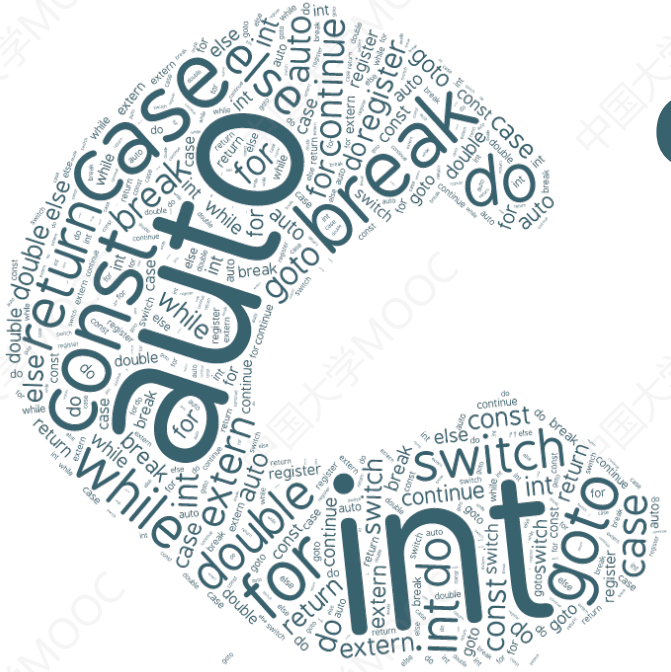

auto
break
case
char
const
continue
default
do
double
else
enum
extern
float
for
goto
if
int
long
register
return
short
signed
sizeof
static
struct
switch
typedef
union
unsigned
void
volatile
while

chapter 5

数组

目录
content

5.1 数组的定义和应用

数组：数组是有序数据的集合,即具有一定顺序关系的若干变量的集合体.组成数组的变量称为该数组的元素变量,简称元素,用数组名后跟带有方括号“[]”的下标来唯一确定数组中的元素。

C语言中有一维数组和多维数组

本教材中多维数组只介绍二维数组相关内容

5.1.1 一维数组的定义和应用

一维数组的定义:

[存储类型] 数据类型 数组名[常量表达式];

例： int a[10];

它表示数组名为a，数组有10个元素。

数组必须先有定义，然后再使用。C语言规定只逐个引用数组元素而不能一次引用整个数组。

数组在内存中存储时，是按下标递增的顺序连续存储各元素变量的值。定义语句中的常量表达式表示元素个数，即数组长度。

5.1.1 一维数组的定义和应用

应用要点:

- 数组的下标从0开始，下标必须是整型量。

  a[10]中，10表示a数组有10个，下标从0开始，这10个元素是a[0],a[1],a[2],a[3], a[4],a[5],a[6] ,a[7],a[8],a[9]，注意不能使用数组元素a[10]

- 数组名表示数据存储区域的首地址。数组的首地址也是第一个元素变量的地址

  int data[5];

  首地址是data或&data[0]

  数组名是一个地址常量，不能向它赋值，也不能对它自加自减等对变量进行操作的运算

5.1.1 一维数组的定义和应用

- 数组的初始化　数组的初始化就是在数据说明时对数组元素赋初值

初始化方式：

- 数组长度与初值的个数相等

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

←→data[0]=2;data[1]=4;data[2]=6;data[3]=8;data[4]=10;

或:　　int data[]={2,4,6,8,10};

- 数组长度与提供初值的个数不相等

```
int b[20]={1,2,3,4,5};
```

数组长度不能省掉,多余的元素都是0;

b[0]=1;b[1]=2;b[2]=3;b[3]=4;b[4]=5;b[5]=b[6]=...=b[19]=0

- 数组下标常量表达式中可以包括常量和符号常量，不能包含变量

错误的定义:

(1)
```
int  n ;
scanf("%d",&n);
int a[n];
```

(2)
```
int n=20;
int a[n];
```

C不允许对数组的长度作动态定义

5.1.1 程序举例

例5.2　编程将一个从键盘输入的整数序列按逆序重新存放并显示，整数个数首先从键盘输入; 如要求输入5个数，原来的顺序为8，6，5，4，1，要求改为1，4，5，6，8

```
#include <stdio.h>
void main( )
{
    int a[100];
    int i,j,n,temp;
    scanf("%d",&n);   //输入整数个数
    printf("input the numbers:\n");
    for(i=0;i<n;i++)    //输入整数序列
    {
        scanf("%d",&a[i]);
    }
   /*将整数序列依次从首尾向中间交换元素,从而实现逆序排列*/
    for(i=0,j=n-1; i<j; i++,j--)
    {
       temp=a[j];
       a[j]=a[i];
       a[i]=temp;
    }
    printf("now the numbers are:\n");
    //输出重排后的整数序列
    for(i=0;i<n;i++)
    {
        printf("%5d",a[i]);
    }
}
```

逐个元素输入

两个数交换

运行结果：
```
5
input the numbers:
8 6 5 4 1
now the numbers are :
1 4 5 6 8
```

5.1.1 程序举例

例5.3： 输入10个整数到一个数组中,调整这10个数组中的排列位置,使得其中最小的一个数成为数组的首元素.

```
include<stdio.h>
#define   SIZE   10
void main( )
{
   int m,k;  //初始化
   int i,j;
   int data[SIZE];
   printf("input the size");
   for ( m = 0;m < SIZE; m++)    //输入数组中的值
   {
      scanf("%d",&data[m]);
   }

   j=0;
   for(i=0;i<SIZE;i++)
  {
      if(data[i]<data[j])   j=i;
   }
    if(j>0)      //如果最小值下标不是0,
则将该值和数组首项中的值交换
   {
      k=data[0];
      data[0]=data[j];
      data[j]=k;
   }
   printf("\n");
   for(m=0;m< SIZE;m++)   //输出调整后
的数组
   {
      printf("%4d",data[m]);
   }
 }
```

符号常量

比较数组中的值，
记下最小值的下标

5.1.2 二维数组的定义和应用

二维数组的定义:

[存储类型] 数据类型 数组名[下标][下标]

例： float a[3][3]={1,2,3,4,5,6,7,8,9};

存放顺序与存储方式:

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

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

二维数组的定义和应用

二维数组的理解:

对于所定义的数组a，可看成是三个一维数组a[0],a[1],a[2];比如a[0]是二维数组中一个特殊元素，它是包含三个元素的一维数组。

$$a\begin{cases} a[0]\begin{cases} a[0][0] \\ a[0][1] \\ a[0][2] \end{cases} \\ a[1]\begin{cases} a[1][0] \\ a[1][1] \\ a[1][2] \end{cases} \\ a[2]\begin{cases} a[2][0] \\ a[2][1] \\ a[2][2] \end{cases} \end{cases}$$

二维数组的定义和应用

二维数组的初始化:

对全部元素都赋予初值

```
int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
int a[ ][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
```

定义数组时对第一维的下标可以不声明，但第二维长度不能省略

可以将所有数据写在一个花括号内，按数组排序的顺序对各元素赋初值

对部分元素赋初值

```
int a[3][3]={{1,2},{1,2,3},{1}}
```

它只对各行前面几列赋初值，其余元素值自动为0

$$\begin{bmatrix} 1 & 2 & 0 \\ 1 & 2 & 3 \\ 1 & 0 & 0 \end{bmatrix}$$

5.1.2 程序举例

例5.4　有一个3×3矩阵，要求编程求出其中最大的那个元素的值，以及其所在的行号和列号。

```
#include <stdio.h>
void main( )
{
   int i,j,max;
   int row=0,column=0;
   int a[3][3] = {{1,2,3},{2,-3,4},{9,4,7}};
   /*数组初始化*/

   max=a[0][0];
```

```
/*以下循环将数组中的元素比较，记下最大值的下标*/
   for (i=0;i<3;i++)
   {
      for (j=0;j<3;j++)
      {
         if (a[i][j]>=max)
         {
            max=a[i][j];
            row=i;
            column=j;
          }
      }
   }
    printf("max=%d,row=%d,column=%d\n",
                                        max,row,column);
    /*输出最大值以及 其行下标和列下标*/
}
```

运行结果：

max=9,row=2,column=0

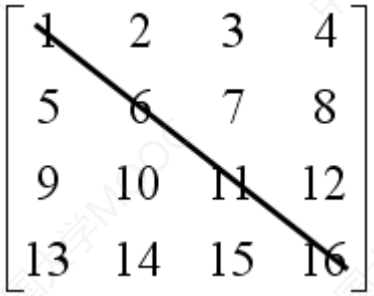
5.1.2 程序举例

例5.5 输入一个4×4的整数矩阵,然后将之转置并显示这个转置后的矩阵。

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}$$

分析:以主对角线为对称轴，交换所有对称点元素

对称点元素：行列位置交换



5.1.2 程序举例

```
#include<stdio.h>
#define SIZE  4
void main( )
{
    int data[SIZE][SIZE],i,j,d;

    for(i=0;i<SIZE;i++)    //输入矩阵中的值
    {
       for(j=0;j<SIZE;j++)
       {
          scanf("%d",&data[i][j]);
       }
    }
```

```
    for(i=0;i<SIZE-1;i++)     //矩阵转置
    {
       for(j=i+1;j<SIZE;j++)
       {
          //交换所有对称点元素
          d=data[i][j];
          data[i][j]=data[j][i];
           data[j][i]=d;
       }
    }

    for(i=0;i<SIZE;i++)
    {
       printf("\n");
       for(j=0;j<SIZE;j++)
            printf("%4d",data[i][j]);
    }
}
```

5.2 数组在函数间的传递

值传递方式 数组元素作为函数实参，用法与变量相同，单向传递

例5.6 求一整型数组的平均值，要求主函数中输入数据，通过调用函数求平均值。在数组元素个数较小的情况下，可以直接传递数组元素。

```
#include<stdio.h>
float aver( float x,float y,float z);
void main( )
{
   float a[3],temp;
   int i;
   printf("input the numbers:\n");
   for(i=0;i<3;i++)      //输入整数序列
   {
      scanf("%f",&a[i]);
   }
   temp = aver( a[0],a[1],a[2]);
   printf("aver = %f",temp);
}
```

实参为数组元素

```
float aver( float x,float y,float z)
{
   return (x + y + z)/3;
}
```

地址传递方式

数组在函数间的传递由于涉及到指针的概念，具体将在指针一章中详细论述，本节仅通过例子来描述数组在函数间传递的过程。

例5.7 从键盘输入十个整数存入一个一维数组中，编写一函数求其最大值。

```
#include <stdio.h>
int findmax( int b[], int n);
void main( )
{
   int a[10];
   int i,temp;
   printf("input the numbers:\n");
   for(i=0;i<10;i++)      //输入整数序列
   {
      scanf("%d",&a[i]);
   }
   temp = findmax(a, 10);
   printf("the max = %d",temp);
}
```

实参为数组首地址

```
 int findmax( int b[], int n)
{
   int max,i ;
   max = b[0] ;
   for(i=1 ; i<n ;i++)
   {
      if(b[i] > max) max = b[i];
   }
   return max;
}
```

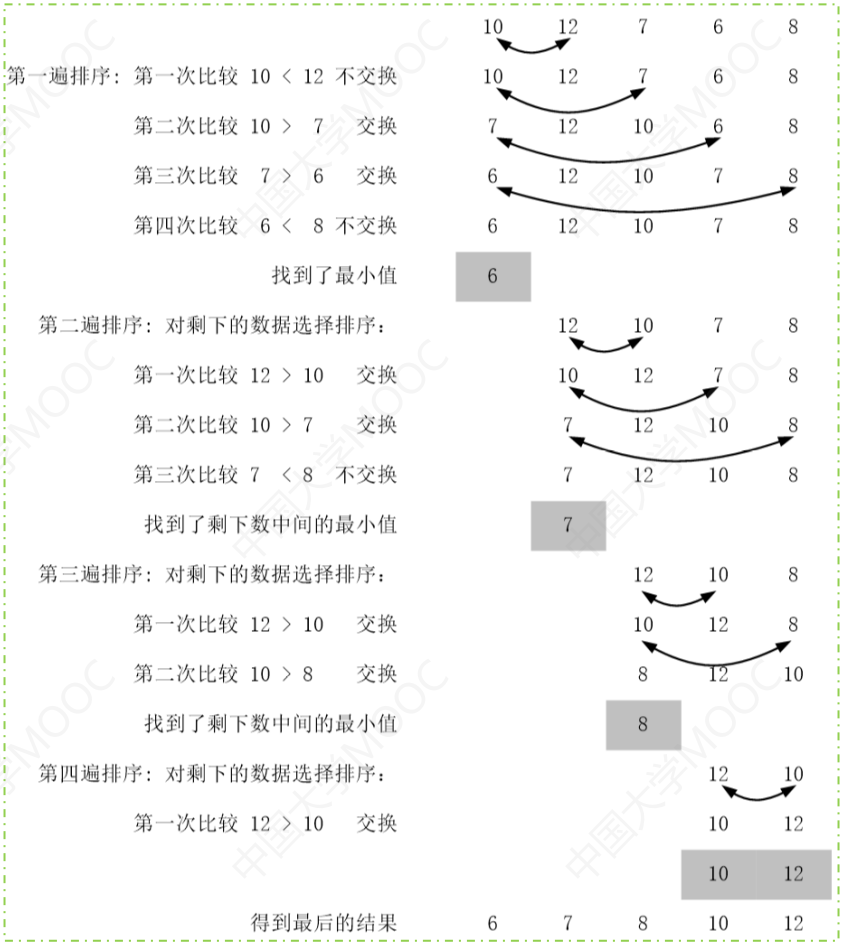
例5.8　编一程序要求对已知的10个数，按从小到大进行排序。

排序方法：（1）选择排序法
（2）冒泡排序法

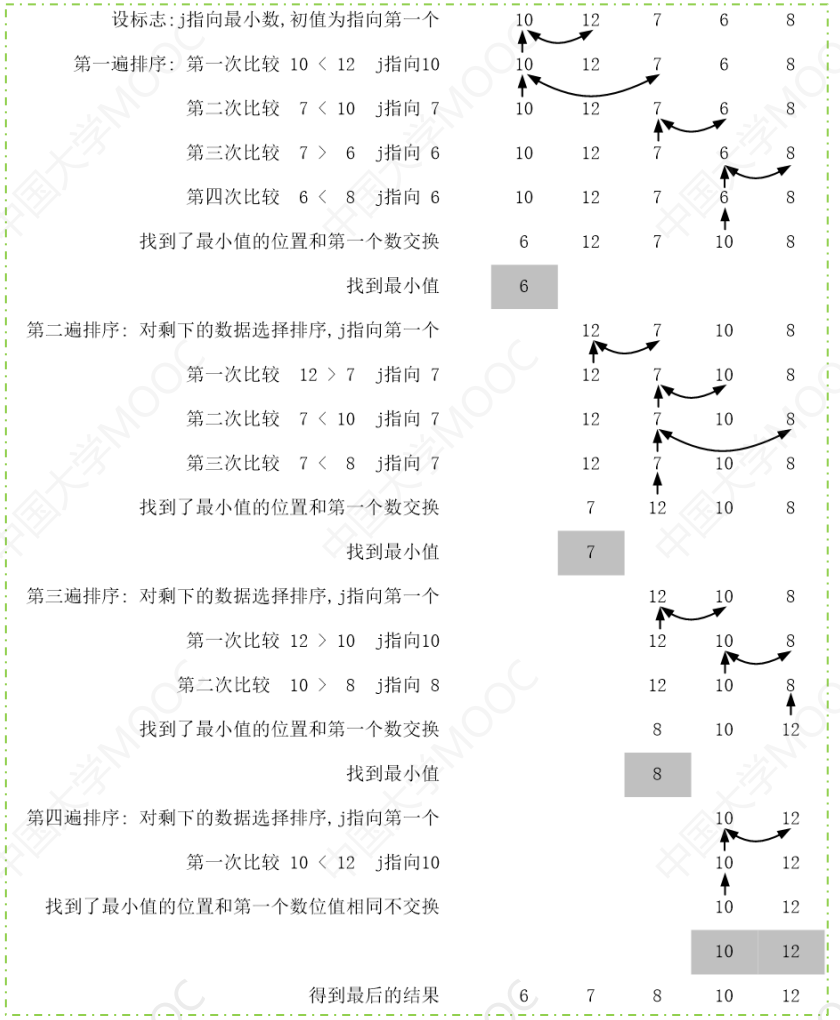
（1）选择排序法

思想:首先进行第一遍排序，确定第一个数为基准数，认为它为最小数，然后依此在所有其它数中找比它小的最小数，如有则交换，这样就找到第一个最小数; 第二遍排序是对除第一个数据外的数据序列进行选择排序，以此类推共进行N-1遍排序，就能将N个数排序;

对N个数要进行N-1遍排序，每一遍的比较次数随遍数的增加而减小(遍数+次数=N) 。



方法1：边比较边交换



方法2：边找边设立标记

5.3 程序设计举例

方法1：边比较边交换

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

剩余元素与第一个元素进行比较，若小于第一个元素，则该元素与第一个元素交换

方法2：边找边设立标记

```
#include <stdio.h>
#define N 10
void main()
{
   int i, j, k,t;
   int a[N];
   for ( i=0 ;i < N ;i++)
         scanf("%d",&a[i]);
    for ( i=0 ;i<N-1; i++)
    {
       j=i;       //记录最小值下标
       for ( k = i+1;k<N;k++)
       {
          if(a[k] < a[j])   j=k;
        }
        if(j!=i)
        {
          t = a[i];      a[i] = a[j];      a[j] = t;
        }
    }
    for ( i=0 ;i < N ;i++)    printf("%d\t", a[i]);
}
```

剩余元素与第一个元素进行比较，若小于第一个元素，则记录该元素的下标

(2) 冒泡排序法

思想: 首先进行第一遍排序，第一个元素开始，将相邻的两个数比较，将小的数字放在前面，大的放在后面，依次往后比较，比较到最后一组后，产生一个最大值; 第二编排序是除去产生的最大值，对剩下的序列进行类似第一遍冒泡排序。依此类推，进行N-1遍排序就能将N个数按从小到大排序;同样的方法也能按从大到小排序。

对N个数要进行N-1遍排序，每一遍比较次数随遍数的增加而减小(遍数+次数=N)，比较的顺序是从前往后。

程序设计举例

冒泡排序法

```
#include <stdio.h>
#define N 10
void main()
{
   int i,j,t;
   int a[N];

   for ( i=0 ;i < N ;i++)
        scanf("%d",&a[i]);
   for ( i=0 ;i<N-1; i++)
   {
      for (j=0;j<N-1-i;j++)
      {
        if(a[j]>a[j+1])
        {
          t = a[j];  a[j] = a[j+1];  a[j+1] = t;
        }
      }
   }
   for ( i=0 ;i < N ;i++)     printf("%d\t", a[i]);
}
```

当前元素与下一个元素进行比较，若大于下一个元素，则该两元素交换

冒泡排序法-main( )

```
#include <stdio.h>
#define N 10
void BubSort(int *a, int n);
void main()
{
   int i,j,t;
   int a[N];

   for ( i=0 ;i < N ;i++)
         scanf("%d",&a[i]);

   BubSort(a, N);

   for ( i=0 ;i < N ;i++)     printf("%d\t", a[i]);
}
```

冒泡排序法-函数实现

```
void BubSort(int *a, int n )
{
   int i,j,t;

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

当前元素与下一个元素进行比较，若大于下一个元素，则该两元素交换

例5.9 计算如下所示两个矩阵的乘积

$$\begin{bmatrix} 2 & 3 & -5 & 0 \\ 12 & -1 & 27 & 8 \\ 91 & 22 & -32 & 1 \end{bmatrix} \quad \begin{bmatrix} 25 & 13 & 65 & 0 & 5 \\ -2 & 0 & 18 & 10 & 45 \\ 53 & 33 & 3 & 9 & 0 \\ 7 & 61 & 26 & -37 & -1 \end{bmatrix}$$

分析：

(1)第一个矩阵的列数必须等于第二个矩阵的行数，一个m行n列的矩阵乘以一个n行p列的矩阵，其结果是一个m行p列的矩阵;

(2)若矩阵A乘以矩阵B，则结果矩阵C中每个元素的值为:

$$C_{i,j} = \sum_{k=1}^{n} a_{i,k} * b_{k,j}$$

$$C_{i,j}^{k+1} = C_{i,j}^{k} + a_{i,k+1} * a_{k+1,j}$$

$$\begin{bmatrix} 2 & 3 & -5 & 0 \\ 12 & -1 & 27 & 8 \\ 91 & 22 & -32 & 1 \end{bmatrix} \begin{bmatrix} 25 & 13 & 65 & 0 & 5 \\ -2 & 0 & 18 & 10 & 45 \\ 53 & 33 & 3 & 9 & 0 \\ 7 & 61 & 26 & -37 & -1 \end{bmatrix}$$

$C_{24} = 12*0+(-1)*10+27*9+8*(-37)$

$=0-10+243-296$

$=-63$

程序设计举例

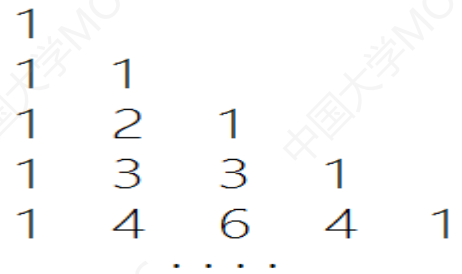
```
#include<stdio.h>
void main( )
{
 int valueA[3][4]={ {2,3,-5,0}, //数组初化
                    {12,-1,27,8},
                    {91,22,-32,1}};
 int valueB[4][5]={ {25,13,65,0,5},
                    {-2,0,18,10,45},
                    {53,33,3,9,0},
                    {7,61,26,-37,-1}};
 int valueC[3][5]={{0}};  //定义结果矩阵
 int i,j,k;
 //按照公式将矩阵相乘
 for(i=0;i<3;i++)
 {
  for(j=0;j<5;j++)
  {
   for(k=0;k<4;k++)
   {
    valueC[i][j]+=valueA[i][k]*valueB[k][j];
   }
  }
 }
```

```
 for(i=0;i<3;i++)   //输出结果矩阵
 {
   printf("\n");
   for(j=0;j<5;j++)
   {
     printf("%5d",valueC[i][j]);
   }
 }
}
```

运行结果：

```
 -221  -139   169   -15   145
 1789  1535  1051   -63     7
  542   188  6241  -105  1444
```

例5.10　输出下列形式的杨辉三角形的前10行



分析： 杨辉三角形是$(a+b)^n$展开后各项的系数，有以下规律：

（1）各行的第一个数都是1,即a[i][0]=1；

（2）各行的最后一个数都是1,即a[i][i]=1；

（3）从第3行起，除上面指出的第一个数和最后一个数外，其余各数是上一行同列和前一列两个数之和，可以这样表示：

a[i][j]=a[i-1][j-1]+a[i-1][j]

其中i为行数，j为列数。

程序设计举例

```
# include<stdio.h>
void main( )
{
  int a[10][10];
  int i,j;
  for(i=0;i<10;i++)
  {
     a[i][0]=1;
     a[i][i]=1;
   }
   for (i=2;i<10;i++)
   {
        for (j=1;j<i;j++)
              a[i][j]=a[i-1][j-1]+a[i-1][j];
   }
   for(i=0;i<10;i++)
   {
       for(j=0;j<=i;j++)
              printf("%5d",a[i][j]);
       printf("\n");
   }
}
```

按规律求数组元数a[i][j]的值

本章小结

本章要掌握的内容有：

- 数组的概念；
- 一维数组和二维数组的定义和应用；
- 数组在函数中的传递方式；
- 选择排序法和冒泡排序法的思想和算法。