第5章 数 组

教学目标

(1) 掌握数组数据结构。

(2) 掌握数组的声明和存放,初始化和数组元素的引用方法。

(3) 掌握数组下标的使用方法。

(4) 了解多维数组声明和操作。

(5) 掌握字符串定义及使用方法。

(6) 初步理解排序和查找等基本算法。

数组是一组类型相同的数据对象构成的集合。数组中的数据存储在一个区域内,引用时用同一名字,而且能够作为一个组或者一个整体访问,从而使得程序的书写变得更简单。数组将数据对象组织成连续的存储单元,最低地址对应于数组的第一个元素,最高地址对应于最后一个元素,每一个数组元素都占用一个或若干连续的存储单元。数组可以是一维的,也可以是多维的,C 语言没有对数组的维数做严格的限制。

数据结构(data structure):在同一个名称下存储的相关的数据项的集合。

数组(array):类型相同的数据的集合。

5.1 问 题 引 导

在第 3 章例 3.28 学生成绩统计中,只是求出了课程考试的最高分、最低分和平均分等信息,如果要把每个学生的成绩信息、排名以及课程的统计信息以短信的方式发送给学生家长,程序又该如何设计呢? 一般来说应进行如下操作。

(1) 保存每个学生的考试成绩。

(2) 比较学生的成绩,找出最高分和最低分。

(3) 将所有学生的成绩累加除以学生人数,计算出课程平均分。

(4) 对学生成绩进行排名,还可以知道学生的名次。

(5) 可以输入学生的学号,查询学生成绩。

通过数组的学习和使用,能够得到合适的存储方式和计算方法实现问题的求解过程。

5.2 一 维 数 组

数组是存放两个或两个以上相邻存储单元的集合,每个存储单元中存放相同数据类型

的数据。这样的单元称为数组元素。在一个具有相同名称和相同类型的连续存储结构中，要引用某个元素，就要指定数组中的元素的位置号或索引(index)。

直观上，我们将数组看作一个矩形框，数组中的每一个元素在这个矩形框中都占有一个方格。图 5-1 描述了一个由 5 个元素组成的双精度浮点型数组。

图 5-1　5 个元素的双精度浮点型数组 x

在 C 语言中，每个数组有以下两个特性。

(1) 数组元素的类型：存储在数组元素中的值的数据类型，简称数组类型。

(2) 数组的大小：数组中能够存放数组元素的个数，也称为数组长度。

在声明数组时，必须反映出数组的这两个特性。

5.2.1　一维数组的定义

一维数组的声明形式如下：

```
数组元素的类型说明符 数组名称[数组的大小];
```

在 C 语言中，要使用数组，必须事先声明，以便编译器为它们分配内存空间。在上面的声明中，类型说明符指明数组的类型，也就是数组中每一个元素的数据类型，数组的大小表示数据元素的个数。每个数组元素都有一个指定字节数的存储单元。一维数组的总字节数可按下式计算：

```
总字节数 = sizeof(数组元素的数据类型) * 数组的大小
```

其中，sizeof 是计算数据类型在存储空间中占用字节数的运算符。图 5-1 中的数组 x 所占用的存储字节数为：

```
sizeof( double ) * 5
```

如果 double 类型占用 8 字节存储，则表达式的值为 40。

例 5.1　数组的声明语句及含义。

数组的声明语句及含义如表 5-1 所示。

表 5-1　数组的声明语句及含义

语　　句	说　　明
int a[10];	定义一个长度为 10 的 int 数组 a
double score[50];	定义一个长度为 50 的 double 数组 score
char studentname[20];	定义一个长度为 20 的 char 数组 studentname
int b[100],x[27];	定义了一个长度为 100 的 int 数组 b 和一个长度为 27 的 int 数组 x

由于 C 语言没有字符串类型，则 char 类型的数组可以存放字符串。字符串及字符数组将在字符串一节中详细讨论。

例 5.2 将学生课程考试成绩数据存入数组相应的数组元素中。

```
/*
  程序名称:ex5_2.c
  程序功能:声明数组,从键盘输入数据存储到数组元素中并输出
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a[10];
    int i;
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    for(i=0; i<10;i++)
        printf("%d",a[i]);
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入: 80 70 95 54 76 90 85 88 77 69
输出: 80 70 95 54 76 90 85 88 77 69
```

说明:

(1) 数组的大小(或长度)是一个整型常量(高版本编译器支持变量,本教材按常量处理),通常可以使用符号常量来表示,这样可以方便改变数组的大小。例如:

```
int a[10];
```

使用符号常量的形式定义数组的大小:

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

(2) 数组的命名方式与其他变量一样,遵循 C 语言标识符的命名规则。

(3) 数组元素的起始下标为 0,在图 5-1 中,x[1]是数组的第二个元素,x[4]是数组的最后一个元素。数组的所有单元使用同一个名称 x,每个元素使用下标来区分。在使用数组元素时,必须要注意下标的使用范围。如果下标错误,会产生一个非法的引用。编译器在语法检查时,并不指出下标的越界错误。尽管有时候会输出一个运行错误信息,但大多数时候并不给出错误提示信息。非法下标引发的不正确结果,编程者有时很难查明原因。

数组在存储时要占用内存空间。在数组声明时必须指定元素的类型和元素的个数,这样编译器才可以为数组分配相应的内存空间。

例如,前面定义的 double 型数组 x,其在内存中的存储如图 5-2 所示。

若有数组的定义为:

```
int a[10];
```

其存储形式如图 5-3 所示。

索引号	存储地址	元素值	数组元素
0	2000	16.0	x[0]
1	2008	2.0	x[1]
2	2016	18.0	x[2]
3	2024	22.0	x[3]
4	2032	12.0	x[4]

图 5-2　double 型数组 x 的存储

索引号	存储地址	元素值	数组元素
0	2000	1	a[0]
1	2004	2	a[1]
2	2008	3	a[2]
3	2012	4	a[3]
4	2016	5	a[4]
5	2020	6	a[5]
6	2024	7	a[6]
7	2028	8	a[7]
8	2032	9	a[8]
9	2036	10	a[9]

图 5-3　int 型数组 a 的存储

定义字符型数组 char str[5],其存储结构如图 5-4 所示。

索引号	存储地址	元素值	数组元素
0	2000	'H'	str[0]
1	2001	'e'	str[1]
2	2002	'l'	str[2]
3	2003	'l'	str[3]
4	2004	'o'	str[4]

图 5-4　字符型数组 str 的存储

5.2.2　一维数组元素的引用

为了处理存储在数组中的数据,一般使用数组名和一个表示下标的整数来引用每一个元素。例如,数组 x 的第三个元素表示为 x[2],称 x[2]为下标变量,方括号内的整数是数组的下标(或者称为索引),它的值必须在 0 到数组元素个数(数组长度)减 1 之间。

图 5-5 所示为一个拥有 8 个元素的整型数组 c。数组中的第一个元素称为第 0 个元素(读作 c 零),记为 c[0];c 数组中的第二个元素为 c[1];c 数组中的最后一个元素为 c[7]。

图 5-5　8 个元素的数组 c

数组的下标表示数组元素的位置,下标必须为整型值。例如,假设 a 等于 2,b 等于 3,则下列语句:

```
c[a+b] = 0;
```

含义为将 0 存入数组元素 c[5]中。

表 5-2 列出了一些常用的数组元素的使用方法。

表 5-2　数组元素的引用

语　　句	说　　明
printf("%.2lf",x[3]);	显示 x[3]的值,其值为 22.00
x[0] = 25.0;	将浮点数 25.0 存入 x[0]中
sum = c[0] + c[1];	将 c[0]和 c[1]的和存储在变量 sum 中
sum += c[2];	将 sum 和 c[2]的和存储在变量 sum 中
c[7] = 100;	将 100 的值存入 c[7]元素中
x[2] + 1 = x[3];	非法的赋值语句,赋值运算符的左边不是变量不能被赋值

一般从数组的第一个元素开始,顺序处理数组的元素。在 C 语言中,使用循环语句可以很容易完成对数组元素的顺序操作。例如,用 for 循环语句,使用计数器作为数组的下标,就可以依次存取每一个数组元素。

例 5.3　保存一个 10 以内整数的所有约数。

分析:一个数的约数的计算过程,是一个不断尝试的过程,约数是从 1 开始的,使用循环变量来表示约数不断增加的过程,若输入的数据 n 能被某一个循环变量整除,则该循环变量即为 n 的一个约数,并将其存入数组中。

```
/*
  程序名称:ex5-3.c
  程序功能:保存一个 10 以内的整数的所有约数
*/
    #include<stdio.h>
    #include<stdlib.h>
    int main()
    {
        int a[5],n;
        int i,j=0;
        printf("输入一个整数(<10): ");
        scanf("%d",&n);
        for(i=1; i<=n;i++)
            if (n%i==0)                 /*计算数据 n 的约数*/
                a[j++]=i;               /*使用循环语句为数组元素赋值*/
        printf( "约数为:\n" );
        for(i=0; i<j; i++)              /*使用循环访问数组*/
            printf( " %6d",a[i] );
        system("pause");
        return 0;
    }
```

程序运行结果:

```
输入:输入一个整数(<10): 8
输出:约数为:
1    2    4    8
```

5.2.3　一维数组的初始化

在声明简单变量时,可以进行初始化:

```
int i = 0;
```

C 语言也允许在声明数组时直接初始化。各元素的值(也可以是表达式)顺序排在一对花括号里,并用逗号分隔,填充到数组元素所在的存储空间中。一般来说有以下几种初始化方法。

1. 定义数组时初始化

例如:

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

其中,数组 b 的各个元素值如图 5-6 所示。

b[0]	b[1]	b[2]	b[3]
1	2	3	4

图 5-6 数组 b 初始化后的元素值

2. 部分初始化

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

定义了一个数组长度为 10 的整型数组 a,其中元素 a[0]~a[4]的值分别是 1,2,3,4,5。而元素 a[5]~a[9]的值都是 0,如图 5-7 所示。

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

图 5-7 数组 a 部分初始化后的元素值

```
char letters[26] = { 'A','B'};
```

语句定义了一个长度为 26 的字符数组 letters,图 5-8 所示为在初始化时只为数组中前两个元素赋了初值的情形,其余的元素值为 0,即'\0'字符。

letters[0]	letters[1]	letters[2]	……	letters[25]
'A'	'B'	'\0'	……	'\0'

图 5-8 字符数组 letters 部分初始化后的元素值

3. 根据初值确定数组长度

```
int d[] = { 0,1,2,3,4,5,6,7,8,9 };
```

对于数组 d,定义时并没有指定数组的长度,但系统会根据初始化数组元素的个数自动计算数组的大小,所以数组 d 的长度是 10。

在定义数组 d 时,由于初值个数比较少,很容易得出数组的长度(元素个数)。如果数组元素更多些,一一计数可能比较麻烦。在 C 语言中,可以采用下面的计算方式确定数组的元素个数,如果数组名为 d,则数组的大小可用如下表达式计算:

```
sizeof(d)/sizeof(d[0])
```

上述表达式的含义是,数组 d 的存储空间的大小除以数组中第一个元素占用存储空间的大小。对于数组来说,其每个元素的数据类型都是相同的,所以计算出来的值就是数组的大小。

应特别指出,这种为数组元素指定值的写法只能用在定义数组时,在语句里不能采用这种写法。例如:

```
int d[10];
d={ 0,1,2,3,4,5,6,7,8,9 };              /*错误的赋值*/
```


例5.4 将学生的成绩信息保存在数组中并求平均分。

分析：用数组a存放最多30名学生的某门课程的考试成绩。为了方便起见，将学生的学号表示为1～30的整数，需要定义长度为31的数组a，即int a[31]。当读入一个有效的成绩(成绩一般大于或等于0，小于或等于100)时，将其保存在数组a中；然后将合法的成绩(不合法的成绩要求重新输入)累加，再除以人数即得平均成绩。

```
/*
  程序名称:ex5-4.c
  程序功能:学生成绩的平均分
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a[31],i;
    double avg=0;
    for(i=1;i<=30;i++)
    {
        do
        {
            printf("第%d个学生的成绩:",i);
            scanf("%d",&a[i]);
        }while(a[i]<0||a[i]>100);
        avg=avg+a[i];
    }
    printf("学生成绩:\n");
    for(i=1;i<=30;i++)
        printf("%d ",a[i]);
    printf("\n平均成绩:\n");
    printf("%.2lf\n",avg);
    system("pause");
    return 0;
}
```

说明：

(1) 如果初始化的元素个数小于数组的长度，则其余元素初始化为0值。

例如，可以用下列声明数组a的元素初始化为0：

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

其显式地将第一个元素初始化为0，隐式地将其余元素自动初始化为0，因为初始化值比数组中的元素少。程序员至少要显式地将第一个元素初始化为0，才能将其余元素自动初始化为0。

(2) 初始化值超过数组元素个数会产生数组定义的错误。例如：

```
int a[5] = {32,27,64,18,95,14};
```

因为初始化了 6 个值，而数组只能存放 5 个元素。

(3) 若数组中的数据已知，使用初始化能提高程序的执行速度。

(4) 当数组声明时，有 static 关键字修饰时，即使没有初始化，系统也能自动为所有元素初始化为 0 值。

```
static int b[5];
```

5.2.4 利用一维数组解决问题

例 5.5 在 N 个学生成绩中找最高分和最低分。

分析：寻找最大值和最小值，需要比较数组中的所有元素，才能找出最值。

```
/*
  程序名称:ex5-5.c
  程序功能:求最值
*/
#include<stdio.h>
#include<stdlib.h>
#define N 10
int main()
{
    int x[N],i,max,min;
    printf("输入 N 个整数:\n");
    for(i=0;i<N;i++)
        scanf("%d",&x[i]);
    max=min=x[0];          /*在输入的数据中确定最大、最小值的初值,一般使用第 1 个元素*/
    for(i=1;i<N;i++)
    {
        if(max<x[i]) max=x[i];
        if(min>x[i]) min=x[i];
    }
    printf("最高分是:%d\n",max);
    printf("最低分是:%d\n",min);
    system("pause");
    return 0;
}
```

例 5.6 已知数组 a 中有 n 个互不相等的元素，数组 b 中有 $m(m<n)$ 个互不相等的元素，而数组 c 中包含那些在 a 中但不在 b 中的元素，编程产生数组 c(产生新数组)。

分析：先用数组 a 中的第一个元素和数组 b 中的元素进行比较，如果和 b 中的元素不相同就存入数组 c 中；然后再用 a 中的其他元素和 b 中的元素进行比较，如果不同即放入数组 c；直到数组 a 中的所有元素都比较结束。

```
/*
  程序名称:ex5-6.c
  程序功能:挑选数据
*/
#include<stdio.h>
#include<stdlib.h>
```

```
#define N 8
#define M 5
int main()
{
    int i,j,k = 0,a[N],b[M],c[N];
    for(i = 0;i < N;i++)
        scanf("%d",&a[i]);
    for(i = 0;i < M;i++)
        scanf("%d",&b[i]);
    for(i = 0;i < N;i++)
    {
        for(j = 0;j < M;j++)
            if(a[i] == b[j]) break;
        if(j >= M) { c[k] = a[i];k++;}
    }
    for(i = 0;i < k;i++)
        printf("%5d",c[i]);
    printf("\n");
    system("pause");
    return 0;
}
```

思考：要求都在两个数组中的数应该如何处理？

例 5.7 对学生成绩按升序(从小到大)输出。

分析：在第 3 章我们对 3 个整数进行从小到大排序，但当数据很多时，这种方法就不适应了，因此必须使用新的排序算法来解决，这里用冒泡排序来实现。冒泡排序的基本思想是两个相邻的数依次比较，将较大的数放在后面。具体做法是：将第一个数和第二个数进行比较，如果前面大，后面小，则交换；然后第二个数和第三个数比较，将较大的数放到后面，直到第 $n-1$ 个数和第 n 个数进行比较，这样就完成了一趟冒泡。这时，最大的数就移到了最后。然后对前 $n-1$ 个数进行第二趟冒泡，这样第二大的数就移到了倒数第二个位置。如此进行直到第 $n-1$ 趟冒泡，就可实现从小到大排序。例如，对序列 49,38,65,97,76,13,27,56 进行从小到大排序，其冒泡排序过程如下：

第一趟：49 38 65 97 76 13 27 56
38 49 65 97 76 13 27 56
38 49 65 76 97 13 27 56
38 49 65 76 13 97 27 56
38 49 65 76 13 27 97 56
38 49 65 76 13 27 56 97

经过第一趟比较后，最大的一个数沉到最后一个位置；

第二趟：38 49 65 76 13 27 56 97
38 49 65 13 76 27 56 97
38 49 65 13 27 76 56 97
38 49 65 13 27 56 76 97

经过第二趟比较后，次大的一个数沉到倒数第二个位置；

……

同理，最后一趟的序列变成 13　27　38　49　56　65　76　97，排序完成。其流程如图 5-9 所示。

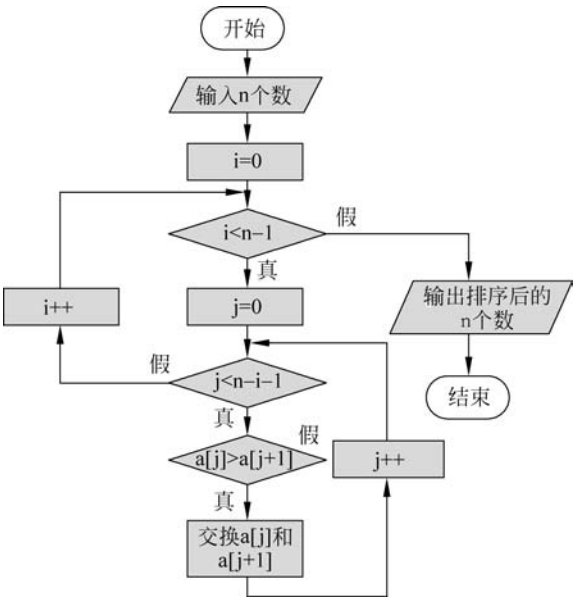

图 5-9　冒泡排序流程图

```
/*
  程序名称:ex5-7.c
  程序功能:冒泡排序
*/
#include<stdio.h>
#define N 8
int main()
{   int a[N],i,j,t;
   for(i=0;i<N;i++)
        scanf("%d",&a[i]);
    printf("\n");
    for(j=0;j<N-1;j++)
        for(i=0;i<N-j-1;i++)
            if(a[i]>a[i+1])
        {
                t=a[i];
                a[i]=a[i+1];
                a[i+1]=t;
        }
    for(i=0;i<N;i++)
        printf("%d ",a[i]);
}
```

程序运行结果：

```
输入:49 38 65 97 76 13 27 30
输出:13 27 30 38 49 65 76 97
```

例 5.8 将一串整数,依次左移一个位置,且原来的第 1 个数移到最后。

分析:首先把下标为 0 的元素值临时保存起来,然后用循环将后面的元素值赋给前面的元素,最后把原来临时保存的下标为 0 的元素值赋给最后一个元素。

```
/*
  程序名称:ex5-8.c
  程序功能:元素移位
*/
#include<stdio.h>
#include<stdlib.h>
#define N 1000
int main()
{
    int i,t,n,a[N];
    scanf("%d",&n);
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    t=a[0];
    for(i=0;i<n-1;i++)
        a[i]=a[i+1];
    a[n-1]=t;
    for(i=0;i<n;i++)
        printf(" %d",a[i]);
    printf("\n");
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入:10
     1 2 3 4 5 6 7 8 9 10
输出:2 3 4 5 6 7 8 9 10 1
```

思考:依次右移一个位置,又该如何实现?

5.2.5 一维数组作为函数参数

数组作为函数参数使用,可以用两类不同的使用方法:

(1) 将数组元素作为函数的参数使用;

(2) 将数组名作为函数的参数使用。

下面依次讨论数组作为函数参数的使用方法。

1. 将数组元素作为函数的参数使用

在例 5.7 和例 5.8 中,就是将数组元素 a[i]作为函数 printf 参数来使用的。实参的值依赖于数组的索引 i 的值。通过如下调用

```
printf("%d",a[i]);
```

输出数组元素 a[i]中的值。当 i 从 0 到 $N-1$ 递增时,数组元素随着索引的增长,从第一个到最后一个元素的值传递到函数 printf 中并显示出来。

例 5.9 利用数组元素作为函数参数，编写两个数交换的程序。

```
/*
  程序名称:ex5-9.c
  程序功能:数组元素作为函数参数
*/
#include<stdio.h>
#include<stdlib.h>
void swap(int x,int y);
int main()
{
    int a[2];
    scanf("%d %d",&a[0],&a[1]);
    printf("调用函数之前:\n");
    printf("a[0]=%d,a[1]=%d\n",a[0],a[1]);
    swap(a[0],a[1]);
    printf("调用函数之后:\n");
    printf("a[0]=%d,a[1]=%d\n",a[0],a[1]);
    system("pause");
    return 0;
}
void swap(int x,int y)
{
    int t;
    t=x;
    x=y;
    y=t;
    printf("调用函数内部:\n");
    printf("x=%d,y=%d\n",x,y);
}
输入 10□20 回车
```

运行结果：

调用函数之前：

```
a[0]=10,a[1]=20
```

调用函数内部：

```
x=20,y=10
```

调用函数之后：

```
a[0]=10,a[1]=20
```

从上面的例题可以看出，数组元素作为函数实参和一般变量作为函数参数一样，也是单向值传递，即形参的值发生了改变，而实参的值没有改变。

2. 将数组名作为函数的参数使用

当数组元素作为参数传递时，函数会接收到数组元素的一个副本，因此函数对复制副本的操作不会影响数组元素原来的值。一般称这种参数传递方法为值传递。除了将数组元素作为函数的参数传递，还可以将数组名作为函数的参数，实现对整个数组的传递。那么形参

和实参之间的关系就发生了变化。这样的函数可以操作与实参数组一致的数组元素。从数组的存储结构中可知,数组名称与普通变量的名称有着很大的不同。数组名实际上标记了一个连续存储区域的首地址。作为参数传递时,不再是简单的值传递的方式了,而是将数组在内存中的存储地址作为参数来传递,这种参数传递的方式称为传地址。函数操作的是同一个存储区域中的数据,而不是数组自身的副本。因此,在函数中通过语句对一个数组元素进行赋值会改变实参数组的内容。

例 5.10 利用数组作为函数参数,编写两个数交换的程序。

```
/*
  程序名称:ex5-10.c
  程序功能:数组名作为函数参数
*/
#include<stdio.h>
#include<stdlib.h>
void swap(int x[2]);
int main()
{
    int a[2];
    scanf("%d %d",&a[0],&a[1]);
    printf("调用函数之前:\n");
    printf("a[0]=%d,a[1]=%d\n",a[0],a[1]);
    swap(a);
    printf("调用函数之后:\n");
    printf("a[0]=%d,a[1]=%d\n",a[0],a[1]);
    system("pause");
    return 0;
}
void swap(int x[2])
{
    int t;
    t=x[0];
    x[0]=x[1];
    x[1]=t;
    printf("调用函数内部:\n");
    printf("x[0]=%d,x[1]=%d\n",x[0],x[1]);
}
输入 10□20 回车
```

运行结果:

调用函数之前:

```
a[0]=10,a[1]=20
```

调用函数内部:

```
x[0]=20,x[1]=10
```

调用函数之后:

```
a[0]=20,a[1]=10
```

从例 5.10 可以看出，数组定义形式作为函数形参、数组名作为函数实参和例 5.9 得出的结果是不一样，即形参数组的元素值发生了改变，而实参数组元素值也随之改变。

例 5.11 定义一个函数 init，将数组中所有元素的值更改为指定的 value。

```
void init( int a[],int n, int value )
{
    int i;
    for ( i = 0; i < n; i++)
        a[i] = value;
}
```

在调用函数 init 时，函数的参数由实参数组名、数组的长度和存储到数组中的值组成，如果实参数组 b 是一个长度为 8 的整型数组，需要将数组元素的值指定为－1，则可以用下面的程序完成：

```
#include < stdio.h >
#include < stdlib.h >
void init(int a[],int,int );
int main()
{
    int b[8],i;
    init(b,8,-1);              /* 函数调用 */
    for(i=0;i<8;i++)
        printf("%d ",b[i]);
    system("pause");
    return 0;
}
```

参数传递的形式如图 5-10 所示。

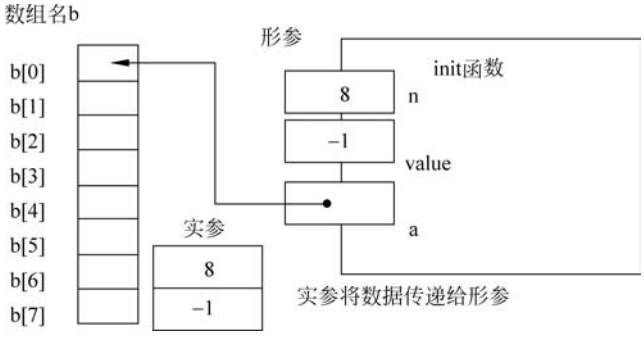

图 5-10 数组作为函数的参数

当实参数组传递给形参时，形参的数组并不分配新的存储单元，而是将形参数组与实参数组共同占用一段存储区域，从而在函数中形参数组的值发生了改变，那么实参数组元素的值也随之改变。在使用形参数组时，要特别留意这个问题。一般情况下，如果不想在函数中改变实参数组的值，可以使用 const 关键字修饰形参数组，这样就可以避免函数在操作数组时发生值更改的意外情形。当然，这种参数传递的方式还是很有效率的，因为节省了大量的存储单元。在第 6 章指针中还会讨论到传地址这种特殊的参数传递形式。

C 语言中提供了 const 关键字,其主要作用就是限定声明的变量值为常量,在程序运行时值不能改动。有时候需要将数组传入函数,但不希望函数体内改变数组元素的值,此时,就可以采用 const 关键字修饰形参数组来实现这样的功能。

例 5.12 定义一个函数 sum,将数组中所有元素值的和返回,即得到一个函数值。

```
int sum(const int a[],int n )
{
    int i,s = 0;
    for ( i = 0; i < n; i++)
        s += a[i];
    return s;
}
```

这里,把函数计算的结果通过 return 返回(只能返回一个值),但 return 语句不能返回整个数组。

例 5.13 用数组表示 ***A*** 和 ***B*** 两个向量,定义函数实现将两个向量的和(对应项相加)存入数组 C 中。

```
void sum (int a[],int b[], int n, int c[] )
{
    int i;
    for ( i = 0; i < n; i++)
        c[i] = a[i] + b[i];
}
```

其主函数为:

```
int main()
{
    int x[10],y[10],z[10],i;
    for(i = 0;i < 10;i++)
        scanf("%d",&x[i]);
    for(i = 0;i < 10;i++)
        scanf("%d",&y[i]);
    sum(x,y,10,z);
    for(i = 0;i < 10;i++)
        printf("%d ",z[i]);
    return 0;
}
```

说明:

(1) 数组作为函数的形参的语法为:

元素类型 数组名[]

或

const 元素类型 数组名[]

两种书写方法是等价的,后一种用法将在指针章节中着重讨论。用关键字 const 声明的数组是一个严格的输入参数,元素的值是不能由函数改变的,这一点很重要。如果没有

const，函数可以改变数组元素的值，具体的操作取决函数的功能需要。

(2) 数组名后的[]不能省略，[]的含义是参数是一个数组，是一个集合的名称，而非普通变量名称。当函数被调用时，实参数组将首地址作为参数传递给形参数组。这样，形参数组和实参数组对应同一个存储区域，而非副本。

(3) 数组作为参数时，实参中只需写实参数组名，不能再取其地址了。例如：

```
sum(x,y,10,z);
sum(&x,&y,10,&z);              /*错误!x,y,z已经是数组名*/
```

5.2.6 一维数组应用

例5.14 短信程序：某班期中C语言课程考试有N名学生参加，输入每名学生的考试成绩，将班级平均分、最高分、最低分和学生的成绩及其排名显示出来。

分析：输入N名学生的考试成绩(0～100)，输入的过程中求出成绩的和，通过比较求出最高分和最低分，然后将学生的原始成绩从大到小排序(同时定义一个数组保存序号，这里把序号作为学号，确保排序过程中成绩交换时，序(学)号也交换)。

```
/*
  程序名称:ex5-14.c
  程序功能:学生成绩短信
*/

#include<stdio.h>
#include<stdlib.h>
#define N 10
int max=0,min=100;
double avg=0.0;
void input(int x[],int y[],int n);    /*输入数据,计算平均值分、最高分、最低分*/
void sort(int x[],int y[],int n);     /*排序*/
int main()
{
    int i,a[N],b[N];                  /*数组a保存学生成绩,数组b保存学号*/
    input(a,b,N);
    sort(a,b,N);
    printf("最高分:%d 最低分:%d 平均分:%.2lf\n",max,min,avg);
    for(i=0;i<N;i++)
        printf("排名:%d 学号:%d 成绩:%d\n",i+1,b[i],a[i]);
    system("pause");
    return 0;
}
void input(int x[],int y[],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
        scanf("%d",&x[i]);
        y[i]=i+1;
        avg=avg+x[i];
```

```
            if(max < x[i])
                max = x[i];
            if(min > x[i])
                min = x[i];
        }
        avg = avg/n;
    }
    void sort(int x[ ],int y[ ],int n)
    {
        int i,j,t;
        for(i = 0;i < n - 1;i++)
            for(j = 0;j < n - 1 - i;j++)
                if(x[j]< x[j + 1])
                {
                t = x[j];                    /* 成绩交换 */
                x[j] = x[j + 1];
                x[j + 1] = t;
                t = y[j];                    /* 学号交换 */
                y[j] = y[j + 1];
                y[j + 1] = t;
                }
    }
```

程序运行结果:

```
输入:
58 35 87 90 78 76 99 25 88 66
输出:
最高分:99 最低分:25 平均分:70.20
排名:1 学号:7 成绩:99
排名:2 学号:4 成绩:90
排名:3 学号:9 成绩:88
排名:4 学号:3 成绩:87
排名:5 学号:5 成绩:78
排名:6 学号:6 成绩:76
排名:7 学号:10 成绩:66
排名:8 学号:1 成绩:58
排名:9 学号:2 成绩:35
排名:10 学号:8 成绩:25
```

例 5.15 将一个数组中的元素首尾依次逆转存储,并输出。

分析:需要注意的是,交换时只需要对一半的元素进行操作,否则结果会还原为原先的顺序。

```
/*
程序名称:ex5 - 15.c
程序功能:数组元素逆转
*/
#include < stdio.h >
#include < stdlib.h >
#define N 10
```

```
void inv (int x[ ],int n);
int main()
{
    int i,a[N] = {8,6,7,1,0,9,3,5,4,2};
    printf("数组元素的原先顺序:\n");
    for (i = 0;i < N;i++)
        printf (" % d ",a[i]);
    printf("\n");
    inv (a,N);
    printf("逆序操作后的顺序:\n");
    for (i = 0;i < N;i++)
        printf(" % d ",a[i]);
    printf("\n");
    system("pause");
    return 0;
}
void inv (int x[ ],int n)
{
    int t,i,j,m = (n - 1)/2;          /* 特别注意,m 的值 */
    for (i = 0;i <= m;i++)
    {
        j = n - 1 - i;
        t = x[i];x[i] = x[j];x[j] = t;
    }
    return;
}
```

程序运行结果:

```
输入: 数组元素的原先顺序:
      8 6 7 1 0 9 3 5 4 2
输出: 逆序操作后的顺序:
      2 4 5 3 9 0 1 7 6 8
```

例 5.16 期中考试后,某专业两个班先分别对计算机课程的考试成绩进行排序,然后按专业来进行排序,确定整个专业的排序情况。

分析:首先将两个班的成绩进行降序排序,然后将排序后的成绩进行合并,这样合并的效率较高。

```
/*
程序名称:ex5 - 16.c
程序功能:合并成绩排序
*/
#include < stdio.h >
#include < stdlib.h >
#define N 5
#define M 4
void sort(int [],int);
void merge(int [],int [],int [],int,int);
int main()
{
    int j;
```

```
    int a[N] = {98,64,75,91,55};
    int b[M] = {90,58,84,61};
    int c[N+M];                          /* 合并后的结果数组 */
    printf("a 数组排序前:\n");
    for(j=0;j<N;j++)
        printf("%5d",a[j]);
    printf("\n");
    printf("b 数组排序前:\n");
    for(j=0;j<M;j++)
        printf("%5d",b[j]);
    printf("\n");
    sort(a,N);
    printf("a 数组排序后:\n");
    for(j=0;j<N;j++)
        printf("%5d",a[j]);
    printf("\n");
    sort(b,M);
    printf("b 数组排序后:\n");
    for(j=0;j<M;j++)
        printf("%5d",b[j]);
    printf("\n");
    merge(a,b,c,N,M);
    printf("合并 a 和 b 后:\n");
    for(j=0;j<N+M;j++)
        printf("%5d",c[j]);
    printf("\n");
    system("pause");
    return 0;
}
void sort(int x[],int k) /* 数组元素排序 */
{
    int temp;
    int i,j;
    for(i=0;i<k-1;i++)
    {
        for(j=0;j<k-i-1;j++)
        {
            if(x[j]<x[j+1])
            {
                temp=x[j];
                x[j]=x[j+1];
                x[j+1]=temp;
            }
        }
    }
}
void merge(int a[],int b[],int c[],int n,int m) /* 合并 a 和 b 数组 */
{
    int ia=0,ib=0,ic=0;
    while(ia<n && ib<m)
    {
        if(a[ia]<b[ib])
            c[ic++]=b[ib++];
        else
```

```
            c[ic++] = a[ia++];
        }
        while(ia < n)
        {
            c[ic] = a[ia];
            ia++;
            ic++;
        }
        while(ib < m)
        {
            c[ic] = b[ib];
            ib++;
            ic++;
        }
}
```

程序运行结果：

```
a 数组排序前:
    98  64  75  91  55
b 数组排序前:
    90  58  84  61
a 数组排序后:
    98  91  75  64  55
b 数组排序后:
    90  84  61  58
合并 a 和 b 后:
    98  91  90  84  75  64  61  58  55
```

5.3 二维数组

可以利用一维数组存储班级学生某一门课程的成绩，但要保存班级学生的多门课程成绩时，若还是采用一维数组来处理，就要定义多个相互独立的一维数组，这样对处理某学生的总分就相对麻烦。可以通过引入二维数组，很容易地解决这个问题。

一维数组中，所有的元素按照顺序依次排列，通常称这种结构为线性结构。线性结构可以直接用于表示和存储数学中的向量、有限序列等数据对象。例如，例 5.13 中利用数组来计算两个向量的和。在实际计算中，有时需要更复杂的结构存储数据来完成相应的计算任务，如表示和处理矩阵(由行和列组成的数阵)。它是一个二维的结构，这种数组在 C 语言中可以用二维数组表示。

5.3.1 二维数组的声明

C 语言提供了很好的方法来定义二维数组和更多维的数组。二维数组的声明语法如下：

```
数组元素的类型说明符 数组名称 [数组的行数][数组的列数];
```

下面定义了两个二维数组：

```
double a[3][2];
int b[4][4];
```


数组 ***a*** 是一个包含三行两列的浮点型数组,若采用矩阵概念,称 ***a*** 是 3×2 的浮点型矩阵。类似地,***b*** 是 4×4 的整型矩阵。

数组 a 的逻辑示意图如图 5-11 所示。

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

图 5-11　二维数组元素的逻辑示意图

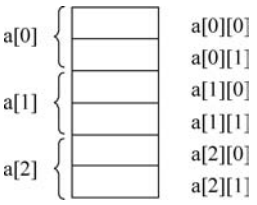

图 5-12　二维数组元素存储示意图

每个元素的表示方法与一维数组类似,也是通过索引(下标)变量的形式引用的,只是增加了一维索引。在计算机内部,C 语言把 a 表示为一个有 3 个元素的数组,而每个数组元素又是有两个整型数的一维数组。因此,数组 a 的存储空间是占用连续的 6 个整型单元,排列次序如图 5-12 所示。

通常,将二维数组看作一维数组的数组。也就是说,二维数组是由多个一维数组(成员类型相同,成员个数也相同)组成的。

在数组 a 对应的二维图表中,第一个下标的值为行号,第二个下标为列号。这种规定是强制的,就像矩阵的二维概念一样。在对应的存储结构中,C 语言强制规定数组的存储顺序为按行序存储,即前一行中的所有元素存储先于后一行中的元素。

5.3.2　二维数组的初始化

和初始化一维数组一样,可以对二维数组进行初始化的操作。常见的初始化方法有下面几种。

(1) 按行全部初始化:定义数组时每行用一对花括号括起来,元素的初值用逗号分隔。如图 5-13 所示是对数组 a 进行按行全部初始化。

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

(2) 按行部分初始化:定义数组时每行用一对花括号括起来,元素的初值用逗号分隔,但每对花括号内数据个数少于列数,这样其余数组元素值自动赋值为 0。如图 5-14 所示是对数组 a 的部分初始化。

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

1	2	3
4	5	6
7	8	9

图 5-13　数组 a 的全部初始化

1	2	0
3	4	0
5	6	0

图 5-14　数组 a 的部分初始化

(3) 顺序初始化:定义数组时,所有数据写在一个花括号内,按数组排列的顺序对各元素赋初值。如图 5-15 所示是数组 α 的顺序初始化,其效果和按行全部初始化一样。

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

(4) 根据初始化确定行数：在定义数组时对全部元素赋初值，即完全初始化，则第一维的长度(行数)可以不指定，但第二维的长度(列数)不能省略。例如：

1	2	3
4	5	6
7	8	9

图 5-15　数组 a 的顺序初始化

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

和

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

其数组的行数都是 3，每个元素的值完全相同。但有时候在定义数组时没有指定行数，也没有对全部元素赋初值，这时就要根据每行有多少个元素来确定行数。例如：

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

1	2	3
4	5	6
7	8	9
10	0	0

图 5-16　根据初始化确定行数

这时，数组 a 的行数为 4，而不是 3 了，如图 5-16 所示。

(5) 全部初值为 0：如果定义数组时，要给所有数组元素赋初值 0，则可以如下定义：

```
int a[3][3] = {0};
```

例 5.17　二维数组元素的访问和初始化。

```
/*
  程序名称:ex5-17.c
  程序功能:二维数组的访问和初始化
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a1[3][3]={{1,2,3},{4,5,6},{7,8,9}};
    int a2[3][3]={{1,2},{3,4},{5,6}};
    int b1[3][3]={1,2,3,4,5,6,7,8,9};
    int b2[][3]={1,2,3,4,5,6,7,8,9};
    int i,j;
    printf( "a1:\n" );
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
            printf("%3d",a1[i][j]);
        printf("\n");
    }
    printf( "\na2:\n" );
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
            printf("%3d",a2[i][j]);
        printf("\n");
    }
```

```
    printf("\nb1:\n");
    for(i = 0;i < 3;i++)
    {
        for(j = 0;j < 3;j++)
            printf(" % 3d",b1[i][j]);
        printf("\n");
    }
    printf("\nb2:\n");
    for(i = 0;i < 3;i++)
    {
        for(j = 0;j < 3;j++)
            printf(" % 3d",b2[i][j]);
        printf("\n");
    }
    system("pause");
    return 0;
}
```

程序运行结果:

```
a1:
  1  2  3
  4  5  6
  7  8  9

a2:
  1  2  0
  3  4  0
  5  6  0

b1:
  1  2  3
  4  5  6
  7  8  9

b2:
  1  2  3
  4  5  6
  7  8  9
```

从该例可以看出,C 语言中二维数组的访问要用二重循环,各种初始化都是按行优先存储的。

5.3.3 二维数组应用

通过二维数组的声明形式、存储方式、元素的访问方法以及初始化的学习,下面通过实例来讲述二维数组的应用。

例 5.18 实现矩阵的输入与输出。

```
/*
  程序名称:ex5 - 18.c
```

```
  程序功能:矩阵的输入与输出
*/
#include<stdio.h>
#include<stdlib.h>
#define M 3
#define N 4
int main()
{
    int a[M][N];
    int i,j;
    for( i = 0; i<M; i++)
        for( j = 0; j<N; j++)
            scanf( "%d",&a[i][j] );
    for( i = 0; i<M; i ++){
        for( j = 0; j<N; j++)
            printf( "%4d",a[i][j] );
        printf("\n"); /*输出完一行后要换行*/
    }
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入:1  2  3  4
     3  4  5  6
     4  5  6  7
输出:
     1  2  3  4
     3  4  5  6
     4  5  6  7
```

例 5.19 编写函数实现两个矩阵的加和乘运算以及矩阵的转置运算。

分析:在实际问题中,通常使用二维数组来定义矩阵结构,矩阵的相加即两个维数相同的二维数组的对应项相加,若要实现矩阵的乘积则须定义出满足矩阵运算要求的合适二维数组,即一个 $m \times n$ 的矩阵与 $n \times p$ 的矩阵相乘,得到一个 $m \times p$ 的矩阵。对于 $m \times n$ 的矩阵,转置运算是将该矩阵中元素的行列互换,得到一个 $n \times m$ 的矩阵。

```
/* 实现矩阵的加法*/
void add(int a[M][N],int b[M][N],int c[M][N])
{
    int i,j;
        for( i = 0; i<M; i ++)
            for( j = 0; j<N; j++)
                c[i][j] = a[i][j] + b[i][j];
}
/* 实现矩阵的转置*/
void transpose( int a[M][N],int t[N][M])
{
    int i,j;
```

```
        for( i = 0; i < M; i ++)
            for( j = 0; j < N; j++)
                t[j][i] = a[i][j];
  }
/ * 实现矩阵的乘积 * /
void product( int a[M][N],int b[N][P],int r[M][P])
{
    int i,j;
    int k = 0;
    for(i = 0;i < M;i++)
        for(j = 0;j < P;j++)
            r[i][j] = 0;
    for( i = 0; i < M; i ++)
        for( k = 0; k < P; k++)
            for( j = 0; j < N; j++)
                r[i][k] += a[i][j] * b[j][k];
}
```

以上 3 个函数的调用如下：

```
/ *
  程序名称:ex5 - 19.c
  程序功能:矩阵运算
* /
# include < stdio.h >
# include < stdlib.h >
# define N 4
# define M 3
int main()
{
    int i,j,x[M][N],y[M][N],z[M][N],s[N][M],t[M][M];
    for(i = 0;i < M;i++)
        for(j = 0;j < N;j++)
            scanf(" % d",&x[i][j]);
    for(i = 0;i < M;i++)
        for(j = 0;j < N;j++)
            scanf(" % d",&y[i][j]);
    add(x,y,z);
    transpose(x,s);
    product(x,s,t);
    for(i = 0;i < M;i++)
    {
        for(j = 0;j < N;j++)
            print(" % d ",z[i][j]);
        printf("\n");
    }
    for(i = 0;i < N;i++)
    {
        for(j = 0;j < M;j++)
            print(" % d ",s[i][j]);
        printf("\n");
```

```
    }
    for(i = 0;i < M;i++)
    {
        for(j = 0;j < M;j++)
            print(" % d ",t[i][j]);
        printf("\n");
    }
    system("pause");
    return 0;
}
```

例 5.20 有 N 个学生，M 门课程，要求计算每个学生的总成绩，并输出总分最高、总分最低的学生成绩信息，统计超过总平均分（含平均分）的人数。

分析：N 个学生 M 门课程可采用 N 行 M 列的二维数组来存储，由于每位学生都有总分，因此可以定义 N 行 $M+1$ 列的二维数组，最后一列保存每个学生的总分，求出总分的同时把每个学生总分累加，以便后面求平均分。再分别求出总分最低和最高的学生所在的行，然后输出总分最高和最低的学生信息以及总平均分以上的人数。

```
/*
  程序名称:ex5 - 20.c
  程序功能:学生成绩信息统计
*/
#include < stdio.h >
#include < stdlib.h >
#define N 5
#define M 3
void caclsum(int x[][M + 1]);
int countavg(int x[][M + 1],double);
int main()
{
    int i,j,a[N][M + 1] = {0},imax,imin,max,min;
    double avg;
    for(i = 0;i < N;i++)
        for(j = 0;j < M;j++)
            scanf(" % d",&a[i][j]);
    caclsum(a);
    max = min = a[0][M];
    imax = imin = 0;
    avg = a[0][M];
    for(i = 1;i < N;i++)
    {
        if(a[i][M]> max)
        {
            max = a[i][M];
            imax = i;
        }
        if(a[i][M]< min)
        {
            min = a[i][M];
            imin = i;
```

```
            }
            avg = avg + a[i][M];
        }
        avg = avg/N;
        printf("学生各科成绩和总分:\n");
        for(i = 0;i < N;i++)
        {
            for(j = 0;j < = M;j++)
                printf(" % 4d",a[i][j]);
            printf("\n");
        }
        printf("总分最高的学生成绩:\n");
        for(j = 0;j < = M;j++)
            printf(" % 4d",a[imax][j]);
        printf("\n 总分最低的学生成绩:\n");
        for(j = 0;j < = M;j++)
            printf(" % 4d",a[imin][j]);
        printf("\n 总平均分: % .2lf\n",avg);
        printf("总分大于或等于总平均分的人数: % d\n",countavg(a,avg));
        system("pause");
        return 0;
}
void caclsum(int x[][M + 1])
{
    int i,j;
    for(i = 0;i < N;i++)
        for(j = 0;j < M;j++)
            x[i][M] += x[i][j];
}
int countavg(int x[][M + 1],double avg)
{
    int i,j;
    int count = 0;
    for(i = 0;i < N;i++)
        if(x[i][M] > = avg)
            count++;
    return count;
}
```

程序运行结果:

```
输入:
1  2  3
4  5  6
7  8  9
10  11  12
13  14  15
输出:
学生各科成绩和总分:
    1  2  3  6
    4  5  6  15
```

```
    7  8  9  24
    10  11  12  33
    13  14  15  42
总分最高的学生成绩:
    13  14  15  42
总分最低的学生成绩:
    1  2  3  6
总平均分:24.00
总分大于或等于总平均分的人数:3
```

例 5.21 生成特殊矩阵：对于一个 n 阶方阵，输入 $n=5(n<100)$，得到如下所示的 n 阶拐角矩阵。

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 3 & 4 & 5 \\ 3 & 3 & 3 & 4 & 5 \\ 4 & 4 & 4 & 4 & 5 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

分析：生成一个特殊矩阵，一般从特殊的下标来找规律，本题定义一个二维数组，从矩阵主对角线(行下标和列下标相同：i==j)出发，分成右上和左下两部分，i<=j 为右上部分(元素值为列号+1)，i>j 为左下部分(元素值为行号+1)。

```
/*
程序名称:ex5-21.c
程序功能:生成特殊矩阵
*/
#include<stdio.h>
#include<stdlib.h>
#define N 100
void fun(int x[][N],int n)
{
    int i,j;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
        if(i<=j)
            x[i][j]=j+1;          /*元素值为列号+1*/
        else
            x[i][j]=i+1;          /*元素值为行号+1*/
}
int main()
{
    int n,i,j,a[N][N];
    printf("输入矩阵阶数:");
    scanf("%d",&n);
    fun(a,n);                     /*函数调用*/
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%d ",a[i][j]);
        printf("\n");
```

```
    }
    system("pause");
    return 0;
}
```


运行结果：

```
输入矩阵阶数:5
1  2  3  4  5
2  2  3  4  5
3  3  3  4  5
4  4  4  4  5
5  5  5  5  5
```

说明：使用数组应特别注意的是，数组下标越界的问题。首先，数组的下标以 0 作为起点，其最大下标与数组长度之间相差一个数值。当使用的下标超出了数组声明的范围时，就产生了一个数组元素超出范围引用错误，可是大多数编译器并不对此类错误提示编译错误，很多情况下也不会出现运行错误，但会产生不正确的结果。

5.4 字 符 数 组

C 语言中只有字符变量，没有字符串变量，字符串的应用非常广泛，在 C 语言中用字符类型的数组来存储和处理字符串。字符数组就是用来存储字符数据的数组，如姓名、单词、电话号码等文本数据。C 语言提供了非常强大的处理字符数据功能，这些功能都是从存储字符型数据开始的。

5.4.1 字符数组的定义

字符数组也是数组，其定义方式与其他数组相同。下面是一维字符数组的声明形式：

```
char 数组名称[长度];
```

例如：

```
char line[ 80 ];
char text[ 1000 ];
```

上面分别定义了两个字符型数组 line 和 text。数组都用常量表达式指定了长度，即可以存放字符数据的个数，也就是说，系统分配了指定的存储空间来存放字符型数据。

也可以定义二维的字符型数组，如：

```
char 数组名称[行数][列数];
```

例如：

```
char name[ 5 ][ 20 ];
```

这个 name 数组可以理解为定义了 5 名同学的姓名，每个姓名的存储字节数不超过 20 个字符。二维字符数组能够存储多个字符串，若是处理多个字符串，则“行数”表示字符串的个数，“列数”表示这些字符串中最长字符串的长度再加 1(即结束符)。

5.4.2 字符数组元素的引用

引用字符数组元素的方式，与其他数组相同，即通过数组名和下标，利用循环实现对数组元素的访问。例如，当我们定义了数组 line 后，可以通过如下方式完成对数组元素的赋值。

```
for ( i = 0; i < 80 && ( line[i] = getchar() ) != '\n'; i++)
;                                        /* 空循环体，即什么也不做的循环体 */
```

这个循环将一行(最多 80 个)字符读入数组 line。值得注意的是，代码中使用了严格的条件保证对数组 line 的访问不越界。既保证了数组长度不越界，又能够实现读入回车符作为输入数据的结束标志。

5.4.3 字符数组的初始化

定义字符数组时也可以像其他数组一样进行初始化，主要有以下初始化方法。

1. 用字符实现初始化

```
char str[12] = { 'N','a','n','j','i','n','g'};
```

上面用字符常量初始化了一个长度为 12 的字符数组，仅对前 7 个元素指定了初始值。初始化后的 str 的存储内容如图 5-17 所示。

[0]						[6]					[11]
N	a	n	j	i	n	g	\0	\0	\0	\0	\0

图 5-17　字符数组的初始化

与前面讲述的数值型数组的初始化相似，字符型数组的部分初始化后，对未赋初值的元素使用\0 表示，而不是其他字符。执行 printf("%s",str)；后得到结果为：Nanjing

若对下面定义的字符数组进行初始化：

```
char str[7] = { 'N','a','n','j','i','n','g'};
```

执行 printf("%s",str)；后得到结果为：Nanjing□，这是因为 str[6]后面不是结束符\0，因此在处理时要特别注意。

2. 用字符串对字符数组初始化

```
char str[12] = {"Nanjing"};
```

或者

```
char str[12] = "Nanjing";
```

其在内存中的存储内容如图 5-18 所示。

[0]						[6]	[7]				[11]
N	a	n	j	i	n	g	\0	\0	\0	\0	\0

图 5-18　字符串的初始化

3. 根据初值确定字符数组的长度

```
char str1[] = {'N','a','n','j','i','n','g'};
```

```
char str2[] = "Nanjing";
```

则字符数组 str1 的长度为 7,但 str2 数组的长度为 8(这是因为字符串后面有一个结束符\0)。

4. 二维字符数组的初始化

和二维数值型数组赋初值类似,可以按行初始化,也可以用多个字符串进行初始化。例如:

```
char str[][5] = {{' ',' ','*'},{' ','*',' ','*'},{'*',' ',' ',' ','*'},{' ','*',' ','*'},{' ',' ','*'}};
char weekday[7][10] = { "Sunday","Monday","Tuesday","Wednesday","Thursday","Friday",
"Saturday" };
```

str[0]			*	\0	\0
str[1]		*		*	\0
str[2]	*				*
str[3]		*		*	\0
str[4]			*	\0	\0

图 5-19　二维字符数组初始化

数组 str 的存储形式如图 5-19 所示。

对于字符数据来说,是将对应字符的 ASCII 码值保存在内存中,字符'\0'是字符串的结束符,其 ASCII 码为 0(称为空字符)。要注意空字符与'0'字符或空格字符的区别。字符'0'的 ASCII 编码值为 48,空格字符的 ASCII 码为 32。

5.4.4　字符串的表示

字符串常量,是用双引号对" "括起来的部分。字符型数据指的是一个字符数据或转义字符,用单引号对''表示。在 C 语言中,二者的区别很大。

1. 字符串的存储

在 C 语言中,字符串的存储是利用字符数组来实现的,编译系统将字符串以字符数组的形式存储,即从内存空间中分配连续的若干存储单元,依次保存字符串中的字符,且每个字符占用一字节。在字符串常量中所有字符之后,增加一个空字符'\0'(ASCII 码值为 0)作为字符串的结束标志。

例如,定义字符数组 str 数组:

```
char str[] = "Nanjing";
```

其存储表示如图 5-20 所示。

[0]						[6]	[7]
N	a	n	j	i	n	g	\0

图 5-20　字符串 str 的存储

虽然它只有 7 个字符,其内部表示却需要占 8 字节存储,尤其是元素 str[7],存储了一个空字符'\0',以表示字符串的结束。

值得注意的是,当用常量字符串初始化默认长度的字符数组时,数组的长度必为字符串字符个数再加一个存放空字符的字节数。C 语言规定:常量字符串会自动包含一个空字符。

2. 字符串数组

一个字符数组可以存储一个字符串,而字符串数组是一个二维字符数组(可以存储多个

字符串),该数组中的每一行都是一个字符串。下面的说明语句声明了一个数组来存放星期名称。每个名称的长度小于 10 个字符:

```
#define DAY 7
#define LEN 10
char weekday[DAY][LEN] = {
    "Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday" };
```

5.4.5 字符数组的输入与输出

1. 用循环实现字符数组的输入与输出

与数值型数组一样,字符数组也是从索引号为 0 的元素开始访问的。使用循环语句很容易访问到数组中所有元素。通常使用循环计数器作为数组的下标,依次地遍历数组中的所有元素,可用 getchar()或 scanf("%c",&str[i])与循环来实现。

例 5.22 从标准输入设备(键盘)中输入一行英文文字,统计这行文字有多少个英文单词。

分析:一行文字中单词间的分隔符,一般认为是空格。也就是说如果前一个字符是空格,当前位置字符不是空格,就是新单词的开始,这时记录单词数的计数器增加 1。因此,可以设计一个标志来记录前一个位置字符是否为空格字符。如此反复,直到该行结束。

```
/*
  程序名称:ex5-22.c
  程序功能:统计单词数
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char str[81];
    int i=0,num=0,word=0;
    char c;
    while((str[i++]=getchar())!='\n');
    str[i]='\0';
    for(i=0;(c=str[i])!='\0';i++)
        if(c==' ') word=0;
        else if(word==0)
        {
            word=1;
            num++;
        }
    printf("%d\n",num);
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入:This is a long stentence for testing.
输出:7
```

2. 用%s 格式实现字符串的输入与输出

在格式控制字符串中,使用%s 格式控制,在 scanf()和 printf()函数中实现对字符串的输入与输出。

```
scanf("%s",str);
printf("%s","hello,world!");
```

和其他接受字符串参数的标准库函数一样,printf()函数通过在字符数组中寻找空字符来标记字符串结束。如果 printf()函数被传入一个不包含'\0'的字符数组,会首先将每个数组元素的内容当作字符显示。但会继续将内存中位于数组参数之后的内容当作字符显示,直到遇到空字符或者试图访问没有分配给该程序的内存单元,从而导致运行时错误或显示一些不可预知的内容。所以处理字符串时,必须确保在每个字符串结尾插入空字符。printf()函数格式字符串中的格式符%s 可以和最小字段宽度一起使用,如下所示:

```
printf("%8s,%-3s\n","Short","Strings");
```

屏幕显示结果:

```
□□□Short
Strings
```

第一个字符串占 8 列宽度以右对齐形式显示,第二个字符串比指定的字段宽度长,因此字段被扩展到刚好容纳该字符串而没有填充空格,并以左对齐形式显示。一般我们更习惯于字符串以左对齐方式输出。

scanf()函数可以用于输入字符串。例 5.23 中给出了使用 scanf()函数和 printf()函数执行字符串 I/O 的简单函数。

例 5.23 要求用户输入一个代表课程名称的字符串、表示星期几的整数以及表示该课程上课节次的字符串,输出该课程上课节次对应的课程表。

```
/*
  程序名称:ex5-23.c
  程序功能:字符串的输入与输出
*/
#include<stdio.h>
#include<stdlib.h>
#define DAY 7
#define LEN 10
char weekday[DAY][LEN] = {"Sunday","Monday",
        "Tuesday","Wednesday","Thursday","Friday","Saturday"};
int main()
{
    char course_name[20];
    int days;
    char time[6];
    int i;
    scanf("%s",course_name);
    scanf("%d",&days);
    scanf("%s",time);
    for(i = 0; i<DAY; i++)
```

```
        printf( " %10s",weekday[i] );
    printf( "\n %s",time );
    for( i = 0; i < 10 * days; i++)
        printf( " " );
    printf( " %s\n",course_name );
    system("pause");
    return 0;
}
```

程序运行结果：

```
输入:Math 3 5～6
输出:Sunday   Monday   Tuesday   Wednesday   Thursday   Friday   Saturday
5～6                              Math
```

scanf 函数读入字符串的方法与读入数值型数据的方法类似。当 scanf 函数扫描字符串时，数据间的分隔符可以是空格符、换行符和制表符。从第一个非分隔字符开始，scanf 函数将字符复制到它的字符数组参数的连续内存单元中。如图 5-21 所示，当遇到分隔字符时，停止读入，同时 scanf 函数将空字符'\0'存入数组参数中的字符串末尾处。

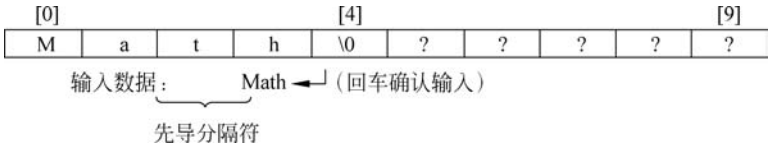

图 5-21　执行 scanf()函数

注：scanf()函数会在字符串的结尾处自动添加一个空字符'\0'

由于 scanf 函数是以默认分隔符作为数据结束的标志，对于字符串中确实存在着诸如空格之类的数据时，就会出现如下情形：

```
char str[50];
scanf( " %s",str );
```

当输入：

```
C Programming Language ↵
```

str 中的字符串如图 5-22 所示。

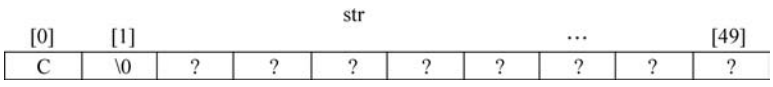

图 5-22　scanf()函数使用%s 格式读入字符串时遇到分隔符结束

3. 用 gets()和 puts()函数实现字符串的输入与输出

C 语言中，可使用 gets()和 puts()函数来实现字符串的输入与输出。

例如：

```
char str[50];
gets( str );
puts( str );
```

使用 gets()函数可以读入具有空格分隔符的字符串到指定的字符数组中。图 5-23 所示为 gets()函数读入字符串的结果。也就是说,gets()函数可以输入带分隔符的字符串,而 scanf()函数用%s 输入字符串时,遇到分隔符就结束了。puts()函数与之类似,输出以'\0'为结束符的字符串。

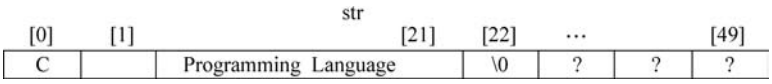

图 5-23 使用 gets()函数读入字符串遇到回车符结束

例 5.24 输入一句英文,输出该句中的最长单词。

分析:一般英文句子中单词之间是用空格(可能有的单词之间不止一个空格)分开的,先将英文句子作为一个字符串输入,从左向右读取字符,并记录字符个数,直到某个单词结束,这样就得到了某个单词的长度;再与前面找出的最长单词进行比较,得到目前为止的最长单词,并记下最长单词中最后一个字符的位置(下标),如此反复,直到字符串结束;然后用最长单词下标减去最长单词的长度得到输出的开始位置,再输出"最长单词"的长度字符,就得到最长单词。

```
/*
  程序名称:ex5-24.c
  程序功能:求最长单词
*/

#include<stdio.h>
#include<stdlib.h>
int main()
{
   char line[1000];
   int maxlen=0,i=0,max=0,end=0;      /*maxlen 为当前单词的长度,max 为最长单词长度,
end 为到目前为止,最长单词的最后一个字符位置(下标)*/
   gets(line);
   while(line[i])
   {
     while(line[i]!=''&&line[i])
     {
        i++;
        maxlen++;
     }
     if (max<maxlen )
     {
        max=maxlen;
        end=i;
     }
     while(line[i]==' ')
     {
        i++;
        maxlen=0;
     }
   }
```

```
    for (i = end - max; i < end;i++)/ * 输出最长单词 * /
        printf( " % c",line[i] );
    printf("\n");
    system("pause");
    return 0;
}
```

5.4.6 常用的字符串处理函数

字符串不是标准数据类型,一般情况下不能直接对字符串进行操作。如果都通过循环对每个字符进行操作,就使字符串的操作比较烦琐。为此,C 语言提供了专门处理字符串的函数,这些函数的原型包含在头文件 string.h 中,下面介绍几个常用的字符串处理函数。

1. 字符串的赋值函数 strcpy()

一般形式:

```
strcpy( 字符数组,字符串 );
```

函数功能:将字符串复制到字符数组中。如果字符串的长度小于数组的长度,其余部分用空字符'\0'填补,返回处理完成的字符串。

说明:不能用赋值符号=对字符串进行复制。字符数组的长度足够大,未使用的部分是空字符'\0'。

例如:

```
char str[20];
strcpy( str,"C Programming" );
```

则 str 中的数据如图 5-24 所示。

C		P	r	o	g	r	a	m	m	i	n	g	\0	\0	…	\0

图 5-24 strcpy()函数执行的结果

2. 字符串的长度

一般形式:

```
strlen(字符串);
```

函数功能:返回字符串的有效字符数(除空字符以外的字符个数)。例如:

```
strlen("Hello"); / * 结果是 5 * /
```

说明:strlen()和 sizeof()是有区别的,sizeof("Hello")的值是 6。

3. 字符串的连接

一般形式:

```
strcat(字符数组 1,字符串);
```

函数功能:把字符串连接到字符数组 1 所表示的字符串的后面,结果放在字符数组 1 中。

说明:字符数组 1 的长度要足够大,以便容纳新的字符串。

例如：

```
char C[40] = "C Programming",str[12];
strcpy( str," Language" );
strcat( C,str );
```

完成上述操作后，字符串 C 中的内容为："C Programming Language"。

4. 字符串的比较

一般形式：

```
strcmp(字符串 1,字符串 2);
```

函数功能：比较两个字符串的大小，返回一个整数值。

说明：比较的过程是对两个字符串自左向右逐个字符按 ASCII 码值大小比较，直到出现不同的字符或遇到结束符'\0'为止。若全部字符相同，则字符串 1 和字符串 2 相等；若出现不相同的字符，则以第一个不相同的字符比较的结果为准。比较的结果由函数值带回：

- 若函数值为 0，则字符串 1 与字符串 2 相等。
- 若函数值大于 0，则字符串 1 大于字符串 2。
- 若函数值小于 0，则字符串 1 小于字符串 2。

例如：

```
char str1[] = "C Programming Language";
char str2[] = "Basic Programming Language";
if ( strcmp(str1,str2) > 0 )
   printf( "str1 字典序中排在 str2 的后面\n" );
else
   printf( "str1 字典序中排在 str2 的前面\n" );
```

上述代码段的执行结果为：str1 字典序中排在 str2 的后面。

附录 E 列出了 C 语言常用的字符串处理函数。

在使用字符串时要注意和数值型数组的区别，特别是在比较和交换中的操作：数值型数据直接通过关系运算符和赋值就能完成的操作，字符串或者字符数组则必须借助于函数才能完成。例如，排序中的比较和交换操作，数值型数据和字符串的操作对比如表 5-3 所示。

表 5-3　交换排序中数值型数据和字符串间的操作对比

交换排序		
	数值型数组	字符型数组
数据定义	int list[M];	char list[M][N];
比较操作	if (list[i] < list[first])	if (strcmp(list[i],list[first]) < 0)
	first = i;	first = i;
元素交换	temp = list[min];	strcpy(temp,list[min]);
	list[min] = list[fill];	strcpy(list[min],list[fill]);
	list[fill] = temp;	strcpy(list[fill],temp);

5.4.7 字符数组应用

下面以一个实例介绍字符数组的应用。

例 5.25 编写一个函数,完成字符串的复制操作,即将一个字符串复制到一个字符数组中。

分析:该问题中有一个假设,即用于存储字符串的目标数组必须有足够的空间,足以存放被复制字符和空字符。函数定义本身很简单,source 字符串的内容不被修改,可以用 const 关键字修饰:

```
void str_copy(char target[],char source[])
{
  int i = 0;
  while( source[i] != '\0'){
    target[i] = source[i];
    i++;
  }
  target[i] = '\0';
}
```

该函数的功能类似于 strcpy()函数。利用 source[i]值为空字符时作为循环的结束条件,需要特别注意的是,循环语句后的赋值语句,为 target[i]赋一个空字符,以表示字符串的结束。

此外,字符串复制的基本操作是赋值运算,所以上述程序常被写成:

```
void str_copy ( char target[],char source[])
{
  int i = 0;
  while ( ( target[i] = source[i] ) != '\0' )
    i++;
}
```

while 循环及后面的代码可以进一步简化为:

```
while (target[i] = source[i]) ++i;
```

因为空字符'\0'的 ASCII 码的值为 0,正好可以作为循环结束条件。

例 5.26 假定输入的字符串中只包含字母和 * 。请编写程序将字符串中的前导符 * 全部删除,字符串中间和后面的 * 不删除。例如,字符串中的内容为“**** A * BC * DEF * G *******”,删除前导 * 后,字符串中的内容应当是“A * BC * DEF * G *******”。

分析:首先需要从字符串开始找到不是 * 的字符,然后从该位置开始,将后面所有字符保存并输出。

```
/*
  程序名称:ex5-26.c
  程序功能:字符串变换
*/
#include<stdio.h>
#include<stdlib.h>
```

```
#define N 80
int main()
{
    int i = 0,k;
    char a[N];
    gets(a);
    puts(a);
    while(a[i] == '*')
     i++;
     k = i;
    for(i = 0;a[k]!= 0;i++,k++)
     a[i] = a[k];
    a[i] = '\0';
    printf("%s\n",a);
     system("pause");
    return 0;
}
```

程序运行结果:

```
输入: ****A*BC*DEF*G*******
输出: A*BC*DEF*G*******
```

5.5 多维数组

5.5.1 多维数组的定义

多维数组的定义与一维或者二维数组的定义类似:必须指明数组的名称、数据类型和数组的长度。下面定义了一个三维数组:

```
int a1[3][2][4];
```

也可以用这种形式定义更高维的数组。高维数组的声明形式语法如下:

```
数组元素的类型说明符 数组名称 [第一维][ 第二维]…[第 N 维];
```

例如:用一个数组 table,来描述学校各学院、年级和课程的选课人数。

假设学校有 10 个学院,每个学院的在校生分成 4 个年级,每个年级需要修 50 门课程,那么描述这样一个数据结构,可以存储每一门课程的选课人数。

```
int table[10][4][50];
```

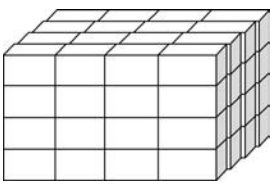

图 5-25 三维数组的逻辑结构

table[2][2][10]数组元素表示的含义为:学院索引号为 2,年级索引号为 2,选修课程索引号为 10 的人数。

其存储结构如图 5-25 所示。利用该结构,可以处理许多问题。如确定一门课程学生的总人数;某个学院 2 年级学生选择课程 2 的人数等问题。

例 5.27 在定义 table 数组中寻找和输出每个学院每

门课程的选修人数。

```
for( course = 0; course < 50; course++)
{
   for( campuse = 0; campuse < 10; compuse++)
   {
       stu_sum = 0;
       for( rank = 0; rank < 4; rank++)
          stu_sum += table[campus][course][rank];
       printf( "Campus %d Course: %d: Students Number: %d",
               campus,course,stu_sum );
   }
}
```

在处理多维数组时存在一个隐患：同一个程序中声明了多个多维数组，那么内存空间很快就会耗尽。因此，在程序中应注意每个数组所需要的内存空间的大小。例如，数组 a1 是一个三维数组，大小为 sizeof(int)×3×2×4，即 96 字节。由于占用大量内存的原因，三维或更多维数组较少使用。

5.5.2 多维数组的初始化

多维数组初始化形式如下：

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

数值表是一个由逗号分隔的常量表。数组 b 是由两个 2×2 的数组构成的存储结构。由于数组在计算机内部的存储是线型的，数组 b 的元素存储结构如图 5-26 所示。

b[0][0][0]	b[0][0][1]	b[0][1][0]	b[0][1][1]	b[1][0][0]	b[1][0][1]	b[1][1][0]	b[1][1][1]
1	2	3	4	5	6	7	8

图 5-26　三维数组初始化后元素的存储结构

说明：

（1）对于三维数组，整型常数 1、整型常数 2、整型常数 3 可以分别看作“深”维（或“页”维）、“行”维、“列”维。可以将三维数组看作一个元素为二维数组的一维数组。三维数组在内存中先按页，再按行，最后按列存放。

（2）多维数组在三维空间中不能用形象的图形表示。多维数组在内存中排列顺序的规律是：第一维的下标变化最慢，最右边的下标变化最快。

（3）多维数组的数组元素的引用：数组名[下标 1][下标 2]…[下标 k]。多维数组的数组元素可以在任何相同类型变量可以使用的位置被引用。只是要注意，不要越界。

5.6 变长数组

5.6.1 不指定维长的数组初始化

使用数组初始化的方法建立如下字符数组：

```
char e1[12] = "read error\n";
```

```
char e2[13] = "write error\n";
char e3[18] = "cannot open file\n";
```


如果用手工去计算每一条信息的字符数以确定数组的长度,这是件很麻烦的事。所以在定义数组时,可以利用 C 语言提供的变长数组初始化的方法,让编译系统自动地计算数组的长度。变长数组建立一个不指明长度的、足够大的数组以存放初始化数据。使用这种方法,以上信息可以定义为:

```
char e1[] = "read error\n";
char e2[] = "write error\n";
char e3[] = "cannot open file\n";
```

给定上面的初始化,如下语句:

```
printf( "%s has length %d\n",e2,sizeof( e2 ) );
```

将打印出:

```
write error has length 13
```

除了可以减少麻烦外,通过应用变长数组初始化,程序员还可以修改任何信息,而不必担心随时可能发生的计算错误。

5.6.2 可变长数组及定义

在 C 语言中,数组维数必须用常数表达式来声明。而 C99 扩充了这一局限性,它为数组增加了一个强大的新功能——可变长度(variable length)。在 C99 中,可以声明这样的数组:其长度可以用任何有效的表达式来指定,这就是变长数组(variable-length array)。然而,只有本地数组(即带有块域或原型域的数组)可以是可变长度的数组,并要求表达式的值已确定。以下是变长数组的实例:

```
void f( int dim)
{
    char str[ dim ];                    /*可变长字符数组 str */
        /* … */
}
```

从数组定义的角度出发,上面的例子违反了数组长度必须是常量的原则。C99 标准对 C 语言进行了很好的扩充,使得上述定义合法化。这仅限于动态结构的数组定义,不能使用在全局数组、外部数组等具有静态特征的数据结构中。通过上面的例子可以看出:数组 str 的大小用 dim 表示,由传递给 f()的值决定。这样,每调用一次 f()均可产生不同长度的局部数组 str。

C99 增加可变长数组的支持,使得程序可以灵活地在运行时改变数组的长度,这是一个具有广泛适用性的特性。注意,在 C99 之前标准中,均不支持可变长数组。

5.7 数组应用举例

例 5.28 模拟掷骰子的游戏,验证骰子的 6 个面出现的概率基本相同。

分析：在这样的模拟问题中，通常使用随机数函数，随机生成一组数据。对于骰子来说，其6个面对应的数值为1～6，如何将随机数转换成骰子各面对应的数值，是本例题的关键所在。这里选用%求余运算符将随机数变换到0～5的数，即实现了功能，接下来的任务就是计数了。C语言提供了一组随机数函数，可直接在程序中使用，具体的说明参见附录。

```
/*
  程序名称:ex5-28.c
  程序功能:模拟掷骰子的游戏
*/
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
#define N 60000
int main()
{
    int ludo[6] = {0};
    int i;
    srand(time(NULL));                    /*以现在的系统时间作为随机数的种子来产生随机数*/
    for(i = 0; i<N; i++)
        ludo[rand()%6]++;                 /*产生随机数的函数*/
    for(i = 0; i<6; i++)
        printf("第%d面的个数%d,出现概率%8.4f\n",i+1,ludo[i],(double)ludo[i]/N);
  return 0;
}
```

程序运行结果：

```
第1面的个数9987,出现概率0.1664
第2面的个数9925,出现概率0.1654
第3面的个数9885,出现概率0.1648
第4面的个数10039,出现概率0.1673
第5面的个数10147,出现概率0.1691
第6面的个数10017,出现概率0.1669
```

例5.29 统计子串b在母串a中出现的次数，如果母串为asd asasdfg asd as zx67 asd mklo，子串为as，则子串的个数为6。

分析：母串用i作为循环控制变量，子串用j作为控制变量。对于每一个s[i]，扫描其后面的字符是否构成子串。若是，则计数加1；否则i移至下一个位置。具体做法是：j从i开始，比较母串与子串中的字符。若s[i]及其后续有strlen(t)个字符与子串中对应字符相等，则表示s[i]即为子串的开始；否则s[i]开头的就不是子串。这时，i移至下一位置继续比较，直到字符串结束。

```
/*
  程序名称:ex5-29.c
  程序功能:查找子串
*/
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
```

```
int count(char [],char []);
int main()
{
    char a[80],b[10];
    int k;
    gets(a);
    gets(b);
    k = count(a,b);
    if(k == 0)
        printf("没有找到!\n");
    else
        printf(" % d\n",k);
        system("pause");
    return 0;
}
int count(char s[],char t[])
{
    int i,j,k,m = 0;
    for(i = 0;s[i]!= '\0';i++)
    {
        k = 0;
        for(j = i;s[j] == t[k]&&k < strlen(t);j++)
            k++;
        if(t[k] == '\0')
            m++;
    }
    return m;
}
```

输入:asd asasdfg asd as zx67 asd mklo
as

输出:6

例 5.30 将一个 $N\times N$ 的矩阵,顺时针旋转 90°输出。

分析:可以将旋转后的矩阵用另外一个二维数组保存,这样关键就是要寻找两个矩阵行和例的对应关系。下面以 3×3 的矩阵为例来说明:

```
旋转前:                                    旋转后:
a[0][0]  a[0][1]  a[0][2]                a[2][0]  a[1][0]  a[0][0]
a[1][0]  a[1][1]  a[1][2]                a[2][1]  a[1][1]  a[0][1]
a[2][0]  a[2][1]  a[2][2]                a[2][2]  a[1][2]  a[0][2]
```

用 b 数组来存放旋转 90°后的 a 数组:

```
b[0][0]  b[0][1]  b[0][2]                a[2][0]  a[1][0]  a[0][0]
b[1][0]  b[1][1]  b[1][2]                a[2][1]  a[1][1]  a[0][1]
b[2][0]  b[2][1]  b[2][2]                a[2][2]  a[1][2]  a[0][2]
```

可以看出:旋转后的矩阵 ***b*** 和原来的矩阵 ***a*** 有这样的对应关系,b[i][j]=a[N−1−j][i]。

```
/*
  程序名称:ex5 - 30.c
  程序功能:矩阵旋转
```

```
*/
#include<stdio.h>
#include<stdlib.h>
#define N 3
int main()
{
    int i,j,a[N][N],b[N][N];
    for(i=0;i<N;i++)
     for(j=0;j<N;j++)
        scanf("%d",&a[i][j]);
    for(i=0;i<N;i++)
     for(j=0;j<N;j++)
        b[i][j]=a[N-1-j][i];
    for(i=0;i<N;i++)
    {
        for(j=0;j<N;j++)
            printf("%4d",b[i][j]);
        printf("\n");
    }
   system("pause");
   return 0;
}
```

运行结果：

```
输入：                          输出：
    1 2 3                       7 4 1
    4 5 6                       8 5 2
    7 8 9                       9 6 3
```

例 5.31 设有 n 个正整数，将它们连成一排，组成一个最大的多位数。例如，输入 $n=3$ 以及 3 个正整数 13，312，343，则连成的最大数为 34331213。

分析：可以将这些整数看成数字字符串来处理，这样问题就转化为对这些数字字符串降序排序，然后输出即可。

```
/*
  程序名称:ex5-31.c
  程序功能:组成最大数
*/
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define N 3
#define M 10
void sort(char x[][M],int);
int main()
{
    int i,j;
    char a[N][M],t[M];
    for(i=0;i<N;i++)
     scanf("%s",a[i]);
```

```
    sort(a,N);
    for(i = 0;i < N;i++)
     printf(" % s",a[i]);
  printf("\n");
  system("pause");
  return 0;
}
void sort(char x[ ][M],int n)
{
    int i,j;
    char t[M];
    for(i = 0;i < n - 1;i++)
     for(j = 0;j < n - 1 - i;j++)
        if(strcmp(x[j],x[j + 1])< 0)
        {
            strcpy(t,x[j]);
            strcpy(x[j],x[j + 1]);
            strcpy(x[j + 1],t);
        }
}
```

本 章 小 结

数组是程序设计中最常用也最为重要的数据结构。数组从元素值的角度,可分为数值数组(整型数组、浮点型数组)、字符数组以及后面将要介绍的指针数组、结构体数组等。从数组的维数来看,数组可以分为一维数组、二维数组、更多维的数组。对于不同的问题,需要根据实际需要选择合适的结构存储数据。数组类型说明由类型说明符、数组名、数组长度(数组元素个数)3 部分组成。数组元素又称为下标变量,同一数组的数组元素是连续分配内存的。数组的类型是指数组元素中值的类型。对数组的赋值可以用数组初始化赋值、输入函数动态赋值和赋值语句赋值三种方法实现。对数值数组不能用赋值语句整体赋值、输入或输出,而必须用循环语句逐个对数组元素进行操作。

习 题 5

1. 单项选择题

(1) 对于定义 int a[10]; 的正确描述是(　　)。

A. 定义一个一维数组 a,共有 a[1]到 a[10]10 个数组元素

B. 定义一个一维数组 a,共有 a(0)到 a(9)10 个数组元素

C. 定义一个一维数组 a,共有 a[0]到 a[9]10 个数组元素

D. 定义一个一维数组 a,共有 a(1)到 a(10)10 个数组元素

(2) 以下数组声明合法的是(　　)。

A. int x(10);　　B. int x[10]　　C. int x[10];　　D. int n,x[n];

(3) 若有定义:

```
double a[] = { 2.1,3.6,9.5};
double b = 6.0;
```

则下列错误的赋值语句是(　　)。

A. b = a[2];　　B. b = a + a[2];

C. a[1] = b;　　D. b = a[0] + 7;

(4) 下列语句中,(　　)定义了一个能存储 20 个字符的数组。

A. int a[21];　　B. char b[20];　　C. char c[21];　　D. int d[20];

(5) 已知函数 isalpha(ch)的功能是判断自变量 ch 是否为字母,若是,则该函数值为 1,否则为 0。以下程序的输出结果是(　　)。

```
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
void fun(char str[ ])
{
  int i,j;
  for (i=0,j=0;str[i];i++)
    if (!isalpha(str[i])) str[j++]=str[i]; /*isalpha()判断是否是字母的函数*/
  str[j]='\0';
}
int main()
{
char str[100]="Current date is Thu 02-12-2008.";
  fun(str);
  printf("%s\n",str);
  system("pause");
  return 0;
}
```

A. 02-12-2008　　B. 02122008

C. Current date is Thu　　D. Current date is Sat 02-12-2008.

(6) 若对数组 a 和数组 b 进行初始化:

```
char a[]="ABCDEF";
char b[]={'A','B','C','D','E','F'};
```

则下列叙述正确的是(　　)。

A. a 与 b 数组完全相同　　B. a 与 b 数组长度相同

C. a 与 b 数组都存放字符串　　D. 数组 a 比数组 b 长度长

(7) 已知下列程序段:

```
char a[3],b[]="Hello";
a=b;
printf("%s",a);
```

则(　　)。

A. 运行后将输出 Hello　　B. 运行后将输出 He

C. 运行后将输出 Hel　　D. 编译出错

(8) 下列程序的运行结果为(　　)。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char a[] = "morning";
    int i,j = 0;
    for(i = 1; i < 7; i++)
      if(a[j]< a[i]) j = i;
    a[j] = a[7];
    puts(a);
    system("pause");
    return 0;
}
```

A. mogninr　　B. mo　　C. morning　　D. mornin

(9) 有以下程序:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int i,t[][3] = {9,8,7,6,5,4,3,2,1};
    for(i = 0;i < 3;i++)
     printf("%d",t[2 - i][i]);
    system("pause");
    return 0;
}
```

程序运行的结果是(　　)。

A. 3 5 7　　B. 7 5 3　　C. 3 6 9　　D. 7 5 1

(10) 下面二维数组定义并初始化中错误的是(　　)。

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

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

C. int x[][3]={{1},{2},{3,4,5}};

D. int x[][3]={1,2,3,4};

(11) 以下程序运行的结果是(　　)。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    char str[][20] = {"One * World","One * Dream!"};
    printf("%d,%s\n",strlen(str[1]),str[1]);
    system("pause");
    return 0;
}
```

A. 10,One * Dream!　　B. 11,One * Dream!

C. 9,One * World　　D. 10,One * World

2. 填空题

(1) 若有 int a[10]；则数组 a 的第一个元素的下标是_____，最后一个元素是_____。

(2) 写出如下数组变量的声明：

① 一个含有 100 个浮点数的数组 realArray _____。

② 一个含有 16 个字符型数据的数组 strArray _____。

③ 一个最多含有 1000 个整型数据的数组 intArray _____。

(3) 在 C 语言中，_____确定某一数据所需的存储字节数。

(4) 设有数组定义：char array [ ]="China"；则数组 array 所占的空间为_____。

(5) 设有数组定义：char a [12]="Nanjing"；则数组 a 所占的空间为_____。

(6) 字符串"ab\n012\\\""的长度为_____。

3. 阅读程序，并写出程序的运行结果

(1) 从键盘输入：

```
aa bb<CR>
cc dd<CR>
```

<CR>表示回车，则下面程序的运行结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char a1[6],a2[6],a3[6],a4[6];
    scanf("%s%s",a1,a2);
    gets(a3); gets(a4);
    puts(a1); puts(a2);
    puts(a3); puts(a4);
    system("pause");
    return 0;
}
```

(2) 从键盘输入：

```
ab<CR>
c<CR>
def<CR>
```

<CR>表示回车，则下面程序的运行结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
#define N 6
int main()
{
   char c[N];
   int i=0;
```

```
    for(;i<N;c[i]=getchar(),i++);
    for(i=0;i<N;i++) putchar(c[i]);
    system("pause");
    return 0;
}
```

(3) 从键盘输入：AhaMA　Aha<CR>(<CR>表示回车)，则下面程序的运行结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char s[80],c='a';
    int i=0;
    scanf("%s",s);
    while(s[i]!='\0')
    {
      if(s[i]==c) s[i]=s[i]-32;
      else if(s[i]==c-32) s[i]=s[i]+32;
      i++;
    }
    puts(s);
    system("pause");
    return 0;
}
```

(4) 从键盘输入18时，下面程序的运行结果是_____。

```
//本程序求输入数的二进制数,结果存放在a数组中
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int x,y,i,a[8],j,u,v;
    scanf("%d",&x);
    y=x;i=0;
    do{
       u=y/2;
       a[i]=y%2;
       i++;y=u;
    }while(y>=1);
    for(j=i-1;j>=0;j--)
    printf("%d",a[j]);
    system("pause");
    return 0;
}
```

(5) 下面程序是将k值按数组中原来的升序找到合适的位置插入，其运行结果是_____。

```
#include<stdio.h>
```

```
#include<stdlib.h>
int main()
{
    int i=1,n=3,j,k=3;
    int a[5]={1,4,5};
    while(i<=n&&k>a[i])i++;
    for(j=n-1;j>=i;j--)
       a[j+1]=a[j];
    a[i]=k;
    for(i=0;i<=n;i++)
       printf("%3d",a[i]);
    system("pause");
    return 0;
}
```

(6) 下面程序的运行结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,j;
    int big[8][8],large[25][12];
    for (i = 0; i<8; i++)
     for (j = 0; j<8; j++)
          big[i][j] = i * j;
    for (i = 0; i<25; i++)
     for (j = 0; j<12; j++)
          large[i][j] = i + j;
    big[2][6] = large[24][10] * 22;
    big[2][2] = 5;
    big[big[2][2]][big[2][2]] = 177;
    for (i = 0; i<8; i++) {
     for (j = 0; j<8; j++)
          printf("%5d",big[i][j]);
          printf("\n");
    }
    system("pause");
    return 0;
}
```

(7) 下面程序运行的结果是_____。

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int main()
{
    char s[4][20]={"JAVA","C#","PHP","Objective-C"};
    int i,k=0;
    for(i=1;i<4;i++)
     if(strcmp(s[k],s[i])<0)
```

```
    k = i;
    puts(s[k]);
    system("pause");
    return 0;
}
```

4. 程序填空题

(1) 以下程序用来检查二维数组是否对称(即：对所有 i,j 都有 a[i][j]=a[j][i]),请填空使程序完整。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a[4][4] = {1,2,3,4,2,2,5,6,3,5,3,7,8,6,7,4};
    int i,j,found = 0;
    for(j = 0; j < 4; j++){
      for(i = 0; i < 4; i++)
        if(_____________)
        {
          found = _____________;
          break;
        }
        if(found) break;
    }
    if(found) printf("不对称\n");
    else printf("对称\n");
    system("pause");
    return 0;
}
```

(2) 以下程序用来输入 5 个整数,并将其存放在数组中；再找出最大数与最小数所在的下标位置,将两者对调；然后输出调整后的 5 个数。请填空使程序完整。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a[5],t,i,maxi,mini;
    for(i = 0; i < 5; i++)
      scanf("%d",&a[i]);
    mini = maxi = _____________;
    for(i = 1; i < 5; i++)
    {
      if(_____________)
         mini = i;
      if(a[i]> a[maxi])
           _____________;
    }
    printf("最小数的位置是:%3d\n",mini);
    printf("最大数的位置是:%3d\n",maxi);
```

```
    t = a[maxi];
    _____;
    a[mini] = t;
    printf("调整后的数为: ");
    for(i = 0; i < 5; i++)
      printf("%d ",a[i]);
    printf("\n");
    system("pause");
    return 0;
}
```

(3) 建立函数 arraycopy(),将数组 a[]的内容复制到数组 b[]中。请填空使程序完整。

```
#include <stdio.h>
#include <stdlib.h>
int arraycopy(_____)
{
    int i = 0;
      while(a[i]!= -999)
      {
      _____;
        i++;
      }
      b[i] = _____;
      return 0;
}
int main()
{
    int a[] = {1,2,3,4,5,6,7,8,9,10, -999};
    int b[100],i = 0;
    _____;
    while(b[i]!= -999)
      printf("%d ",_____);
    system("pause");
    return 0;
}
```

(4) 以下程序将数字字符串转换成数值。请填空使程序完整。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char ch[] = "600";
    int a,s = 0;
    for(a = 0;_____;a++)
      if(ch[a]>= '0'&&ch[a]<= '9')
        s = 10 * s + ch[a] - '0';
    printf("\n%d",s);
    system("pause");
    return 0;
}
```

5. 编程题

(1) 用循环将 a[3][4]的第一行与第三行对调。

```
a
    0  2   9  7        27  11  1  3
    5  13  6  8   →    5   13  6  8
    27 11  1  3        0   2   9  7
```

(2) 编程实现如下形式的数字。

```
1 0 0 0 0 0
2 1 0 0 0 0
3 2 1 0 0 0
4 3 2 1 0 0
5 4 3 2 1 0
6 5 4 3 2 1
```

(3) 编程输出 n 阶左上拐矩阵,如 $n=5$ 时有:

```
1  1  1  1  1
1  2  2  2  2
1  2  3  3  3
1  2  3  4  4
1  2  3  4  5
```

(4) 编程输出如下 n 阶蛇形矩阵,如 $n=5$ 时有:

```
15  7  6  2  1
16  14  8  5  3
22  17  13  9  4
23  21  18  12  10
25  24  20  19  11
```

(5) 输入正整数 m 和 k,编写程序:将大于 m 且紧靠 m 的 k 个非素数,保存在数组中,然后输出。

(6) 编写程序,计算具有 NROWS 行和 NCOLS 列的二维数组中指定列的平均值以及数组各行的和的最小值。

(7) 输入一段文字,统计文字中指定字符的个数。

(8) 假定输入的字符串中只包含字母和 * 号。请编写函数 fun(),它的功能是:除了字符串前导的 * 之外,将字符串中其他 * 号全部删除。在编写函数时,不得使用 C 语言提供的字符串函数。例如,若字符串中的内容为 **** A * BC * DEF * G ******* ,删除后,字符串中的内容则应当是 **** ABCDEFG。

(9) 编写程序,寻找输入字符串中字符 ASCII 码值大的字符,并统计其位置和出现的次数。