAAB與ABAB與BABA都是相同的字串AB。

輸入 m, n (前m個字元，n個字串)	輸出這N個字串互補的對數
7 4 (前7個字元-ABCDEFG, 4 個字串) AABAAB ABCABCDE CDECD GFFGF	1 (ABCABCDE: GFFGF)
4 3 ABB AB C CC	4 (ABB:C, ABB:CC, AB:CC, AB:C)

# 範例一 互補字串I

## □ 互補字串

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int isIn(char *s, char c, int n) { //判斷c 是否在 s 裡面
    int i=0;
    for (i=0; i<n; i++)
        if (c==s[i]) return 1;
    return 0;
}
char *erase(char *s, int m) { //將重複的部分刪除，變成集合
    int i=0, j=0;
    char *p = (char*) malloc(strlen(s)*sizeof(char));
    char ch = 'A' + m - 1;
    for (; s[i]!='\0'; i++)
        if ((s[i]<=ch)&&(isIn(p, s[i], j)==0)) p[j++] = s[i];
    p[j]='\0';
    return p;
}
```

# 範例一 互補字串I

## □ 互補字串

```
int isDupilcate(char *s1, char *s2) { //判斷 s1, s2 是否有重複字元
    int i=0, j=0;
    printf("%s, %s\n", s1, s2);
    for (i=0; s1[i]!='\0'; i++) {
        for (j=0; s2[j]!='\0'; j++) {
            if (s1[i]==s2[j]) return 1;
        }
    }
    return 0;
}

int isFitLength(char *s1, char *s2, int m) { // 判斷s1+s2的長度是否剛好符合
    if (m==(strlen(s1)+strlen(s2))) return 1;
    else return 0;
}
```

# 範例－互補字串I

## □ 互補字串

```
void compute(char s[][10], int n, int m) {
    char *p[10];
    int i=0, j=0, sum=0;
    for (i=0; i<n; i++) p[i] = erase(s[i], m);
    for (i=0; i<n; i++) {
        for (j=i+1; j<n; j++) { // p[i], p[j] 沒有重複，總和長度又等於 m
            if ((isDupilcate(p[i], p[j])==0) && (isFitLength(p[i], p[j],m)))
                { sum++; }
        }
    }
    printf("sum = %d\n", sum);
}

void test01() {
    char s[][10] = {"AABAAB", "ABCABCDE", "CDECD", "GFFGF"};
    compute(s, 4, 7);
    char s1[][10] = {"ABB", "AB", "C", "CC"};
    compute(s1, 4, 3);
}
```

# 範例一 互補字串II

## □ 互補字串 C++ STL

```
#include <iostream>
#include <string.h>
#include <map>      // map 函式庫
#include <algorithm> // sort 函式庫
#define F first
#define S second
using namespace std;
int solve(int m, int n, char s[][10]) {
    map < string, int > lib;
    int ans = 0;
    string str, basic;
    char c = 'A';
    for ( int i = 0 ; i < m ; i++, c++ )
        basic += c;    //前 m 個形成的基本集合
    cout<<basic<<endl;
```

# 範例一 互補字串II

## □ 互補字串C++ STL

```
while ( n-- ){
    //cin >> str;
    str.assign(s[n]);
    cout<<str<<endl;
    sort ( str.begin(), str.end() ); // 排序統一順序
    //unique會將重複放在最後面，回傳n為前n個字元沒重複
    //erase會刪除(i,j) i~j字元，兩者配合，可刪除重複字元
    str.erase ( unique ( str.begin(), str.end() ), str.end() );
    lib[str]++;    //放入map，計數加一
}
for (auto j:lib) {    //一一尋訪 map lib
    cout<<"j:("<<j.F<<","<<j.S<<")"<<endl;
    for ( auto i: j.F ) //針對尋訪到的 map 元素，取 first，一一尋訪
        cout<<"i:"<<i<<endl;
}
// 例如 {{"ABC", 1},{CD}} 會取出 ABC，再取出 A, B, C，
// 第二次取出 CD，再取出 C, D
```

# 範例一 互補字串II

## □ 互補字串C++ STL

```
for ( auto j: lib ){
    str = basic;
    // 把有出現從基本集合中刪除，即其互補，
    // 比對是否有互補資料，將出現個數相乘，即配對個數
    for ( auto i: j.F )
        str.erase ( lower_bound ( str.begin(), str.end(), i ) );
    ans += j.S * lib[str];
}
return (ans/2); // 重複計算兩次
}
int main(){
    ios::sync_with_stdio ( false );
    cin.tie ( 0 );
    cout.tie ( 0 );
    int n = 4, m = 3; //cin >> n >> m;
    char s[][10] = { "ABB", "AB", "C", "CC" };
    cout<<solve(m, n, s);
    return 0; }
```



# 範例一 互補字串III

## □ 互補字串C

```
#include <stdio.h> #include <string.h>
void print(int n) {
    for (;n>0; n=n/2) printf("%d ", n%2);
    printf("\n");
}
int encode(char *s) { // 編碼 A***E = 10001
    int i=0, code=0; // 向左位移，重複不算，使用 |
    for (i = 0; i != strlen(s); ++i) code |= (1LL << (s[i] - 'A'));
    return code;
}
int match(int code[], int c, int n, int mask) {
    int answer = 0;
    for (int i=0; i<n; i++) {
        //判斷互補 10001 + 01110 = 11111
        // 10001 ^ 11111 = 01110 (XOR)
        //if ((code[i]+c)==mask) answer++;
        if (code[i]==(c^mask)) answer++;
    }
    return answer;}
}
```

# 範例一 互補字串III

## □ 互補字串C

```
int solve(int charSet, int amount, char s[][10]){
    int basic = 0, answer=0, i=0, code[10];
    // 基礎集合編碼 ABCDE = 11111，1 向左移後累加
    for (int i = 0; i < charSet; ++i) basic += (1LL << i);
    print(basic);
    while (i<amount) {
        code[i] = encode(s[i]);
        print(code[i]);
        // 目前處理的輸入，跟之前的比對互補
        answer+= match(code, code[i], i, basic);
        i++;
    }
    return answer;
}
```

# 範例一 互補字串III

## □ 互補字串C

```
int main() {  
    char s1[][10] = {"ABB", "AB", "C", "CC"};  
    printf("%d\n", solve(3, 4, s1));  
    char s2[][10] = {"AABAAB", "ABCABCDE", "CDECD", "GFFGF"};  
    printf("%d\n", solve(7, 4, s2));  
    return 0;  
}
```

# Homework I—互補字word

## □ 互補字串

- 互補字S1, S2的定義是字串S1, S2沒有重複出現的字。字串是英文字的一個集合，亦即，元素有重複只算一個，也不管排列情況。例如"Happy Happy Day"與"Day Happy Day"是相同的字串。輸入N個字串，輸出這N個字串互補的個數。
- 輸入 (m, n) n 個字串
- 輸出

# 指向指標的指標

## □ 處理字串

- 可用來指向任意個任意長度的字串
- compiler 編譯函數的指向指標的指標時，會給足夠空間

```
#include <stdio.h>
void f() {
    char **p;
    char *pa[4]={"YOUUD", "ME", "HI", "STAR"};
    p = &pa[0];          //同 p = pa
    printf("%c\n", **p);  //Y
    printf("%s\n", *p);   //YOUUD
    printf("%s\n", *(p+1)); //ME
    printf("%c\n", *(*p+1)); //O
    printf("%s\n", *p+2);  //UD
    printf("%c\n", *(*p+2)+1)); //I
}
```

			+0	+1	+2	+3	
*(p+0)	*(pa+0)	pa[0]	Y	O	U	D	\0
*(p+1)	*(pa+1)	pa[1]	M	E	\0		
*(p+2)	*(pa+2)	pa[2]	H	I	\0		
*(p+3)	*(pa+3)	pa[3]	S	T	A	R	\0

$*(*(p+0)+1)=*(*p+1)$

# 指向指標的指標

## □ 一般作為函數的引數

- 長度及個數不固定，例如 \* argv[ ]
- 宣告：char \*\*p; 或 char \*p[ ];

```
#include <stdio.h>
#include <string.h>
void f(char **p) {           // 同 char *p[]
    printf("%c\n", **p);    // 印出 Y
    printf("%s\n", *p);     // 印出 YOUD
}
void test() {
    char *pa[4]={"YOUD", "ME", "HI", "STAR"};
    f(pa);
}
```

# 指向指標的指標

## □ 要修改內容

- 宣告成字串陣列 d[][5]，而非字串常數

```
#include <stdio.h>
#include <string.h>
void f(char *p[]) {
    printf("%c\n", **p);    //Y
    strcpy(*(p+1), "PK"); //ME -> PK
    *(* (p+2)+1) = 'Y';    //HI -> HY
}
void test() {
    char d[][5]={"YOUR", "ME", "HI", "STAR"};
    char *pa[4]={d[0], d[1], d[2], d[3]};
    f(pa);
    printf("%s\n", pa[1]);
    printf("%s\n", pa[2]);
}
```

f			+0	+1	+2	+3	
*(p+0)	*(pa[0])	d[0]	Y	O	U	D	\0
*(p+1)	*(pa[1])	d[1]	M	E	\0		
*(p+2)	*(pa[2])	d[2]	H	I	\0		
*(p+3)	*(pa[3])	d[3]	S	T	A	R	\0

# 指向指標的指標

- ❑ 字串存在字元陣列，要交換字串須要另外配置空間。

```
#include<string.h>
/* Swaps strings by swapping data*/
void swap(char *str1, char *str2) {
    int strLen = strlen(str1)>strlen(str2)?strlen(str1):strlen(str2);
    char *temp = (char *)malloc((strLen + 1) * sizeof(char));
    strcpy(temp, str1);
    strcpy(str1, str2);
    strcpy(str2, temp);
    free(temp);
}
int main() {
    char str1[] = "geeks";
    char str2[] = "forgeeks";
    swap2(str1, str2);
    printf("str1 is %s, str2 is %s", str1, str2);
    return 0;
}
```



# Exercise

- ❑ 字串存在字元陣列，要交換字串，以下Code問題？

```
#include<string.h>
void swap2(char *x, char *y) {
    char buf[80];
    int i, len;
    len = strlen(x);
    for (i=0; i<len; i++)    buf[i] = x[i];
    for (i=0; i<len; i++)    x[i] = y[i];
    for (i=0; i<len; i++)    y[i] = buf[i];
}
int main() {
    char str1[] = "geeks";
    char str2[] = "forgeeks";
    swap2(str1, str2);
    printf("str1 is %s, str2 is %s", str1, str2);
    return 0;
}
```

# Exercise

- ❑ 字串存在字元陣列，要交換字串，以下Code問題？

```
#include<string.h>
void swap(char *x, char *y) {
    char tmp;
    tmp = *x;
    *x = *y;
    *y = tmp;
}
int main() {
    char name1[] = "hello";
    char name2[] = "world";
    int i;
    printf("name1=%s\nname2=%s\n", name1, name2);
    for (i=0; i<strlen(name1); i++)
        swap(&name1[i], &name2[i]);
    printf("name1=%s\nname2=%s\n", name1, name2);
    return 0;
}
```

# Homework II

## 問題描述

一個字串如果全由大寫英文字母組成，我們稱為大寫字串；如果全由小寫字母組成則稱為小寫字串。字串的長度是它所包含字母的個數，在本題中，字串均由大小寫英文字母組成。假設  $k$  是一個自然數，一個字串被稱為「 $k$ -交錯字串」，如果它是由長度為  $k$  的大寫字串與長度為  $k$  的小寫字串交錯串接組成。

舉例來說，「StRiNg」是一個 1-交錯字串，因為它是一個大寫一個小寫交替出現；而「heLLow」是一個 2-交錯字串，因為它是兩個小寫接兩個大寫再接兩個小寫。但不管  $k$  是多少，「aBBaaa」、「BaBaBB」、「aaaAAbbCCCC」都不是  $k$ -交錯字串。

本題的目標是對於給定  $k$  值，在一個輸入字串找出最長一段連續子字串滿足  $k$ -交錯字串的要求。例如  $k=2$  且輸入「aBBaaa」，最長的  $k$ -交錯字串是「BBaa」，長度為 4。又如  $k=1$  且輸入「BaBaBB」，最長的  $k$ -交錯字串是「BaBaB」，長度為 5。

請注意，滿足條件的子字串可能只包含一段小寫或大寫字母而無交替，如範例二。此外，也可能不存在滿足條件的子字串，如範例四。

例一： 1 aBBdaaa 例二： 3 DDaasAAbbCC 例三： 2 aafAXbbCDCCC 例四： 3 DDaaAAbbCC

範例一： 2 範例二： 3 範例三： 8 範例四： 0

# Homework

```
#include <stdio.h>
#include <string.h>
void print(int b, int length){
    for (int i=0; i<length; i++) {        printf("%d ", b&1);        b = b>>1;    }
    printf("\n");
}
int match(int data, int bit, int mask) {
    if ((bit ==0) &&                                ;
    else if ((bit==1)                                return 1;
    else return 0;
}
void f(int data, int k, int length) {
    int mask= (1<<k)-1;
    int currentCount=0, maxCount=0, bit =0;
    print(mask,length);
    print(data,length);
    while (length>0) {
        if ((currentCount ==0) && (match(data, 0, mask)==1)) {    bit = 1;        currentCount++;    }
        else if ((currentCount ==0) && (match(data, 1, mask)==1)) {    bit = 0;        currentCount++;    }
        else if ((currentCount>0) && (match(data, bit, mask)==1)) {    currentCount++;    bit = !bit;    }
        else currentCount=0;
    }
}
```

# Homework

```
if (currentCount==0) {      data = data>>1;      length=length-1;      }
else {      data = data>>k;      length=length-k;      }
if (currentCount>maxCount) maxCount=currentCount;
}
printf("%d %d %d %d\n", data, mask, currentCount, maxCount);

}
int input(char s[]) {

}

int main() {
    //int k=2, length = 6, data=51;
    int k=3, length = 6, data=7;
    //int k=2, length = 6, data=50;
    f(data, k, length);
    //printf("%d\n", match(51, 1, 3));
    print(input("AAaaaBBB"), 8);
    return 0;
}
```