# 第四章

# 数组

# 重难点

重点

1. 理解数组类型的概念
2. 熟练掌握一维数组的使用
3. 熟练掌握二维数组的概念的使用
4. 熟练掌握查找、排序、求极值等算法
5. 熟练掌握字符数组及字符串处理的使用

难点

1. 理解数组类型和数组名
2. 设计常见成批数据处理算法
3. 编程实现字符串处理算法

# 4.1 顺序数据处理

```
/* Program: input a,b,c, and sort them.      */
#include <stdio.h>
void main( void )
{
   float a, b, c, t;

   printf("Input a,b,c:");
   scanf("%f,%f,%f", &a, &b, &c);
   if( a>b )
     t=a, a=b, b=t;
   if( a>c )
     t=a, a=c, c=t;
   if( b>c )
     t=b, b=c, c=t;

   printf("%5.2f,%5.2f,%5.2f\n", a, b, c);
}
```

如何对输入的10个整数排序并求平均呢?

## ★初识数组：

在 C 语言中，除了整型、实型和字符型等基本数据类型之外，还可以将基本数据类型按照一定的规则组合起来构成较为复杂的数据类型，称为构造数据类型，又称导出数据类型，主要包括数组、结构体、共用体等。

数组中顺序存放了一批相同数据类型的数据，这些数据不仅数据类型相同，而且在计算机内存里连续存放，地址编号最低的存储单元存放数组的起始元素，地址编号最高的存储单元存放数组的最后一个元素。通过数组名标识和序号（C 语言称为下标）可以引用这些数组元素。

**数组是一组有序的数据的集合——数组中的元素类型相同，并由数组名和下标唯一地确定。**

```
short num[10]={ 12 , -23 , 5,0,45 , -81 , 72,56,90,28 };
```

# 4.2 一维数组

## 4.2.1、一维数组的定义

语法： type ArrayName[size]；

**说明**：1. 定义一个含size个type类型元素的数组。

int a[10]; /*定义的整型数组a含10个元素*/

2. size必须是大于0的整型常量表达式。
3. ArrayName必须是合法
4. 编译时分配连续内存:

字节数=size* sizeof(type)

~~char name[0];~~
~~float weight[10.3];~~
~~float array[-100];~~

~~int n=5;~~
~~int a[n]~~

## 4.2.2、一维数组的初始化

定义时赋初值

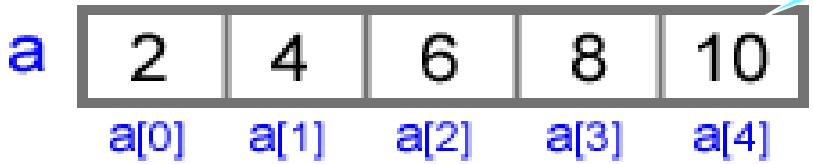
语法： type ArrayName[size]={value-list};

int a[5]={ 2 , 4 , 6 , 8 , 10 };

int a[3]={6,2,3,5,1}; (×)

存储单元

存储形式：




❖说明：

- 数组如果不初始化，其元素值为随机数
- 可以只初始化部分数组元素,余者自动赋0值。

int a[5]={ 2 , 4 };⇔ int a[5]={ 2 , 4, 0, 0, 0 };

❖说明：

- 当全部数组元素赋初值时，可不指定数组长度

```
int a[]={1,2,3,4,5,6};
```

编译系统根据初值个数确定数组维数

- 全局数组或static数组元素自动初始化成0值

```
static int a[5];⇔static  int a[5]={ 0 , 0, 0, 0, 0 };
```

## 例：数组元素初始化赋值测试

```c
#include <stdio.h>
void main(void)
{
    int i,a[5];
    for(i=0; i<5; i++)

        printf("%d\n", a[i]);
}
```

```
-858993460
-858993460
-858993460
-858993460
-858993460
请按任意键继续. . .
```

```c
#include <stdio.h>
void main(void)
{
    int i,a[5]={1,2};
    for(i=0; i<5; i++)

        printf("%d\n", a[i]);
}
```

```
1
2
0
0
0
请按任意键继续. . .
```

## 使用其它方法赋初值：

(1)利用赋值语句初始化

```
char str[26],ch;
str[0]= 'A';
str[1]= 'B';
…
str[25]= 'Z';
```

(2)利用赋值语句初始化

```
char str[26],ch;
    for(ch='A';ch<='Z';ch++)
        str[ch-‘A’]=ch;    //将26个大写字母依次存放到str[0]
                                、 str[1]、 ……str[25]中
```

(3)利用输入语句初始化

```
char  str[26];    int i ;
   for(i=0; i<26; i++) scanf("%c",&str[i]);
   ……
```

## 4.2.3. 一维数组的引用：

a. 数组必须先定义，后使用

b. 只能逐个引用数组元素，不能一次引用整个数组

例 `int a[10];`
`printf("%d",a);` (×)

必须
```
for(j=0;j<10;j++)
    printf("%d\t",a[j]);
```
(✓)

c. 数组元素表示形式： 数组名[下标]

数组元素的下标：0～size-1。

d. C语言对数组不作越界检查，使用时要注意

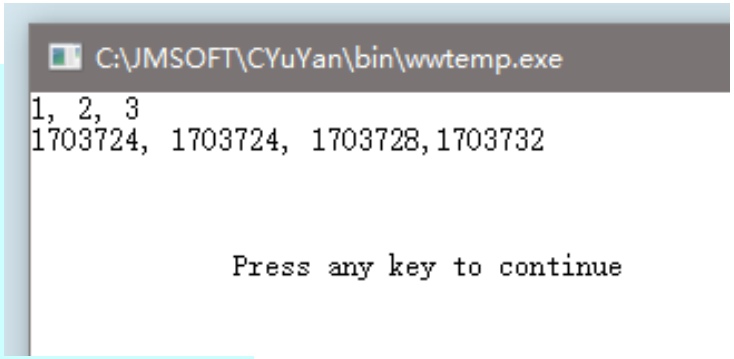
例 int data[5];
data[5]=10;

e.数组名a代表的是数组a在内存中的首地址，也即数组元素a[0]的地址。

scanf(“%d”,&a[0]); ⟹ scanf(“%d”,a);

等价于

```
#include<stdio.h>
void main(void)
{
  int a[5]={1,2,3,4,5};
printf("%d, %d, %d\n",a[0],a[1],a[2]);
printf("%d, %d, %d,%d\n",a,&a[0],&a[1],&a[2]);
}
```



# 习题讲解

1、若有以下数组说明，且 i=10；则 a[a[i]]元素数值是（ ）。

int a[12]={1,4,7,6,2,5,8,11,3,10,9,12};
A）10 B）9 C）6 D）5

答案：A。

2．若有 short Num[3]={24};则 Num[1]*10 的值是( )。
A）24 B）0 C）240 D）10

答案： B

3．数组中的元素类型必须_____，并由数组名和_____唯一确定。

答案：相同 下标

# 4.2.4、一维数组应用举例

1. 顺序或逆序访问数组元素

【例4-1】 用C语言编程输入10个整数，要求逆序打印其中的自然数，例如，输入12，-23，5，0，45，-81，72，56，90，28，程序输出的结果为28，90，56，72，45，5，12。

**分析：根据题意，需要定义一个能存放 10 个整数的数组 num 和用于控制下标变化的计数器 i，利用循环，让计数器 i递增，依次输入各个初值，然后再利用计数器 i 递减逆序访问每个数组元素，最后打印其中的非零正数。**

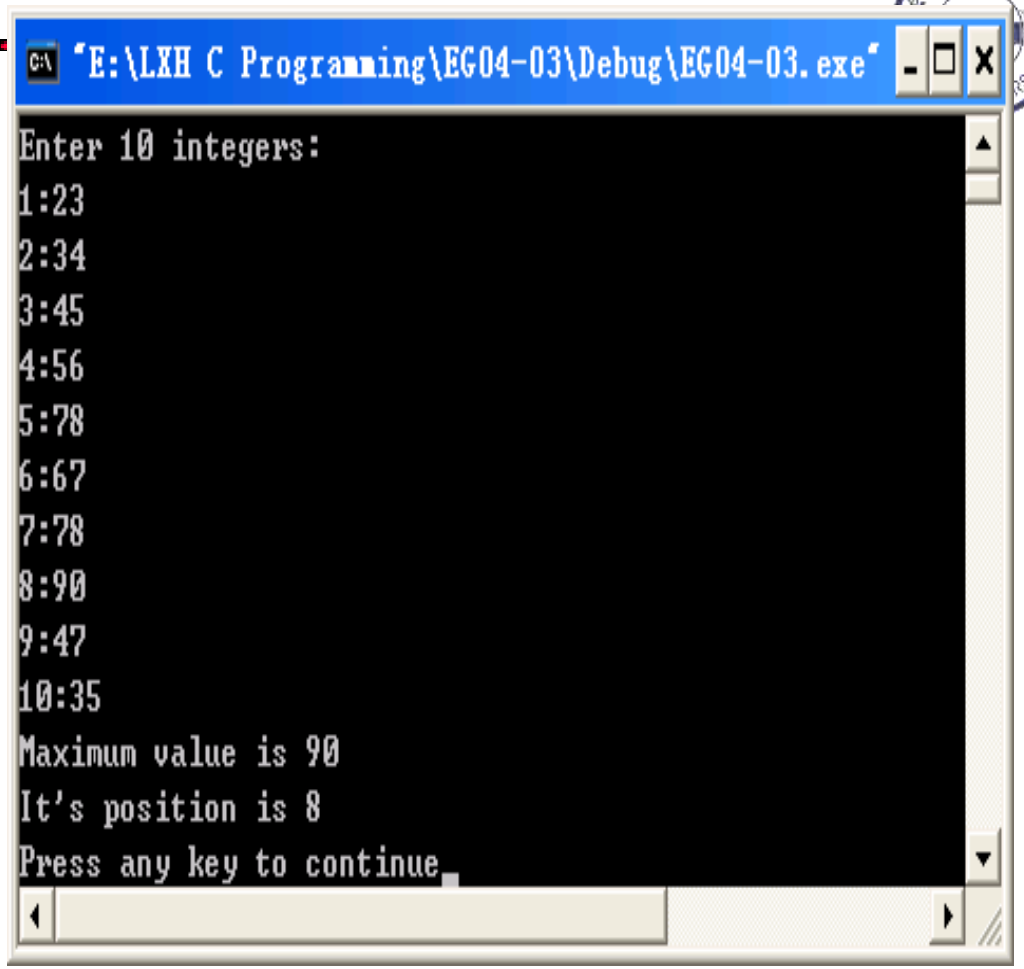
```
// Program: EG04-01.C
// Description: 输入10 个整数，要求逆序打印其中的自然数。
#include <stdio.h>
void main( void )
{
    short num[10], i ;

    printf("请输入10个整数:");
    for(i=0; i<10; i++)                    // 顺序输入10个数据
        scanf("%d,",&num[i]);

    printf("逆序打印其中的自然数:");
    for(i=9; i>=0; i--)                    //逆序访问各数组元素
        if(num[i]>0)
            printf("%d,", num[i]);         //打印正数
}
```

## 2. 寻找最大值、最小值和它们的位置

【例4-3】 用C语言编程输入10个整数存入数组num，找出其中的最大数和它所在的位置。

**分析：**根据题意，必须先定义一个能存放输入的 10 个整数的数组 num、计数器 i，利用循环，让计数器 i从 0 递增到 9，依次输入 10 个整数存到 num[i]中。当然，还要定义一个存放最大数的变量 Max、一个存放最大数所在位置的变量 MaxPos。对于查找极值的问题，初值的选取是非常重要的，不能随便定，如果选 0 作为最小值，就无法发现输入数据中比 0 更小的负数。查找极值的算法通常选取数组的起始元素和起始位置作为初值。然后利用计数器 i 从 1到 9，依次用 num[i]和 Max 比较，若 Max<num[i],令 Max=num[i]。最后将 Max、MaxPos 打印出来。

```
#include <stdio.h>
#define SIZE 10
void main( void )
{
   int num[SIZE];
   int i, Max, MaxPos;
   printf("Enter 10 integers:\n");
   for(i=0; i<SIZE; i++)
   {
      printf("%d:",i+1);   //方便阅读
      scanf("%d",&num[i]);
   }
   Max=num[0];   // Max要设置初值
   MaxPos= 0;
   for(i=1; i<SIZE; i++)
      if( Max < num[i] )
      {
         Max=num[i];
         MaxPos=i;
      }
   printf("Maximum value is %d\n", Max);
   printf("It's position is %d\n", MaxPos+1); //方便阅读
}
```



# 3. 排序

用选择法对10个整数排序

排序过程：

（1）首先通过n-1次比较，从n个数中找出最小的， 将它与第一个数交换—第一趟选择排序，结果最小的数被安置在第一个元素位置上

（2）再通过n-2次比较，从剩余的n-1个数中找出关键字次小的记录，将它与第二个数交换—第二趟选择排序

（3）重复上述过程，共经过n-1趟排序后，排序结束

selectionSort.gif (811×248) (runoob.com)

【例4-4】 用C语言编程实现对10个整数按从小到大顺序进行选择法排序，最后输出排序结果。

分析：根据题意，必须先初始化一个存放了 10 个整数的数组 Num，定义两个计数器 i、j，利用两重循环，让计数器 i 负责扫描 n−1 趟，让计数器 j 在内循环中进行 n−i 次比较。在 i 次循环时， 把第 i 个元素的位置 i 赋予变量 pos。 内循环从 Num[i+1]起一直比到最后一个元素，逐个与位置 pos 对应元素 Num[pos]作比较，将更小元素的位置记录到 pos 中。内循环结束后， pos 对应了最小元素， 如果 i=pos 表示 Num[i]就是最小的， 否则交换 Num[i]和 Num[pos]之值。排好 Num[i]后转入下一轮外循环。对 i+1 以后各个元素排序。最后将排序结果打印出来。

```
#include <stdio.h>
#define SIZE 10
void main(void)
{
   int Num[SIZE]={12,32,28,45,67,48,18,72,90,68};
   int Pos, tmp, i, j;
   printf("Normal numbers:");
   for(i=0; i<SIZE; i++)        //该for循环功能是?
    printf("%d ",Num[i]);
   printf("\n");
   for(i=0; i<SIZE-1; i++)
   {
       Pos=i;
       for( j=i+1; j<SIZE; j++)
         if(Num[j]<Num[Pos])
              Pos=j;
       if(i!=Pos)            //if语句的作用是?
       tmp=Num[i], Num[i]=Num[Pos], Num[Pos]=tmp;
   }
   printf("Sorted numbers:");
   for(i=0; i<SIZE; i++)
       printf("%d ",Num[i]);
   printf("\n");
}
```

# 冒泡法（bubble sort） - - 相邻元素比较交换法：

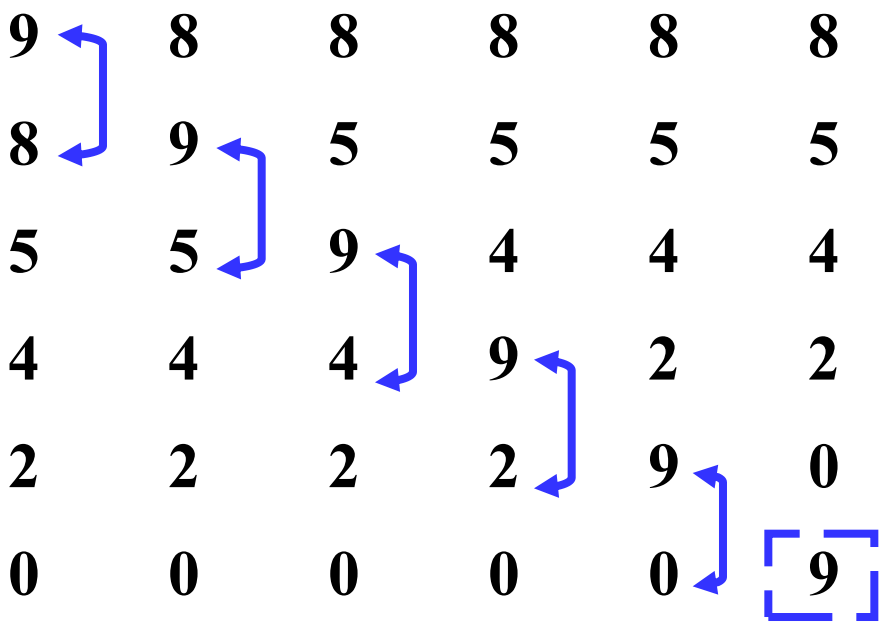
排序过程：

（1）第一趟扫描：从i=1开始，依次比较A[i]与A[i+1]，若为逆序(A[i]>A[i+1])，则交换其值；依次类推，直至第n-1个数和第n个数比较为止——第一趟扫描使最大的数被安置在第n个元素位置上。

（2）对前n-1个数进行第二趟扫描，结果使次最大的数被安置在第n-1个元素位置。

（3）重复上述过程，最多经过n-1趟扫描后，排序结束

bubbleSort.gif (826×257) (runoob.com)




第一趟扫描，比较了n-1次，最大的数➔第n个元素

第二趟扫描，比较前n-1个数，n-2次，最大的数➔第n-1个元素

……

【例4-5】 用C语言编程实现对10个浮点数按从小到大顺序进行冒泡法排序，最后输出排序结果。

```
// Program: EG0405.C
// Description: 对10个浮点数按从小到大顺序进行冒泡法排序，将SIZE改成10，便于理解
#include <stdio.h>
void main(void)
{
   float Num[10]={11.2,32.1,25.8,4.5,6.7,4.8,18,27.2,19,6.8}, tmp;
   int i, j;
   printf("Normal numbers:");
   for(i=0; i<10; i++)
       printf("%5.1f",Num[i]);
   printf("\n");

   for(i=0; i<9; i++)
   {
       for( j=0; j<9-i; j++)
           if(Num[j]>Num[j+1])
              tmp=Num[j], Num[j]=Num[j+1], Num[j+1]=tmp;
   }

   printf("Sorted numbers:");
   for(i=0; i<10; i++)
       printf("%5.1f",Num[i]);
   printf("\n");
}
```

```
for(i=0; i<SIZE-1; i++)
{
    for( j=0; j<SIZE-i-1; j++)
        if(Num[j]>Num[j+1])
           tmp=Num[j], Num[j]=Num[j+1], Num[j+1]=tmp;
}
```

```
Normal numbers: 11.2 32.1 25.8  4.5  6.7  4.8 18.0 27.2 19.0  6.8
Sorted numbers:  4.5  4.8  6.7  6.8 11.2 18.0 19.0 25.8 27.2 32.1
```

# 4. 二分查找(Binary Search，又称折半查找)

【例 4-6】 用 C 语言编程实现在按从大到小顺序排好序的 15 个整数中查找输入的数 x，如果找到 x，打印其位置，否则输出“未找到 x！ ”的提示。

分析：设数组Num存放按从大到小顺序排好序的SIZE个数，Begin代表查找区间的起点（初值为0）、End代表终点（初值为SIZE-1）

1）Mid代表当前查找区间的中间点位置：
Mid=( Begin + End )/2。
2）若x >Num[Mid]，则x必定在区间[0，Mid-1]中，
需要将End修正为Mid-1。
3）若x <Num[Mid]，则x必定在区间[Mid+1…n-1]中
需要将Begin修正为Mid+1。
4）重复这一过程直至找到x为止。
如果当前查找区间为空（End<Begin），查找失败。

注意：二分查找的查找对象是已经排好序的数据。

```
#include <stdio.h>
#define SIZE 15
void main( void )
{
   int Num[SIZE]={81,72,68,66,56,48,36,33,22,12,10,9,6,3,1};
   int Begin=0,End=SIZE-1, Mid, x;
    printf("Input a int:");
   scanf("%d",&x);
   while( Begin<=End )
   {
      Mid=(Begin+End)/2;
      if(x==Num[Mid])
      {
         printf("It's position is %d \n", Mid+1);     break;
      }
      if(x<Num[Mid])   Begin=Mid+1;
      else             End=Mid-1;
   }
   if(Begin>End)
      printf("Cann't find it\n");
}
```

```
Input a int:72
It's position is 2
```

```
Input a int:67
Cann't find it
```

# 5. 插入、删除、逆序、合并等。

我们经常要插入一个新数到排好序的数据序列中，要求新的序列仍然保持原有顺序。常见的操作还有删除、逆序、合并等。

【例 4-7】 用 C 语言编程实现在按从小到大顺序排好序的 9 个浮点数中插入一个输入的数，要求新的序列仍然保持原有顺序。

**分析：**由于要插入一个新数到原数据序列中，我们在初始化一个存放按从小到大顺序排好序的浮点数数组 Num时，**数组 SIZE 长度至少要设置成 9+1 = 10个，这是插入问题的关键所在。**最后一个数据元素在初始化时不要赋值。可以将要插入的数存到这个位置上。将它和前面的SIZE−1 个数据依次比较，如果发现Num[i]>Num[SIZE-1]，违背了从小到大排列的规律，互换其值。如果比较到最后都没有发现比 Num[SIZE-1]大的数，那就将插入的新数放在最后。

```
// Program: EG04-07.C
#include <stdio.h>
#define SIZE 10
void main( void )
{
   float Num[SIZE]={12,22,33,36,48,56,68,72,81};
   int i; float tmp;
   printf("Normal numbers:");
              for(i=0; i<SIZE-1; i++)
    printf("%3.0f",Num[i]);
   printf("\n");
   printf("Input new number:");
   scanf("%f",&Num[SIZE-1]);
   for( i=0; i<SIZE-1; i++ )
      if( Num[i]>Num[SIZE-1] )
        tmp=Num[i],Num[i]=Num[SIZE-1],Num[SIZE-1]=tmp;
    printf("New numbers:");
   for(i=0; i<SIZE; i++)
      printf("%3.0f",Num[i]);
   printf("\n");
}
```

```
Normal numbers: 12 22 33 36 48 56 68 72 81
Input new number:75
New numbers: 12 22 33 36 48 56 68 72 75 81
```

# 4.3 多维数组

## 4.3.1、二维数组的定义

**类型标识符 数组名[常量表达式SIZE 1][常量表达式 SIZE2];**

例 int a[3][4];

int score[3,4]; (×)

- 数组在内存中按行列顺序存放，最右下标变化最快。
- 字节数=sizeof(type) * size1* size2

二维数组a是由3个元素组成

int a[3][4];

| | | | | |
|---|---|---|---|---|
| a[0] | a[0][0] | a[0][1] | a[0][2] | a[0][3] |
| a[1] | a[1][0] | a[1][1] | a[1][2] | a[1][3] |
| a[2] | a[2][0] | a[2][1] | a[2][2] | a[2][3] |

行名

每个元素a[i]由包含4个元素的一维数组组成

| | |
|---|---|
| a[0][0] | a[0] |
| a[0][1] | |
| a[0][2] | |
| a[0][3] | |
| a[1][0] | a[1] |
| a[1][1] | |
| a[1][2] | |
| a[1][3] | |
| a[2][0] | a[2] |
| a[2][1] | |
| a[2][2] | |
| a[2][3] | |

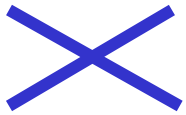
若二维数组 Num共有 R行 C列;则 Num[i][j]之前有____个元素。

i*C+ j

二维数组的许多性质与一维数组是类似的。

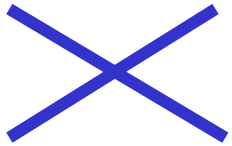
**size必须是大于0的整型常量表达式。**

```
int a[0][3];
```



不允许对数组的大小作动态定义。

```
int  i=3 , j=4 ;
int  a[i][j] ;
```



## 4.3.2、二维数组的初始化

全部初始化

- 分行初始化：

例 int a[2][3]={{1,2,3},{4,5,6}};

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| a[0][0] | a[0][1] | a[0][2] | a[1][0] | a[1][1] | a[1][2] |

建议格式：

```
int a[2][3]={ {1,2,3},
              {4,5,6} };
```

- 分行初始化：

部分初始化

例 int a[2][3]={{1,2},{4}};

| 1 | 2 | 0 | 4 | 0 | 0 |
|---|---|---|---|---|---|
| a[0][0] | a[0][1] | a[0][2] | a[1][0] | a[1][1] | a[1][2] |

- 分行初始化：

第一维长度省略初始化

例 int a[][3]={{1},{4,5}};

| 1 | 0 | 0 | 4 | 5 | 0 |
|---|---|---|---|---|---|
| a[0][0] | a[0][1] | a[0][2] | a[1][0] | a[1][1] | a[1][2] |

- 分行初始化：
- 按元素排列顺序初始化

全部初始化

例 int a[2][3]={1,2,3,4,5,6};

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| a[0][0] | a[0][1] | a[0][2] | a[1][0] | a[1][1] | a[1][2] |

- 分行初始化：
- 按元素排列顺序初始化

部分初始化

例 int a[2][3]={1,2,4};

| 1 | 2 | 4 | 0 | 0 | 0 |
|---|---|---|---|---|---|
| a[0][0] | a[0][1] | a[0][2] | a[1][0] | a[1][1] | a[1][2] |

- 分行初始化：
- 按元素排列顺序初始化

第一维长度省略初始化

例 int a[][3]={1,2,3,4,5};

| 1 | 2 | 3 | 4 | 5 | 0 |
|---|---|---|---|---|---|
| a[0][0] | a[0][1] | a[0][2] | a[1][0] | a[1][1] | a[1][2] |

例. 下列数组定义错误的是 D (第二维不能省略)

A) int Score[3][3];　　B) int Score[][3]={{1},{2},{3}};

C) int Score[3][3]={1};　D) int Score[3][]= {{1},{2},{3}};

# 4.3.3、二维数组的引用

表示形式： **数组名[行下标][列下标]**

**行下标、列下标可以是常量、变量、函数或表达式。**
**如果下标带有小数，C 编译系统将自动对其取整。**
**例如，若有定义：float Num[3][10];**
**则Num[2][1]、Num[i][j]、Num[j][i++]都是对数组 Num中数组元素的合法引用。**

行下标和列下标均从0开始的至size-1。

例如：若有 int a[2*5][3*4], i=15;

则使用 a[3*3][0], a[1][i-5]都是合法的。

C 语言对数组元素的引用有以下几种规定：

1） 行下标、 列下标均从 0 开始计算。 数组下标表示了数组元素在数组中的顺序号。 因此，如果定义：

float Num[2][4]={ 0,1,2,3,4,5,6,7 };

则可以访问的行下标为 0、1，可以访问的列下标为 0、1、2、3。不能越界引用数组元素Num[1][8]、Num[5][3]，因为 C 语言从不检查数组下标是否越界，程序员要自己确认数组元素的正确引用，不能越界访问。

2）在 C 语言中只能逐个引用数组元素，而不能一次引用整个数组。

例如，不能用一个语句输入整个数组： scanf("%f",Num);

必须使用循环语句逐个输入数据：

```
for(i=0; i<2; i++)
    for(j=0; j<4; j++)
        scanf("%f", &Num[i][j]);
```

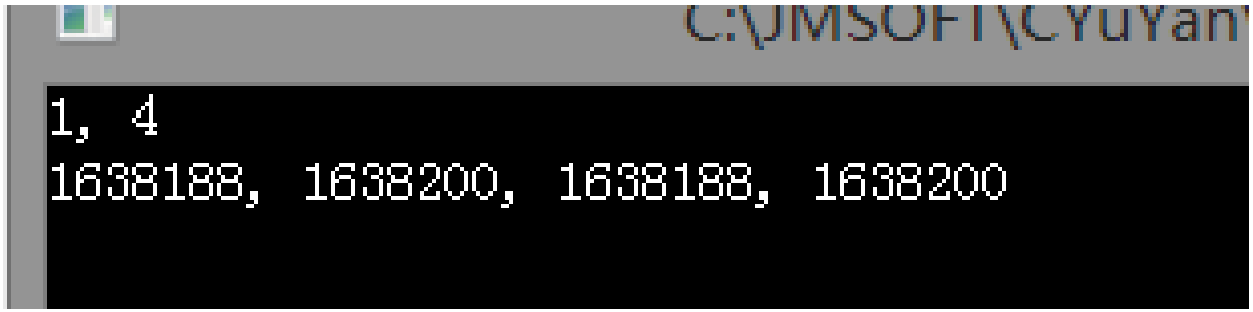
3）数组元素和普通变量一样，可以出现在任何合法的 C 语言表达式中。例如，数组元素可以出现在函数调用：printf(“%d”, Num[i][j]);中。

4）定义数组时，方括号中出现的是某一维的长度，只能是常量或常量表达式；

引用数组元素时，方括号中出现的是该元素在数组中的位置标识，可以是常量，变量或任何合法的C语言表达式。

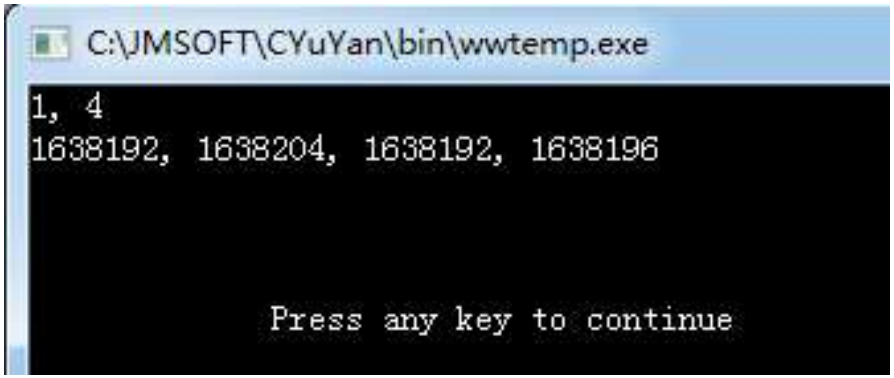
# 分析程序结果：

```
#include<stdio.h>
void main(void)
{
int a[2][3]={{1,2,3},{4,5,6}};

printf("%d, %d\n",a[0][0],a[1][0]);

printf("%d, %d, %d, %d\n",a,a+1,&a[0][0],&a[1][0]);
}
```



# 分析程序结果：

```
#include<stdio.h>
void main(void)
{

int a[2][3]={{1,2,3},{4,5,6}};

printf("%d, %d\n",a[0][0],a[1][0]);

printf("%d, %d, %d, %d\n",a,a+1,&a[0][0],&a[0][0]+1);

}
```



## 4.3.4、二维数组程序举例

**【例4-9】 编程实现指定二维数组的转置。**

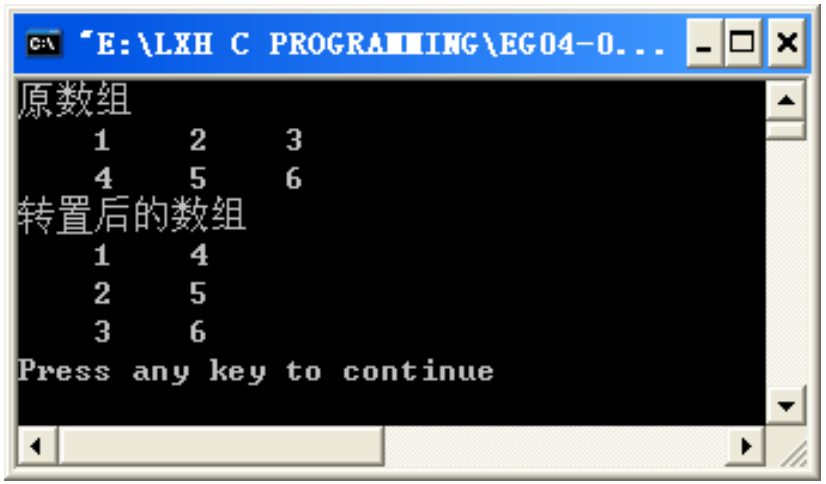
**解题思路：二维数组的转置是指将一个二维数组行和列的元素互换，存到另一个二维数组中去。**

$$a=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \qquad b=\begin{bmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{bmatrix}$$

将上述思路转换成C语言源程序：

```
#include <stdio.h>
void main( void )
{
    int a[2][3]={{1,2,3},
                 {4,5,6}};
    int b[3][2], i, j;
    printf("原数组\n");
    for(i=0;i<=1;i++)
    {
        for(j=0;j<=2;j++)
            printf("%5d",a[i][j]);
        printf("\n");
    }
    for(i=0;i<=1;i++)
        for(j=0;j<=2;j++)
            b[j][i]=a[i][j];
}
```

```
printf("转置后的数组\n");
    for(i=0;i<=2;i++)
    {
        for(j=0;j<=1;j++)
        printf("%5d",b[i][j]);
      printf("\n");
    }
```



## 【想一想】

以下编程将一个3×3 的数组转置：

```
for(i=0; i<3; i++)
for(j=0; j<3; j++)
tmp = a[j][i], a[j][i] = a[i][j] , a[i][j] = tmp ;
```

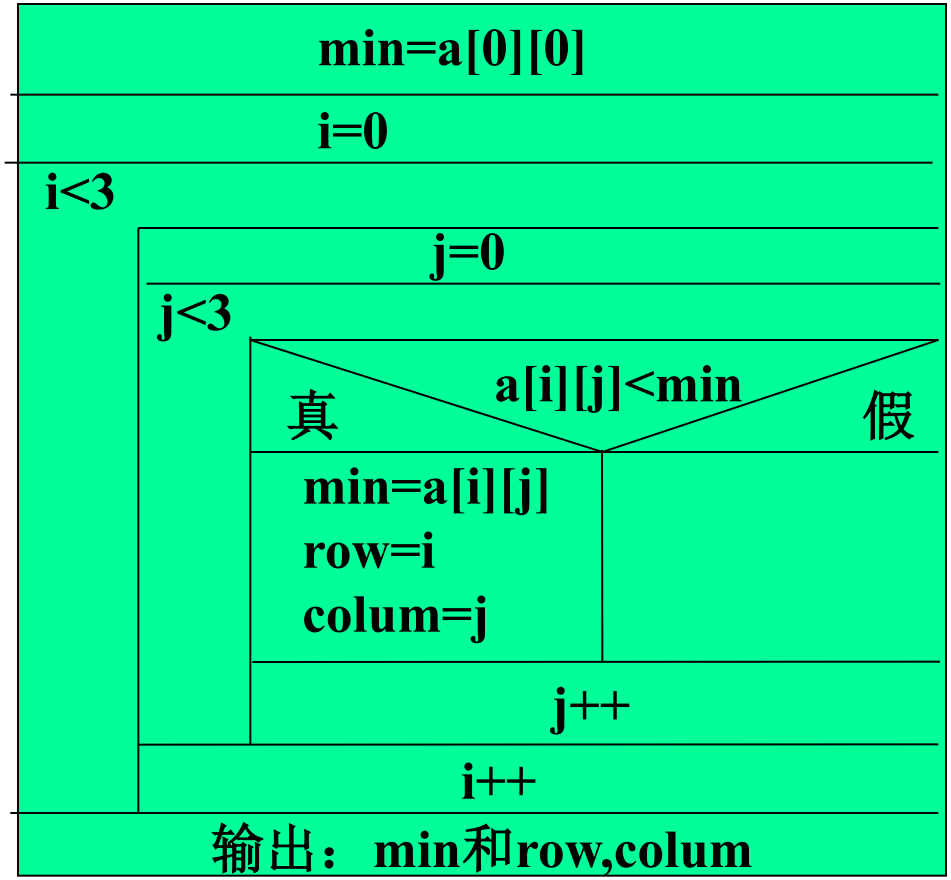
但打印出来的转置数组根本没有变化，这是为什么呢？

**分析：** **[例4-9] 是将转置结果保存到另一个二维数组中，而这里是将转置结果保存在原数组中，这就不能简单地模仿 [例4-9] 了。因为当i=0、j=1 时已经将a[1][0]、a[0][1]互换了，如果继续循环到i=1、j=0时，就不能再次将a[0][1]、a[1][0]互换了，否则就又换回去了。正确的程序应该是**

```
for(i=0; i<3; i++)
for(j=0; j<i; j++)
tmp = a[j][i], a[j][i] = a[i][j] , a[i][j] = tmp ;
```

## 【例4-12】 编程求指定二维数组中最小元素的值及其位置（行列号）。

**解题思路：** 这是二维数组的求极值问题。先初始化一个二维数组Matrix、计数器i、j，还要定义一个存放最小数的变量Min、一个存放最小数所在位置的变量MaxPosi、MaxPosj。选取数组的起始元素和起始位置作为上述变量的初值。然后利用循环依次用Matrix[i][j]和Min比较，若Min>Matrix[i][j]，令Min= Matrix[i][j]。最后打印最小元素的值及其位置。



```
// Program: EG04-10.C
// Description: 求指定二维数组中最小元素的值及其位置。
#include <stdio.h>
#define SIZE 10
void main( void )
{
   int Matrix[3][4]={{1, 2, 3,4},
                     {9, 8, 7,6},
                     {-10,10,-5,2}};
    int i, j, MinPosi, MinPosj, Min;
   Min=Matrix[0][0]; MinPosi=0; MinPosj=0;          //一定要赋初值
   for(i=0; i<3; i++)
       for(j=0; j<4; j++)
           if( Min>Matrix[i][j] )
           {
               Min=Matrix[i][j];
               MinPosi=i;
               MinPosj=j;
           }
   printf("Min=%d, row=%d, colum=%d\n", Min,MinPosi, MinPosj );
}
```

**说明：**

**数组名 M代表的是数组在内存中的首地址，也即数组元素 M[0][0]的地址。二维数组可以看作是一维数组的嵌套:一维数组的每个数组元素都又是一个一维数组。**

**由此，可以将一个二维数组分解成多个一维数组。**

**例如，int M[3][4];可以分解成三个一维数组：M[0]、M[1]、M[2]，这三个一维数组都有 4 个元素，例如，一维数组 M[0]的元素为 M[0][0], M[0][1], M[0][2], M[0][3]。**

# 4.4 字符数组和字符串处理

## 4.4.1、字符型数据的概念与存储

•字符串常量：

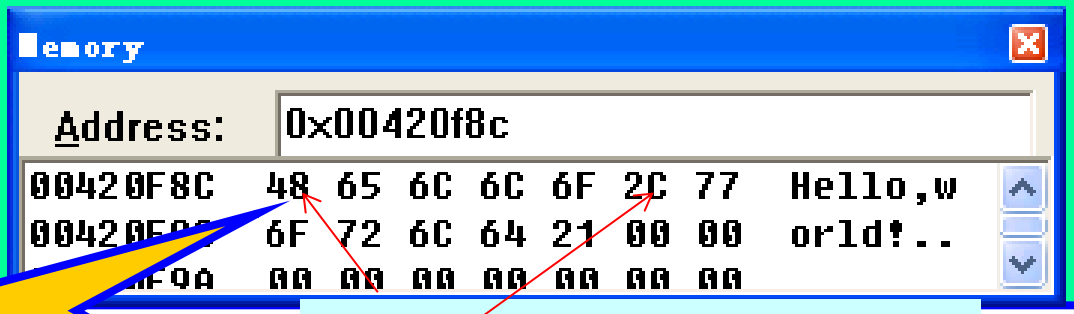
a. 用一对双引号括起来的若干字符序列。

b. 字符串长度：字符串中字符的个数。

c. 字符串的结束标志： NULL ⇔'\0',ASCII值为0。

d. “ ”表示空字符串，长度为0，占一个字节，即NULL。

例 “Hello”共5个字符，在内存占6个字节 字符串长度5




内存存放字符ASCII码

$48_{16}$----$72_{10}$----- H

$2C_{16}$---$44_{10}$ ---- ,

## 4.4.2. 字符数组的定义与初始化：

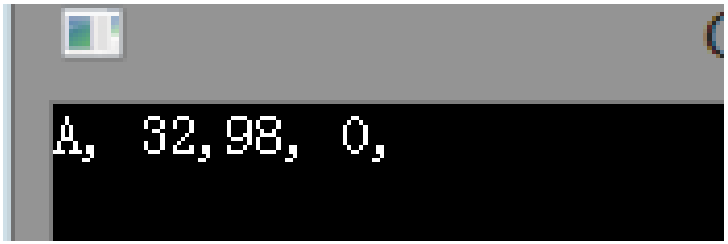
字符数组的定义与一维数组、二维数组基本相同，类型标识符为 char。

说明：1. 字符数组的每一个元素都是字符变量

2. 处理n个字符，size必须≥n+1

char c[10]; ⇔int c[10]; （×）

```
#include "stdio.h"
void main()
{
  char a[5]={65, ' ',98};
 printf("%c, %d,%d, %d,",
          a[0],a[1],a[2],a[3]);
}
```



## 4.4.2. 字符数组的定义与初始化：

### 1. 逐个数值赋给字符数组的元素

```
char str[10]={112,114,111,103,114,97,109,0};
```

### 2. 逐个字符赋给字符数组的元素

```
char str[10]={'p','r','o','g','r','a','m','\0'};
```

### 3. 用字符串常量直接初始化数组

```
char str[10]={"program"};
char str[10]="program";
```

## 4.4.2. 字符数组的定义与初始化：

4.字符数组定义初始化时可省略第一维长度，由系统编译时根据初值个数自动确定数组长度：

```
char Str[ ]="program";  // Str数组长度自动确定为 8
```

5. 通常一维字符数组用于存放一个字符串，二维字符数组用于存放多个字符串，而且至少要按最长的字符串长度加1设定第二维长度。二维字符数组定义初始化时同样只能省略第一维长度。

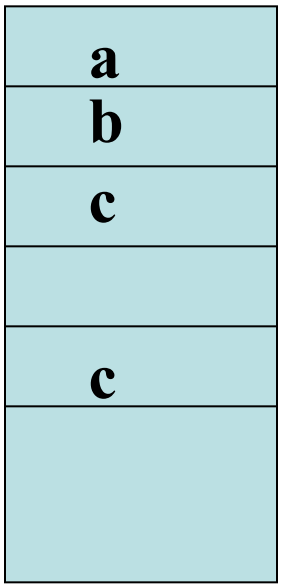
```
char Subject[3][15]={"C programming","Java","Authorware"};
char Subject[][15] ={"C programming","Java","Authorware"};
```

## 4.4.3、字符数组的输入输出

1. 逐个字符输入输出

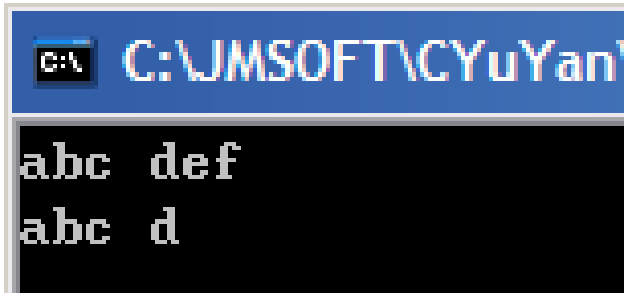
用格式符“%c”输入或输出一个字符。

```
#include <stdio.h>
void main(void )
{
   char a[5]; int i=0;
   while(i<=4)
     scanf("%c",&a[i++]);
   for(i=0;i<5;i++)
         printf("%c", a[i]);
}
```




a[3]
a[4]

注意：空格的ASCII码值为32!



2. 将整个字符串一次输入或输出。

用格式符“%s”输入输出字符串。

```
#include <stdio.h>
void main(void )
{
    char a[5];
    int i=0;
    scanf("%s",a);
    printf("%s",a);
}
```

| |
|---|
| a |
| b |
| c |
| \0 |
| |
| |

a[0]
a[1]
a[2]
a[3]
a[4]

scanf中%s输入时，遇空格、TAB或回车结束

```
abcde
abcde
```

```
ab cd
ab
```

3. 用“%s”格式符输入字符串时，字符数组名前不要再加地址符&，因为数组名代表该数组的起始地址。

如： char str[10]; ~~scanf(“%s”,&str);~~

4. 用“%s”格式符输入字符串时，输入的字符串应≤字符数组的长度－1。系统自动在后面加 个‘\0’结束符。

```
void main( void )
{
    char a[]={'h','e','l','\0','l','o','\0'};
    printf("%s",a);
}
```

输出：

```
hel
```

数组中有多个‘\0’时，
遇第一个结束

## 4.4.4、字符串处理函数

（1）字符串输入输出函数

语法：char *gets(char *string)
功能：从stdin输入一个字符串(直到回车)到string中
说明：输入串可包含空格，长度应小于字符数组维数

语法：int puts(char *string)
功能：输出string到stdout
说明：字符串结束标志将转换成‘\n’，即输出完字符串后换行。

注意：使用puts和gets函数前，要用#include “stdio.h”。
但其它字符串处理函数包含在头文件：string.h

scanf函数使用空格、TAB或回车作为输入数据的分隔符，无法输入带空格的字符串，此时需要用到gets函数！

例 若准备将字符串“This is a string.”记录下来，错误的输入语句为：

（A）scanf(“%20s”,s);

（B）for(k=0;k<17;k++)
　　s[k]=getchar();

（C）while((c=getchar())!='\n')
　　s[k++]=c;
　　s[k]='\0';

（D）gets(s);

scanf(“%s”,str1)读取字符串内容，遇空格结束；

puts(str)与 printf(“%s\n”, str)功能类似；

一般scanf()后面不使用gets()

```
#include <stdio.h>
void main( void  )
{
char str1[20];
printf("Input a sentence:");
scanf("%s",str1);
printf("%s\n",str1);
}
```

```
Input a sentence:Hello world!
Hello
请按任意键继续. . .
```

```
#include <stdio.h>
void main( void  )
{
char str1[20];
printf("Input a sentence:");
gets(str1);
puts(str1);
}
```

```
Input a sentence:Hello world!
Hello world!
请按任意键继续. . .
```

# 练习

1. 若定义charName[20]; 则（ B ） 可以输入带空格的字符串。

A）scanf("%s",&Name);

B）gets(Name);

C）scanf("%s", Name);

D）gets(Name[20]);

2. 对字符数组s 赋值，不合法的一个是（ C ） 。

A）char s[]="Beijing";

B）char s[20]={"beijing"};

C）char s[20]；s="Beijing";

D）char s[20]=
{'B','e','i','j','i','n','g'};

# 字符串处理函数

除了字符串的输入、输出以外，C 语言在标准库函数里提供了丰富的字符串处理函数：字符串的长度测试、复制、比较、连接等。

使用这些库函数可大大减轻编程的负担。在使用前必须包含头文件"string.h"。

# 转义字符回顾

**定义：由“反斜杠字符”开始，后跟单个字符或若干个字符组成，每个转义字符算一个字符！**

## 转义字符表

| 转义字符 | 对应字符 | 转义字符 | 对应字符 |
|---|---|---|---|
| \n | 回车换行符 | \a | 响铃符号 |
| \t | Tab符 | \' | 单引号 |
| \v | 垂直制表符 | \" | 双引号 |
| \b | 左退一格符 | \\ | 反斜杠 |
| \r | 回车符 | \ddd | 1～3位8进制数ddd对应的符号 |
| \f | 换页符 | \xhh | 1～2位16进制数hh对应的符号 |

## （2）求字符串长度strlen( )函数

语法： unsigned int strlen(char *str)

功能：计算字符串长度（不包括‘\0’字符）。

例如： char str[80]={"ab\n\\012/\\\""} ;

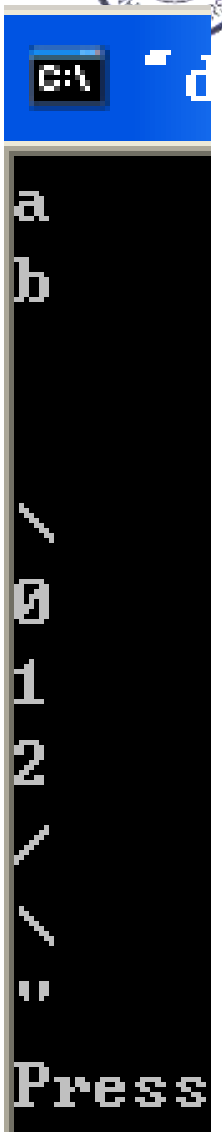
printf("%d",strlen(str)) ;

输出： 10

如果该字符串是“ab\n\012/\\\”” ， 结果呢? 7

```
char str[80]={"ab\n\0y\012/\\\""} ;
printf("%d",strlen(str)) ;
```

输出： 3



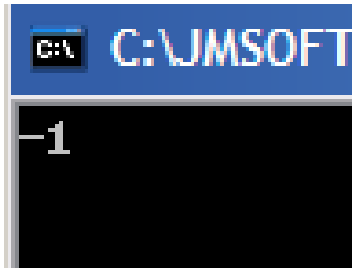
## （3）字符串比较strcmp( )函数

语法：int strcmp(char *str1, char *str2);

功能：对两串从左向右逐个字符比较（ASCII码），直到遇到不同字符或‘\0’为止

返值：
- >0; 串1>串2
- =0; 串1=串2
- <0; 串1<串2

```
#include <stdio.h>
#include <string.h>
void main( void  )
{
  printf("%d\n",strcmp("China","Chinese")) ;
}
```




！！字符串比较不能用“==”,必须用strcmp函数

# （4）字符串复制strcpy()函数

语法：char * strcpy(char *str1, char *str2);

功能：将字符串str2复制到str1中

返值：返回str1的首地址

说明：a. 定义str1时，size必须足够大

b. 拷贝时‘\0’一同拷贝

```
例如：char str1[20], str2[15] ;
      strcpy(str1,“hello world”);
      strcpy(str2,str1);
      puts(str1); puts(str2);
```

输出：hello world
hello world

c. 不能使用赋值语句为一个字符数组赋值

str1=“hello” ; (错误)

只能初始化时赋值或复制： char str1[ ]=“hello”; 或
strcpy(str1,“hello world”); （正确）

# （5）字符串连接strcat()函数

语法：char * strcat(char *str1, char *str2);

功能：将字符串str2连接到str1后

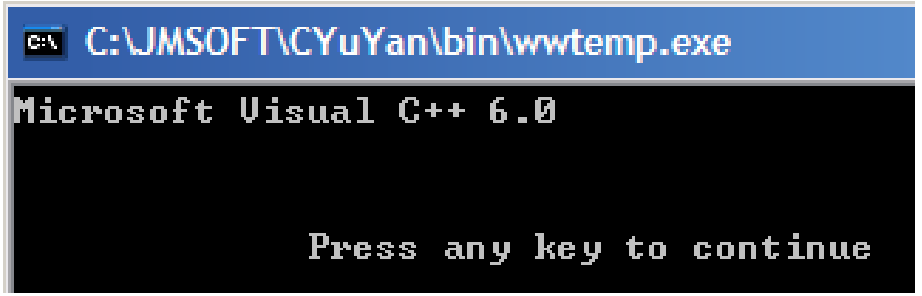
返值：返回str1的首地址

说明：a. 定义str1时，size≥strlen(str1)+strlen(str2)

b. 删除str1的字符串结束符NULL，然后将str2指向的字符串连同str2的结束符一起复制到str1的后面，构成新串，连接时只在新串最后保留一个‘\0’。

**例如：**

```
char str[25];
strcpy(str, "Microsoft ");
strcat(str, "Visual C++ 6.0");
printf(str);
```



（6）将字符串中大写字母换成小写字母strlwr()函数

语法：

**char * strlwr**(char *str )

（7）将字符串中小写字母换成大写字母strupr函数

语法：

**char *strupr(char *str)**

```
#include <string.h>
#include <stdio.h>
void main()
{
    char str[25];
    strcpy(str, "Turbo C");
    printf(strupr(str));
}
```

输出：
TURBO C

# 练 习

1.下列程序的输出结果是 （ C ）。

```
char str[80]= "this\0is\0abook";
printf("%d,%d", sizeof(str), strlen(str));
```

A）5,4　　B）4,4　　C）80,4　　D）14,4

2. 若定义: char Array[][8]={“China”, “USA”, “UK”}; 则数组 Array所占的内存为（ D ）字节。

A）15　　B）10　　C）18　　D）24

3. 若定义char str1[10], str2[10];… 则 （ C ）可以判断字符串str1是否大于字符串str2。

A）if(str1>str2)　　B）if(strcmp(str1,str2))

C）if(strcmp(str1,str2)>0)　　D）if(strcmp(str2,str1)>0)

4. 下面程序的输出结果是 。 D

```
#include <stdio.h>
#include <string.h>
void main(void)
{
char str[100] ="How do you do";
strcpy( str + strlen(str)/2, "es she");
printf("%s\n", str);
}
```

A）How do you do　　B）es she

C）How are you　　D）How does she

5、程序运行结果为。

```
#include <stdio.h>
#include <string.h>
void main( void )
{
char str[30];
strcpy(&str[0], "CHINA");
strcpy(&str[1], "DEFINE");
strcpy(&str[2], "ARM");
printf("%s\n", str);
}
```

结果：CDARM

为便于理解，将程序修改为：

```
#include <stdio.h>
#include <string.h>
void main( void )
{
char str[30];
strcpy(&str[0], "CHINA");
printf("%s\n", str);
strcpy(&str[1], "DEFINE"
printf("%s\n", str);
strcpy(&str[2], "ARM");
printf("%s\n", str);
}
```

## 4.4.5 字符串应用举例

【例4-11】 不用strcmp函数编程实现对输入两个字符串的比较。（字符串处理）

**分析：首先要根据题意定义两个一维字符数组，接受输入的两个字符串。然后借助计数器 i 从左向右对两个字符串第 i 个字符逐个比较其 ASCII 码值，直到存在差异或遇到字符串结束符NULL 为止，并由两个字符 ASCII码值差的三种情况显示相应比较结果。**

```
#include <stdio.h>
void main( void )
{
    char string1[20], string2[20];
    short i;
    printf("输入两个字符串:\n");
    gets(string1);
    gets(string2);
    for (i = 0; string1[i] == string2[i]; i++)
        if (string1[i] == '\0')
                break;       //表示string1到结尾了，此时跳出循环，直接判断
    if( string1[i] - string2[i] >0 )
        printf("%s>%s\n", string1, string2);
    else
       if( string1[i] - string2[i] ==0 )
            printf("%s==%s\n", string1, string2);
       else
            printf("%s<%s\n", string1, string2);
}
```

试着实现strcat（）函数

【例4-12】 不用itoa函数，编程实现将一个输入的整数转换为字符串。（字符转换）

分析：库函数 atoi可以把一个数字字符串转换成对应整数，而 itoa 可以将一个输入的整数转换为字符串。根据题意定义一个整型变量 Num存放输入的整数，一个一维字符数组 Str，借助计数器 i从右向左求 Num逐位数字， 并将其转换成 ASCII字符， 直到处理完符号位为止，将所得字符串逆序并显示即可。

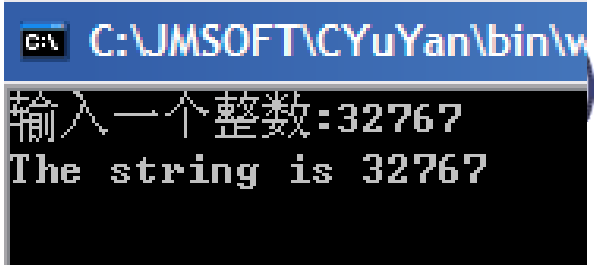
注意：字符 ‘0’ 与整数0的ASCII值关系：

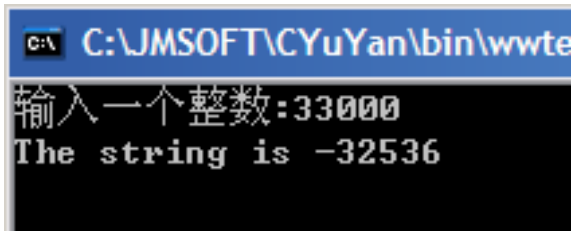
‘0’=48=0+48= 0+ ‘0’

‘1’=49=1+48= 1+ ‘0’

‘2’=50=2+48= 2+ ‘0’

……



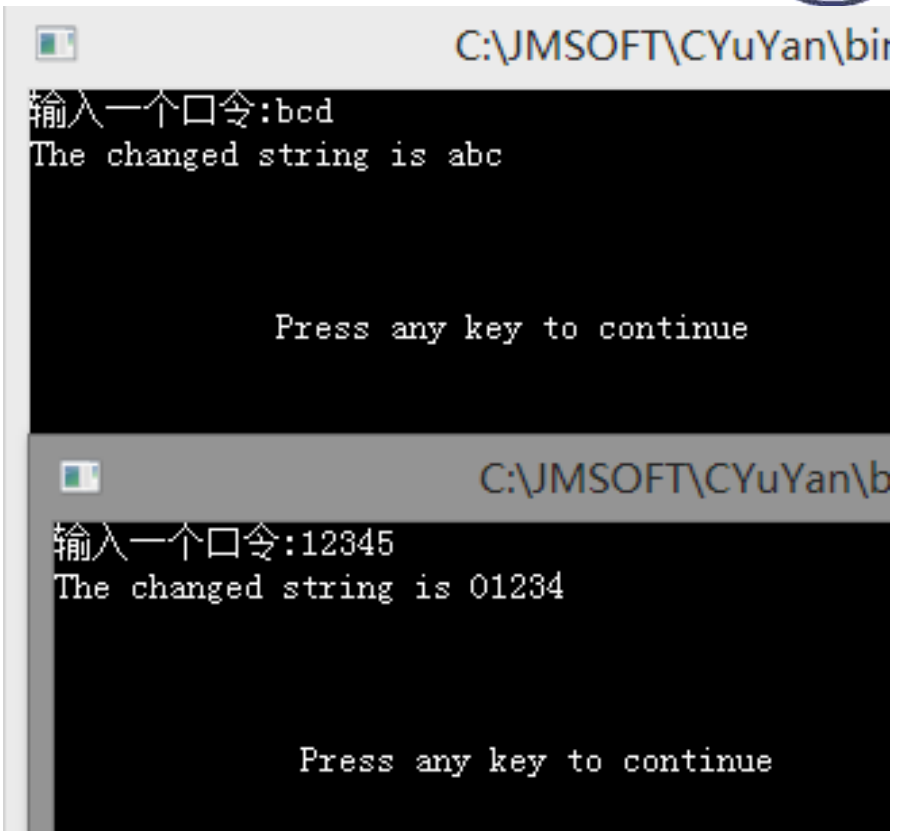

```
#include <stdio.h>
void main( void )
{
   char Str[20], tmp;
   short Num, i, j, Sign;
   printf("输入一个整数:");
   scanf("%d", &Num);
   if((Sign=Num)<0)          //记录负号
     Num=-Num;               //num取绝对值
   i=0;
   do
       Str[i++]=Num%10+'0';      //取下一位数字，+‘0’即为对应的字符
   while ((Num/=10)>0);          // 循环后获得的字符为逆序

 if(Sign<0)
     Str[i++]='-';
   Str[i] = '\0';
   for (j = i-1, i = 0; i < j; i++, j--)
   tmp = Str[i], Str[i] = Str[j], Str[j] = tmp;        //将字符前后调换，即逆序打出来即可
   printf("The string is %s\n", Str );
}
```

## 【例 4-13】 编程实现将一个输入的口令字符串加密。（基于字符的加密、解密）

字符串加密是指按照一定的规律将代表秘密的字符串，如口令，变换成没有意义的内容。加密过的文字，即使被别有用心的人得到，也无从得知里面的内容。合法用户则可以根据对应的规律将密文还原成原文使用。

字符串加密、解密的方法有很多，简单一点的是直接对字符加上或减去一个数，从而将字符变成另一个数。更进一步是将这个数换成一串复杂的密文（密钥），或干脆使用一张字符变换对照表……

```c
#include <stdio.h>
void main( void )
{
   char Str1[20], Str2[20];
   unsigned short i;

   printf("输入一个口令:");
     scanf("%s", Str1);      //gets(Str1);

   i=0;
   while (Str1[i])
   {
    Str2[i]=Str1[i]-1;
   i++;
   }
   Str2[i]='\0';

   printf("The changed string is %s\n", Str2 );
}
```



【例4-14】 已知10 名同学的姓名和考试成绩，编程实现按姓名的字典顺序将其递增排序。

本题首先要根据题意初始化一个二维字符数组Names[10][10]存放10 名同学的姓名、一个一维数组Scores[10]存放10名同学的考试成绩，然后借助冒泡法、选择法排序算法的思想对其排序，需要注意的是：

首先，字符串不能直接比较大小，需要调用strcmp 函数，交换顺序时要调用strcpy函数，为此需要嵌入string.h。

其次，在交换姓名的同时，必须将对应的成绩也要进行交换。

将上述思路转换成C源程序：

【例4-14】 已知10名同学的姓名和考试成绩，编程实现按姓名的字典顺序将其递增排序。

```
#include <stdio.h>
#include <string.h>
void main( void )
{   char Names[10][10]={"zhang","wang","li","zhao","qian","sun",
                        "wan", "zao","wu","zheng"}, NameTmp[10];
   short Scores[10]={95,74,83,90,66,89,70,92,73,86}, ScoreTmp;
   unsigned short i, j;
    for(i=0; i<10; i++)
     for( j=0; j<10-i-1; j++)
         if(strcmp(Names[j], Names[j+1])>0)
         {
               strcpy(NameTmp, Names[j]);
               strcpy(Names[j], Names[j+1]);
               strcpy(Names[j+1], NameTmp);
               ScoreTmp=Scores[j];
               Scores[j]=Scores[j+1];
               Scores[j+1]=ScoreTmp;
         }
   printf("The shorted data is :\n");
   for(i=0; i<10; i++)
       printf("%s\t%d\n", Names[i], Scores[i]);
}
```

```
The shorted data is :
li      83
qian    66
sun     89
wan     70
wang    74
wu      73
zao     92
zhang   95
zhao    90
zheng   86
```

# 小结

学习数组这一章要注意的几个问题：

1. 在C语言中数组的下标是从0开始；
2. C语言不进行下标的越界检查。
3. 数组名是地址。
4. 字符数组在定义时必须考虑到串结束符的位置，因为它要占一个字符的位置.
5. 要注意数组初始化的方法与简单变量的区别，特别是字符数组的初始化方法。
6. 注意掌握操作字符串的专用函数。