第 5 章　数　　组

在解决实际问题中，经常需要对批量数据进行处理，例如对一批数据进行求和、求平均值或排序等运算。

在 C 语言中，为了便于处理批量数据，将具有相同类型的若干变量有序地组织起来，这些被有序组织起来的同类型数据的集合称为数组。

5.1　数 组 引 例

【例 5-1】读入某位同学 8 门课的期末考试成绩，求这位同学期末考试每门课的平均分。

编程提示：根据前面章节学习的知识，用以下程序可以实现其功能。

```
#include <stdio.h>
int main()
{
      float score1,score2,score3,score4,score5, score6,score7,score8,sum;
      scanf("%f",&score1);
      scanf("%f",&score2);
      scanf("%f",&score3);
      …           /* 省略了 4 条语句，如果要运行程序，必须补充完整 */
      scanf("%f",&score8);
      sum= score1+score2+score3+score4+score5+score6+score7+score8;
      printf("The average score is %f",sum/8.0);
      return 0;
}
```

在上述程序中定义了 8 个整型变量并分别读入 8 门课的成绩，最后求平均值。若有 20 门课，难道要定义 20 个变量吗？显然不可取。可以发现这 8 个变量具有相同数据类型，如果引入数组知识，程序在处理这些批量数据时将更加灵活方便。程序改写如下：

```
#include <stdio.h>
int main()
{
      int i;
      float score[8],sum=0;
      for (i=0;i<8;i++)
      {
          scanf("%f",&score[i]);
          sum=score[i]+sum;
      }
```

```
    printf("The average score is %.1f",sum/8.0);
    return 0;
}
```

程序运行结果：

```
80 85 90 95 100 75 70 84 ↙
The average score is 84.9
```

【同步练习】 某件商品分 10 次进货，每次进货价格不同，读入 10 次进货价格，求该件商品的平均进货价格。

提示：定义长度为 10 的一维数组，依次读入每次进货的价格，求和后再求平均值。

5.2 一 维 数 组

数组是一组具有相同数据类型的有序的数据的集合。数组中各个数据的排列具有一定规律，下标代表数据在数组中的序号。一维数组是数组中最简单的一种，它的数组元素只有一个下标。

5.2.1 一维数组的定义

数组和变量一样，必须先定义，然后才能使用。

一维数组的一般形式如下：

```
类型名 数组名[常量表达式]
```

例如：

```
int a[5];
```

表示定义了一个整型数组，数组名为 a，定义的数组元素为 a[0]、a[1]、a[2]、a[3]和 a[4]。

```
char b[8];
```

表示定义了一个由 8 个元素组成的一维数组。数组名为 b，其元素均为字符型，分别为 b[0]、b[1]、b[2]、b[3]、b[4]、b[5]、b[6]和 b[7]。

```
float c[10];
```

表示定义了一个由 10 个元素组成的一维数组，数组名为 c，其元素均为浮点型，分别为 c[0]、c[1]、c[2]、c[3]、c[4]、c[5]、c[6]、c[7]、c[8]和 c[9]。

注意：

（1）“[ ]”不能错用为“()”。

（2）数组元素的下标从 0 开始。

（3）数组中存储的数据均为同一个数据类型。

（4）定义数组长度时，方括号中只能是常量或常量表达式。

例如：

```
int i,a[i];     /* 错误！定义数组长度时 i 为变量 */
int a[2+3];     /* 正确！ */
```

（5）一维数组中各元素按顺序连续存放，即相邻元素的地址相差 sizeof(int)个字节。

5.2.2 一维数组的引用

定义数组之后，就可以引用数组中的元素。

一维数组的引用形式如下：

```
数组名 [下标];
```

其中，下标可以是常量、变量或表达式。由于下标是 0～*n*–1 的数，因此引用数组元素时可以与循环语句结合，即通过循环变量改变下标的值。例如：

```
int n=1;
int a[3];
a[0]=1;
a[n]=2;
a[2]=5;
```

【例 5-2】 输入 10 个学生的成绩，将低于 60 分的成绩输出。

编程提示：定义长度为 10 的数组，依次将 10 个成绩赋值给数组元素。利用循环将各个元素与 60 分比较，小于 60 分的成绩输出。

```
#include <stdio.h>
int main()
{
    int score[10],i;
    for (i=0;i<10;i++)
       scanf("%d",&score[i]);
    printf("小于 60 分的成绩有：\n");
    for (i=0;i<10;i++)
       if(score[i]<60)
          printf("%3d",score[i]);
    return 0;
}
```

程序运行结果：

```
10 20 30 40 50 60 70 80 90 100↙
小于 60 分的成绩有：
10 20 30 40 50
```

【同步练习】 输入 10 个学生的成绩，求平均值。

提示：定义长度为 10 的数组，依次将 10 个成绩赋值给数组元素。利用循环将各个元素累加求和，计算平均值输出。

5.2.3 一维数组的初始化

所谓数组的初始化是指在定义数组时就对各个数组元素进行赋值。

（1）在定义数组时对全部元素进行初始化。

例如：

```
int a[6]={0,1,2,3,4,5};
```

利用花括号对各元素按序赋值，数据元素间用逗号分隔。等价于 a[0]=0, a[1]=1, a[2]=2, a[3]=3, a[4]=4, a[5]=5。

（2）对全部元素初始化时，可以不指定数组的长度。

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

等价于

```
int a[6]={0,1,2,3,4,5};
```

（3）对部分元素进行初始化。

例如：

```
int b[6]={0,1,2};
```

只对 b[0]、b[1]和 b[2]进行赋初值，未赋值的整型自动初始化为 0。等价于 b[0]=0, b[1]=1, b[2]=2, b[3]=0, b[4]=0,b[5]=0。字符型数据自动初始化为'\0'；浮点型数据自动初始化为 0.0。

5.2.4 一维数组的应用

【例 5-3】 将 10 个数组元素依次赋值为 0,1,2,3,4,5,6,7,8,9，并按逆序输出。

编程提示：首先定义一个长度为 10 的数组，利用循环语句逐个将数组元素赋值为 0~9。输出时，按逆序输出，即输出 a[9],a[8],…, a[0]。

```
#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]);
     printf("\n");
     return 0;
 }
```

运行结果：

```
9 8 7 6 5 4 3 2 1 0
```

【例 5-4】 输入 10 个整数，查找其中的最大值与最小值。

编程提示：定义长度为 10 的数组 a，将最大值 max 和最小值 min 都赋值为 a[0]，利用循环依次将各个数组元素与 max 和 min 比较，如果发现 a[*i*]大于 max，则令 max=a[*i*]；如发现 a[*i*]小于 min，则令 min=a[*i*]，循环结束后 max 与 min 中值即为最大值与最小值。

```
# include <stdio.h>
int main()
{
    int a[10], i, max, min;
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    max=a[0];
    min=a[0];
    for(i=1;i<10;i++)
    {
       if(max<a[i])
           max=a[i];
        if(min>a[i])
           min=a[i];
    }
    printf("最大值为%d\n",max);
    printf("最小值为%d\n",min);
    return 0;
}
```

程序运行结果：

```
11 22 33 44 55 66 77 88 99 100↙
最大值为 100
最小值为 11
```

【例 5-5】 利用一维数组输出斐波那契数列的前 10 项。

编程提示：斐波那契数列为 1,1,2,3,5,8,…

生成公式为

$$\begin{cases} f_1 = 1, & n = 1 \\ f_2 = 1, & n = 2 \\ f_n = f_{n-1} + f_{n-2}, & n \geqslant 3 \end{cases}$$

定义长度为 10 的数组 f，将 $f[0]=1$，$f[1]=1$，从第 3 项开始令 $f[i]=f[i-1]+f[i-2]$;

```
#include <stdio.h>
int main()
 {
     int i;
```

```
    int f[10]={1,1};
    for(i=2;i<20;i++)
        f[i]=f[i-1]+f[i-2];
    for(i=0;i<10;i++)
    {
        if(i%5==0) printf("\n");
        printf("%10d",f[i]);
    }
    printf("\n");
    return 0;
}
```

程序运行结果：

```
1    1    2    3    5
8   13   21   34   55
```

【例 5-6】 输入 10 个整数，分别统计奇数和偶数的个数。

编程提示：定义一维数组存储数据，利用 if 语句依次对输入的数据判断其奇偶性。

```
#include<stdio.h>
int main()
{
    int a[10],i,n=0,k=0;            /*数组上限定为 10 */
    printf("请输入数值:");
    for(i=0;i<10;i++)
       scanf("%d",&a[i]);
    printf("数组中的偶数为:");
    for(i=0;i<10;i++)
    {
       if(a[i]%2==0)
       {
           printf("%d ",a[i]);      /*输出偶数*/
           n++;                     /*统计偶数的个数*/
       }
    }
    printf("\n");
    printf("数组中的奇数为:");
    for(i=0;i<10;i++)
    {
      if(a[i]%2!=0)
      {
         printf("%d ",a[i]);        /*输出奇数*/
         k++;                       /*统计奇数的个数*/
       }
    }
```

```
    printf("\n");
    printf("数组中偶数的个数:%d",n);
    printf("数组中奇数的个数:%d",k);
    return 0;
}
```

程序运行结果：

```
请输入数值：1 2 3 4 5 6 7 8 9 10↙
数组中的偶数为：2 4 6 8 10
数组中的奇数为：1 3 5 7 9
数组中的偶数个数为：5
数组中的奇数个数为：5
```

【同步练习】

1．有一个数组 a[5]={0,1,2,3,4}，要求不借助其他数组，将其逆序为 a[5]={4,3,2,1,0}。

提示：定义长度为 5 的数组，将 a[0]与 a[4]交换值，a[1]与 a[3]交换值，a[2]不变。

2．在数组 a[10]中，查找变量 *m*，若找到输出对应的元素下标；若没有找到输出查无此数据。

提示：利用循环将 a[*i*]与 *m* 依次比较，若发现 a[*i*]与 *m* 相等，输出下标 *i*；若数组元素全部比较结束无相等出现，输出查无此数据。

5.3 二维数组及多维数组

一维数组只有一个下标，但在实际问题中有很多数据是二维的或多维的，因此 C 语言允许构造多维数组。多维数组元素有多个下标，以标识它在数组中的位置，所以也称为多下标变量。本节只介绍二维数组，多维数组可由二维数组类推而得到。

5.3.1 二维数组的定义

定义二维数组的一般形式如下：

```
类型名 数组名[常量表达式 1][常量表达式 2];
```

其中，常量表达式 1 表示第 1 维下标的长度，常量表达式 2 表示第 2 维下标的长度。

例如：

```
int a[3][4];
```

表示定义了一个 3 行 4 列的数组，数组名为 a，其元素共有 3×4 个，均为整型：

```
          第 0 列    第 1 列    第 2 列    第 3 列
第 0 行   a[0][0], a[0][1], a[0][2], a[0][3]
第 1 行   a[1][0], a[1][1], a[1][2], a[1][3]
第 2 行   a[2][0], a[2][1], a[2][2], a[2][3]
```

（1）二维数组可以看作是一个特殊的一维数组，即一维数组的每个元素又是一个一维数组。例如二维数组 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]
```

先将二维数组 a[3][4]看作一个由 3 个元素组成的一维数组，它的数组名为 a，元素分别为 a[0]、a[1]和 a[2]，因为这 3 个元素并不是真的一维数组的元素，所以它们的值当然也不是整型数据。实际上，这 3 个元素又分别是由 4 个元素组成的一维数组，因此 a[0]、a[1]和 a[2]分别代表这 3 个一维数组的数组名（即数组的起始地址）。也就是数组 a[0]的 4 个元素分别为 a[0][0]、a[0][1]、a[0][2]和 a[0][3]；数组 a[1]的 4 个元素分别为 a[1][0]、a[1][1]、a[1][2]和 a[1][3]；数组 a[2]的 4 个元素分别为 a[2][0]、a[2][1]、a[2][2]和 a[2][3]。

（2）二维数组以一维数组为基本单位，一维数组的各元素按顺序连续存放，因此二维数组中的各元素也按顺序连续存放。这样，在处理二维数组 a[3][4]时，也可以将它看作一个一维数组 a[3×4]来处理。

5.3.2 二维数组的引用

二维数组的元素表示的形式如下：

```
数组名[下标 1][下标 2]
```

其中，下标可以是常量、变量或表达式。这与一维数组元素在引用时对下标的要求相同。例如，a[3][4]表示数组中第 3 行第 4 列的元素，注意：下标 1 和下标 2 皆从 0 开始。

【例 5-7】 求一个二维数组 a[2][3]全部元素之和。

编程提示：利用两层 for 循环依次将数组元素读入 a，累加求和保存至 sum，输出 sum。

```
#include <stdio.h>
int main()
{
   int a[2][3],i,j,sum=0;
    for (i=0;i<2;i++)
    {
       for (j=0;j<3;j++)
       {
            scanf("%d",a[i][j]);
            sum=sum+a[i][j];
         }
         }
   printf("sum is %d",sum);
   return 0;
}
```

程序运行结果：

```
1 2 3↙
4 5 6↙
sum is 21
```

【同步练习】 求一个二维数组 a[2][3]全部元素的平均值。

提示：利用两层 for 循环依次将数组元素读入 a，累加求和保存至 sum，再求平均值。

5.3.3 二维数组的初始化

二维数组初始化可以通过以下方式：

（1）对全部元素赋初值，通过使用大括号分成若干行，赋予不同的行。例如：

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

赋值后第 1 对花括号代表第 1 行的数据，第 2 对花括号代表第 2 行数据，第 3 对花括号代表第 3 行数据。

也可按顺序赋值将所有的数据写在一对花括号内，例如：

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

如果对全部数组元素置初值，则第 1 维的长度可以省略。例如：

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

以上 3 种赋初值的结果完全相同。

（2）只对部分元素赋初值，未赋初值的元素系统自动把它们初始化为 0，过时字符型数据，则初始化为'\0'，弱势浮点型数据，则初始化为 0.0。例如：

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

是对每行的第 1 列元素赋值，未赋值的元素取 0 值。 赋值后各元素的值为

```
1  0  0
2  0  0
3  0  0
```

又如：

```
int a [3][3]={{0,1},{0,0,2},{3}};
```

赋值后的元素值为

```
0  1  0
0  0  2
3  0  0
```

【例 5-8】 将一个 3×4 的矩阵，按行列形式输出。

编程提示：读入时利用两层 for 循环依次为二维数组元素赋值，外层循环控制行下标为 0~2，内层循环控制列下标为 0~3。同理输出时利用两层 for 循环按 3 行 4 列的要求输出，

每输出一行后，利用'\n'换行。

程序代码：

```
#include <stdio.h>
int main()
{
   int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}},i,j;
   for (i=0;i<3;i++)
   {
      for (j=0;j<4;j++)
         printf("%5d",a[i][j]);
      printf("\n");
   }
   return 0;
}
```

程序运行结果：

```
1    2   3   4
5    6   7   8
9   10  11  12
```

【同步练习】 有一个二维数组 a[4][4]，编程将其每一行的第一个元素输出。

提示：每一行第一个元素的列下标为 0，利用循环将 a[*i*][0]依次输出。

5.3.4 二维数组的应用

【例 5-9】 将一个二维数组 a[2][3]={{1,2,3}，{4,5,6}}行和列的元素互换，保存在 b[3][2]中。

编程提示：创建一个数组 b[3][2]，将数组 a 中的行和列元素互换可以看作将数组元素行下标与列下标互换，即 b[*j*][*i*]=a[*i*][*j*]，然后利用两层 for 循环输出二维数组 b。

```
#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++)      /*按行列式格式输出数组 a，并把 a[i][j]赋值给 b[j][i]*/
   {
      for (j=0;j<=2;j++)
      {
         printf("%5d",a[i][j]);
         b[j][i]=a[i][j];     /*把 a[i][j]赋值给 b[j][i]*/
       }
       printf("\n");
```