第 5 章　数　　组

第 2 章介绍了 C 语言的简单数据类型，其特点是值域中的值都是单值。本章介绍 C 语言中的一种构造数据类型——数组。数组是 C 语言中常用的一种数据组织形式，它是变量的一个有序集合，其中所有变量具有相同的数据类型。同一数组中的数据元素按顺序占有一块连续的存储空间，数组中的首元素存放在该空间的最低地址，后续元素依次存放，最后一个元素存放在该空间的最高地址，因而各数组元素在该空间的存储位置是相对固定的。本章主要介绍一维数组、二维数组和字符数组的定义、访问及初始化，并通过实例对数组的应用进行介绍。

5.1　一维数组

一维数组是数组中最简单的，这里讲的一维是指元素的下标只有一个。以此类推，可以得到二维数组、三维数组、……、n 维数组的定义。一维数组中的元素可以表示为数组名加一个下标，例如，如果定义数组 x 表示计算机学院 12 级的全部班级，则计算机学院 12 级 1 班可以表示为 x_1。类似地，二维数组中的元素可以表示为数组名加两个下标，如计算机学院 12 级 1 班学号为 2 的学生可以表示为 $x_{1,2}$，其他维的数组元素可以类似表示。

5.1.1　一维数组的定义

与其他数据类型类似，要想在计算机中使用一维数组，必须首先对一维数组进行定义。定义一维数组时，需要说明两点：①数组中元素的类型；②数组中元素的个数。编译系统会根据这两点向内存申请数组的存储空间。例如，当数据类型为 int 类型、数组元素个数为 10 时，系统将申请 $4\times10=40$ 字节的存储空间。

定义一维数组的一般格式如下：

```
类型标识符 数组名[整型常量表达式];
```

例如：

```
int x[10];              //定义一个包含 10 个整数的数组 x
char name[20];          //定义一个包含 20 个字符的数组 name
double score[20];       //定义一个包含 20 个浮点数的数组 score
```

与变量名一样，数组名是用户定义的标识符，命名规则也相同。与变量不同的是，数组表示的不是一个元素，而是具有相同数据类型的一组元素，同时，数组名是数组存储空间的首地址，也是第一个元素的存储地址，其他元素的地址可以通过数组名和元素的下标来获得。这里需要说明的是，数组中元素的下标是从 0 开始的，因此，定义数组 x[10]后，系统将在内存中分配如下的存储空间：

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

在分配的存储空间中，数组的第 1 个元素 x[0]存放在存储空间的最低地址，第 10 个元

素 x[9]存放在存储空间的最高地址。

需要说明的是，在定义一维数组时，元素的个数[①]必须是整数常量或常量表达式，不能包含任何变量。如下定义的数组是不合法[②]的：

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

但 C 语言允许借助符号常量对数组加以定义，如：

```
#define N 10
int a[N];
```

5.1.2 一维数组的访问

对数组进行定义后，可以通过数组名和元素的下标对数组中的元素进行访问，形式如下：

```
数组名[下标]
```

例如，在定义了数组 x[10]后，它的 10 个元素可以通过如下的方式来访问：

```
x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7], x[8], x[9]
```

在这里，数组的下标从 0 开始，这是在编写程序时需要注意的地方。

例 5-1 平均成绩。

【问题描述】

从键盘输入 5 位同学的成绩(整数)，输出它们的平均分，保留小数点后一位。

【输入格式】

一行，5 位同学的成绩。

【输出格式】

一个数，表示 5 位同学的平均成绩，保留到小数点后一位。

【问题分析】

本题旨在考查数组元素的基本操作。本题可以采用循环结构来处理，但这样处理只保留了最后一名同学的成绩，如果后续有对成绩的其他处理要求，无法进一步处理，这里采用数组实现。

题目要求输入 5 名同学的成绩，以整数存储，因此，只需要定义数组 score[5]即可，利用循环变量 i 作为数组的下标进行成绩的读取和累加(可以在读取时进行，也可以单独处理)。最后除以 5 即可。

【参考代码】

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
```

① 需要说明的是，如果数组定义在 main()函数中，属于局部变量，数组是由栈(stack)存储的；如果数组在 main()函数外定义，属于全局变量，则存储在静态存储区中。通常情况下，全局变量存储时可以定义的数组元素个数远多于局部变量存储时定义的数据元素个数，具体要根据操作系统分配的空间来确定。

② 在第 7 章的指针部分将采用动态申请空间的方式处理这种问题。

```
3.   int main() {
4.       int score[5],sum=0;
5.       int i;
6.       for(i=0; i<5; i++) {
7.           cin>>score[i];
8.           sum+=score[i];                          //在读入时直接处理
9.       }
10.      printf("%.1lf",sum/5.);
11.      return 0;
12.  }
```

【代码分析】

上述程序有两点需要注意，一是平均成绩的计算，由于要求录入的成绩是以整数存储，因此计算平均分数时，需要注意整数的除法，在第 10 行，用 sum/5.有效解决这一问题；第二个问题是确定数组的大小，本题要求对 5 个同学的成绩进行处理，如果同学的数量未知，又应该如何处理呢？通常预先定义一个较大的数组进行处理。当然在设置数组时，最好使数组中元素的个数与实际输入的个数大致相同，否则会造成存储空间的极大浪费。

从上述例子可以看出，数组中的元素类型都是一致的，它们存放在内存空间中连续的区域中。在对数组元素进行访问时，可以通过改变下标来访问不同的数组元素，因而对数组元素进行逐一访问时，通常借助循环来完成。

需要特别注意的是，编译器在编译时不会检查数组元素的下标是否越界，当下标越界时不会出现编译错误，但程序在运行时会出现问题。例如，对数组 a[10]来说，如果在程序中访问元素 a[10]，不会出现编译错误，但在程序运行时可能会出现运行错误。

5.1.3 一维数组的初始化

数组的初始化是指定义数组时对数组中的元素赋初值。一般地，对数组进行初始化有以下 5 种方式。

（1）对数组的全部元素进行赋值，例如：

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

（2）对数组中的部分元素进行赋值。此时将根据提供的值的个数为数组中前面的元素进行赋值，对于数组中的其他元素赋缺省值，例如：

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

等价于

```
int a[10]={1,2,3,4,5,0,0,0,0,0};
```

说明：当对部分元素初始化时，凡未被初始化列表指定初始化的数组元素，对于数值型数组，系统会自动把它们初始化为 0，如上例；对于字符数组，则自动初始化为'\0'；对于指针型数组，则自动初始化为 NULL，即空指针。

（3）如果要对一个数组中的全部元素赋值为 0，可以按如下方式赋初值：

```
int a[10]={0};
```

这种赋值方式等价于

```
int a[10]={0,0,0,0,0,0,0,0,0,0};
```

（4）如果对数组中所有元素赋初值，可以不指定数组的长度，此时初值的个数即为数组的长度，例如：

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

等价于

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

注意：如果既指定了数组的长度，同时对数组元素进行初始化，则初始值的个数必须小于数组的长度，否则编译时会报错，例如，以下初始化语句是错误的：

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

与变量不同的是，数组不能进行整体赋值，如果要把一个数组的值赋给另一个数组，需要逐个元素进行赋值。例如，定义如下两个数组：

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

进行赋值时，不能利用如下的语句进行：

```
b=a;
```

而应按照如下的语句进行：

```
int i;
for(i=0;i<10;i++)
    b[i]=a[i];
```

（5）在头文件 string.h 中提供了 memset()函数对数组进行整体赋值，其基本用法是

```
memset(void *buffer, int c, int count);
```

其中，buffer 为数组名或指针，c 为预设的值，count 是 buffer 的长度。

具体地，memset()函数是将指定的值 c 复制到 buffer 所指向的内存区域的前 count 字节中，这可以用于将内存块清零或设置为特定值按字节填充，例如：

```
int a[10];
memset(a,0,sizeof(a));          //0 的补码为 0x00,因此 a 的所有元素预设为 0x00000000,即
                                //十进制的 0
memset(a,-1,sizeof(a));         //-1 的补码为 0x11,因此 a 的所有元素预设为 0x11111111,
                                //即十进制的-1
memset(a,1,sizeof(a));          //1 的补码为 0x01,因此 a 的所有元素预设为 0x01010101,即
                                //十进制的 16843009
```

5.1.4 一维数组的应用

例 5-2 斐波那契数列。

【问题描述】

编写程序计算斐波那契数列的第 n 项($n<40$)。其中斐波那契数列 $f(n)$定义为

$$f(n)=\begin{cases}1, & n=1,2\\ f(n-1)+f(n-2), & n\geqslant 3\end{cases}$$

【输入格式】

整数 n($0<n<40$)。

【输出格式】

第 n 项的值。

【问题分析】

本题旨在考查运用数组计算斐波那契数列。在第 4 章中介绍循环时曾介绍过求斐波那契数列的方法,但求得的斐波那契数列的每一项无法保存。当输入多组数据时,可能会存在算法效率低下的问题。基于一维数组计算斐波那契数列,从算法的角度上属于用空间换时间,通过分析数列中的项之间的关系,可以提前计算出数列中的项,当多组数据输入时,直接查表输出相应的结果即可。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    long long a[41]={0,1,1};
    int n;
    for(int i=2;i<41;i++)
        a[i]=a[i-1]+a[i-2];
    cin>>n;
    cout<<a[n];
    return 0;
}
```

例 5-3 查找整数。

【问题描述】

给出一个包含 n 个整数的数列,整数 a 在数列中的第一次出现是第几个?

【输入格式】

第一行包含一个整数 n(n<10000);第二行包含 n 个非负整数,为给定的数列,数列中的每个数都不大于 10000;第三行包含一个整数 a,为待查找的数。

【输出格式】

如果 a 在数列中出现了,输出它第一次出现的位置(位置从 1 开始编号),否则输出 -1。

【问题分析】

本题旨在考查数组元素的遍历和查找。查找是程序设计中的基本问题之一,其基本思想是从第一个元素开始查找,判断它与待查找的数字是否相等。本题要求只输出第一次出现的位置,因此当找到第一个时,直接中断即可。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
```

```
3.  int main() {
4.      int arr[10001], a;
5.      int n,i;
6.      cin>>n;
7.      for(i=0; i<n; i++)
8.          cin>>arr[i];
9.      cin>>a;
10.     for(i=0; i<n; i++) {
11.         if(arr[i]==a)
12.             break;              //找到第一个直接中断
13.     }
14.     if(i<n)                     //在数组 arr 中查找到元素 a,中断循环,因此 i<n
15.         cout<<i+1;
16.     else                        //没有找到
17.         cout<<-1;
18.     return 0;
19. }
```

【代码分析】

对于是否查找到元素的判断，通常有两种方式，一是题目中的方式，二是通过设置标志位 flag。在第二种方式中，预设 flag＝0，在第 10 行的循环条件中，修正为“!flag && i<n”，第 12 行中的 break 语句可以用 flag＝1 替代，也可以达到只查找第一个元素的目的。

【问题扩展】

(1) 如果要找到所有等于待查找元素的位置，该如何处理？

(2) 如果给定的数组有序，又该如何处理？

例 5-4　折半查找法。


例题 5-4

【问题描述】

在一排按编号由小到大排序的数中，快速地查找到某个给定的数字。

【输入格式】

第一行是 n(n<10000)，表示有 n 个元素，第二行是 n 个数，第三行是 m，表示要查找的数。

【输出格式】

一个数，若找到该数，则输出所在位置，否则输出－1。

【问题分析】

本题旨在考查利用数组元素的有序性提高元素查找的效率。

与例 5-3 相比，本题的特点是给定的元素是有序的，虽然可以采用例 5-3 中的思路进行顺序查找，但这样做并没有有效利用元素的有序性。考虑到给定的 n 个元素已经由小到大排好顺序，因此可以首先取中间元素与待查找的元素进行比较，存在以下 3 种情况。

(1) 中间元素与待查找元素相等，返回其所在位置即可。

(2) 中间元素小于待查找元素，如果元素存在，则必存在于右半侧。

(3) 中间元素大于待查找元素，如果元素存在，则必存在于左半侧。

上述思路每次都可以将查找范围缩小一半，因此又称为**二分查找法**。

【参考代码】

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  int main() {
```

```
    int a[10001],m;
    int n,left,right,mid,i;
    cin>>n;
    for(i=0;i<n;i++)
        cin>>a[i];
    cin>>m;
    left=0, right=n-1;
    while(left<=right){
        mid=(left+right)/2;
        if(a[mid]==m)          //中间元素与待查找元素相等
            break;
        if(a[mid]<m)           //中间元素小于待查找元素,如果存在,必位于右半侧
            left=mid+1;
        else                   //中间元素大于待查找元素,如果存在,必位于左半侧
            right=mid-1;
    }
    if(left>right)             //没有找到
        cout<<-1;
    else
        cout<<mid+1;
    return 0;
}
```

【代码分析】

折半查找法是一种高效的数据查找方式,但它要求待查找的元素是有序的,并且查找的一组数据要采用一维数组存储。如果数组中有 n 个待查找的元素,则程序最多需要$\lceil \log_2 n \rceil$次查找。

例 5-5 排序。


例题 5-5

【问题描述】

给定 n(n≤100)个整数,按照从小到大的顺序排序后输出。

【输入格式】

第一行是一个正整数 n,第二行有 n 个整数。

【输出格式】

从小到大输出这 n 个数,中间用空格隔开。

【问题分析】

本题旨在考查利用数组实现数据的排序。排序问题是计算机科学中的经典问题之一,这里介绍两种基本的思路。

(1) 每次将小的元素往前放,最小的元素放在第一个位置,第二小的元素放在第二个位置,……,第 n−1 小的元素放在第 n−1 个位置。具体步骤如下。

第 1 步:将第一个元素与后面的每一个元素进行比较,如果第一个元素比后面的元素大,则将两个元素进行交换。

第 2 步:将第二个元素与后面的每一个元素进行比较,如果第二个元素比后面的元素大,则将两个元素进行交换。

……

第 n−1 步:将第 n−1 个元素与最后一个元素进行比较,如果第 n−1 个元素比最后一个元素大,则将两个元素交换。

这种方法每次将最小的元素尽可能地放在前面，如同碳酸饮料中二氧化碳的气泡最终会上浮到顶端一样，称为**冒泡排序（bubble sort）**。

（2）每次将大的元素往后放，即将最大的元素放在最后一个，第二大的元素放在倒数第二个位置，……，第 n－1 大的元素放在第 2 个位置。这种操作可以采取类似上述思路进行，也可以采取将元素与相邻元素进行比较的方式进行，具体步骤如下。

第 1 步：从第一个元素开始，将该元素与它的相邻元素进行比较，如果前面的元素比后面相邻的元素大，则将二者进行交换，直至第 n－1 个元素。

第 2 步：继续从第一个元素开始，将该元素与它的相邻元素进行比较，如果前面的元素比后面相邻的元素大，则将二者进行交换，直至第 n－2 个元素。

……

第 n－1 步：从第一个元素开始，将该元素与它相邻的元素进行比较，如果前面的元素比后面相邻元素大，则将二者进行交换，直至第 1 个元素。

这种排序的方法每一步都将大的元素尽可能往后放，如同重的石头沉到池塘底部，称为**沉底排序（bottom sort）**。

【参考代码 1】 冒泡排序。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    long long a[101],temp;
    int n,i,j;
    cin>>n;
    for(i=0;i<n;i++)
        cin>>a[i];
    for(i=0;i<n;i++){
        for(j=i+1;j<n;j++){
            if(a[i]>a[j]){
                temp=a[i]; a[i]=a[j]; a[j]=temp;      //swap(a[i],a[j]);
            }
        }
    }
    for(i=0;i<n;i++)
        cout<<a[i]<<" ";
    return 0;
}
```

【参考代码 2】 沉底排序。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    long long a[101],temp;
    int n,i,j;
    cin>>n;
    for(i=0;i<n;i++)
        cin>>a[i];
    for(i=0;i<n-1;i++){
        for(j=0;j<n-1-i;j++){
            if(a[j]>a[j+1]){
                temp=a[j]; a[j]=a[j+1];a[j+1]=temp;  //swap(a[j],a[j+1]);
```

```
            }
        }
    }
    for(i=0;i<n;i++)
        cout<<a[i]<<" ";
    return 0;
}
```

【代码分析】

上述程序的主要工作是进行元素的交换。在最坏情况下，每一次都需要将两个元素进行交换。

查找最小元素时，需要将第 1 个元素与后续的 $n-1$ 个元素进行交换，赋值的次数为 $3(n-1)$。

查找第 2 小元素时，赋值的次数为 $3(n-2)$。

……

查找第 $n-1$ 小元素时，赋值的次数为 3。

因此，上述排序算法在最坏情况下的时间复杂度为

$$3(n-1)+3(n-2)+\cdots+3=\frac{3n(n-1)}{2}=O(n^2)$$

例 5-6 进制转换。

【问题描述】

输入一个十进制正整数 m，将该数转换成 n 进制数(n≤16)，大于或等于 10 的数依次用 A 代替。

【输入格式】

正整数 m 和 n。

【输出格式】

将 m 转换为 n 进制的结果。

【问题分析】

本题旨在考查利用数组进行逆序变换，后续学习中，还可以运用递归函数、栈等方式处理。进行进制转换时，首先求出的是最低位，为了从高位到低位输出，这里借助数组进行逆序。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[100];
    int m,n,i,index=0;
    cin>>m>>n;
    while(m){
        a[index]=m%n;
        m=m/n;
        index++;
    }
    for(i=index-1;i>=0;i--){
        if(a[i]<10)
```

```
14.             cout<<a[i];
15.         else                    //将大于或等于10的数字按大写字母A～F输出
16.             cout<<(char)((a[i]-10)+'A');
17.     }
18.     return 0;
19. }
```


例题 5-7

例 5-7 统计难题。

【问题描述】

给定 n 个数字(n＜10^5),统计个位分别是 0,1, 2, 3, 4, 5, 6, 7, 8, 9 的数字个数。

【输入格式】

第一行一个数字 n,第二行 n 个数字。

【输出格式】

输出 10 行,分别代表个位数为 0 1 2 3 4 5 6 7 8 9 的数字的个数。

【问题分析】

本题旨在考查对数组的灵活应用,特别是下标表示的物理意义。本题要求计算个位数是 0～9 的数字个数,可以将个位数直接作为数组下标来处理。

【参考代码】

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  int main(){
4.      int n,i,num;
5.      int a[10]={0};
6.      cin>>n;
7.      for(i=0;i<n;i++){
8.          cin>>num;
9.          a[num%10]++;                    //个位数为num%10,直接作为下标处理
10.     }
11.     for(i=0;i<10;i++)
12.         cout<<a[i]<<endl;
13.     return 0;
14. }
```


例题 5-8

例 5-8 高精度阶乘。

【问题描述】

输入一个数 n(n＜10000),求 n 的阶乘 n!。例如,当 n＝5 时,n!＝5×4×3×2×1＝120。

【输入格式】

一个数 n。

【输出格式】

n 的阶乘 n!。

【问题分析】

本题旨在考查基于数组实现高精度运算。当数字很大时,现有的数据类型无法存储较大的数据,在这种情况下,通常采用数组存储计算的结果。此时用数组的每一位表示计算结果的每一位,因此需要根据给定的数据预判数据的大小。以本题为例,计算 10000!时,如果