# 第5章　数　　组

数组通常用来存放成批的具有相同类型的数据。在实际问题中，如果涉及批量数据的顺序或位置关系，或者要多次使用同一批量数据等情况时，就要使用数组。例如，在统计学生成绩时经常要统计高于平均成绩的人数或者按成绩排名次时，就需要使用数组处理。


数组引例

**数组**是指一批具有相同类型数据的有序集合，用一个名字（称为**数组名**）来表示，数组属于构造数据类型。数组中的元素用位置号（称为**下标**）表示它的顺序。有一个下标的数组称为**一维数组**，有两个下标的数组称为**二维数组**，多个下标的数组称为“多维数组”。

应用本章的知识可以完成学生成绩管理的程序，根据输入的学生成绩统计每个人、每门课程的最高分、最低分、平均成绩、超过平均成绩的人数以及按每个人的平均成绩排序，查找不及格的学生的成绩等。

## 5.1　一维数组

一维数组通常用来表示具有相同数据类型的一组数，可以对这组数进行排序和查询等。一维数组的常用算法包括顺序排序法、选择排序法和冒泡排序法等排序的算法，以及数组逆序存放和二分查找法等。

### 5.1.1　一维数组的定义和引用


一组数组的定义和引用

#### 1. 一维数组的定义

数组必须先定义后使用。定义一维数组的一般形式是：

类型标识符 数组名[数组长度]

**【说明】** 方括号[ ]内的数组长度表示数组中元素的个数，必须是整数或整型常量表达式，不能包含变量。

例如，int cj[30]，a[10]；float b[20]；，都是合法的数组定义。而 int n＝10；int a[n]；，这个数组 a 的定义是不合法的。

数组在内存中占据一片连续的存储单元。若有定义 int a[10]；，则表示数组 a 中包括 10 个**元素**，分别为 a[0]～a[9]。这 10 个元素在内存中是连续存放的，如图 5.1 所示。只要知道数组 a 的起始元素 a[0]的地址，后面元素的相应地址就可以计算出来。由于 C 语言中用数组名代表数组的起始地址，所以此处的数组名 a 也代表数组 a 的起始地址，与 a[0]的地址相同。

数组的下标从 0 开始。若有数组定义 int a[10]；，则数组 a 的最小下标为 0，最大可

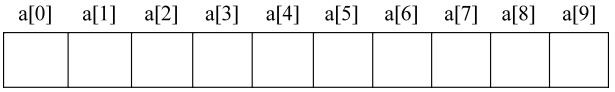


图 5.1　数组中 10 个元素在内存中连续存放

用下标为 9，C 的编译器不对下标越界进行检查，因此编程者必须保证程序中所用数组的下标是不越界的。

### 2. 一维数组的引用

引用一维数组元素的一般形式是：

数组名[下标]

【说明】　[]内的下标必须是整型表达式，可以只是一个常量、变量，也可以是表达式。

例如，a[1]，a[i]，a[9－i](i 是整型)等都是合法的数组元素的引用，数组元素也称为下标变量。

数组元素的处理往往与循环分不开，输入和输出都要结合循环进行。假设 int 型数组 cj 有 30 个元素，输入所有元素时可以使用下面的循环语句对 cj 的所有元素进行遍历。

```
for(i=0;i<30;i++)
    scanf("%d",&cj[i]);
```

如果数组中的数组元素的值是有某种计算规律的，有时也可以使用循环直接给数组元素赋值，不必一个一个地输入。

【例 5-1】　数组元素的引用。

```
#include "stdio.h"
main()
{  int i,b[10];
   for(i=0;i<10;i++)          /*通过循环遍历数组元素*/
      b[i]=i+1;               /*赋值后，数组 b 中各元素的值依次为 1,2,3,…,10*/
   for(i=0;i<10;i++)          /*再次遍历数组元素*/
      printf("%5d",b[i]);     /*格式%5d 的作用是控制每个数据占 5 列*/
                              /*如果使用格式%d，输出的数将连在一起*/
   printf("\n");
}
```

运行结果：

```
    1    2    3    4    5    6    7    8    9   10
```

## 5.1.2　一维数组的初始化

定义数组时直接给数组的一些元素赋值，称为对数组的初始化，也可称为对数组赋

初值。

对数组的初始化可以用以下 3 种方法实现。

(1) 定义数组时,对全部数组元素赋以初值。例如:

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

(2) 定义数组时,只给一部分元素赋初值。例如:

```
int a[10]={1,2,3,4,5};
```

表示只给数组 a 的前 5 个元素赋初值,后边元素自动被赋值为 0。

(3) 在对全部数组元素赋初值时,可以不指定数组长度。例如:

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

与下面的写法等价:

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

一组数组的初始化

【注意】 许多情况下,赋初值是针对循环变量、累加(累乘)变量、数列的初始项等而言的,所以对数组、变量赋初值的方法不是唯一的,既可以通过定义时的初始化实现,也可以通过其他方法(如赋值语句、for 语句中的表达式 1 等)实现。

【例 5-2】 用数组求 Fibonacci 数列的前 20 项。

【分析】 例 4-13 求 Fibonacci 数列时,在循环中求出一项输出一项,不能同时保存 20 项的数据;而用数组表示 Fibonacci 数列时,则可以同时保存求出的所有的数。

```
#include "stdio.h"
main( )
{  int i;
   long int f[20]={1,1};                    /*前两项都赋初值 1*/
   for(i=2;i<20;i++)                        /*循环 18 次,循环变量 i 从 2 到 19*/
      f[i]=f[i-2]+f[i-1];                   /*后一项等于前两项的和*/
   for(i=0;i<20;i++)                        /*对数组中的 20 个元素遍历*/
      {  printf("%12ld",f[i]);             /*输出元素 f[i]*/
         if((i+1)%5==0)printf("\n");        /*控制每行输出 5 个数*/
      }
}
```

运行结果:

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

【说明】 程序中采用每输出 5 个数换一次行的方法,也可以用以下程序段先换行后再输出 5 个数(输出第一个数前先换一次行)的方法。

```
for(i=0;i<20;i++)
   {  if(i%5==0)printf("\n");
      printf("%12ld",f[i]);
   }
```

编程时,一般采用先输出若干个数再换行的方法并且列对齐输出。

### 5.1.3 随机函数 rand 和 random


随机函数

可以用随机函数产生一组随机整数赋值给数组中的各个元素,C 中用 rand()函数实现。

#### 1. rand 函数

rand 函数的一般形式是:rand()。它产生 0 到整型最大值之间的一个随机整数,如要产生[a,b]的随机整数可以使用 rand()%(b−a+1)+a 实现。

rand 函数位于标准库 stdlib.h 中,使用该函数时要先用文件包含命令 #include "stdlib.h"(或 #include <stdlib.h>)。

#### 2. random 函数

在 Turbo C 中,random 是包含在头文件 stdlib.h 中的宏定义:

```
#define random(num) (rand()%(num))
```

宏定义(详见 6.7.1 节)的参数用一对括号括起来,使用时和函数类似,所以也可把 random 当作函数使用。使用 random 函数时也需要文件包含命令 #include "stdlib.h",调用的一般形式是:random(n)。它产生[0,n−1]的随机整数,如要产生[a,b]的随机整数可以使用 random(b−a+1)+a 实现。

在 Turbo C 中可以用 random 函数代替 rand 函数,但在 Visual C++中只能使用 rand 函数,不能使用 random 函数。

#### 3. srand 函数和 randomize 函数

srand 函数的一般形式是:srand(unsigned seed)。它产生随机整数的种子,能使程序中每次运行 srand 函数时得到不同的随机整数序列。通常的使用方法是:srand(time(0)),用当前的时间作为随机数的初始种子。time 函数在时间库 time.h 中,srand 函数在标准库 stdlib.h 中。

在 Turbo C 中,randomize 是包含在头文件 stdlib.h 中带参数的宏定义:

```
#define randomize() srand((unsigned)time (NULL))
```

调用形式是:randomize(),类似于一个无参函数。在 Turbo C 中可以用 randomize 函数代替 srand 函数,但在 Visual C++中只能使用 srand 函数,不能使用 randomize 函数。

产生随机整数的具体方法参见例 5-6 和例 5-7。由于 random 和 randomize 函数不是 C 中真正意义的库函数，为了通用，建议使用 rand 和 srand 函数。

### 5.1.4 一维数组的简单应用

考虑设计一个学生成绩管理系统，可以完成成绩统计、查询、排序等功能。例如，成绩统计包括求平均成绩、最高分、最低分等。

当要解决一个实际问题时，首先要研究采用什么样的解决方案，使用什么样的算法，以及是否使用过类似的算法等，要把实际问题变成计算机能够解决的问题。

1）选择或者创造一个抽象

例如，通过抽象，可以将学生的成绩用一组数据表示，然后选择合适的数据结构来处理它。显然，学生成绩的数据用数组来表示比较恰当。

2）采用逐步求精的方法设计解决问题的方案

解决问题需要遵循一定的过程或规则，培养解决问题的能力是伴随着整个程序设计的过程逐步实现的。简单地说，解决问题的方法就是采用逐步求精、各个击破的思想。当面对比较复杂的问题时，要设法将其分解为若干个比较简单的问题，并逐步分解，使这些比较简单的问题最终分解成为简单的、最好是与已经解决的问题类似的基本问题。把这些基本问题逐一解决之后，再拼接在一起去解决原来的复杂问题。

以此成绩管理系统为例，可将它分解为成绩统计、成绩排序、成绩查询等若干个较小的问题。当较小的问题还不够简单时就继续简化，例如成绩统计这个问题可以再细分为求平均成绩、求最高分、求最低分等更小的问题，由此形成各种基本问题。当这些基本问题解决之后，可以逐步拼接成较大的程序去解决原来的复杂问题。本章将逐步介绍这些基本问题的程序设计方法，最后组成解决综合性问题的例 5-30。

3）按照算法编写程序代码实现

同样的问题可以采用不同的程序设计语言编程实现，采用的算法也会有细微差别，但应尽量使用通用的算法。

4）运行测试结果

采取各种测试方法对程序输入各类数据进行测试性运行，以保证程序的正确性。

下面给出一些基本问题的解决方法，从这些例子中体会用数组解决问题的大致思路。

**【例 5-3】** 求 10 个学生成绩的平均成绩。

例 5-3

**【分析】** 在例 4-9 中编程求解过若干个学生的平均成绩。本例中已知学生个数为 10，这样就可以定义一个包含 10 个元素的一维数组，同时也不必使用终止标记法输入数据了。

编程解决具体问题时，为了保证程序的易读性，一般将程序划分为 3 个部分。

（1）输入数据，或用其他方式给数组、变量等赋值。

（2）处理数据，也就是计算过程，例如求和、比较大小等。

（3）输出数据，即输出计算结果。

```
#include "stdio.h"
main( )
{  int i,a[10];
   float average,sum;
   for(i=0;i<=9;i++)                    /* 循环变量 i 从 0 到 9,遍历 10 个元素 */
      scanf("%d",&a[i]);                /* 输入第 i 个成绩放到 a[i]中 */
   sum=0;                               /* sum 存放成绩和,赋初值为 0 */
   for(i=0;i<10;i++)                    /* for 循环用于累加 10 个元素 */
      sum+=a[i];                        /* 将 a[i]累加到 sum 中 */
   average=sum/10;                      /* for 循环结束之后再求平均值 */
   printf("平均成绩=%7.2f\n",average);  /* 输出平均成绩 */
}
```

运行结果：

```
63 88 89 90 66 73 61 92 72 78↙
平均成绩=   77.20
```

【说明】 如果 sum 是整型的变量，求平均成绩的语句要改为

```
average=sum/10.0;
```

例 5-4

【例 5-4】 求 10 个数的最小值和最大值。

【分析】 求最大值的算法和例 4-8 类似，只是把 10 个数都保存在数组中。

```
#include "stdio.h"
main( )
{  int i,a[10],min,max;    /* 用数组 a 存放 10 个数,min、max 分别存放最小值和最大值 */
   for(i=0;i<=9;i++)       /* 循环变量 i 从 0~9,遍历 10 个元素 */
      scanf("%d",&a[i]);
   max=min=a[0];                    /* a[0]为最大值和最小值的初值 */
   for(i=1;i<10;i++)                /* 遍历 a[1]到 a[9],求最大值和最小值 */
     {  if(a[i]<min) min=a[i];      /* 若 a[i]比 min 还小,到目前为止 a[i]是最小值 */
        if(a[i]>max) max=a[i];      /* 若 a[i]比 max 还大,到目前为止 a[i]是最大值 */
     }
   printf("最大值=%d,最小值=%d\n", max,min);
}
```

运行结果：

```
63 88 89 90 66 73 61 92 72 78↙
最大值=92,最小值=61
```

【说明】 例 5-3 和例 5-4 程序的功能不用数组也可以实现，但是下面例 5-5 的问题如果不用数组仅用变量是不好实现的，因为它不但涉及要记住一个数在一组数中的位置，而且求出该位置后还要再次使用这批数。


例 5-5

【例 5-5】 求 10 个数的最小值，并将该最小值与最前面的元素互换。

【分析】 因为要将最前面的数换到原来最小值所在的位置，所以求最小值时要记住最小值的位置，数组中的位置通常指的就是其下标。

此问题的核心部分可以分解为两个问题：

(1) 求最小值 min，并记下最小值在原数组中的位置 k。

(2) 将最小值 a[k]和 a[0]交换。

第(1)个问题只是比例 5-3 中的程序多一步，所以在例 5-3 中的程序中将求最小值部分改为 if(a[i]<min){ min=a[i];k=i;}，同时在循环之前给 k 赋初值为 0。

注意到程序中 min 和 a[k]都是表示最小值，而且一直相等，所以只要记住一个 k 就够用了。例 5-5 算法的 N-S 图如图 5.2 所示。

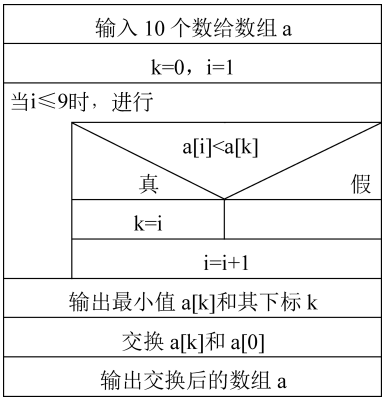

图 5.2 例 5-5 算法的 N-S 图

```
#include "stdio.h"
main( )
{  int i,t,a[10],k;
   for(i=0;i<=9;i++)
      scanf("%d",&a[i]);
   k=0;                          /*设 a[0]最小,k 为最小值的下标,所以 k 赋初值为 0*/
   for(i=1;i<10;i++)             /*a[0]已设为最小,所以循环从下标 1 开始*/
     if(a[i]<a[k])k=i;           /*如果 a[i]比当前最小值 a[k]还小,就将 i 赋值给 k*/
   printf("最小值是:%d\n",a[k]);     /*第 9 行*/
   printf("最小值的下标为:%d\n",k);
   if(k!=0)                      /*如果 k=0,最小值就是 a[0],不必交换*/
      {t=a[0]; a[0]=a[k]; a[k]=t;}  /*将 a[0]和 a[k]交换,即将最小值换到最前面*/
   for(i=0;i<=9;i++)             /*用 for 循环输出变化后的数组*/
      printf("%3d",a[i]);
   printf("\n");
}
```

运行结果：

```
67 98 76 64 89 58 52 69 91 87↙
最小值是:52
最小值的下标为:6
 52 98 76 64 89 58 67 69 91 87
```

【说明】 程序的一般步骤是输入数据、处理数据和输出数据，但三者也常常交叉。例如，本例中输出数据和处理数据(a[0]和 a[k]交换)就交叉了。把程序中从第 9 行开始的 4 行用以下程序段替换：

```
if(k!=0)
```

```
    {t=a[0]; a[0]=a[k]; a[k]=t;}
printf("最小值是:%d\n",a[0]);
printf("最小值的下标为:%d\n",k);
```

就可以避免输出数据和处理数据部分的交叉，但输出的最小值应是交换后的 a[0]，其易读性弱了。可见，程序总体上分成 3 步就可以，没有必要一定划分为严格的 3 步，还需要具体问题具体对待。

**【问题扩展】** 这个程序完成了将 10 个数的最小值交换放到 a[0]中的功能。按照同样的做法，可以找出从 a[1]到 a[9]中的最小值交换放到 a[1]的位置，再找出从 a[2]到 a[9]中的最小值交换放到 a[2]的位置，以此类推，最后从 a[8]到 a[9]中找出最小值交换放到 a[8]中，剩下的 a[9]就是 10 个数中的最大值。这样，就完成了将 10 个数按照由小到大的次序排序的任务，这种排序的方法称为**选择排序法**。简单地说，对 10 个数的选择排序法，就是重复 9 轮次求最小值的位置，并将该位置的最小值交换放到相应位置的过程。

对 n 个数选择排序的算法描述：

（1）给数组 a 中元素输入数据。

（2）对 i＝0，1，…，n－2 做

① k＝i；

② 对 j＝i＋1，…，n－1 做

如果 a[j]＜a[k]，则 k＝j；

③ 交换 a[k]和 a[i]。

（3）输出数组 a。

例 5-6

**【例 5-6】** 产生 10 个[40，100]的随机整数，并用选择排序法按由小到大的顺序排序后输出。

**【分析】** 使用例 5-5 问题扩展中讨论的方法排序。

```
#include "stdio.h"
#include "stdlib.h"                  /* 函数 rand、srand 包含在 stdlib.h 中 */
#include "time.h"                    /* 函数 time 包含在 time.h 中 */
main( )
{  int i,j,t,a[10],k;
   srand(time(0));   /* 用时间作为随机数的种子，每次运行得到不同的随机数序列 */
   for(i=0;i<10;i++)
      {  a[i]=rand()%61+40;          /* 产生[40,100]的随机整数赋给数组元素 a[i] */
         printf("%5d",a[i]);         /* 输出随机产生的数据(排序前的) */
      }
   printf("\n");
   for(i=0;i<9;i++)                  /* 循环变量 i 从 0~8，共 9 轮 */
      {   k=i;          /* 求 a[i]~a[9]中最小值的位置 k，先假定 a[i]最小，将 k 赋初值 i */
         for(j=i+1;j<10;j++)         /* 遍历 a[i+1]~a[9]，求最小值下标赋值给 k */
            if(a[j]<a[k]) k=j;
         if(k!=i)                    /* 如果 k=i，最小值就是 a[i]，不必交换 */
            {t=a[i]; a[i]=a[k]; a[k]=t;}
```

```
                                        /*将a[i]与此轮求出的最小值a[k]交换位置*/
    }
  for(i=0;i<10;i++)                     /*输出排序后的10个随机整数*/
    printf("%5d",a[i]);
  printf("\n");
}
```

某一次运行结果：

```
  67   82   63   45  100   58   99   71   84   51
  45   51   58   63   67   71   82   84   99  100
```

【说明】 此程序在 Visual C++和 Turbo C 环境下都可正常运行，在 Turbo C 环境下还可以使用 random 函数和 randomize 函数组合，具体用法参考例 5-7。

对一组数排序的方法有很多种，除了选择排序法外，还有顺序排序、冒泡排序、插入排序和快速排序等方法。

【例 5-7】 用顺序排序法实现例 5-6 的排序功能。


例 5-7

【分析】 顺序排序法的循环变量的变化过程与选择排序相同。按升序进行顺序排序时，也是先安排好第一个数，即设法将最小的数交换到第一个数的位置(元素的下标为 0)，方法是将后面的每一个数都和第一个数比较，如果后面的数比第一个数还小就立即交换相比较的两个数，一轮循环完成后就将最小的数放在了最前边。第二轮循环按照同样的方法将剩下的 9 个数(第一个数不再参与比较)中的最小数交换放在第二个数的位置(元素的下标为 1)。以此类推，直到将最后两个数中较小的数交换到倒数第二个数的位置(元素的下标为 8)，整个顺序就排好了。

```
#include "stdio.h"
#include "stdlib.h"
#include "time.h"
main()
{  int i,j,t,a[10];
   randomize();                         /*随机数种子，保证每次运行得到不同的随机数序列*/
   for(i=0;i<10;i++)
      {  a[i]=random(61)+40;            /*随机产生[40,100]的随机整数*/
         printf("%5d",a[i]);            /*排序前输出一次*/
      }
   printf("\n");
   for(i=0;i<9;i++)                     /*通过9轮比较排序*/
      for(j=i+1;j<10;j++)               /*遍历a[i]~a[9]找最小值，交换到a[i]中*/
         if(a[j]<a[i])                  /*如果a[j]比a[i]小*/
            {t=a[i]; a[i]=a[j]; a[j]=t;}        /*将a[j]和a[i]交换*/
   for(i=0;i<10;i++)                    /*输出排好序的数组*/
      printf("%5d",a[i]);
   printf("\n");
}
```

某一次运行结果:

```
  78   69  100   99   74   89   41   95   86   64
  41   64   69   74   78   86   89   95   99  100
```

【说明】 本例程序在 Turbo C 上运行。

例 5-8

【例 5-8】 将一维数组中的 n(n≤50)个数按逆序存放。

【分析】 当处理的数据个数 n 不确定时,可以先定义一个足够大的数组,再输入 n,通过 n 来控制输入 n 个数。数组逆序存放的方法可以采用前后对应元素对换的方法,例如,当 n=8 时,a[0]和 a[7]交换,a[1]和 a[6]交换,a[2]和 a[5]交换,a[3]和 a[4]交换。为了防止同一对数据交换两次(等于没交换),只需要遍历其中一半元素,即循环变量 i 从 0 到(n-1)/2 变化时,a[i]和 a[n-1-i]交换。

```
#include "stdio.h"
main()
{  int a[50],t,n,i;              /*最多有 50 个数,所以定义用 int a[50]*/
   scanf("%d",&n);               /*输入元素个数 n*/
   for(i=0;i<n;i++)              /*循环 n 次,遍历 a[0]~a[n-1]*/
       {  scanf("%d",&a[i]);     /*输入第 i 个数,放到 a[i]中*/
          printf("%5d",a[i]);    /*输出逆序前的每一个数*/
       }
   printf("\n");
   for(i=0;i<=(n-1)/2;i++)       /*遍历数组 a 的前一半元素,让前一半和后一半交换*/
      {  t=a[i];                 /*这 3 行完成 a[i]和 a[n-1-i]交换*/
         a[i]=a[n-1-i];
         a[n-1-i]=t;
      }
    for(i=0;i<n;i++)             /*输出逆序后的数*/
       printf("%5d",a[i]);
    printf("\n");
}
```

运行结果:

```
8↙
1 2 3 4 5 6 7 8↙
    1    2    3    4    5    6    7    8
    8    7    6    5    4    3    2    1
```

【说明】

(1) 第一个 for 语句既用于输入 n 个数,也做了逆序前的数据输出。如果 n 比较大时有些不方便,也容易使输入和输出数据交叉在一起,所以最好将这个 for 语句分成两个:一个用于输入 n 个数,另一个用于输出逆序前的 n 个数。

(2) n 是偶数时,数据正好成对地交换。n 是奇数时,本程序遍历的最后一对数是自己和自己交换。例如,n=7 时,a[0]和 a[6]交换,a[1]和 a[5]交换,a[2]和 a[4]交换,最