5

数 组

5.1 引言
5.2 一维数组
5.3 字符数组
5.4 多维数组
5.5 数组类型的应用

5.1 引言

数组

- 构造数据类型之一.
- 数组:有序数据的集合,用数组名标识.
- 元素:属同一数据类型,用数组名和下标确定.

5.2 一维数组

一维数组的定义

- 数组的名称
- 数组的大小（长度/数组元素的个数）
- 数组的基类型（元素的类型）

[ ] :数组运算符
单目运算符
优先级(1)
左结合
不能用( )

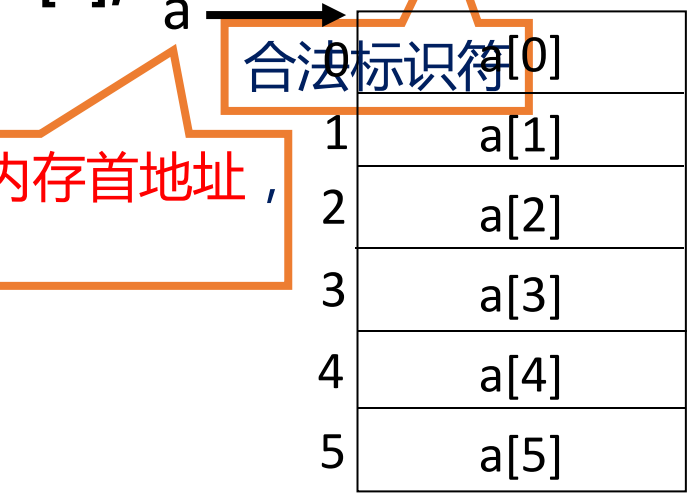
定义方式： **数据类型 数组名[整形常量表达式]；**

例 **int a[6];**

合法标识符

表示元素个数
下标从0开始

数组名表示内存首地址，是地址常量



编译时分配连续内存
内存字节数=
数组长度*sizeof(元素数据类型)

一维数组的引用

- 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]　　下标可以是整形常量或整型表达式

例　int i=15;
　　int data[i];　✕ 不能用变量定义数组长度

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

例　int a[10];
　　printf(''%d'',a);　✕ 不能直接用数组名输出数值数组元素

```
for(j=0;j<10;j++)
    printf(''%d\t'',a[j]);
```

【例5.1】 一头母牛，每年初生一头小母牛。每头小母牛从第四年开始，每年初也生一头小母牛。问在第20年时，共有多少头牛？

思路：

年数	一	二	三	四	五	六	…
牛的数量	2	3	4	6	9	13	

```
#include <stdio.h>
main()
{
    int i,ncow[20+1];
    ncow[1]=2;
    ncow[2]=3;
    ncow[3]=4;
    for(i=4;i<=20;i++)
        ncow[i]=ncow[i-1]+ncow[i-3];
    for(i=1;i<=20;i++)
        printf("%12d%c",ncow[i],i%6?' ':'\n');
}
```

一维数组的初始化

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;

在定义数组时，为数组元素赋初值

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长度

只给部分数组元素赋初值
int a[5]={6,2,3}; a[0]=6; a[1]=2;a[2]=3; a[3]=0; a[4]=0;

1、未初始化，同时数组中的某些元素未被赋值时，其值有两种可能，一种是全局数组，被编译器初始化为0，一种是局部数组，为随机数。在vc6.0的debug版程序中一般是0xcccc的值。
2、已初始化，不管是局部数组还是全局数组，编译器都会将其未被赋值的元素初始化为0。

【例5.2】 读10个整数存入数组，求总和和平均值，并找出其中最大值、最小值

```
#include<stdio.h>
main()
{
    int array[10],i,sum=0,max,min;
    for(i=0;i<=9;i++)
    {
        printf("enter %dth number:",i+1);
        scanf("%d",&array[i]);
        putchar("\n");
    }
    for(max=min=array[0],i=0;i<=9;i++)
    {
        sum=sum+array[i];
        if(array[i]>max)     max=array[i];
        if(array[i]<min)     min=array[i];
    }
    printf("sum=%-5d average=%-.1f max=%-5d min=%-5d\n",sum,(double)sum/10,max,min);
}
```

数组定义、变量定义

通过循环逐一输入数组数据

通过一次循环即可求出总和
和最大值、最小值

【例5.3】计算2019年的某月某日是当年的第几天？这一天是一个星期中的第几天（已知2019年元旦是星期二）？

天数%7+1即是该天是一个星期中的第几天


```c
#include<stdio.h>
int main()
{
    int _month[13]={0,31,29,31,30,31,30,31,31,30,31,30,31};
    int month,day,i,week;
    while(1)
    {
        printf("input month:");
        scanf("%d",&month);
        if(month<1||month>12){printf("month error!");putchar('\n');continue;}
        else break;
    }
    while(1)
    {
        printf("input day:");
        scanf("%d",&day);
        if((day<1)||((month==1||month==3||month==5||month==7||month==8||month==10||month==12)&&day>31)||((month==4||month==6||month==9||month==11)&&day>30)||(month==2&&day>29))
            {printf("day error!\n");putchar('\n');continue;}
        else break;
    }
    for(i=1;i<month;i++)
        day+= _month[i];
    week=day%7+1;
    printf("it's %dth day in 2019.\n",day);
    printf("it's %dth day in a week.\n",week);
}
```

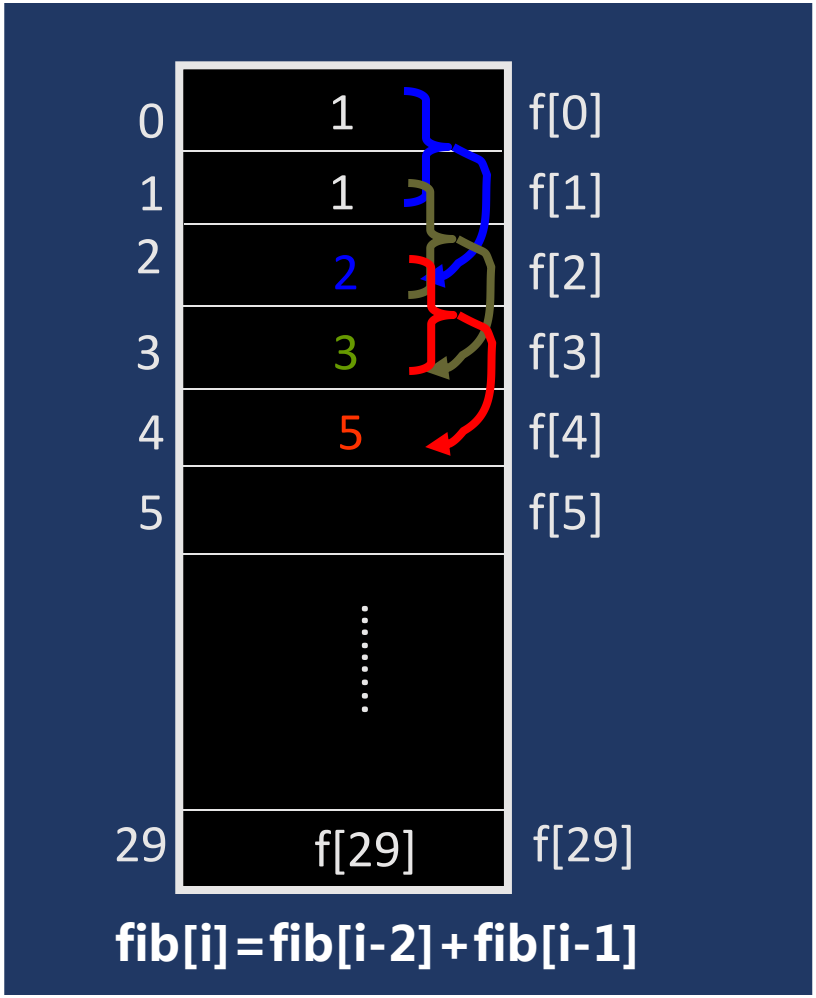
输入月份并对输入数据合法性进行判断

输入日期并对输入数据合法性进行判断

计算当年的第几天和一星期中的第几天

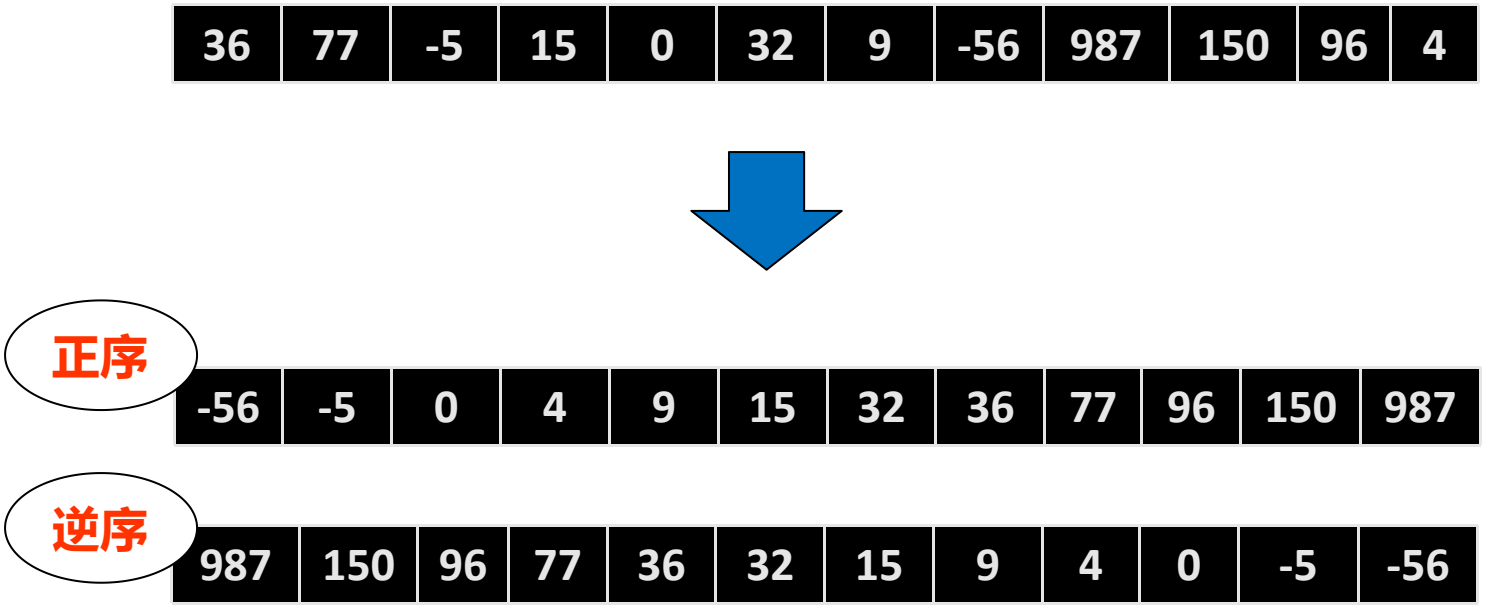
【例5.4】 用数组求Fibonacci数列前30个数

```
#include <stdio.h>
main()
{    int i;
     long int fib[30]={1,1};
     for(i=2;i<30;i++)
        fib[i]=fib[i-2]+fib[i-1];
     for(i=0;i<30;i++)
     {
        if(i%5==0)  printf("\n");
        printf("%12ld",fib[i]);
     }
}
```



排序

❖排序是计算机内经常进行的一种操作，其目的是将一组“无序”的记录序列调整为“有序”的记录序列。



常用的排序算法

- 冒泡排序
- 选择排序
- 插入排序
- 快速排序
- 堆排序
- 归并排序
- 基数排序
- 希尔排序

常用的排序算法

- ❖冒泡排序
- ❖选择排序
- ❖插入排序
- ❖快速排序
- ❖堆排序
- ❖归并排序
- ❖基数排序
- ❖希尔排序

冒泡排序

- 冒泡排序（Bubble Sort），是一种较简单的排序算法。
- 通过无序区中相邻记录关键字间的“比较”和位置的“交换”，实现关键字较大（或较小）的记录向序列“一端”移动，从而达到记录按关键字递增（或递减）顺序排列的目的。
- 这个算法的名字由来是因为越大（或越小）的元素会经由交换慢慢“浮”到数列的顶端，故名。

n=8

初始序列	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟	第七趟	排序结果
38	38	38	38	13	13	13	13	13
49	49	49	13	27	27	27	27	27
65	65	13	27	30	30	30		30
76	13	27	30	38	38			38
13	27	30	49	49				49
27	30	65	65					65
30	76	76						76
97	97							97

排序过程：

（1）比较第一个数与第二个数，若为逆序a[0]>a[1]，则交换

（2）依次类推，比较第二个数与第三个数，直至第n-1个数和第n个数比较为止——第一趟冒泡排序结束，结果最大的数被安置在最后一个元素位置上。

（3）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。

（4）重复上述过程，共经过n-1趟冒泡排序后，排序结束。

```c
#include<stdio.h>
void main()
{
    int a[11],i,j,t;
    printf("Input 10 Numbers:\n");
    for(i=1;i<=10;i++)
        scanf("%d",&a[i]);
    putchar('\n');
    for(j=1;j<=9;j++)
        for(i=1;i<=10-j;i++)
            if(a[i]>a[i+1])
            {t=a[i];a[i]=a[i+1];a[i+1]=t;}
    printf("output:\n");
    for(i=1;i<=10;i++)
        printf("%5d",a[i]);
}
```

定义数组、输入数组元素数据

利用双重循环，两两比较数组相邻元素，9轮过后排序结束

```c
#include<stdio.h>
void main()
{
    int a[11],i,j,t;
    printf("Input 10 Numbers:\n");
    for(i=1;i<=10;i++)
        scanf("%d",&a[i]);
    putchar('\n');
    for(j=1;j<=9;j++)
        for(i=1;i<=10-j;i++)
            if(a[i]>a[i+1])
            {t=a[i];a[i]=a[i+1];a[i+1]=t;}
    printf("output:\n");
    for(i=1;i<=10;i++)
        printf("%5d",a[i]);
}
```

```c
#include<stdio.h>
void main()
{
    int a[11],i,j,t;
    int flag;
    printf("Input 10 Numbers:\n");
    for(i=1;i<=10;i++)
        scanf("%d",&a[i]);
    putchar('\n');
    for(j=1;j<=9;j++)
    {
        flag=1;
        for(i=1;i<=10-j;i++)
            if(a[i]>a[i+1])
            {
                t=a[i];a[i]=a[i+1];a[i+1]=t;
                flag=0;
            }
        if(flag==1) break;
    }
    printf("output:\n");
    for(i=1;i<=10;i++)
        printf("%5d",a[i]);
}
```

定义flag变量

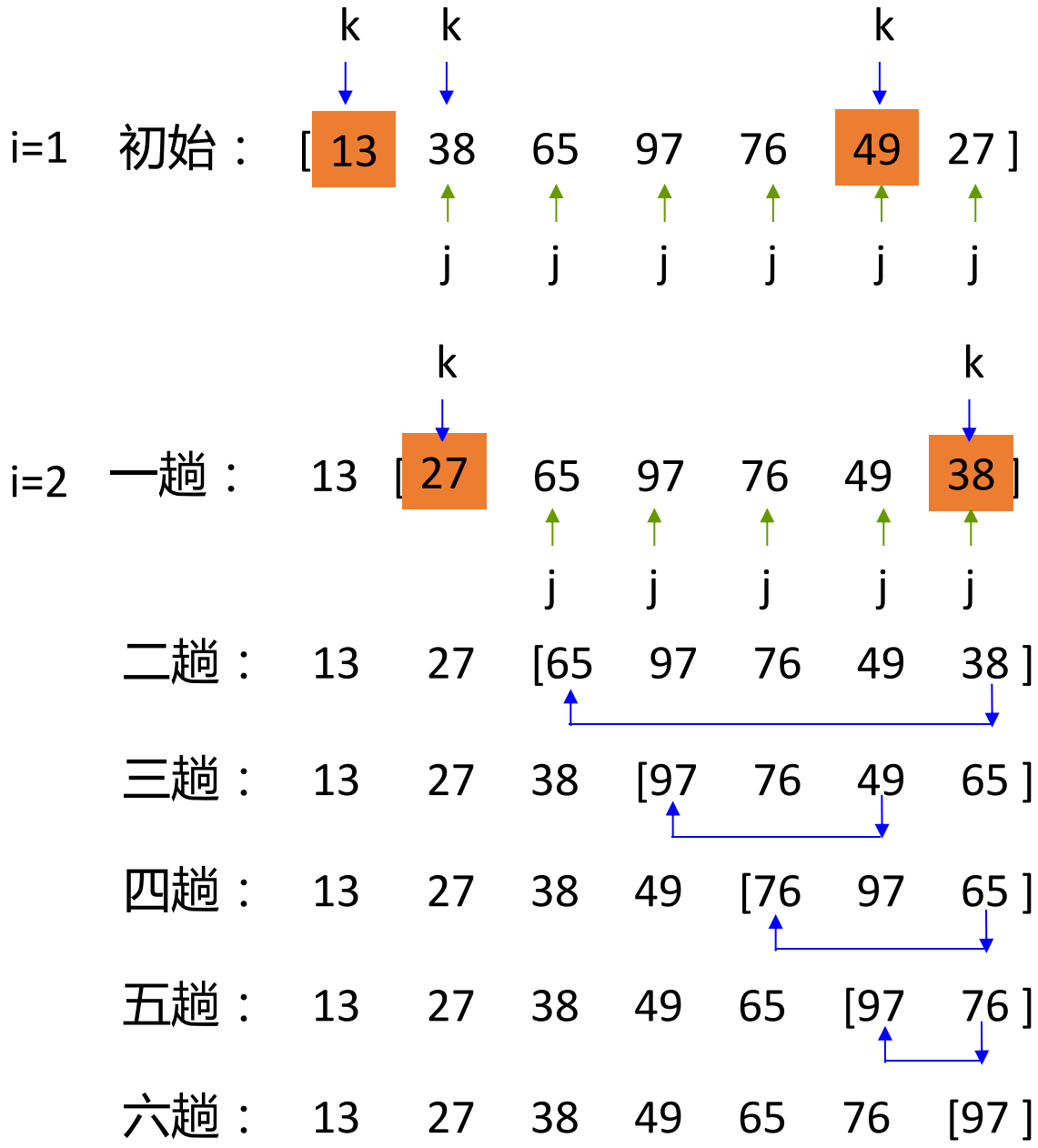
利用flag变量判断是否有交换从而结束排序

选择排序

（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。


k k k
i=1 初始： [13 38 65 97 76 49 27]
j j j j j j
k k
i=2 一趟： 13 [27 65 97 76 49 38]
j j j j j
二趟： 13 27 [65 97 76 49 38]
三趟： 13 27 38 [97 76 49 65]
四趟： 13 27 38 49 [76 97 65]
五趟： 13 27 38 49 65 [97 76]
六趟： 13 27 38 49 65 76 [97]

```c
#include <stdio.h>
void main()
{    int a[11],i,j,k,x;
     printf("Input 10 numbers:\n");
     for(i=1;i<11;i++)
        scanf("%d",&a[i]);
     printf("\n");
    for(i=1;i<10;i++)
    {   k=i;
        for(j=i+1;j<=10;j++)
            if(a[j]<a[k])  k=j;
        if(i!=k)
        {  x=a[i]; a[i]=a[k]; a[k]=x;}
    }
     printf("The sorted numbers:\n");
     for(i=1;i<11;i++)
      printf("%5d ",a[i]);
}
```

每次开始内层循环前将k复位到i

内层循环将k的值置于该轮最小元素的位置

判断最小元素是否就在该轮第一个位置上

5.3 字符数组

字符数组和字符串的关系

- 定义
- 字符数组的初始化
  - 逐个字符赋值
  - 用字符串常量
- 用库函数对字符数组赋值

例　char c[10], ch[3][4];

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

有问题!

```
char ch[5]={'B','o','y'};
```

```
char ch[6]={"Hello"};  char ch[6]="Hello";
char ch[]="Hello";
```

```
char name[16];
gets(name);
```

```
gets()
```

```
%s
    int i=0;
    while((c=getchar())!='\n')
        name[i++]=c;
    printf("%s",name);
}
```

? 是否能实现一串字符串的输入

字符数组应用（常用的字符串处理函数）

包含在**string.h**头文件里

字符串输出函数**puts**
格式：puts(字符数组数组名)
功能：向显示器输出字符串（
说明：字符数组必须以 '\0'

字符串输入函数**gets**
格式：gets(字符数组数组名)
功能：从键盘输入一以回车结
并自动加 '\0'
说明：输入串长度应小于字符

例
```
#include <stdio.h>
main( )
{
    char string[80];
    printf( "Input a string:" );
    gets(string);
    puts(string);
}
```
输入: How are you?
输出: How are you ?

字符串连接函数**strcat**

格式：strcat(字符数组1,字符数组2)

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’

字符串比较函数**strcmp**

格式：strcmp(字符串1,字符串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**

字符串拷贝函数**strcpy**

格式：strcpy(字符数组1,字符数组2/字符串常量)

功能：将字符数组2中的字符串，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1一般比字符数组2长

②拷贝时 ‘\0’ 一同拷贝

③不能使用赋值语句为一个字符数组赋值

```
char str1[20],str2[20];
str1={''Hello!''};
str2=str1;
```

×

字符串拷贝函数**strncpy**

格式：strcnpy(字符数组1,字符数组2,非负整型变量n)

功能：将字符串2中的n个字符拷贝到字符数组1中去

返值：返回字符数组1的首地址

说明：①参数n必须小于等于字符数组1的长度

②满足条件①的情况下，如果n>字符数组2长度，则将字符数组2全部复制到字符数组1中（包括'\0'）.

strcpy与strcat举例

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

【例5.13】演示使用strncpy()

```
#include<stdio.h>
#include<string.h>
main()
{
    char src[]="one world,one dream!";
    char dest1[40]={0},dest2[40]={0};
    strncpy(dest1,src,10);
    printf("The string dest1 now is \"%s\"\n",dest1);
    strncpy(dest2,src+strlen(src)-10,10);
    printf("The string dest1 now is\"%s\"\n",dest2);
}
```

字符串长度函数**strlen**

格式：strlen(字符数组数组名)

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

例 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

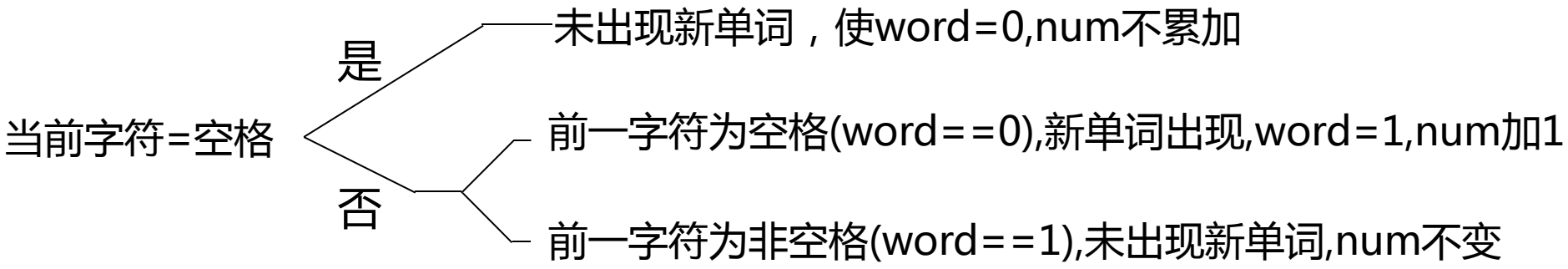
【例5.14】对一个特定文本中特定的某个单词进行统计.

```
#include<stdio.h>
#include<string.h>
main()
{
    int i=0,sum=0,length;
    char word[20];
    char text[]="10 plus 5 is 15,10 minus 5 is 5,10 multiplied by 5 is 50,10 divided by 5 is 2.";
    char temp[20]={0};
    printf("please input the word whick you want to count:");
    gets(word);
    length=strlen(word);
    while(text[i]!='\0')
    {
        strncpy(temp,text+i,length);
        if(strcmp(word,temp)==0)
        {
            sum++;
            i+=length;
        }
        i++;
    }
    printf("the word \"%s\" appears %d times in the text.\n",word,sum);
}
```

求目标单词的长度

根据目标单词利用strncpy逐个向后匹配查找

另外一种统计单词的方法




```c
#include <stdio.h>
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)
       {   word=1;  num++;   }
    printf("There are %d words  \
       in the line\n",num);
}
```

5.3 二维数组和多维数组

二维数组的定义

定义方式：

数据类型　数组名[常量表达式][常量表达式]；

行数　列数

- 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

int c[2][3][4]

0	c[0][0][0]
1	c[0][0][1]
2	c[0][0][2]
3	c[0][0][3]
4	c[0][1][0]
5	c[0][1][1]
6	c[0][1][2]
7	c[0][1][3]
	c[0][2][0]
	c[0][2][1]
	c[0][2][2]
	c[0][2][3]
	c[1][0][0]
	c[1][0][1]
	c[1][0][2]
	c[1][0][3]
	c[1][1][0]
	c[1][1][1]
	c[1][1][2]
	c[1][1][3]
20	c[1][2][0]
21	c[1][2][1]
22	c[1][2][2]
23	c[1][2][3]

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

int a[3][2]

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

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

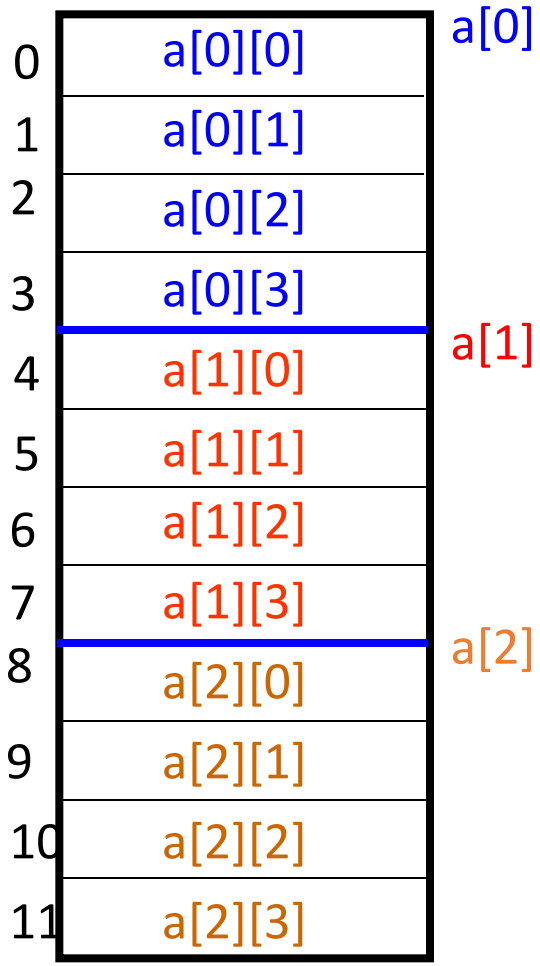
二维数组理解

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]

每个元素a[i]由包含4个元素的一维数组组成



例 将二维数组行列元素互换，存到另一个数组中

```
#include <stdio.h>
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");
   }
```

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

```
printf("array b:\n");
   for(i=0;i<=2;i++)
   {   for(j=0;j<=1;j++)
             printf("%5d",b[i][j]);
         printf("\n");
   }
}
```

【例5.15】 计算两个矩阵相乘

矩阵相乘的条件：A×B **矩阵A的列数要与B的行数相等** A[i][k]×B[k][j]=C[i][j]

记作C=AB,其中矩阵C中的第i行第j列元素可以表示为：

$$(AB)_{ij}=\sum_{k=1}^{p}a_{ik}b_{kj}=a_{i1}b_{1j}+a_{i2}b_{2j}+\cdots+a_{ip}b_{pj}$$

$$C=AB=\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}\begin{pmatrix}1&4\\2&5\\3&6\end{pmatrix}=\begin{pmatrix}1\times1+2\times2+3\times3&1\times4+2\times5+3\times6\\4\times1+5\times2+6\times3&4\times4+5\times5+6\times6\end{pmatrix}=\begin{pmatrix}14&32\\32&77\end{pmatrix}$$

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

}
```

三层循环计算矩阵相乘

【例5.15】 实现矩阵的转置，构成一个新的二维数组

转置：将一个矩阵的行列互换

初始矩阵的赋值和输出

行列互换

```
#include<stdio.h>
int main()
{
    int i,j,k=1,temp;
    int a[4][4]={0};
    for(i=0;i<4;i++)
        for(j=0;j<4;j++)
            a[i][j]=k++;
    printf("a矩阵转置前:\n");
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
            printf("%6d",a[i][j]);
        printf("\n");
    }
    for(i=0;i<4;i++)
        for(j=0;j<i;j++)
        {
            temp=a[i][j];
            a[i][j]=a[j][i];
            a[j][i]=temp;
        }
    printf("a矩阵转置后:\n");
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
            printf("%6d",a[i][j]);
        printf("\n");
    }
}
```

转置完成后输出

例 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
8	23	3
15	7	9
2	5	17

感觉很简单？

程序怎么写？

简单粗暴的写法

```
int array[4][3]={12,4,6,8,23,3,15,7,9,2,5,17};
int row_sum[4],column_sum[3],all_sum;
row_sum[0]=12+4+6;
row_sum[1]=8+23+3;
row_sum[2]=15+7+9;
row_sum[4]=2+5+17;
column_sum[0]=12+8+15+2;
column_sum[1]=4+23+7+5;
column_sum[2]=6+3+9+17;
all_sum=12+4+6+8+23+3+15+7+9+2+5+17;
```

例 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	
8	23	3	
15	7	9	
2	5	17	

有没有问题？

```
#include<stdio.h>
main()
{
    int array[5][4],i,j;
    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
        {
            printf("%d row,%d column:",i+1,j+1);
            scanf("%d",&array[i][j]);
        }

    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
        {
            array[i][3]+=array[i][j];
            array[4][j]+=array[i][j];
            array[4][3]+=array[i][j];
        }
    for(i=0;i<5;i++)
        {
            for(j=0;j<4;j++)
                    printf("%5d\t",array[i][j]);
            putchar('\n');
        }
}
```

例 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	
8	23	3	
15	7	9	
2	5	17	

```
#include<stdio.h>
main()
{
    int array[5][4],i,j;
    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
        {
            printf("%d row,%d column:",i+1,j+1);
            scanf("%d",&array[i][j]);
        }
    for(i=0;i<4;i++)     array[i][3]=0;
    for(j=0;j<4;j++)     array[4][j]=0;
    for(i=0;i<4;i++)
        for(j=0;j<3;j++)
        {
            array[i][3]+=array[i][j];
            array[4][j]+=array[i][j];
            array[4][3]+=array[i][j];
        }
    for(i=0;i<5;i++)
        {
            for(j=0;j<4;j++)
                    printf("%5d\t",array[i][j]);
            putchar('\n');
        }
}
```

```c
#include<stdio.h>
int main()
{
    int a[11],i,x,flag,j;
    for(i=1;i<=19;i=i+2)
        a[(i-1)/2]=i;
    printf("array:\n");
    for(i=0;i<=9;i++)
        printf("%3d",a[i]);
    printf("\ninput x:");
    scanf("%d",&x);
    flag=0;
    for(i=0;i<10;i++)
        if(a[i]==x)
        {
            flag=1;break;
        }
        else if(a[i]>x) break;
    if(flag==1)
    {
        printf("\nx is in array\n");
        for(i=0;i<=9;i++)
            printf("%3d",a[i]);
        printf("\n");
    }
    else if(i<=10)
    {
        printf("x is not in array,inserting x.\n");
        for(j=9;j>=i;j--)
            a[j+1]=a[j];
        a[i]=x;
        for(i=0;i<=10;i++)
            printf("%3d",a[i]);
        printf("\n");
    }
}
```

寻找x要插入的位置，并用flag的值标识数组中是否有待查数据元素

若flag==1,说明数组中存在待查元素，直接输出即可

若flag==0,根据之前的定位，移动数组元素，将x插入到定位好的位置i

【例5.26】实现数据的折半查找

```c
int bin_search(int A[],int n,int key)
{
    int low,high,mid;
    low=0;
    high=n-1;
    while(low<=high)
    {
        mid =(low + high)/2;
        if(A[mid]==key) return mid;
        if(A[mid]<key)
            low=mid+1;
        if(A[mid]>key)
            high=mid-1;
    }
    return -1;
}
main()
{
    int a[20],i,k;
    for(i=0;i<=19;i++)
        a[i]=i*2;
    printf("array is:");
    for(i=0;i<=19;i++)
        printf("%3d",a[i]);
    printf("\ninput number:");
    scanf("%d",&k);
    i=bin_search(a,20,k);
    if(i!=-1)
        printf("a[%d]=%d\n",i,a[i]);
    else
        printf("not found!");
}
```

折半查找函数
用low、high、mid定位数组折半查找，找到后返回位置mid，未找到返回1

调用折半查找函数并返回位置输出