信息与电气工程学院

C程序设计案例教程

数组

主要内容

- 一维数组
  - 定义
  - 引用
  - 初始化
- 二维数组
  - 定义
  - 引用
  - 初始化
- 字符数组
  - 定义
  - 初始化
  - 字符串
  - 字符数组的输入输出
  - 字符串处理函数

用基本数据类型可以解决所有问题吗？

例如：对学生的成绩按由高到低的次序进行排序。

3 名？stud01,stud02,stud03 

300 名？stud001,stud02,……stud300 ？ 

- 数组属于构造类型。
- 数组:是具有一定顺序关系的若干相同类型变量的集合，用数组名标识。
- 元素:组成数组的变量，用数组名和下标确定

§7.1 一维数组的定义和引用

★一维数组的定义

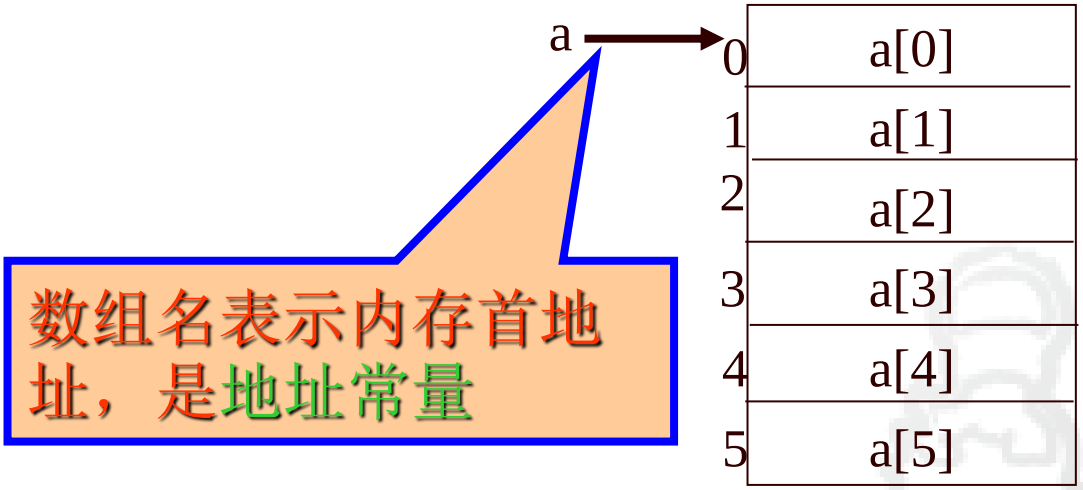
❖定义方式： 数据类型 数组名[常量表达式];

表示元素个数
下标从0开始

例 int a[6];

合法标识符

[ ]：数组运算符
单目运算符
优先级(1)，左结合，
不能用( )



数组名表示内存首地址，是地址常量

例 int data[5]; //C语言对数组不作越界检查，
data[5]=10; //错误写法，使用时要 注意

注意：

常量表达式中可以包括常量和符号常量，不能包含变量。也就是说，c不允许对数组的大小作动态定义，即数组的大小不依赖于程序运行过程中变量的值。

例如，下面这样定义数组是不行的：

```
int n;
scanf("%d", &n);
int a[n];
```

★一维数组元素的引用

- ❖数组必须先定义，后使用
- ❖只能逐个引用数组元素，不能一次引用整个数组。
- ❖数组元素表示形式： 数组名[下标]

  其中：下标可以是常量或整型表达式

例 int a[10];

printf("%d",a); (×)

必须 for(j=0;j<10;j++)

printf("%d\t",a[j]); (✓)

运行结果：

9 8 7 6 5 4 3 2 1 0

例7.1

```
#include <stdio.h>
int main()
{ int i,a[10];
  for(i=0;i<=9;i++)
     a[i]=i;
  for(i=9;i>=0;i--)
     printf("%d",a[i]);
}
```

在定义数组时，为数组元素赋初值
(在编译阶段使之得到初值）

★一维数组的初始化

实现的方法：

❖在定义数组时对数组元素赋初值。

```
int a[5]={1,2,3,4,5};
等价于：a[0]=1; a[1]=2; a[2]=3; a[3]=4; a[4]=5;
```

❖只给一部分元素赋值。

```
如   int a[5]={6,2,3};
等价于：  a[0]=6; a[1]=2;a[2]=3; a[3]=0; a[4]=0;
如   int a[3]={6,2,3,5,1};   (×)
```

❖数组元素值全部为0

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

❖对整个数组元素赋初值时，可以不指定长度。

```
int a[]={1,2,3,4,5,6};
编译系统根据初值个数确定数组大小
```

★一维数组程序举例

例1 读10个整数存入数组，找出其中最大值和最小值

步骤:
1. 输入:for循环输入10个整数
2. 处理:
(a) 先令max=min=x[0]
(b) 依次用x[i]和max,min比较(循环)
若max<x[i],令max=x[i]
若min>x[i],令min=x[i]
3. 输出:max和min

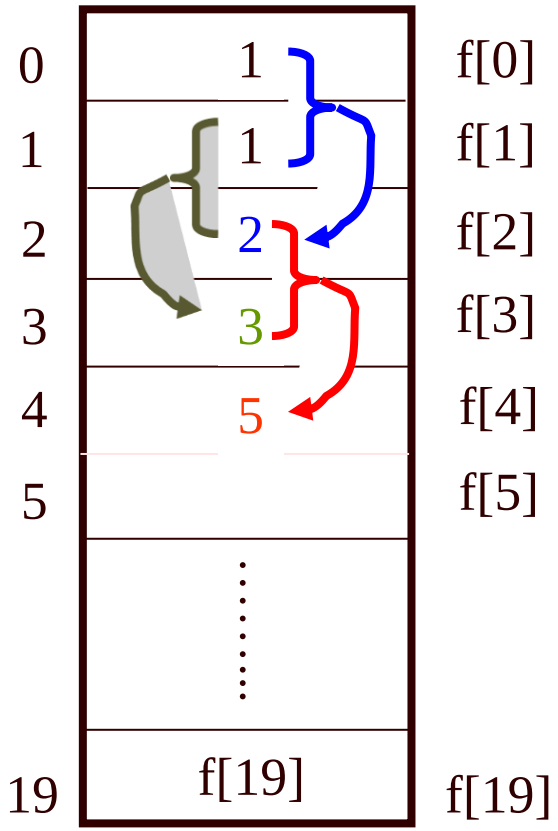
★一维数组程序举例

例1 读10个整数存入数组，找出其中最大值和最小值

```
#include <stdio.h>
#define SIZE 10
int main()
{ int x[SIZE],i,max,min;
  printf("Enter 10 integers:\n");
  for(i=0;i<SIZE;i++)
  {   printf("%d:",i+1);
        scanf("%d",&x[i]);
  }
  max=min=x[0];
  for(i=1;i<SIZE;i++)
  {  if(max<x[i])  max=x[i];
     if(min>x[i])  min=x[i];
  }
  printf("Maximum value is %d\n",max);
  printf("Minimum value is %d\n",min);
}
```

例2 用数组来处理求Fibonacci数列问题

$$F1 = 1 \quad (n = 1)$$
$$F2 = 1 \quad (n = 2)$$
$$F_n = F_{n-1} + F_{n-2} \quad (n \geq 3)$$




```
#include <stdio.h>
int main()
{   int i;
    int f[20]={1,1};
   for(i=2;i<20;i++)
     f[i]=f[i-2]+f[i-1];
   for(i=0;i<20;i++)
   {  if(i%5==0)  printf("\n");
      printf("%12d",f[i]);}
}
```

运行结果：

```
           1           1           2           3           5
           8          13          21          34          55
          89         144         233         377         610
         987        1597        2584        4181        6765
```

例3 用冒泡法对10个数排序（由小到大）

排序过程：

（1）比较第一个数与第二个数，若a[0]>a[1]，则交换；然后比较第二个数与第三个数；依次类推，直至第n-1个数和第n个数比较为止——第0趟冒泡排序，结果最大的数被安置在最后一个元素位置上

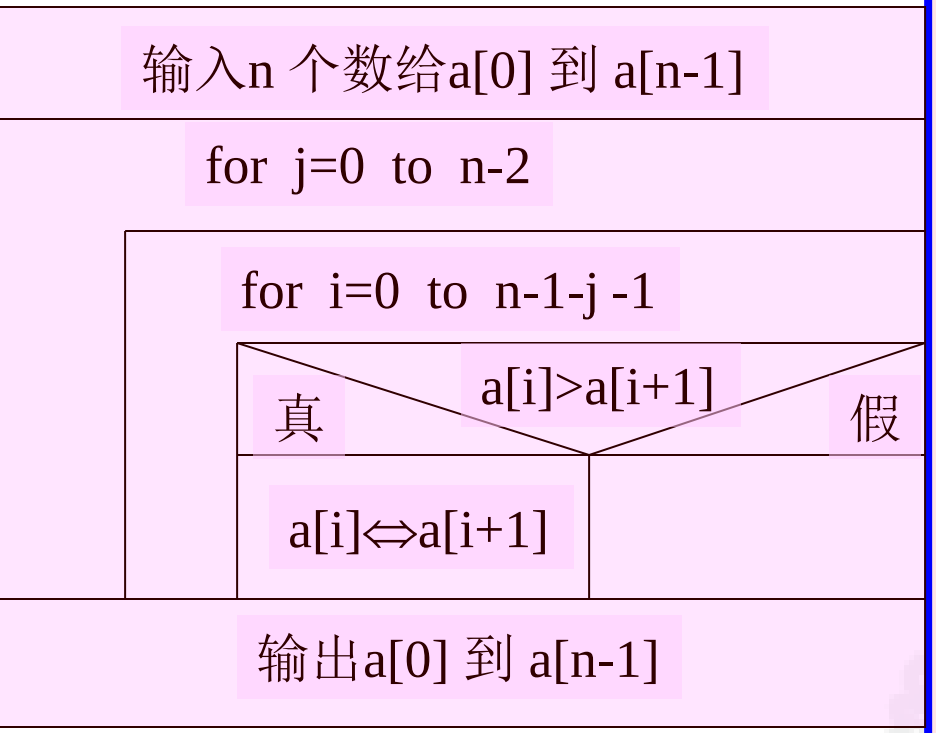
（2）对前n-1个数进行第二趟冒泡排序，结果使次大的数被安置在第n-1个元素位置

（3）重复上述过程，共经过n-2趟冒泡排序后，排序结束

此处：n=6

9 8 8 8 8 8	8 5 5 5 5	5 4 4 4	4 2 2	2 0
8 9 5 5 5 5	5 8 4 4 4	4 5 2 2	2 4 0	0 2
5 5 9 4 4 4	4 4 8 2 2	2 2 5 0	0 0 4	
4 4 4 9 2 2	2 2 2 8 0	0 0 0 5		
2 2 2 2 9 0	0 0 0 0 8			
0 0 0 0 0 9				
第0轮大数沉底	第1轮	第2轮	第3轮	第4轮

外层循环j(0~n-2)次　　内层循环i (0~n-j-1)次




```
#include <stdio.h>
int main()
{ int a[10],i,j,t;
   int n=10;
  for(i=0;i<n;i++)
     scanf("%d",&a[i]);
  for(j=0;j<(n-1);j++)
     for(i=0;i<(n-1-j);i++)
        if(a[i]>a[i+1])
         {t=a[i]; a[i]=a[i+1]; a[i+1]=t;}
  for(i=0;i<10;i++)
     printf("%d ",a[i]);
}
```

例4 用简单选择法对10个数排序

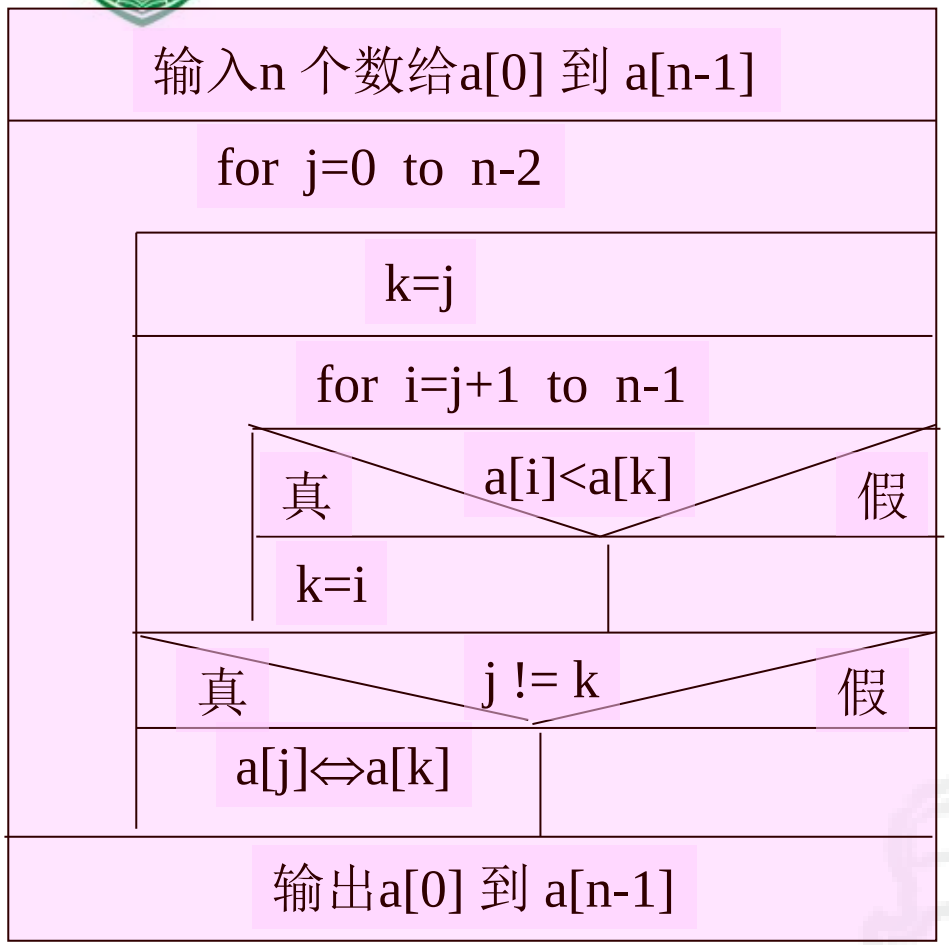
排序过程：

（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

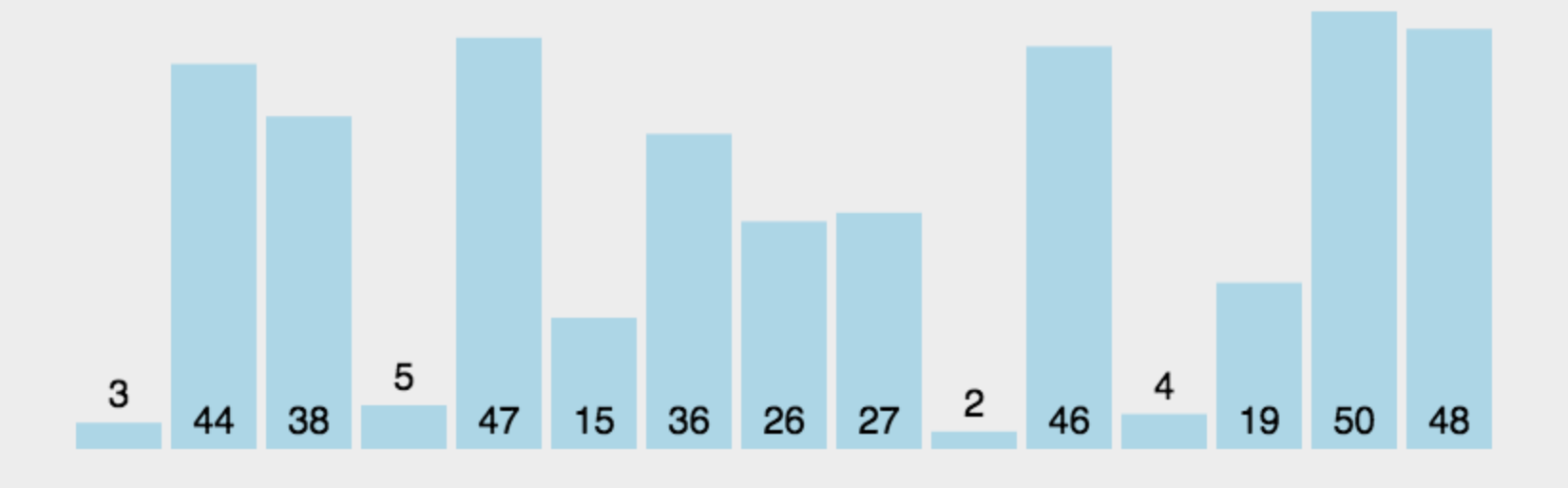
（3）重复上述过程，共经过n-1趟排序后，排序结束

j=0	0趟：	[49	38	65	97	76	13	27]
j=1	1趟：	13	[38	65	97	76	49	27]
	2趟：	13	27	[65	97	76	49	38]
	3趟：	13	27	38	[97	76	49	65]
	4趟：	13	27	38	49	[76	97	65]
	5趟：	13	27	38	49	65	[97	76]
	6趟：	13	27	38	49	65	76	[97]

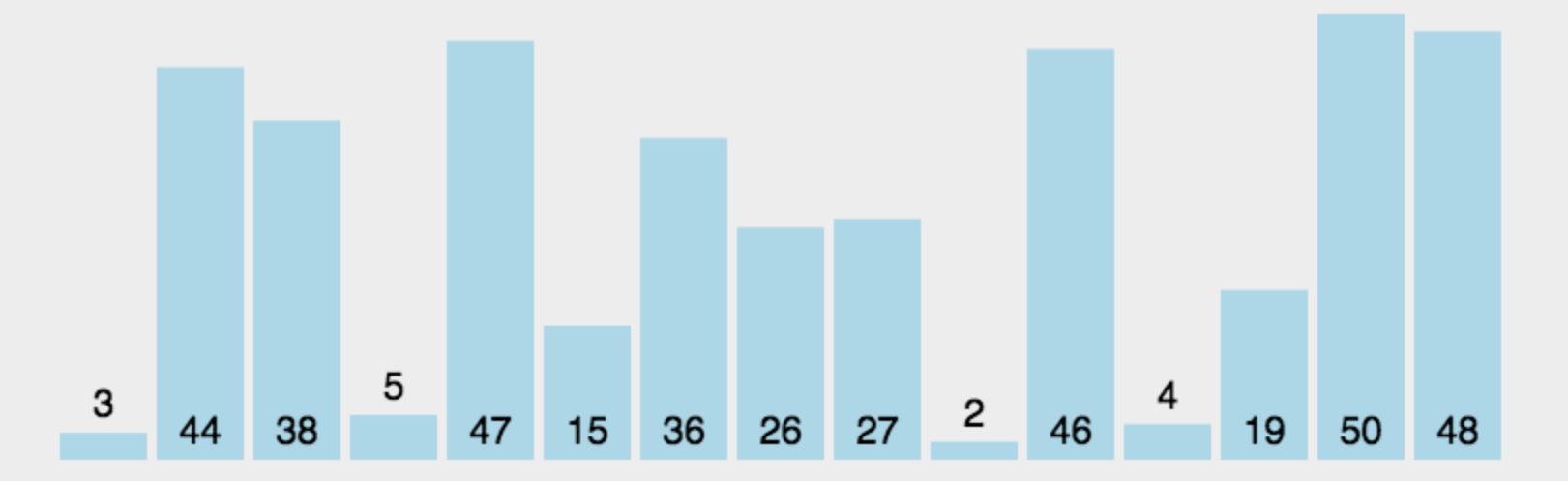



```
#include <stdio.h>
int main()
{ int a[10],i,j,k,x;
  int n=10;
  for(i=0;i<n;i++)
    scanf("%d",&a[i]);
  for(j=0;j<(n-1);j++)
  { k=j;
    for(i=j+1;i<n;i++)
          if(a[i]<a[k])  k=i;
    if(j!=k)
    { x=a[j]; a[j]=a[k]; a[k]=x;}
  }
  for(i=0;i<n;i++)
    printf("%d ",a[i]);
}
```

冒泡排序法示意动图



简单选择法示意动图




```
for(j=0;j<(n-1);j++)
  { k=j;
    for(i=j+1;i<n;i++)
            if(a[i]<a[k])  k=i;
    if(j!=k)
    { x=a[j]; a[j]=a[k]; a[k]=x;}
  }
```

j 橙色
k 红色
i 绿色

§7.2 二维数组的定义和引用（多维数组）

★二维数组的定义

❖定义的一般形式

行数

元素个数=行数*列数

类型说明符 数组名[常量表达式][常量表达式];

❖数组元素的存放顺序

- 原因:内存是一维的
- 二维数组：按行序优先
- 多维数组：最右下标变化最快

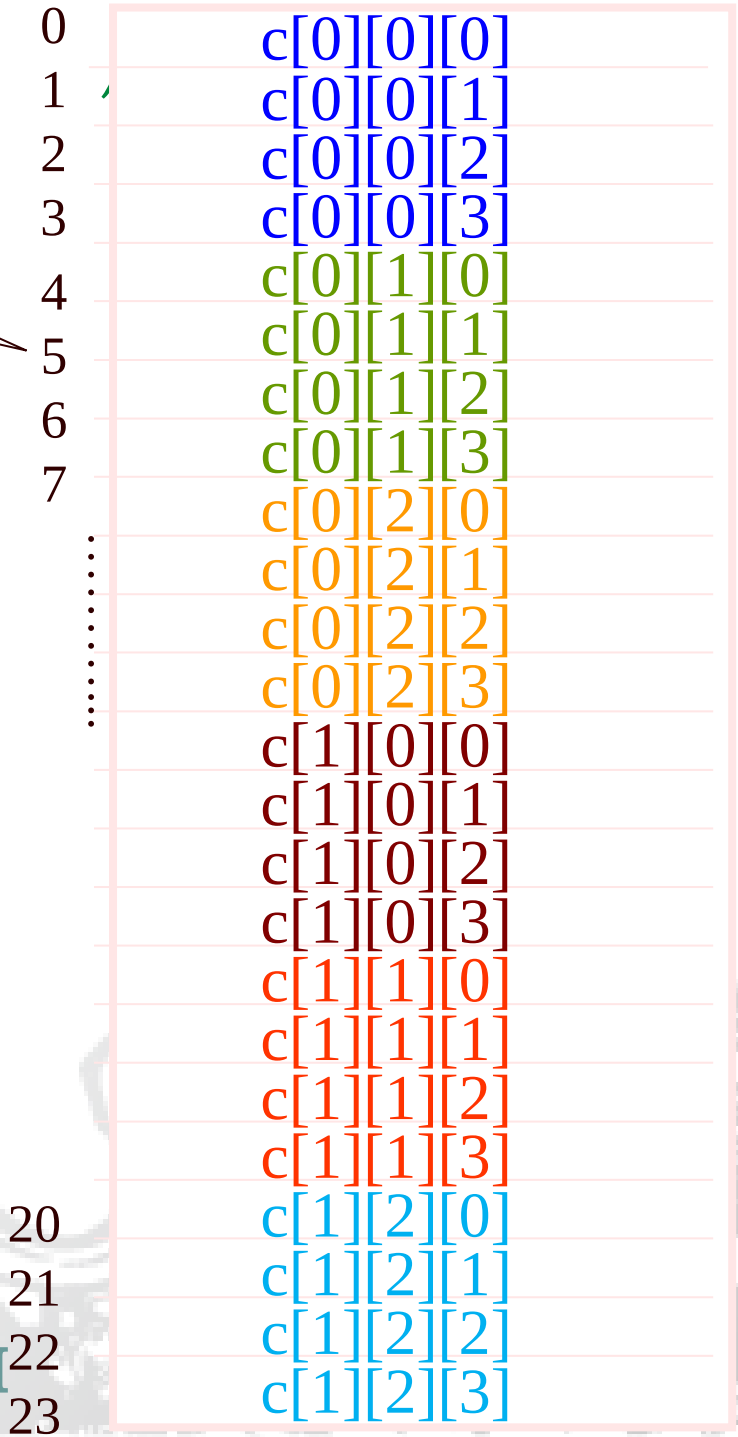
例 int a[3][4];
float b[2][5];
int c[2][3][4];
int a[3,4]; ×

int a[3][2]

a[0][0]	a[0][1]
a[1][0]	a[1][1]
a[2][0]	a[2][1]

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

多维数组元素在内存中的排列顺序：第一维的下标变化最慢，最右边的下标变化最快。



❖二维数组理解

二维数组a是由3个元素组成

例　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

0	a[0][0]	a[0]
1	a[0][1]	
2	a[0][2]	
3	a[0][3]	
4	a[1][0]	a[1]
5	a[1][1]	
6	a[1][2]	
7	a[1][3]	
8	a[2][0]	a[2]
9	a[2][1]	
10	a[2][2]	
11	a[2][3]	

★二维数组的引用

❖形式： 数组名[下标][下标]

- 下标是整型或字符型的常量，变量或表达式。（定义时不能使用变量）

  如： a[1][2]　　a[i][j]

- 数组元素可出现在表达式中，如：a[1][2]=a[2][2]/2
- 使用数组元素时，应注意不要超出其定义的范围；

  如： int a[2][3];　　a[2][3]=5;

★二维数组的初始化

❖分行初始化

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]

★二维数组的初始化

❖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]

★二维数组的初始化

❖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]

★二维数组的初始化

- ❖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]

★二维数组的初始化

- ❖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]

★二维数组的初始化

- ❖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]

★二维数组程序举例

例5 将二维数组行列元素互换，存到另一个数组中

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

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

例6 读入下表中值到数组，分别求各行、各列及表中所有数之和

12	4	6	22
8	23	3	34
15	7	9	31
2	5	17	24
37	39	35	111

```
#include <stdio.h>
int main()
{  int x[5][4],i,j;
   for(i=0;i<4;i++)
      for(j=0;j<3;j++)
         scanf("%d",&x[i][j]);
   for(j=0;j<3;j++)
      x[4][j]=0;
   for(i=0;i<5;i++)
      x[i][3]=0;
   for(i=0;i<4;i++)
      for(j=0;j<3;j++)
       { x[i][3]+=x[i][j];
         x[4][j]+=x[i][j];
         x[4][3]+=x[i][j];
       }
```

```
  for(i=0;i<5;i++)
   { for(j=0;j<4;j++)
        printf("%5d\t",x[i][j]);
     printf("\n");
   }
  return 0;
}
```

例7　从一个三行四列的整型二维数组中查找第一个出现的负数。

分析

算法要点：

用两层嵌套的 for 循环来遍历数组元素，判断是否为负数。当找到第一个负数时就应该退出循环，为此，应定义一个标记变量，用于标记找到与否的状态，并将此标记加入循环控制条件中，以控制循环在适当时候退出。

break语句只能跳出一重循环

```
#include <stdio.h>
int main()
{ int i,j,found,num[3][4];
  printf("Enter 12 integers:\n");
  for(i=0;i<3;i++)
       for(j=0;j<4;j++)
     scanf("%d",&num[i][j]);
  found=0;
  for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
            {  found=num[i][j]<0;
               if (found == 1)  break;
            }
        if (found == 1)  break;
     }
  if(!found)
     printf("not found\n");
  else
     printf("minus number num[%d][%d]:%d\n",
                        i,j,num[i][j]);
}
```

例8　请输出杨辉三角（前十列）。

分析

算法要点：

```
if(j==0||j==i)
{
    a[i][j]=1;
}
a[i][j]=a[i-1][j-1]+ a[i-1][j];
```

```
1
1    1
1    2    1
1    3    3    1
1    4    6    4    1
…
```

§ 7.3 字符数组

字符数组：存放字符数据的数组。

一维字符数组：存放一个字符串（每个数组元素存放一个字符）

二维字符数组：存放多个字符串（行数是字符串的个数）

★字符数组的定义

❖形式：

- char 数组名[常量表达式]
- char 数组名[常量表达式][常量表达式]
- 常量表达式：整数、字符、符号常量

例 char c[10], ch[3][4];

❖可以用整型数组存放字符型数据，但浪费存储空间。

★字符串和字符串结束标志

- ❖字符串：用双引号括起的若干字符，如： "china"

  可将其存放在一维或两维字符型数组中。

  无字符串变量，用字符数组处理字符串

- ❖字符串结束标志：'\0'，(既无动作,又不显示)
  - 字符串的长度：第一个'\0'以前字符的个数
  - 在字符型数组或字符串中遇'\0'，即认为该字符串结束。
  - 系统对字符串常量自动加一个'\0'作为结束符。

```
printf("china");
```

c	h	i	n	a	\0

★字符数组的初始化

- ❖逐个字符赋值
- ❖用字符串常量

逐个字符赋值

例 char ch[5]={'H', 'e', 'l', 'l', 'o'};

H	e	l	l	o
ch[0]	ch[1]	ch[2]	ch[3]	ch[4]

★字符数组的初始化

- ❖逐个字符赋值
- ❖用字符串常量

逐个字符赋值

例 char ch[4]={'H', 'e', 'l', 'l', 'o'};

H	e	l	l	o
ch[0]	ch[1]	ch[2]	ch[3]	ch[4]

有问题!

★字符数组的初始化

- ❖逐个字符赋值
- ❖用字符串常量

逐个字符赋值

例 char ch[5]={'B', 'o', 'y'};

B	o	y	\0	\0
ch[0]	ch[1]	ch[2]	ch[3]	ch[4]

★字符数组的初始化

- ❖逐个字符赋值
- ❖用字符串常量

逐个字符赋值

例 char ch[ ]={'H', 'e', 'l', 'l', 'o'};

H	e	l	l	o
ch[0]	ch[1]	ch[2]	ch[3]	ch[4]

★字符数组的初始化

- ❖逐个字符赋值
- ❖用字符串常量

二维字符数组初始化

例 char diamond[][5]={{' ',' ','*'},{' ','*',' ','*'},
{'*',' ',' ',' ','*'},{' ','*',' ','*'},{' ',' ','*'}};

diamond[0]			*	\0	\0
diamond[1]		*		*	\0
diamond[2]	*				*
diamond[3]		*		*	\0
diamond[4]			*	\0	\0

❖用字符串常量初始化字符数组

用字符串常量

例 char ch[6]="Hello";
char ch[]="Hello";

H	e	l	l	o	\0
ch[0]	ch[1]	ch[2]	ch[3]	ch[4]	ch[5]

❖用字符串常量初始化字符数组

二维字符数组初始化

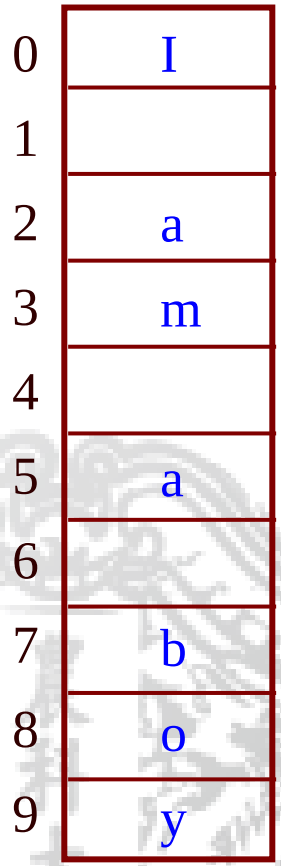
例 char fruit[][7]={"Apple", "Orange",
"Grape", "Pear", "Peach"};

fruit[0]	A	p	p	l	e	\0	\0
fruit[1]	O	r	a	n	g	e	\0
fruit[2]	G	r	a	p	e	\0	\0
fruit[3]	P	e	a	r	\0	\0	\0
fruit[4]	P	e	a	c	h	\0	\0

★字符数组的引用

例9 输出一个字符串

```
#include <stdio.h>
int main()
{ char c[10]={'I',' ','a','m',' ','a',' ','b','o','y'};
   int i;
   for(i=0;i<10;i++)
      printf("%c",c[i]);
   printf("\n");
}
```



例10 输出一个钻石图形

```
#include <stdio.h>
void main()
{ char diamond[ ][5]={{' ',' ','*'},{' ','*',' ','*'},{'*',' ',' ',' ','*'},
                      {' ','*',' ','*'},{' ',' ','*'}};
  int i,j;
  for(i=0;i<5;i++)
    {for(j=0;j<5;j++)
       printf("%c",diamond[i][j]);
     printf("\n");
     }
}
```

运行结果：

```
  *
 * *
*   *
 * *
  *
```

★字符数组的输入输出

- ❖逐个字符I/O：　%c
- ❖整个字符串I/O：　%s

用%c

```
int main()
{    char   str[5];
     int i;
     for(i=0;i<5;i++)
         scanf("%c", &str[i]);
     for(i=0;i<5;i++)
         printf("%c", str[i]);
}
```

输入：China ↵
输出：China
输入：Program ↵
输出：Progr

★字符数组的输入输出

- ❖逐个字符I/O：　%c
- ❖整个字符串I/O：　%s

用字符数组名,不要加&
输入串长度<数组维数
遇空格或回车结束
自动加‘\0’

用%s

```
int main()
{    char   str[5];
     scanf("%s", str);
     printf("%s", str);
}
```

用字符数组名,
遇‘\0’结束,
输出字符不包括结束符‘\0’

其它注意事项：

```
#include <stdio.h>
int main()
{
    char a[ ]={'h','e','l','\0','l','o','\0'};
    printf("%s",a);
}
```

输出：hel

h	e	l	\0	l	o	\0

数组中有多个‘\0’时，
遇第一个结束

```
#include <stdio.h>
int main()
{ char a[15],b[5],c[5];
  scanf("%s%s%s",a,b,c);
  printf("a=%s\nb=%s\nc=%s\n",a,b,c);
  scanf("%s",a);
  printf("a=%s\n",a);
}
```

运行情况：
输入：How are you?
输出：a=How
b=are
c=you?
输入：How are you?
输出：a=How

运行情况：
输入：How are you?

scanf中%s输入输入多个字符串时,遇空格或回车分隔

H	o	w	\0	
a	r	e	\0	
y	o	u	?	\0

例11　若准备将字符串“This is a string.”记录下来，错误的输入语句为：

（A）char s[20];
　　scanf("%20s",s);

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

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

（D）char a[5],b[5],c[5],d[10];
　　scanf("%s%s%s%s",a,b,c,d);

★字符串处理函数

包含在头文件 string.h 中

❖字符串输出函数 puts

- 格式： puts（字符数组或字符串）
- 功能：向显示器输出一个字符串（输出完，换行）
- 说明：字符数组必须以 '\0'结束。可以包含转义字符。

输出时 '\0' 转换成 '\n'，即输出字符后换行，原'\0'后的内容不输出。

这里是将
'\0' ➔ '\n'
因此光标移
到下行

运行结果：
China
Beijing
China
WUHAN

例12：

```
#include <stdio.h>
#include <string.h>
int main( )
{ char  a1[  ]="China\nBeijing" ;
  char  a2[  ]="China\0Beijing" ;
  puts(a1);  puts(a2);
  puts("WUHAN" );
}
```

❖字符串输入函数gets

- 格式：gets (字符数组)
- 功能：从键盘输入一个以回车结束的字符串放入字符数组中，并自动加 '\0' 。
- 说明：输入串长度应小于字符数组维数

例13：gets和scanf输入比较

```
#include <stdio.h>
#include <string.h>
int main( )
{ char  a1[15], a2[15] ;
   gets(a1);
   scanf("%s",a2);
   printf ("a1=%s\n",a1);
   printf ("a2=%s\n",a2);
}
```

注意： puts和gets函数只能输入输出一个字符串。

错 puts(str1,str2)　gets(str1,str2)

输入：China Beijing ↵
　　China Beijing ↵
输出：a1=China Beijing
　　a2=China

❖字符串连接函数strcat

- 格式：strcat (字符数组1,字符数组2或字符串)
- 功能：把字符数组2连到字符数组1后面
- 返值：返回字符数组1的首地址
- 说明：①字符数组1必须足够大

  ②连接前,两串均以'\0'结束;连接后,串1的'\0'取消,新串最后加'\0'。

例14：

```
#include <stdio.h>
#include <string.h>
int main( )
{ char  str1[30]={"People's Republic of "};
  char str2[]={"China"};
  printf ("%s\n",strcat(str1,str2));
}
```

问题： char str1[]={"People's Republic of "}; ？

str1: People's Republic of \0
str2: China\0
str1: People's Republic of China\0

❖字符串拷贝函数strcpy

- 格式：strcpy(字符数组1,字符串2或字符串)
- 功能：将字符串2，拷贝到字符数组1中去
- 返值：返回字符数组1的首地址
- 说明：①字符数组1必须足够大，>字符串2

  ②字符数组1必须是数组名形式（str1），
  字符串 2可以是字符数组名或字符串常量。

  ③拷贝时‘\0’一同拷贝

  ④不能使用赋值语句为一个字符数组赋值

例
```
char str1[20],str2[20];
str1={"Hello! "};          (×)
str2=str1;                 (×)
```

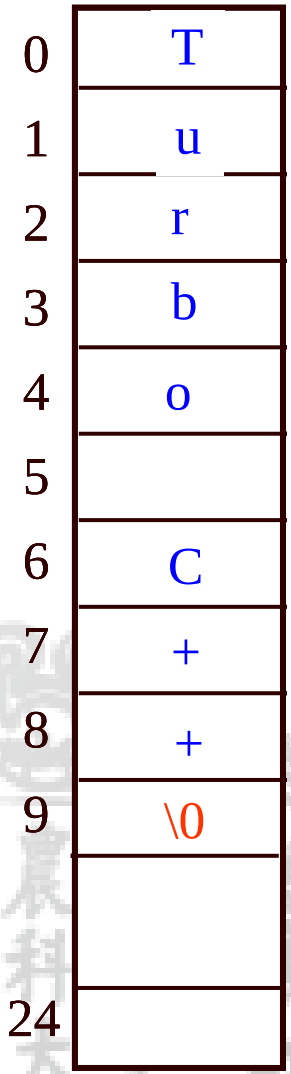
⑤可以只复制字符串2中的前几个字符，来取代字符数组1的前几个字符。

strncpy(str1,str2,2) —— 复制前2个。

例15 strcpy与strcat应用举例

```
#include <stdio.h>
#include <string.h>
int main()
{  char destination[25];
   char blank[] = " ", c[]= "C++",
        turbo[] = "Turbo";
   strcpy(destination, turbo);
   strcat(destination, blank);
   strcat(destination, c);
  printf("%s\n", destination);
}
```

Turbo C++



❖字符串比较函数strcmp

- 格式：strcmp(字符串1或字符数组1,字符串2或字符数组2)
- 功能：比较两个字符串
- 比较规则：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。
- 返回值：返回int型整数。
  a. 若字符串1< 字符串2， 返回负整数
  b. 若字符串1> 字符串2， 返回正整数
  c. 若字符串1== 字符串2， 返回零
- 说明：字符串比较不能用“==”,必须用strcmp，虽然编译无错，但结果不对

错

```
if(str1==str2) printf("yes");
```

```
if(strcmp(str1,str2)==0)  printf("yes");
```

对

例16：字符比较

```
#include <stdio.h>
#include <string.h>
int main( )
{ int  i,j,k;
  char  a1[]="wuhan",  a2[]="beijing" ;
  i=strcmp(a1,a2);
  j=strcmp("china", “korea");
  k=strcmp(a2, "beijing" );
  printf("i=%d\nj=%d\nk=%d\n",i,j,k);
}
```

运行结果：

i=21　　i=w-b=119-98=21

j=-8　　j=c-k=99-107=-8

k=0　　k=b-b=98-98=0……

❖字符串长度函数strlen

- 格式：strlen(字符数组或字符串)
- 功能：计算字符串长度
- 返值：返回字符串实际长度，不包括‘\0’在内；不是字符数组的长度

例17 对于以下字符串，strlen(s)的值为：

（1）char s[10]={'A', '\0', 'B', 'C', '\0', 'D'};

（2）char s[ ]="\t\v\\\0will\n";

（3）char s[ ]="\x69\082\n";

答案：1 3 1

例18：测试字符串长度

```
#include <stdio.h>
#include <string.h>
int main( )
{ char a1[10]="china" ;
  printf ("%d\n",strlen(a1));
  printf ("%d\n", strlen("beijing\0wuhan"));
}
```

运行结果：
5
7

- 大写字母转换成小写字母函数strlwr
  - 格式：strlwr(字符串)
- 小写字母转换成大写字母函数strupr
  - 格式：strupr(字符串)

例19：字符转换

```
#include <stdio.h>
#include <string.h>
int main(   )
{ char  a1[6]="CHinA", a2[ ]="wuHAn" ;
   printf ("%s\n",strlwr(a1)); printf ("%s\n",strupr(a2));
}
```

运行结果：china
WUHAN

注意”\0”对字符串函数的影响

```
#include <stdio.h>
#include <string.h>
int main()
{
   int i,result,length;
   char destination[25];
   char upper[]= "ABCDE",
        lower[] = "abcdefghijklm";
   strcpy(destination, lower);
   printf("%s\n", destination);//"abcdefghijklm\0"

   strcpy(destination, upper);//"ABCDE\0ghijklm\0"
   for(i=0;i<10;i++)
   {
              printf("%c",destination[i]);
   }
   printf("\n");
   printf("%s\n",destination);
```

```
//destination:  "ABCDE\0ghijklm\0"
   //upper :       "ABCDE\0"
   result = strcmp(destination,upper);
   printf("%d\n",result);
   //
   length = strlen(upper);
   printf("length:%d\n",length);
   strupr(destination);
   printf("%s\n",destination);
   for(i=0;i<10;i++)
   {
              printf("%c",destination[i]);
   }
}
```

总结 （1）

1. 字符串输入方法
   1. scanf + %c
   2. while((c=getchar())!='\n') 或者
      while((c=getchar())!= EOF)
   3. scanf + %s
   4. gets()

2. 字符串输出方法
   1. printf + %c
   2. printf + %s
   3. puts()

总结 （2）

1. 字符串问题的解题思路：
   1. 是否可以进行字符串整体操作？使用各种string函数
   2. 否则，需要对每个字符进行遍历（或操作）？对字符变量进行操作
2. 字符串问题的解题思路：
   1. 输入数据和输出结果用什么数据结构？变量还是数组？类型？
   2. 算法 及中间使用的数据结构
3. 寻找字符数组str的尾的方法
   或者对字符数组str中的每个字符进行遍历（或操作）的方法？
   1. i= 0; while(str[i]!='\0')
      { 对str[i]进行操作；　　i++;}
   1. for(i = 0; i< strlen(str); i++) {对str[i]进行操作；}
   2. for(i=0;(c=str[i])!='\0';i++) {对c进行操作；}

例20 接受键盘输入的两个字符串，并将其首尾相接后输出。每个字符串内部不含空格。

分析：

数据结构：

字符串的存储需要用字符数组

算法要点：

字符串输入，可以用具有词处理功能的scanf()函数。

字符串拼接方法：先找到第一个字符串的末尾，然后将第二个串的字符逐个添加到末尾。

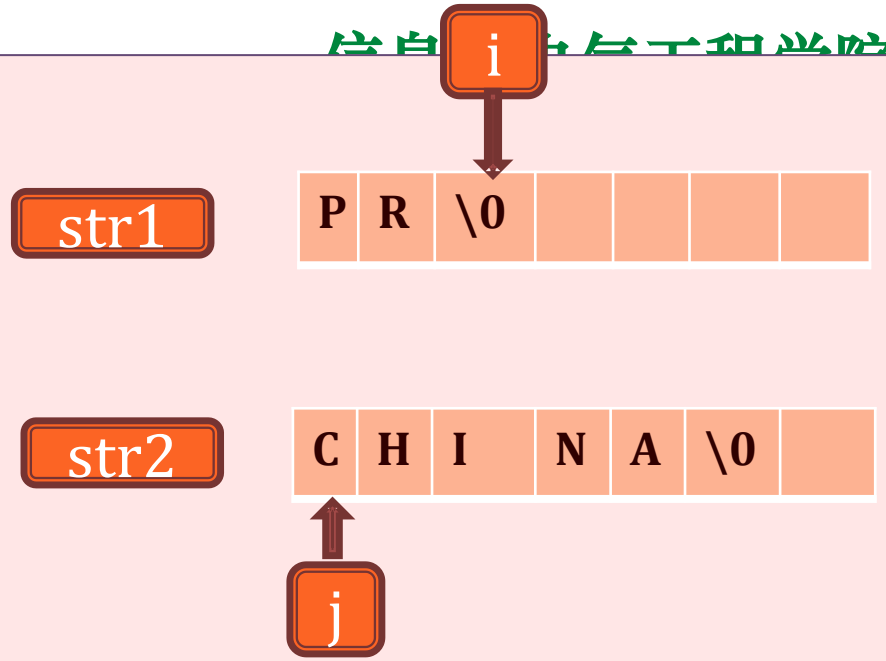
注意，要去掉第一个串的结束符‘\0’，但第二个串的结束符‘\0’要添加进去。

程序1：

```
#include <stdio.h>
#include<string.h>
int main()
{ char str1[50],str2[20];
  int i,j;
  scanf("%s",str1);
  scanf("%s",str2);
    strcat(str1,str2); /*最直接的方式，使用strcat函数*/
    printf(str1);
}
```

程序2：

```
#include <stdio.h>
int main()
{ char str1[50],str2[20];
  int i,j;
  scanf("%s",str1);
  scanf("%s",str2);
  i=j=0;
  while(str1[i]!='\0')  i++;          /*寻找字符串str1的尾*/
  while(str2[j]!='\0')                /*str2加入到str1*/
   {
      str1[i]=str2[j];i++; j++;
   }
   str1[i]=str2[j];                   /*可以去掉吗？ */
                                       /*如果去掉，如何初始化 */
  printf("string No.1->%s\n",str1);
}
```

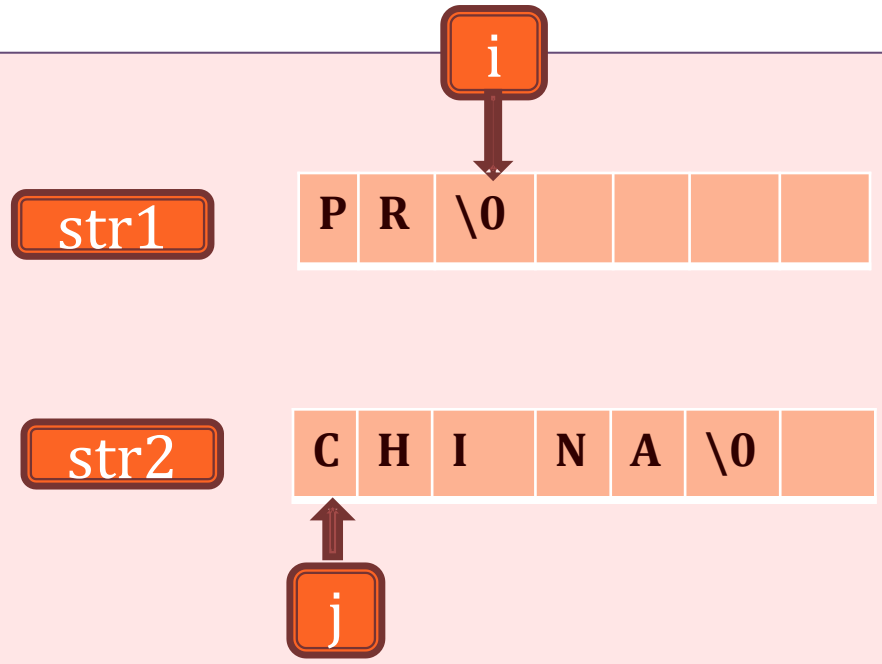


程序3：

```
#include <stdio.h>
#include<string.h>
int main()
{ char str1[50],str2[20];
  int i,j;
  scanf("%s",str1);
  scanf("%s",str2);
  i=0;
  while(str1[i]!='\0')  i++;          /*寻找字符串str1的尾*/
  for(j = 0; j< strlen(str2); j++,i++)              /*    */
   {
      str1[i]=str2[j];
   }
   str1[i]=str2[j];
  printf("string No.1->%s\n",str1);
}
```

程序4：

```
#include <stdio.h>
int main()
{ char str1[50],str2[20];
  int i,j;
  scanf("%s",str1);
  scanf("%s",str2);
  i=j=0;
  while(str1[i]!='\0')  i++;          /*寻找字符串str1的尾*/
  while((str1[i++]=str2[j++])!='\0');   /*str2加入到str1*/
  printf("string No.1->%s\n",str1);
}
```



例21 从键盘输入若干行文本，每行以回车结束，以 ctrl+z 作为输入结束符，统计其行数。

分析：

数据结构：

由于只统计行数，所以不必使用数组存储文本内容，只须定义一个字符变量暂存读入的字符。

算法要点：

读入字符可以用 getchar() 函数。

每读入一个字符，要判断是否输入结束

要判断读入的是否回车符，定义一个整型变量对回车符进行计数，以实现统计行数的功能。

```
#include<stdio.h>
int main()
{
   int nl;
   char c;
   nl=0;
   while((c=getchar())!=EOF)
         if(c=='\n')                /*如果是换行，则nl加1*/
                ++nl;
   printf("%d\n",nl);
}
```

EOF （ end of file，“文字流”结束），标准输入时，Ctrl-Z 代表 EOF

例22　把输入的字符串逆序排列，并显示。

分析：

数据结构：

输入的字符串用字符数组存放。

算法要点：

逆序排列用交换算法，求出字符串最后一个字符的下标，然后将第一个和最后一个交换，第二个和倒数第二个交换，

…。

```
#include <stdio.h>
#include<string.h>
int main()
{  char str[80];
   int temp,i,j;
   scanf("%s",str);
   for(i=0,j=strlen(str)-1;i<j;i++,j--)
   {
     temp=str[i];
      str[i]=str[j];                    /*交换i,j两个元素*/
      str[j]=temp;
   }
   printf("\nReversed string:\n%s\n",str);
}
```

例23　从键盘输入字符，以 ctrl+z 结束，统计输入的数字0～9、空白符和其它字符的个数。

分析：

数据结构：

定义一个具有 10 个元素的整型数组来存放数字 0～9 的个数

定义两个整型变量来存放空白符和其它字符的个数。

算法要点：

计数用的数组和变量要初始化为0。

用循环结构处理字符读入，内嵌分支结构处理计数。

```
#include<stdio.h>
int main()
{ int i,nwhite,nother,ndigit[10]; /*存放相应的计数值*/
  char c;
  nwhite=nother=0;
  for(i=0;i<10;i++)                    /*初始化这些计数值*/
     ndigit[i]=0;
  while((c=getchar())!=EOF)
     if(c>='0'&&c<='9')
           ++ndigit[c-'0';              /*相应的ndigit[i]加1*/
     else if(c==' '||c=='\n'||c=='\t')
           ++nwhite;
     else
           ++nother;
  for(i=0;i<10;i++)
     printf("digit '%d':%d\n",i,ndigit[i]);
  printf("white space:%d\n",nwhite);
  printf("other character:%d\n",nother);
}
```

例24　从键盘输入一个字符串（长度不超过20，其中不含空格），将其复制一份，复制时将小写字母都转换成为大写字母）。

分析：

数据结构

定义两个数组，存放字符串。

算法要点

将小写字母 转换为大写字母：小写字母-'a'+'A'。

```
#include <stdio.h>
int main()
{ char a[20],b[20];
  int i;
  printf("Enter a string:\n");
  scanf("%s",a);
  i=0;
  while(a[i]!='\0')
  {
  b[i]=(a[i]>='a'&&a[i]<='z')?a[i]-'a'+'A':a[i];
    /*将a数组中小写字母转为大写并对b数组赋值*/
    i++;
  }
  b[i] ='\0';
 printf("Copyed string:\n%s\n",b);
}
```

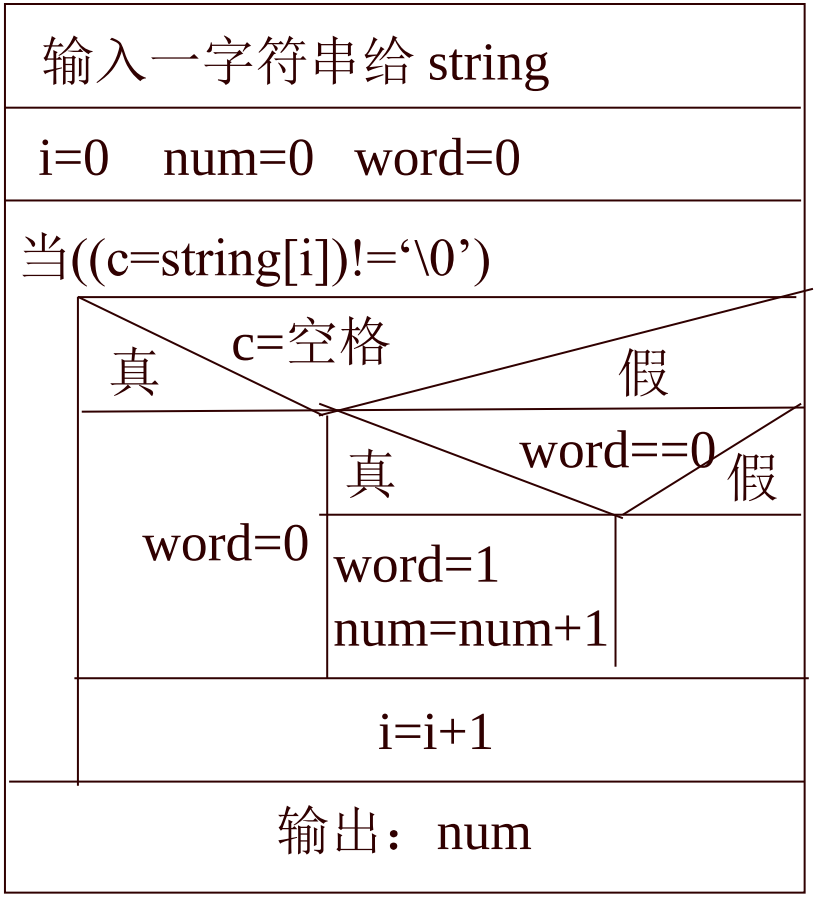
例25 输入一行字符，统计其中的单词个数，单词间空格分开。

分析：根据题目要求，可以用一个字符数组来存储输入的这行字符。要统计其中单词数，就是判断该字符数组中的各个字符，如果出现非空格字符，且其前一个字符为空格，则新单词开始，计数num加1。
但这在第一个单词出现时有点特殊，因为第一个单词前面可能没有空格，因此在程序里我们可以人为加上一个标志word，并初始化为0。该标志指示前一个字符是否是空格，如果该标志值为0则表示前一个字符为空格。

当前字符=空格
- 是：未出现新单词，使word=0,num不累加
- 否：
  - 前一字符为空格(word==0),新单词出现,word=1,num加1
  - 前一字符为非空格(word==1),未出现新单词,num不变

例 输入：I ␣am ␣a ␣boy.

当前字符	I	␣	a	m	␣	a	␣	b	o	y	.
是否空格	否	是	否	否	是	否	是	否	否	否	否
word原值	0	1	0	1	1	0	1	0	1	1	1
新单词开始否	是	未	是	未	未	是	未	是	未	未	未
word新值	1	0	1	1	0	1	0	1	1	1	1
num值	1	1	2	2	2	3	3	4	4	4	4




```
#include <stdio.h>
int main()
{char string[81];
  int i,num=0,word=0;
  char c;
  gets(string);
  for(i=0;(c=string[i])!='\0';i++)
       if(c==' ')  word=0;
       else if(word==0)
                {num++;
                  word=1; }
  printf("There are %d words
                    in the line\n",num);
}
```

例26 有3个字符串，要求找出其中最大者。

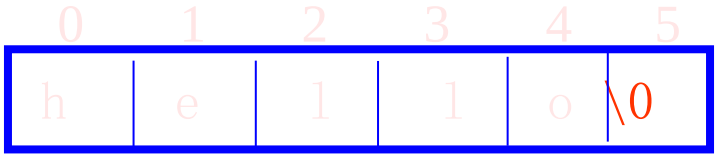
str[0]	H	o	w	\0					
str[1]	H	e	l	l	o	\0			
str[2]	H	i	g	h	\0				

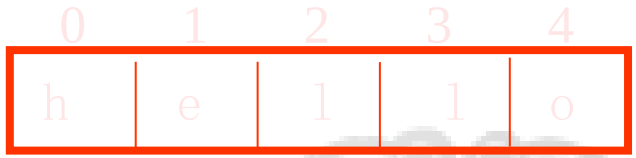
```
#include <stdio.h>
#include <string.h>
int main()
{ char string[20],str[3][20];
  int i;
  for(i=0;i<3;i++)
     gets(str[i]);
  if(strcmp(str[0],str[1])>0)
          strcpy(string,str[0]);
  else  strcpy(string,str[1]);
  if(strcmp(str[2],string)>0)
          strcpy(string,str[2]);
  printf("\nThe largest string
              is:\n%s\n",string);
}
```

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

例 char str[]="Hello";
char str[]={'H', 'e', 'l', 'l', 'o'};

例 int a[10];
float i=3;
a[i]=10;





例 int a[5];
a={2,4,6,8,10};

例 int a[][10];
float f[2][]={1.2 ,2.2};

例 比较 int a[2][3]={{5,6},{7,8}};
与 int a[2][3]={5,6,7,8};

5	6	0
7	8	0

5	6	7
8	0	0