第5章 数组

编写程序会涉及数据的处理和保存，在前面的章节已经解决了将单个数据存储到相应类型变量中的问题，但是在一些实际应用中往往需要处理和保存大量的数据。通常，这类数据具有两个特点，一是数量大，二是类型相同。例如，气象站的百叶箱每 30 分钟测一次气温、湿度和风力，每天将产生气温、湿度和风力数据各 48 个；一个开设了 C 语言程序设计课程有 75 名学生的班级，学期末将产生考试成绩 75 个，诸如此类的例子不胜枚举。如果要存储和处理这类数据，用前面的办法就要定义大量名字不同的变量，还要编写大量功能相似甚至完全相同的语句处理这些数据，从而使程序变得相当臃肿。于是 C 语言引入了数组这种重要的数据类型。

5.1 厨师选鱼（一维数组）


视频讲解

厨师要用千岛湖大鱼为客人做一桌美味的全鱼宴，需要从 10 条大鱼中挑选出质量最大的一条鱼，要求程序运行时可以输入 10 条鱼的质量，再找出最大的。

5.1.1 分析与设计

一种简单的方法是将 10 条鱼的质量输入并保存到一个数组中，与此同时通过逐步比较的过程找出其中质量最大的数组成员。

例 5-1 厨师选大鱼（从 10 条大鱼中找质量最大的鱼）。

```
#include "stdio.h"
void main()
{
    float fish[10],max,w;                          /*定义数组*/
    int n=0,max_n=0;
    printf("请输入 10 条鱼的质量:\n");
    while(n<10)                                    /*输入数组成员*/
    {
        scanf("%f",&w);
        if(w<=0)
            printf("数据非法,重来!\n");
        else
        {
```

```
            fish[n] = w;
            n++;
        }
    }
    max = fish[0];                          /* 不妨先假设第一条鱼质量最大 */
    for(n = 1;n < 10;n++)                   /* 找出质量最大成员 */
    {
        if(max < fish[n])
        {
            max = fish[n];
            max_n = n;
        }
    }
    printf("第 %d 条鱼质量最大,鱼质量: %.2f\n",max_n,max);
}
```

运行结果：

```
请输入 10 条鱼的质量:
1
2
0
数据非法,重来!
-3
数据非法,重来!
3
4
5
6
7
8
9
10
第 9 条鱼质量最大,鱼质量: 10.00
```

图 5.1　运行结果

程序运行结果如图 5.1 所示。

用户也可以优化以上程序，改用一个循环语句，控制程序在循环输入数据的同时就将最大数据找出来。

5.1.2　一维数组

1. 一维数组的定义

数组和普通变量是一样的，都要遵循先定义后使用的原则，定义一维数组的一般形式如下：

类型名 数组名[常量表达式]

其中：

(1)“数组名”的命名要符合 C 语言标识符的命名规则。

(2) “类型名”用来指定数组存储数据的类型,可以为 int、float、char 等基本数据类型,也可以为指针或结构体等复合型的数据类型。

(3) “[常量表达式]”用来指定数组的长度,即这个数组能够存储的数据的个数。指定数组长度必须用常量表达式,可以是常量也可以是符号常量,但是不能包含变量。“[ ]”不能省略。

例如:

```
int count[13];
```

这是一个标准的数组定义语句,其中包含的信息有数组的名字为 count;数组的类型为 int 型,即数组的所有成员都是 int 型数据,数组包含 13 个成员。

2. 一维数组的存储

编译器根据变量的定义类型为它们分配相应大小的存储空间。那么数组具体是怎样存储的呢? 为了管理数组中的元素,为每个元素分配一个下标,以数组定义语句 int a[13];为例,数组元素的下标由 0(下标下限)到 12(下标上限),数组中的元素分别为 $a[0]$、$a[1]$、$a[2]$、…、$a[12]$,共计 13 个,编译器在内存中分配一片连续的存储空间来存放这 13 个元素,它们占用的内存空间大小为 4bytes × 13 = 52bytes(当开发环境为 VC++ 6.0 时,每个整型元素占用 4 字节),共 52 字节。这些数组元素在内存中是顺序存放的,如图 5.2 所示。

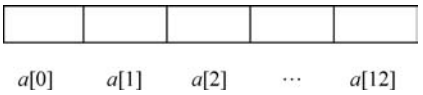

图 5.2 一维数组的存储示意图

从以上分析可知,“int a[13];”中的 13 是指数组中有 13 个元素,因下标是从 0 开始的,所以可引用的最大下标为 12,而不是 13,这一点读者需要特别注意。

3. 一维数组元素的引用

通过一维数组中元素的下标来引用数组元素,这种方法称为下标法。引用形式如下:

数组名[下标值]

其中:

(1) 下标值必须在数组元素下标值范围内,假如数组长度为 N(N 为常量),那么下标值要在$[0,N-1]$才被视为有效引用。

(2) 下标值的形式多样,可以是常量,也可以是变量或者表达式。

例如:

```
int a[100];
```

以下形式都是合法的引用,前提是下标值不超过规定范围:a[0]、a[99]、a[4 + 5]、a[4 * 5]、a[i]、a[i + 3]、a[i + j]。

(3) 一个数组元素实质上就是一个同类型的基本数据类型变量,在使用数组元素时可以把它们看作同类型的变量来使用,因此基本数据类型变量能够进行的操作也适用于同种类型的数组元素,如数组元素可以进行算术运算、逻辑运算、关系运算等。

(4) 数组名不代表任何一个数组元素。例如,a 不代表 $a[0]$~$a[99]$中的任何一个元

素。数组名实质上是一个地址常量，它代表数组的起始地址。

4. 一维数组的初始化

在定义变量时给变量赋初始值称为变量的初始化，同样在定义数组时为数组元素赋初始值就是数组元素的初始化。数组在定义时，系统分配的存储单元内没有确定的数值，通过初始化为数组元素分配确定值。

一维数组的初始化有以下两种方法。

（1）给数组的全部元素赋值，例如：

```
int a[5] = {10,11,12,13,14};
```

该语句把 10～14 分别赋给 $a[0]$～$a[4]$这 5 个数组元素。在给全部元素赋初值的情况下也可以省略数组的长度，例如：

```
int a[ ] = {10,11,12,13,14};
```

这种赋初值的方法，其效果和上面的语句是等同的，编译器会根据赋值的个数来确定数组的长度，但是这种方法只适用于给数组中全部元素赋初值的情况，不能滥用。

（2）给数组的部分元素赋值，例如：

```
int a[5] = {10,11,12};
```

该语句给数组 a 中的部分元素赋初值，分别把 10、11、12 赋给 $a[0]$、$a[1]$、$a[2]$这 3 个元素，那么数组中剩余的两个元素则自动赋初值为 0。

注意：这种初始化方法是在数组定义时进行的，如果定义时不需要初始化，可以在程序运行过程中进行，只是要注意不能使用没有被初始化的数组元素，因为元素值是不确定的。

5. 一维数组元素的输入/输出

因为数组是由若干元素按序排列而成的，所以对数组元素的访问可通过循环语句来实现。

例 5-2 定义一个一维数组，由键盘先输入数据到数组元素中，再把数组元素从头到尾顺序输出到屏幕。

```
#include "stdio.h"
void main()
{
    int a[5];
    int i;
    for(i = 0;i < 5;i++)            /* 用循环控制,向数组中的 a[0]～a[4]输入数值 */
        scanf("%d",&a[i]);
    for(i = 0;i < 5;i++)            /* 用循环控制,把数组中的元素值输出到屏幕 */
        printf("%d ",a[i]);
    printf("\n");
}
```

运行结果：

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

说明：在本例中用循环控制变量来控制数组元素的下标变化，从头至尾按顺序给数组元素赋值，再把数组元素的值按顺序输出到终端。

下面再举一些应用一维数组的例子。

例 5-3 查找并替换一维整数数组中部分元素的值。例如，将数组中元素值为 x 的所有元素替换为 y，其中 x、y 的值通过输入获得。

```
#include "stdio.h"
void main()
{
    int i,x,y,n = 0,a[12] = {5,6,7,8,9,10,11,12,6,13,6,20};
    printf("请输入要查找和修改的数据：");
    scanf("%d%d",&x,&y);
    if(x == y)                    /* 两个数相同就不用换 */
    {
        printf("两个数相同,数组仍然保持原样\n");
        return;
    }
    for(i = 0;i < 12;i++)
    {
        if(a[i] == x)             /* 查找数组元素 */
        {
            a[i] = y;             /* 修改数组元素 */
            n++;
        }
        printf("%d\n",a[i]);
    }
    if(n > 0)
        printf("有%d个元素被置换\n",n);
    else
        printf("数组中没有找到%d\n",x);
}
```

运行结果：

```
请输入要查找和修改的数据：6 -6
5
-6
7
8
9
10
11
12
-6
13
-6
20
有3个元素被置换
```

图 5.3 查找并修改数组的元素

程序运行效果如图 5.3 所示。

例 5-4 比较两个成员数量相同的整型数组是否相等(如果两个数组的成员个数相同，且下标相同的数组成员的数值也相同，则称这两个数组相等)。

```
#include "stdio.h"
#define N 6
#define M 6
void main()
{
    int i,a[N] = {5,6,7,8,9,10},b[M] = {5,6,7,10,11,12};
    if(N!= M)
    {
        printf("这两个数组不相等\n");
        return;
    }
    for(i = 0;i <= 5;i++)
        if(a[i]!= b[i])                    /* 比较相同下标的数组成员 */
            break;                         /* 遇到不相等元素就停止,0≤i≤5 */
    if(i > 5)
        printf("这两个数组相等\n");        /* i > 5 表明所有元素比较完成 */
    else
        printf("这两个数组不相等\n");
}
```

运行结果：

```
这两个数组不相等
```

例 5-5 有一个含有 6 个元素的整型数组，程序要完成以下功能。

(1) 调用 C 语言库函数中的随机函数给所有元素赋以 0～49 的随机数。

(2) 输出数组元素的值。

(3) 对下标为奇数的数组元素求和。

分析：调用 C 语言函数库中的随机函数 rand()可给数组元素赋随机值，该函数会产生一个 0～32767 的随机数，为避免每次运行时产生的随机数序列相同，可使用 srand()函数，每次运行时输入不同的整数就会产生不同的随机数序列，调用这两个函数需包含 stdlib.h 文件。另外要注意对下标的控制，下标初始值应为 1，每次的增值为 2，而不是 1。

```
#include <stdio.h>
#include <stdlib.h>
#define N 6
void main()
{
    int i,a[N],sum = 0,seed;
    printf("input a integer(seed): ");
    scanf("%d",&seed);
    srand(seed);                    /* 新的随机数系列种子,避免每次产生的随机数序列相同 */
    printf("随机产生的数组为: \n");
    for(i = 0;i < N;i++)
    {
        printf("%d\n",a[i] = rand() % 50);          /* 产生 0～49 的随机数 */
```

```
        if(i%2)
            sum+=a[i];
    }
    printf("下标为奇数的数组元素的和为: %d\n",sum);
}
```

运行结果：

```
input a integer(seed): 100
随机产生的数组为:
15
16
15
4
4
4
下标为奇数的数组元素的和为: 24
```

程序运行结果如图 5.4 所示。

图 5.4 下标为奇数的数组元素的和

例 5-6 找出 100～999 中最大的 3 个水仙花数，并求这 3 个数的和。提示：水仙花数是指一个 n 位数($n \geqslant 3$)，它的每位上的数字的 n 次幂之和等于它本身(例如，$1^3 + 5^3 + 3^3 = 153$)。

```
#include "stdio.h"
void main()
{
    int n,f,s,t,a[3],i=0,sum=0;
    printf("100～999 中最大的 3 个水仙花数:\n");
    for(n=999;n>=100;n--)                          /*从大往小找*/
    {
        f=n/100;
        s=(n-f*100)/10;
        t=(n-f*100)-s*10;
        if(f*f*f+s*s*s+t*t*t==n)
        {
            a[i]=n;
            printf("%d\n",a[i]);
            sum+=a[i++];
```

```
        }
        if(i>=3)
            break;
    }
    printf("3个水仙花数的和:%d\n",sum);
}
```

例 5-7 某城市一天中发生交通事故的记录如表5.1所示，按性别统计肇事司机人数和损失。

表 5.1 交通事故的记录

肇事司机	肇事损失(万元)	肇事司机	肇事损失(万元)
女	5.2	女	0.5
男	10.5	男	1.2
男	13.0	男	0.03
女	2.5	男	3.2
男	0.7	女	0.05

```
#include<stdio.h>
void main()
{
    char gender[10]={'f','m','m','f','m','f','m','m','m','f'};      /*定义数组*/
    float m[10]={5.2,10.5,13.0,2.5,0.7,0.5,1.2,0.03,3.2,0.05},fm=0,mm=0;
    int n,fn=0,mn=0;
    for(n=0;n<10;n++)
    {
        if(gender[n]=='f')
        {
            fm+=m[n];
            fn++;
        }
        else
        {
            mm+=m[n];
            mn++;
        }
    }
    printf("女司机肇事%d起,损失: %.2f\n",fn,fm);
    printf("男司机肇事%d起,损失: %.2f\n",mn,mm);
}
```

运行结果：

```
女司机肇事4起,损失: 8.25
男司机肇事6起,损失: 28.63
```

例 5-8 水果罐头生产线加工一批产品，共计100瓶，要求每瓶罐头的标准质量为500g，因工艺的原因工人在装料时可能未按产品标准装料(多装或少装)。为保证质量，检验员必须视情况为不合格的罐头减少或添加物料以达到标准要求。编写程序，输入每

瓶罐头的现有质量数据，然后将该数据与标准质量数据进行比较，提示该瓶罐头需要添加或减少物料的数量，当这批罐头数据输入完毕后给出总共为这批罐头添加或减少了多少物料。

```
#include <stdio.h>
#define STD 500                                    /*产品标准质量 500g*/
void main()
{
    float pd[100],sum1 = 0,sum2 = 0;               /*设置产品数据数组和物料累计变量*/
    int n;
    for(n = 0;n < 100;n++)
    {
        printf("输入罐头质量:");
        scanf("%f",&pd[n]);
        if(STD - pd[n]> 0)
        {
            printf("本件产品物料不足,少装: %.2f\n",STD - pd[n]);
            sum1 += STD - pd[n];
        }
        else
        {
            if(STD == pd[n])
                printf("本件产品物料质量符合标准\n");
            else
            {
                printf("本件产品物料超量,多装: %.2f\n",pd[n] - STD);
                sum2 += pd[n] - STD;
            }
        }
    }
    if(sum2 - sum1 > 0)
        printf("这批产品总共添加物料:%.2f\n",sum2 - sum1);
    else
        printf("这批产品总共减少物料:%.2f\n",sum1 - sum2);
}
```

5.1.3　实战演练

将一个长度为 N 的一维数组中的元素按颠倒的顺序重新存放，注意操作时只能借助一个临时变量而不得另外开辟数组。

分析：题目要求不是逆序打印数据，而是要变换数值的存储位置。例如，原数组如图 5.5 所示，变换后的数组如图 5.6 所示。

1	3	5	7	9

图 5.5　原数组

9	7	5	3	1

图 5.6　变换后的数组

虽然数组元素还在同一个数组中，但元素的位置发生了变化。解决的方法就是把数组中前后对应位置的元素相互调换。可以用 i、j 两个变量记录要交换的两个数组元素的下标，i 的初始值是 0，j 的初始值是最后一个元素的下标 $N-1$。

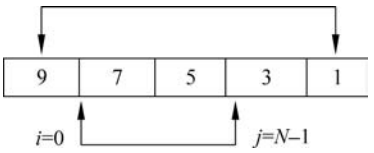

图 5.7　数组元素交换的示意图

交换过程如图 5.7 所示。

对应位置的数组元素每交换一次，i 的数值加 1，指向后一个元素；j 的数值减 1，指向前一个元素，然后继续交换，直到 i 和 j 的值不满足条件 $i<j$ 为止。该程序使用循环实现。程序流程图如图 5.8 所示，请补全下面的程序并运行得出结果。

```
#include<stdio.h>
#define N 5
main()
{
    int a[N]={1,3,5,7,9};
    int i,j,temp;
    printf("原数组:\n");
    for(i=0;i<N;i++)
        printf("%d",a[i]);
    printf("\n");
    for(;;)                          /*调换数组元素*/
    {
        ________;
        ________;
        ________;
    }
    printf("调换后数组:\n");
    for(i=0;i<N;i++)
        printf("%d",a[i]);
    printf("\n");
}
```

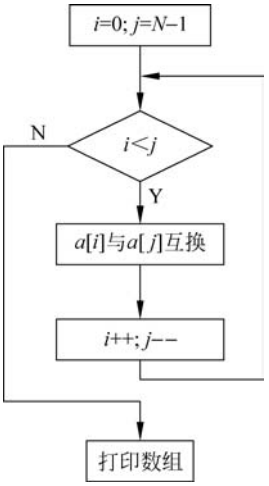

图 5.8　程序流程图


视频讲解

5.2　果园里的竞赛（二维数组）

某果园举行摘果子比赛，参赛选手有 3 名，比赛规则是在规定时间内摘 3 种水果——葡萄、鸭梨和桃子，摘得果子总质量最多的人赢得比赛。3 人的比赛成绩表如表 5.2 所示，请编写一个程序来计算谁是冠军。

表 5.2　摘果子比赛成绩表

选手编号	葡萄(kg)	鸭梨(kg)	桃子(kg)	总质量(kg)
1	57	68	40	
2	60	83	72	
3	40	56	69	

5.2.1 分析与设计

首先要将表5.2中的15个数据保存到数组中，分别是3位选手的序号、所摘的每种水果的质量和3种水果的总质量。这个数组显然与一维数组不同，因为如果用某种手段能将此表逻辑上映射到存储器中，那么每个数据的位置要由它所在的行和列来确定，这与一维数组存储的数据只需一个下标就能确定元素位置不同。下面采用一种新的数据类型——二维数组来保存这些数据。这里定义一个3行5列的二维数组存储数据，它们在内存中的抽象状态（实际保存状况不是这样）如图5.9所示，每一行存储一个选手的所有数据，每一列表示所有人的同样的数据，分别是所有参赛选手的编号、所有选手摘的同种水果的质量，以及所有选手摘水果的总质量。

	第0列	第1列	第2列	第3列	第4列
第0列	1	57	68	40	0
第1行	2	60	83	72	0
第2行	3	40	56	69	0

图5.9 二维数组的数据存储示意图

如果要计算出冠军得主，只需要对二维数组的每行从列号为1的元素到列号为3的元素求和，存放到列号为4的元素中，再通过比较第4列的元素的值就可以得出谁是冠军了。

例 5-9 果园里的竞赛（二维数组的应用）。

```
#include <stdio.h>
void main()
{
    int i,j,id,max = 0;
    //定义二维数组保存水果质量,并初始化
    int s[3][5] = { {1,57,68,40,0}, {2,60,83,72,0},{3,40,56,69,0} };
    printf(" NO. GRAPE PEAR PEACH TOTAL\n");
    for(i = 0;i < 3;i++)
    {
        printf(" %4d ",s[i][0]);            /*输出参赛选手的编号*/
        for(j = 1;j < 4;j++)
        {
            s[i][4] += s[i][j];             /*求第i个选手摘得水果的总质量*/
            printf(" %6d",s[i][j]);         /*输出各种水果的质量*/
        }
        printf(" %8d\n",s[i][4]);           /*输出第i个选手摘得水果的总质量*/
        if(s[i][4]> max)
        {
            id = s[i][0];
            max = s[i][4];
        }
    }
    printf("冠军编号: %4d,摘水果的总质量: %8d kg\n",id,max);
}
```

运行结果：

```
NO. GRAPE      PEAR     PEACH    TOTAL
1     57       68       40       165
2     60       83       72       215
3     40       56       69       169
冠军编号:     2,摘水果的总质量:     215kg
```

在本例中采用二维数组来存储数据，这些数据有以下特点：由行属性和列属性描述位置。在二维数组的行和列上进行相应的运算可以得出有特定意义的数据。例如，计算每个选手的总成绩是在每一行上取第 1、2、3 列元素求和。如果要求出所有人采摘某种水果的总质量，应该如何做呢？

5.2.2 二维数组

1. 二维数组的定义

从以上例子可以知道，二维数组的定义类似一维数组，但是要在一维数组的基础上增加一个维度。在 C 语言中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，二维数组名的命名规则、“类型名”与一维数组相同，不同的是“[常量表达式 1]”用来指定二维数组的行数；“[常量表达式 2]”用来指定二维数组的列数。在指定数组的行数和列数时，可以是常量（整型、字符型），也可以是符号常量或常量表达式，但是不能包含变量。与一维数组相比，二维数组多了一个维度的定义——[常量表达式 2]，使二维数组可以存储矩阵和二维表数据（如例 5-9 所示）。

例如：

```
int s[3][5];
```

这是一个二维数组定义语句，包含的信息如下：

数组的名字为 s，数组的类型为 int 型，是整型数组。数组为 3 行 5 列的二维数组，数组 s 能够存储的 int 型数据应该有 $3\times5=15$ 个，该数组的逻辑结构可视为 3 行 5 列的矩阵或二维表格，但要注意数组元素的类型必须一致，否则不能定义成数组，其形式如图 5.10 所示。

	第0列	第1列	第2列	第3列	第4列
第0行	s[0][0]	s[0][1]	s[0][2]	s[0][3]	s[0][4]
第1行	s[1][0]	s[1][1]	s[1][2]	s[1][3]	s[1][4]
第2行	s[2][0]	s[2][1]	s[2][2]	s[2][3]	s[2][4]

图 5.10 数组 s 的逻辑结构

一维数组的存储是按元素顺序依次存放在内存中的。二维数组的存储也类似，按行顺序依次存储数组元素。二维数组的每个元素有两个下标，分别代表元素所在的行和列，下标值的范围规定与一维数组类似，这里以数组 s 为例，下标值的取值范围分别为 0～2 行和

0～4 列。数组 s 在内存中的存储形式如图 5.11 所示。

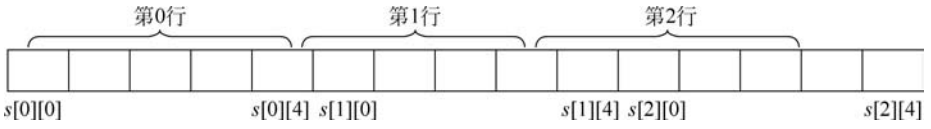

图 5.11　数组 s 在内存中的存储

2. 二维数组元素的引用

二维数组元素的引用也采用下标法，具体引用形式如下：

数组名[下标值 1][下标值 2]

其引用的规则同一维数组。例如，定义一个数组：

```
int a[20][50];
```

则以下形式都是合法的引用：

```
a[0][0]、a[19][49]、a[4+5][8]、a[4*4][0]、a[i][j]、a[i+3][j*4]、a[i+j][5%2]
```

以下形式为非法的引用：

```
a[20][50]、a(3)(4)、a{m}{n}、a[2,3]
```

读者能否说出这些引用存在哪种语法错误？和一维数组名类似，二维数组名不代表任何一个数组元素，实质上是一个地址常量。例如，a 不代表 $a[0][0]$～$a[19][49]$中的任何一个元素，它代表数组的起始地址。

通常，二维数组还可以看作特殊的一维数组，如图 5.12 所示。

a	$a[0]$	$a[0][0]$	$a[0][1]$	$a[0][2]$	$a[0][3]$	$a[0][4]$
	$a[1]$	$a[1][0]$	$a[1][1]$	$a[1][2]$	$a[1][3]$	$a[1][4]$
	$a[2]$	$a[2][0]$	$a[2][1]$	$a[2][2]$	$a[2][3]$	$a[2][4]$

图 5.12　特殊的一维数组

把 a 看作一个一维数组，它有 3 个元素，即 $a[0]$、$a[1]$、$a[2]$，其中每个元素又是一个有 5 个元素的一维数组。

C 语言还支持多维数组，读者在二维数组的基础上可以理解多维数组。例如，定义一个三维数组：

```
int threeDim[4][5][6];
```

3. 二维数组的初始化

二维数组在定义的同时也可以对数组元素进行初始化。二维数组的初始化方式主要有以下两种。

1）对数组的全部元素赋初始值

（1）整体赋初始值：将所有数据写在一对大括号内，按照它们的排列顺序给各元素赋值。

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

（2）分行赋初始值：以行为单位将数据用一对大括号括起来，最外层再用一对大括号把所有数据括起来。这种赋值方式更加直观。

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

在对全部元素赋初始值时，数组的第一维长度是可以省略的。例如：

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

可写成：

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

2）对数组的部分元素赋初始值

对数组中各行的某些元素赋值：

```
int a[3][4] = {{1},{5},{9}};
int a[3][4] = {{1},{0,6},{0,0,0,11}};
int a[3][4] = {{1},{5,6}};
```

以上 3 条语句的赋值结果如图 5.13 所示。

1	0	0	0
5	0	0	0
9	0	0	0

1	0	0	0
0	6	0	0
0	0	0	11

1	0	0	0
5	6	0	0
0	0	0	0

图 5.13　二维数组的赋值结果

部分赋值是对数组中各行对应位置的元素赋初始值，没有指定值的元素由系统自动赋值。

在对部分元素赋值时同样可以省略数组的第一维长度的指定，例如：

```
int a[3][4] = {{1},{0,6},{0,0,11}};
```

可写成：

```
int a[][4] = {{1},{0,6},{0,0,11}};
```

数组的长度被自动确定为 3，再如：

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

此时，数组的第一维长度的确定方法是用数值的总数和第二维长度之商作为依据，若能整除，该商即为第一维长度（即为数组的所有元素赋值时省略第一维长度的情况）；若不能整除，该数组的第一维长度为商值加 1。所以本例数组的第一维长度为 3。

注意：在定义二维数组和赋值时，不管是哪种情况，第二维长度都不能够省略。下面的

赋值语句是错误的。

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

4. 二维数组的输入/输出

一维数组用一个循环语句控制数组元素的输入/输出，二维数组有行和列，所以要用二重循环来访问。

例 5-10 定义一个二维数组，由键盘输入数据为数组元素赋值，再把数组元素值输出到屏幕。

```
#include <stdio.h>
main()
{
    int a[3][2];
    int i,j;
    for(i = 0;i < 3;i++)                    /* 给数组元素输入数据 */
        for(j = 0;j < 2;j++)
            scanf("%d",&a[i][j]);
    for(i = 0;i < 3;i++)                    /* 将数组元素的值输出 */
    {
        for(j = 0;j < 2;j++)
            printf("%d ",a[i][j]);
        printf("\n");                       /* 每输出数组的一行元素，数据就换下一行 */
    }
    printf("\n");
}
```

运行结果：

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

说明：在本例中用循环控制数组的下标，依次给数组元素赋值或把数组元素值输出到终端。由于二维数组有行和列两个下标，要遍历二维数组中的每个元素需要采用二重循环，一般外层循环控制行，内层循环控制列。

下面再举一些应用二维数组的例子。

例 5-11 将表 5.3 中小于 0 的数据显示出来，其他数据用 0 代替。

表 5.3 实数表

20.0	−3.5	−6.0
−10.0	2.2	50.0
−9.1	71.0	−20.0
3.0	8.0	−11.0

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

```
void main()
{
    float a[4][3] = {{20.0, -3.5, -6.0},{-10.0,2.2,50.0},{-9.1,71.0,-20.0},{3.0,8.0,
-11.0}};
    for(int i = 0;i < 4;i++)
    {
        for(int j = 0;j < 3;j++)
            if(a[i][j]< 0)
                printf("%5.1f ",a[i][j]);
            else
                printf("%5.1f ",0);
        printf("\n");
    }
}
```

运行结果：

```
  0.0  -3.5  -6.0
-10.0   0.0   0.0
 -9.1   0.0 -20.0
  0.0   0.0 -11.0
```

程序运行结果如图 5.14 所示。

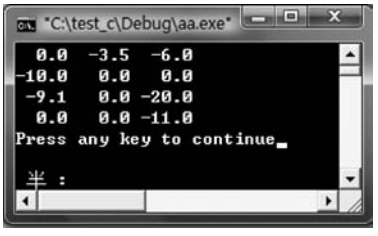

图 5.14　显示负实数

例 5-12　某乡鼓励农民通过办农家乐来增加收入，因此全乡农家乐已经发展到 10 家，各家的经济来源为住宿与餐饮、种植与养殖两项。年终到了，请编写程序帮乡里找出总收入最高的农户，乡里要给予奖励。

```
#include<stdio.h>
void main()
{
    float a[10][2],max;                    /*10 家农家乐两项经济收入*/
    int i,m;
    for(i = 0;i < 10;i++)
    {
        printf("输入第%d家的收入状况\n",i);
        printf("住宿与餐饮: ");
        scanf("%f",&a[i][0]);
        printf("种植与养殖: ");
        scanf("%f",&a[i][1]);
    }
    m = 0;
    max = a[m][0] + a[m][1];
    for(i = 1;i < 10;i++)
    {
        if(max < a[i][0] + a[i][1])
        {
            max = a[i][0] + a[i][1];
            m = i;
        }
```

```
    }
    printf("第%d家总收入最高,已经达到: %.2f\n", m,a[m][0]+a[m][1]);
}
```

思考：请考虑用一个循环语句实现的方案，即在输入数据的同时找出最大者，另外要排除输入的非法数据该如何处理。

例 5-13　老师布置给道路桥梁工程专业学生的实习任务是测量学校旁一段公路的长度，学生们沿着公路测量，获得如表 5.4 所示的公路的经纬度数据。学生们利用两点间距离公式设计程序来处理这些数据，并估算公路的长度。

表 5.4　公路的经纬度数据

经度	10.57	10.578	10.5792	10.5799	10.58	10.583	10.5785	10.57862	10.5787
纬度	35.775	35.76	35.77	35.771	35.775	35.78	35.782	35.783	35.785

分析：设置二维数组保存各测量点数据，计算各相邻点间的距离并求和，最后算出公路的长度。

```
#include<stdio.h>
#include<math.h>
void main()
{
    float m_xy[10][2];                              /*保存测量数据的二维数组*/
    float roadlength=0,dis;                         /*公路长度*/
    int n;
    printf("输入测量数据(经度和纬度):\n");
    for(n=0;n<10;n++)
    {
        scanf("%f%f",&m_xy[n][0] ,&m_xy[n][1]); /*输入第n个测量点数据*/
        /*计算相邻两点距离*/
        dis=sqrt(pow(m_xy[n][0]-m_xy[n-1][0],2)+pow(m_xy[n][1]-m_xy[n-1][1],2));
        roadlength+=dis;                            /*公路长度求和*/
    }
    printf("这段路长度%.6f: \n",roadlength);
}
```

5.2.3　实战演练

(1) 2015 年国际田联钻石联赛有 14 场比赛，美国队 3 名短跑运动员 A、B、C 参加了全部赛事，教练员将用程序把运动员每场比赛的最好成绩记录下来，最后将 3 人在 14 场比赛中的最好成绩交给体育记者做新闻报道，请完成该程序。

提示：设置二维数组记录 3 人 14 场的最好成绩，同时比较每个运动员每场比赛的成绩，找出 14 场中的最好成绩。

(2) 求 3×4 矩阵 $\boldsymbol{a}$ 的所有外围元素之和。

提示：3×4 矩阵 $\boldsymbol{a}$ 的外围元素如图 5.15 所示，图中加底纹标识的元素是 $\boldsymbol{a}$ 的外围元素。设 i 代表矩阵的行下标、j 代表矩阵的列下标，可以看出外围元素的下标有以下特点：

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

图 5.15　矩阵 ***a*** 的外围元素

① 当 $i=0$ 时，j 为 0～3 中的任何整数；

② 当 $i=1$ 时，j 为 0 或者 3 中的一个整数；

③ 当 $i=2$ 时，j 为 0～3 中的任何整数。

只要 $i=0$ 和 2(即起始行和终结行的下标)，该元素就确定是外围元素；$j=0$ 和 3(即起始列和终结列的下标)，该元素也可以确定为外围元素，可以利用逻辑表达式 `i == 0 || i == 2 || j == 0 || j == 3` 表示该条件。对矩阵中的任意元素的下标 i 和 j 的值进行判断，满足该条件的元素即为外围元素。请补全下面的程序，观察运行结果。

考虑为什么 `i == 0 || i == 2 || j == 0 || j == 3` 可以作为判断外围元素的条件，是否还有其他的解决方法？如果是 $M\times N$ 的矩阵应该怎样改写程序？

程序如下，请填空。

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


视频讲解

5.3　古诗词填空(字符数组)

古诗词填空，编写一个程序让用户将诗句“楼船夜雪瓜州渡，____秋风大散关”中所缺的两个字填上。

5.3.1　分析与设计

在计算机中保存一个汉字一般需要 2 字节，在 C 语言中保存字符串用字符数组。因此，要解决以上问题可以先将整个诗句保存到一个一维字符数组中，当然所缺的汉字要占用的空间应先预留，这里缺两个汉字所以要预留 4 字节，以上诗句有 15 个汉字(“,”也占用 2 字节)要怎样处理呢？一是安排足够的空间，建立字符数组，并保存诗句；二是由用户输入所缺的汉字；三是将汉字数据填入数组中的相应位置；四是输出整个诗句。

例 5-14　古诗词填空(字符数组)。

```
#include<stdio.h>
void main()
{
    char stent[]={"楼船夜雪瓜州渡,  秋风大散关"}; /*注意预留两个汉字的空间*/
    char tmp[5];                                  /*定义字符数组存储输入的缺字*/
```

```
    printf("%s\n",stent);
    printf("输入诗句中的缺字:");
    scanf("%s",tmp);                                    /*输入两个缺字*/
    stent[16] = tmp[0];                                 /*填写第一个缺字*/
    stent[17] = tmp[1];
    stent[18] = tmp[2];                                 /*填写第二个缺字*/
    stent[19] = tmp[3];
    printf("%s\n",stent);
}
```

运行结果：

```
楼船夜雪瓜州渡，  秋风大散关
输入诗句中的缺字:铁马
楼船夜雪瓜州渡,铁马秋风大散关
```

程序运行结果如图 5.16 所示。

图 5.16　程序运行结果

读者也可以用字符串处理函数更简便地实现本例，在后面将介绍一些字符串处理函数。

5.3.2　字符数组

字符数组即用于存储字符的数组，在 C 语言中用字符数组来处理字符串。与前面介绍的数组类似，字符数组中的每个元素都可以看作一个字符型变量。字符以 ASCII 码的形式存放在数组元素中。在第 1 章介绍过字符串常量，字符串是一种字符数组，并且其最后一个单元是'\0'(结束符)，即字符串是一种以'\0'结尾的字符数组。结束符的作用是标识字符串的结束。

1. 字符数组的定义

字符数组的定义方法与一维数组类似。如例 5-14 中的语句：

```
char string[100];
```

其含义为定义一个一维字符数组 string，它具有一维数组的性质。数组名遵循 C 语言的标识符命名规则。该数组长度为 100，即可以存储 100 个字符。

字符数组元素同样采用下标法引用，引用范围为 string[0]～string[99]。string[i]即为对字符数组中下标为 i 的元素的引用。

字符数组在内存中的存储是连续的，它占用一段连续的内存空间，且数组元素按顺序依次存储在内存中。例如，程序段：

```
char ch[5];                                             /*定义字符数组 ch*/
ch[0] = 'h',ch[1] = 'a', ch[2] = 'p',ch[3] = 'p', ch[4] = 'y' ;     /*给数组 ch 赋值*/
```

在内存中的存储如图 5.17 所示。

也可以定义二维字符数组：

```
char diamond[3][5] = {{' ',' ','*'},{' ','*',' ','*'},{'*',' ',' ',' ','*'}};
```

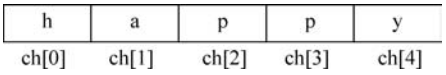

图 5.17　ch 数组的存储示意图

diamond 数组为一个二维字符数组，存储一个字符矩阵，其定义和初始化方法和二维数组一致，请试着画出这个字符矩阵的形状。

2. 字符数组和整型数组的关系

前面介绍过字符型数据和整型数据存在一定的关联，字符型数据在计算机中是以整型数据来存储的，即字符的 ASCII 码，所以在一定范围内字符型数据和整型数据是可以通用的。依据这一点，也可以定义一个整型数组来存放字符型数据，例如：

```
int ch[10];
ch[0] = 'a';
```

上面这样处理是可以的，但是会浪费存储空间，因为存储整型数据占用的字节数多于字符型数据。

3. 字符数组元素的引用

字符数组元素的引用可以通过下标法来实现，如字符数组元素的引用语句：

```
c = string[i];
```

该语句的含义为把数组 string 中的第 i 个元素赋给字符型变量 c。

例 5-15　输出一个字符串。

分析：定义一个字符数组，给字符数组存入数值，然后输出该字符数组。

```
#include <stdio.h>
void main()
{
    char string[10] = {'I',' ','a','m',' ','a',' ','b','o','y'};
    int i;
    for(i = 0;i < 10;i++)
        printf("%c",string[i]);
    printf("\n");
}
```

运行结果：

```
I am a boy
```

4. 字符数组的初始化

在初始化字符数组时，注意存入字符数组中的字符个数或字符串的长度要小于字符数组的长度，对字符数组初始化主要有以下两种方法。

1）逐个元素赋值

```
char string[10] = {'I',' ','a','m',' ','a',' ','b','o','y'};
```

即在定义数组 string 时对该数组中的元素赋初始值，大括号中的字符按照顺序依次存入字符数组中。由于字符数组元素为字符型变量，所以在赋值时元素值要使用单引号，以代表该值是一个字符型常量。各元素值之间用“,”隔开。

2）整体赋值

整体赋值是直接把一个字符串常量赋给字符数组。例如：

```
char string[12] = "I am a boy";
```

或者

```
char string[12] = { "I am a boy"};
```

注意：在采用整体赋值方式赋值时，实际能够赋给字符数组的字符的个数要比字符数组能够容纳的字符个数少一个，这也正是例 5-14 中留给大家的问题，到底是为什么呢？因为 C 语言在处理字符串时会在每个字符串结束的位置加一个结束标志'\0'，也就是"I am a boy"的真正面目是"I am a boy\0"，这个标志在显示字符串的时候不显示出来，但的确是字符串的一部分，所以当字符串被存入字符数组时'\0'占用一个字符的存储空间，因此"I am a boy"存放到数组中后内存如图 5.18 所示。

I		a	m		a		b	o	y	\0	\0

图 5.18 字符串"I am a boy"的内存存储示意图

图 5.18 中第一个'\0'是字符串结束标志，第二个'\0'是由于字符串长度小于数组的长度，最后还空余一个字符的空间，所以最后一个字符空间的值也被自动设定为'\0'。

5. 字符数组的输入/输出

下面介绍几种常用的字符数组输入/输出方法。

1）字符数组的输出

（1）逐个字符输出：在格式输出函数中用格式符"%c"输出一个字符。例如：

```
char string[50] = {"I am a student"};
for(i = 0;i < strlen(string);i++)
    printf("%c",string[i]);
```

与一维数组元素的输出方法相同，用循环语句依次输出字符串中的字符。strlen()函数是计算字符串实际长度的函数，将在后面介绍。

（2）将整个字符串一次输出：在格式输出函数中用格式符"%s"，意思是对字符串的输出。例如：

```
char string[50] = { "I am a student"};
printf("%s",string);
```

这种方法会输出数组 string 中的字符串，直到遇见结束符'\0'，但不输出'\0'。这种输

出的好处是只输出字符串的有效字符"I am a student"，而不是输出字符数组的 50 个字符。如果一个字符数组中包含多个'\0'，则遇到第一个'\0'时输出就结束。

2）字符数组的输入

利用 scanf()函数输入一个字符串，存放在指定的字符数组中。例如：

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

这种输入方法实际上存入数组中的字符还有一个'\0'。

注意：对于包含空格的字符串，在使用 scanf()函数时要把空格隔开的部分单独进行字符串处理。例如，下列语句：

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

从键盘输入：

```
I am a student
```

输出的结果：

```
I
```

结果并没有出现输入的字符串，而是只有字符串的第一个单词“I”，这是因为系统把空格符作为输入的字符串之间的分隔符，因此只将空格前的字符"I"存入数组 string 中，在"I"的后面加'\0'。数组的存储状态如图 5.19 所示。

I	\0	\0	\0	\0	\0	\0	\0

图 5.19　只存入空格前的单词的数组状态

把程序改变一下：

```
char str1[8],str2[8],str3[8],str4[8];
scanf("%s%s%s%s",str1,str2,str3,str4);
printf("%s %s %s %s",str1,str2,str3,str4);
```

从键盘输入：I am a student

输出的结果：I am a student

数组的状态如图 5.20 所示。

I	\0	\0	\0	\0	\0	\0	\0
a	m	\0	\0	\0	\0	\0	\0
a	\0	\0	\0	\0	\0	\0	\0
s	t	u	d	e	n	t	\0

图 5.20　4 个单词存入 4 个字符数组的状态

在使用 scanf()和 printf()函数以字符串的形式输入或输出字符数组值时只需要提供

字符数组名即可，下面的用法是错误的：

```
scanf("%s",&string);
```

因为字符数组名本来就代表了字符数组的起始地址，所以不需要再一次取地址。

3）利用其他字符串输入/输出函数对数组进行输入/输出

因为 scanf()函数不能输入带空格的字符串，所以采用 scanf()函数对于带有空格的字符串的输入操作是不方便的，而利用字符串输入函数就可以输入带空格的字符串。

（1）字符串输入函数 gets()。

其一般形式如下：

```
gets(字符数组名)
```

例如：

```
char string[50];
gets(string);
```

程序运行时可从键盘输入：

```
I am a student
```

整个字符串被送入 string 字符数组中进行存储，不受空格符号的限制。

（2）字符串输出函数 puts()。

其一般形式如下：

```
puts(字符数组名)
```

例如：

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

程序运行时可从键盘输入：

```
I am a student
```

输出：

```
I am a student
```

注意：在使用 gets()、puts()函数时，函数参数只需提供字符数组名即可。

5.3.3 字符串处理函数

在 C 语言的函数库中提供了一些专门处理字符串的函数，使用十分方便，下面介绍几种常用的函数，在使用前要加语句“#include <string.h>”说明。

1. strlen()函数

该函数用来测试字符串长度。其一般形式如下：

strlen(字符数组)

其中：

(1) 该参数会返回一个整数，数值即为字符串的实际长度值(不包含'\0'在内)。

(2) 参数可以写成字符数组名，也可以是一个字符串常量。例如：

```
int len1,len2;
char str = {"I love China!"};
len1 = strlen(str);
len2 = strlen("I love China!");
```

变量 len1、len2 的数值为 13。

2. strcmp()函数

该函数用于比较两个字符串的大小。其一般形式如下：

strcmp(字符串 1,字符串 2)

其中：

(1) 两个参数“字符串 1”和“字符串 2”既可以是字符数组名也可以是字符串常量。例如：

```
strcmp(str1,str2);                    /*字符数组 str1 和 str2 中的字符串进行比较*/
strcmp(str1, "C program");            /*字符数组 str1 和"C program"进行比较*/
strcmp("China","USA");                /*"China"和"USA"进行比较*/
```

(2) 该函数的返回值为整型，通过判断返回值来确定两个字符串的大小关系。

(3) 两个字符串的比较结果与返回值的关系如表 5.5 所示。

表 5.5 strcmp()函数的返回结果

字符串间的关系	函数的返回值
字符串 1 等于字符串 2	0
字符串 1 大于字符串 2	正数
字符串 1 小于字符串 2	负数

(4) 字符串的比较规则。

对两个字符串自左至右逐个字符相比(按 ASCII 码值大小比较)，直到出现不同的字符或遇到'\0'为止。如全部字符相同，则认为相等；若出现不同的字符，则以第一个不同的字符的 ASCII 码值比较结果为准。例如：

"computer"大于"compare"、"a"大于"A"、"DOG"小于"cat"、"12+36"大于"!$&#"

比较两个字符串必须使用 strcmp()函数。例如：

```
if(strcmp(str1,str2)>0)
    printf("yes\n")
else
    printf("no\n");
```

不能写成：

```
if(str1 > str2)
    printf("yes\n")
else
    printf("no\n");
```

这种方式是错误的,这是比较两个字符串的首地址。两个字符串的比较一定要通过调用函数实现,不能直接用关系运算来比较大小。

3. strcat()函数

该函数用于连接两个字符串。其一般形式如下:

strcat(字符数组 1,字符数组 2)

其中:

(1) 把字符串 2 连接到字符串 1 的后面,结果存放在字符数组 1 中,函数会有一个返回值(字符数组 1 的地址)。**字符数组 1 必须写成数组名**,字符数组 2 可以为数组名也可以为字符串常量。

(2) 字符数组 1 的长度必须足够大,以便容纳连接后的字符串。

(3) 在连接前两个字符串的后面都有'\0',连接后去掉了第一个字符串的'\0',在新串后面保留'\0'。

例如:

```
char str1[30] = "I love ";
char str2[] = "C program! ";
strcat(str1,str2);                          /* 连接两个字符串 str1、str2 */
puts(str1);
```

输出:

```
I love C program!
```

4. strcpy()函数和 strncpy()函数

这两个函数是字符串复制函数。strcpy()函数的一般形式如下:

strcpy(字符数组 1,字符数组 2)

把字符数组 2 的内容复制到字符数组 1 中。

strncpy()函数的一般形式如下:

strncpy(字符数组 1,字符数组 2,n)

把字符数组 2 的前 n 个字符复制到字符数组 1 中。

其中:

(1) 字符数组 1 必须足够大,以便容纳复制的字符串。

(2) 函数的返回值为字符数组 1 的地址。

(3) 字符数组 1 要写成数组名的形式,字符数组 2 可以是数组名也可以是一个字符串常量。例如:

```
strcpy(str1,str2);
strncpy(str,"China",3);
```

strcpy()函数把数组 2 中的字符串和它后面的'\0'一起复制到数组 1 中。

例如：

```
char str1[8] = "Chinese";
char str2[] = "USA";
strcpy(str1,str2);
puts(str1);
```

输出结果：

```
USA
```

复制之后 str1 的存储内容如图 5.21 所示。

U	S	A	\0	e	s	e	\0

图 5.21　使用 strcpy()复制 str2 后 str1 的存储内容

当输出数组 1 时，遇到第一个'\0'结束输出，所以后面的字符不再显示，但并不代表数组 1 中原来的字符不存在了，逐个输出数组中的元素，就可以看到复制后的数组中的全部内容。strncpy()函数把数组 2 中的前 n 个字符复制到数组 1 中，不另外附加'\0'。例如：

```
char str1[8] = "Chinese";
char str2[] = "USA";
strncpy(str1,str2,2);
puts(str1);
```

输出结果：

```
USinese
```

复制后数组 1 的状态如图 5.22 所示。

U	S	i	n	e	s	e	\0

图 5.22　使用 strncpy()函数复制 str2 后 str1 的存储内容

注意：不能将一个字符串常量或者字符数组直接赋给一个字符数组。以下语句是错误的：

```
str1 = "student";
str1 = str2;
```

例 5-16　将“aaaaBBccccBBBddddddBBeeeeeefffBB”中的“BB”替换成“AA”，即使其变为“aaaaAAccccBBBddddddAAeeeeeefffAA”。

```
#include<stdio.h>
#include<string.h>
void main()
{
```

```
    char strs[80] = "aaaaBBccccBBBddddddBBeeeeeefffBB",strd[] = "BB";
    int i,n = 0;
    for(i = 0;i <= strlen(strs);i++)
    {
        if(strs[i] == 'B')
            n++;
        else
        {
            if(n == strlen(strd))
            {
                strs[i - n] = 'A';
                strs[i - n + 1] = 'A';
            }
            n = 0;
        }
    }
    puts(strs);
}
```

运行结果：

```
aaaaAAccccBBBddddddAAeeeeeefffAA
```

程序运行结果如图 5.23 所示。

图 5.23　替换结果

5.3.4　实战演练

(1) 以下是一个包含 0 元素的 5×5 方阵，编程序找出“行相邻、列相同”的 3 个 0 元素(已用底纹以示突出)，如图 5.24 所示。

1	1	0	0	1
0	1	1	1	0
0	0	0	1	1
1	0	0	1	0
0	0	1	0	0

图 5.24　行中有连续 0 的矩阵

(2) 输入一行最多包含 80 个字符的英语句子，统计其中大写字母、小写字母、数字、空格和其他字符的数量。程序如下，请填空，并观察运行结果。

提示：对输入的每个字符逐一判断，看属于哪种字符。为每种字符设置一个计数器，记录它们出现的次数(字符出现时对应的计数器加 1)。判断字符的依据为该字符的 ASCII 码值。

```
#include <stdio.h>
void main()
{
    /* upper 为大写字母计数器,lower 为小写字母计数器,digit 为数字字符计数器,space 为空格字符计数器,other 为其他字符计数器 */
    int i,upper = 0,lower = 0,digit = 0,space = 0,other = 0;
    char text[80];
    printf("请输入一行英语句子: \n");
    gets(text);
    for(i = 0;;i++)
    {
```

```
        if(text[i]>='A'&&text[i]<='Z')        /*判断是否为大写字母*/
            ________;
        else if()                             /*判断是否为小写字母*/
            ________;
        else if(&&text[i]<='9') ________      /*判断是否为数字字符*/
            ________;
        else if()                             /*判断是否为空格字符*/
            ________;
        else                                  /*判断是否为其他字符*/
            ________;
    }
    printf("这个英语句子中：大写字母有%d个,小写字母有%d个," ,upper,lower);
    printf("数字字符有%d个,空格字符有%d个,",digit,space);
    printf("其他字符有%d个\n ",,other);
}
```

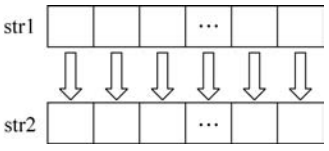

图 5.25　str1 复制给 str2

（3）编写一个字符串复制程序，将字符串 str1 复制到字符串 str2 中（不能使用 strcpy()函数），并显示复制完成后的结果。

提示：将 str1 中的字符串复制给 str2。使用循环逐个将 str1 中的字符复制给 str2 的对应元素，包括字符串结束符'\0'，如图 5.25 所示。

5.4　综合设计

例 5-17　把果园竞赛的问题进一步升级，假设这次比赛有 8 个选手参加，需要存储这 8 个选手的姓名、编号以及摘得的葡萄、梨和桃子的质量，选手的具体信息如表 5.6 所示。需要解决以下 3 个问题。

（1）摘桃最多的选手是谁？质量是多少？他的编号是什么？

（2）整个比赛共摘了多少葡萄？每个选手平均摘了多少葡萄？

（3）计算每个选手摘得的水果的总质量，按照总质量由高到低排序，输出成绩表，格式为“姓名　编号　总质量”。

表 5.6　果园竞赛成绩表

姓　名	编　号	葡萄(kg)	梨(kg)	桃子(kg)
张明	2001	55	51	49
王蒙蒙	2002	60	57	62
赵子山	2003	53	46	66
李晓春	2004	47	59	64
李勇	2005	41	48	56
杨天成	2006	61	58	70
刘晓明	2007	40	35	47
丁言	2008	39	54	63

5.4.1 解决数据的存储问题

本例是综合应用问题，需要将前面所学的基础知识灵活地运用起来。对于这类规模较大的问题，可以把问题分成若干个比较容易解决的子问题后再分别解决，然后再把它们结合起来，整个问题就得到解决了。

这道题首先要解决的是数据的存储问题，通过观察成绩表，发现表中大部分数据间的关系和数据类型符合二维数组的特征，可使用二维数组来存储。编号和几种水果质量的 4 列都是整型数据，可用二维数组保存，但是选手姓名是字符串，要使用一个一维数组存储。注意，一维数组每个元素的下标必须和二维数组的行下标一一对应，否则会造成选手姓名和成绩上的混乱。以下程序的功能是输入并保存表 5.6 中的数据，然后输出查看。

```
#include<stdio.h>
void main()
{
    char name[8][8];                          /*保存选手姓名*/
    int fruitWeight[8][4];                    /*保存编号和几种水果的质量*/
    int i;
    printf("输入 8 位选手的姓名、编号和成绩\n");
    for(i=0;i<8;i++)
    {
        printf("第%d位选手的姓名是:",i+1);
        gets(name[i]);
        printf("编号是:");
        scanf("%d",&fruitWeight[i][0]);
        printf("摘得的葡萄质量是:");
        scanf("%d",&fruitWeight[i][1]);
        printf("摘得的梨质量是:");
        scanf("%d",&fruitWeight[i][2]);
        printf("摘得的桃子质量是:");
        scanf("%d",&fruitWeight[i][3]);
        getchar();
    }
    printf("选手成绩表\n");
    printf("------------------------------------------------------------\n");
    printf("姓名  编号  葡萄  梨  桃子\n");
    for(i=0;i<8;i++)
    {
        puts(name[i]);
        printf("%16d",fruitWeight[i][0]);
        printf("%12d",fruitWeight[i][1]);
        printf("%10d",fruitWeight[i][2]);
        printf("%10d",fruitWeight[i][3]);
        printf("\n");
    }
}
输入 8 位选手的姓名、编号和成绩
第 1 位选手的姓名是:张明
编号是:2001
```

```
摘得的葡萄质量是：55
摘得的梨质量是：51
摘得的桃子质量是：49
第 2 位选手的姓名是：王蒙蒙
编号是：2002
摘得的葡萄质量是：60
摘得的梨质量是：57
摘得的桃子质量是：62
…
选手成绩表
-------------------------------------------------------------------------
姓名            编号            葡萄            梨              桃子
张明            2001            55              51              49
王蒙蒙          2002            60              57              62
赵子山          2003            53              46              66
李晓春          2004            47              59              64
李勇            2005            41              48              56
杨天成          2006            61              58              70
刘晓明          2007            40              35              47
丁言            2008            39              54              63
```

5.4.2 找出摘桃子最多的选手

在所有选手中找出摘桃子最多的选手，然后输出该选手的姓名和编号。该问题是在二维数组的桃子列上做运算，找出该列的最大值，获得最大值对应的选手编号，再根据最大值的行下标在存储选手名字的一维数组中查到该选手姓名的数组元素。

```
#include<stdio.h>
void main()
{
    char name[8][8];
    int fruitWeight[8][4],max=0,pNo;
    int i;
    printf("输入 8 位选手的姓名、编号和成绩\n");
    for(i=0;i<8;i++)
    {
        printf("第%d位选手的姓名：",i+1);
        gets(name[i]);
        printf("编号是：");
        scanf("%d",&fruitWeight[i][0]);
        printf("摘得的葡萄质量是：");
        scanf("%d",&fruitWeight[i][1]);
        printf("摘得的梨质量是：");
        scanf("%d",&fruitWeight[i][2]);
        printf("摘得的桃子质量是：");
        scanf("%d",&fruitWeight[i][3]);
        getchar();
    }
    for(i=0;i<8;i++)                        /*在桃子列上找出质量最大值元素*/
    {
```

```
            if(max < fruitWeight[i][3])
            {
                max = fruitWeight[i][3];
                pNo = i;                                /* 取最大值元素的行下标 */
            }
        }
        printf("摘桃子的冠军是 %s,编号为 %d,桃子的质量是 %d\n",
            name[pNo],fruitWeight[pNo][0],fruitWeight[pNo][3]);
    }
```

运行结果的输入数据部分与 5.4.1 节相同,这里不再给出,只给出桃子信息的显示部分。

运行结果:

```
摘桃子的冠军是杨天成,编号为 2006,桃子的质量是 70
```

5.4.3 计算选手的总成绩

选手总成绩即选手摘得的葡萄、梨和桃子的总质量。要按选手摘得的水果总质量排序,于是需要保存各选手摘得的水果的总质量。可以把数组 int fruitWeight[8][4]变为 int fruitWeight[8][5],即在原数组中增加一列来存储总质量。求选手所摘水果的总质量,只要对 fruitWeight 数组每行的 1～3 列元素求和即可,再存放到第 4 列元素中。对总成绩一列的元素进行排序,排序方法可以选择起泡法,注意当该列数组元素发生位置变换时和它同一列的其他元素也要同步变换,存储姓名的数组元素也要同步变换。

```
#include < stdio.h >
#include < string.h >
void main()
{
    char name[8][8];
    int fruitWeight[8][5];
    int i,j,temp,total = 0;
    char nTemp[8];
    printf("输入 8 位选手的姓名、编号和成绩\n");
    for(i = 0;i < 8;i++)
    {
        fruitWeight[i][4] = 0;
        printf("第 %d 位选手的姓名是: ",i + 1);
        gets(name[i]);
        printf("编号是: ");
        scanf(" %d",&fruitWeight[i][0]);
        printf("摘得的葡萄质量是: ");
        scanf(" %d",&fruitWeight[i][1]);
        printf("摘得的梨质量是: ");
        scanf(" %d",&fruitWeight[i][2]);
        printf("摘得的桃子质量是: ");
        scanf(" %d",&fruitWeight[i][3]);
        fruitWeight[i][4] = fruitWeight[i][1] + fruitWeight[i][2] + fruitWeight[i][3];
```

```
            getchar();
        }
        for(i = 0;i < 7;i++)
            for(j = 0;j < 7 - i;j++)
                if(fruitWeight[j][4]< fruitWeight[j + 1][4])
                { /* 调换总质量 */
                    temp = fruitWeight[j][4];
                    fruitWeight[j][4] = fruitWeight[j + 1][4];
                    fruitWeight[j + 1][4] = temp;
                    /* 调换编号 */
                    temp = fruitWeight[j][0];
                    fruitWeight[j][0] = fruitWeight[j + 1][0];
                    fruitWeight[j + 1][0] = temp;
                    /* 调换葡萄质量 */
                    temp = fruitWeight[j][1];
                    fruitWeight[j][1] = fruitWeight[j + 1][1];
                    fruitWeight[j + 1][1] = temp;
                    /* 调换梨质量 */
                    temp = fruitWeight[j][2];
                    fruitWeight[j][2] = fruitWeight[j + 1][2];
                    fruitWeight[j + 1][2] = temp;
                    /* 调换桃子质量 */
                    temp = fruitWeight[j][3];
                    fruitWeight[j][3] = fruitWeight[j + 1][3];
                    fruitWeight[j + 1][3] = temp;
                    /* 调换选手姓名 */
                    strcpy(nTemp,name[j]);
                    strcpy(name[j],name[j + 1]);
                    strcpy(name[j + 1],nTemp);
                }

        printf("选手成绩表\n");
        printf("------------------------------------------------------------\n");
            printf("姓名  编号  葡萄  梨  桃子  总质量\n");
        for(i = 0;i < 8;i++)
        {
            puts(name[i]);
            printf("%16d",fruitWeight[i][0]);
            printf("%12d",fruitWeight[i][1]);
            printf("%10d",fruitWeight[i][2]);
            printf("%10d",fruitWeight[i][3]);
            printf("%10d",fruitWeight[i][4]);
            printf("\n");
        }
    }
```

运行结果的输入数据部分与5.4.1节相同，这里不再给出，只给出排序后的成绩表显示部分。

选手成绩表

姓名	编号	葡萄	梨	桃子	总质量
杨天成	2006	61	58	70	189
王蒙蒙	2002	60	57	62	179
李晓春	2004	47	59	64	170
赵子山	2003	53	46	66	165
丁言	2008	39	54	63	156
张明	2001	55	51	49	155
李勇	2005	41	48	56	145
刘晓明	2007	40	35	47	122

以上几个小问题全部解决，把这几个小问题的代码综合在一个程序中，便能得到这个问题的最后解答。试着自己综合 4 段代码，在计算机上调试运行，观察运行结果。

5.5 小结

数组是 C 语言组织数据的一种数据类型，它是将一组类型相同的数据按照顺序关系组织起来，用一个名字来命名，并保存在一片连续内存空间的数据保存形式。

本章介绍了 3 种数组，即一维数组、二维数组和字符型数组。其中字符型数组可以是一维数组，也可以是二维数组。单独介绍字符型数组主要是因为它有存储字符串常量的功能，除此之外，字符数组的定义和使用与前两种数组相同。另外，C 语言风格的字符串有一个字符串结束标志'\0'，该标志可作为处理字符串时的依据。

在学习数组时有 4 点需要注意。

(1) 每个数组元素都可以看作一个与数组类型相同的变量来使用。

(2) 数组名是地址常量，它代表数组在内存中的地址，也就是数组的第 0 号元素在内存中的地址。注意，不能改变数组名的值，也不能用数组名来引用整个数组(字符数组除外)。

(3) 有 N 个元素的数组的下标值是 0～N－1，在用下标值引用数组元素时要注意下标值的范围，不要造成下标越界的问题。

(4) 通常使用单重循环处理一维数组，使用双重循环处理二维数组。

习 题 5

1. 选择题

(1) 在 C 语言中引用数组元素时，其数组下标的数据类型允许是(　　)。

A. 整型常量　　B. 整型表达式

C. 整型常量或整型表达式　　D. 任何类型的表达式

(2) 以下能正确定义一维数组的选项是(　　)。

A. `int num[];`　　B. `int num[0…100];`

C. `#define n 100`
`int num[n];int num[n];`　　D. `int n=100;`

(3) 有数组声明 int values[30];,下列下标值引用错误的是(　　)。

A. values[30]　　B. values[20]　　C. values[10]　　D. values[0]

(4) 以下定义不正确的是(　　)。

A. float a[2][] = {1};　　B. float a[][2] = {1};

C. float a[2][2] = {1};　　D. float a[2][2] = {{1},{1}};

(5) 若有定义 int a[][3] = {1,2,3,4,5,6,7};,则数组 a 的第一维大小是(　　)。

A. 2　　B. 3　　C. 4　　D. 5

(6) 对 a 和 b 两个数组进行以下初始化：

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

则下面描述正确的是(　　)。

A. a 和 b 数组完全相同　　B. a 和 b 中都存放字符串

C. sizeof(a)比 sizeof(b)大　　D. sizeof(a)与 sizeof(b)相同

(7) 判断字符串 a、b 是否相等应当使用(　　)。

A. if(a == b)　　B. if(a = b)

C. if(strcpy(a,b))　　D. if(strcmp(a,b) == 0)

(8) 以下选项中能正确赋值的是(　　)。

A. char s1[10];s1 = "Ctest";

B. char s2[] = {'C','t','e','s','t'};

C. char s3[5] = "Ctest";

D. char s4 = "Ctest\n";

(9) 如果定义 int a[] = {1,1,1};int b[3] = {1,1,1};,表达式 a == b 的结果为(　　)。

A. 无法比较　　B. 为真　　C. 为假　　D. 不确定

(10) 以下程序的输出结果是(　　)。

```
#include "stdio.h"
void main ()
{
    int a[8] = {1,2,3,4,5,6,7,8},sum = 0,i;
    for(i = 0;i < 8;i = i + 2)
        sum = sum + a[i];
    printf("sum = %d",sum);
}
```

A. 输出一个不正确的值　　B. sum = 36

C. sum = 20　　D. sum = 16

(11) 以下程序的执行结果为(　　)。

```
#include "stdio.h"
#include "string.h"
void main()
{
    char a[] = "ABCDEFG";
```

```
    int m,n,t;
    m = 0;
    n = strlen(a) - 1;
    while(m < n)
    {
        t = a[m];
        a[m] = a[n];
        a[n] = t;
        m++;
        n--;
    }
    printf("%s",a);
}
```

A. ABCDEFG　　B. ABCGEFD　　C. GFEDCBA　　D. 不正确

(12) 以下程序的输出结果是(　　)。

```
#include <stdio.h>
void main()
{
    int a[3][3] = {{1,2},{3,4},{5,6}},i,j,s = 0;
    for(i = 1;i < 3;i++)
        for(j = 0;j <= 1;j++)
            s += a[i][j];
    printf("%d",s);
}
```

A. 18　　B. 19　　C. 20　　D. 21

(13) 有下面的程序段,则(　　)。

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

A. 运行后将输出 China　　B. 运行后将输出 Ch

C. 运行后将输出 Chi　　D. 编译出错

2. 读程序写结果题

(1) 当从键盘输入 18 时,下面程序的运行结果是_____。

```
#include <stdio.h>
void main()
{
    int y,j,a[8];
    scanf("%d",&y);
    j = 0;
    do
    {
        a[j++] = y%2;
        y++;
```

```
    }while(j<8);
    for(j=7;j>=0;j--)
        printf("%d\n",a[j]);
}
```

（2）下面程序的运行结果是_____。

```
#include<stdio.h>
main()
{
    int i;
    char str[]="12345";
    for(i=4;i>=0;i--)
    printf("%c\n",str[i]);
}
```

（3）下面程序的运行结果是_____。

```
#include<stdio.h>
main()
{
    int i=5;
    char c[6]="abcd";
    do
    {
        c[i]=c[i-1];
    }while(--i>0);
    puts(c);
}
```

（4）下面程序的运行结果是_____。

```
#include<stdio.h>
main()
{
    int i,r;
    char s1[80]="bus",s2[80]="book";
    for(i=r=0;s1[i]!='\0'&&s2[i]!='\0';i++)
        if(s1[i]==s2[i])
        i++;
        else
        {
            r=s1[i]-s2[i];
            break;
        }
    printf("%d",r);
}
```

（5）下面程序的运行结果是_____。

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

```
    int a[4][4] = {{1,2, - 3, - 4},{0, - 12, - 13,14},{ - 21,23,0, - 24},{ - 31,32, - 33,0}};
    int i,j,s = 0;
    for(i = 0;i < 4;i++)
        for(j = 0;j < 4;j++)
        {
            if(a[i][j]< 0)
                continue;
            if(a[i][j] == 0)
                break;
            s += a[i][j];
        }
    printf(" % d\n",s);
}
```

3. 填空题

(1) 下面的程序以每行 4 个数据的形式输出数组 a,请填空。

```
# include < stdio.h >
# define N 20
main()
{
    int a[N],i;
    for(i = 0;i < N;i++)
        scanf(" % d",    ①    );
    for(i = 0;i < N;i++)
    {
        if(    ②    ) printf("\n");
        printf(" % 3d",a[i]);
    }
    printf("\n");
}
```

(2) 以下程序是求矩阵 $\boldsymbol{a}$、$\boldsymbol{b}$ 的和,结果存入矩阵 $\boldsymbol{c}$ 中,并按矩阵的形式输出,请填空。

```
# include < stdio.h >
main()
{
    int a[3][4] = {{3, - 2,7,5},{1,0,4, - 3},{6,8,0,2}};
    int b[3][4] = {{ - 2,0,1,4},{5, - 1,7,6},{6,8,0,2}};
    int i,j,c[3][4];
    for(i = 0;i < 3;i++)
        for(j = 0;j < 4;j++)
            c[i][j] =    ①    ;
    for(i = 0;i < 3;i++)
    {
        for(j = 0;j < 4;j++) printf(" % 3d",c[i][j]);
            ②    ;
    }
}
```

（3）以下程序的功能是删除字符串 s 中的所有数字字符，请填空。

```
#include<stdio.h>
main()
{
    int i,n = 0;
    char s[] = "f45dsa45fas8";
    for(i = 0; s[i]; i++)
      if (________)
          s[n++] = s[i];
    s[n] = '\0';
}
```

（4）以下程序的功能是求矩阵 b（除外围元素）的元素之积，请填空。

```
#include<stdio.h>
main()
{
    int i,j,f = 1;
    int b[][4] = {1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7};
    for(i = 1;    ①    ;i++)
        for(j = 1;j<3;j++)
            f = f * ____②____;
        printf("f = %d\n",f);

}
```

4. 编程题

（1）编写程序实现“查表”功能，即如果若干数据存放在一个数组 data 中，该程序对输入的任意一个数查找数组 data 中是否有与这个数相等的数。若有，则输出该数在数组 data 中的位置，否则输出“没有找到数据！”。

（2）找出二维数组的“鞍点”，即该位置上的元素在该行上最大、在该列上最小。二维数组也可能没有“鞍点”。

（3）输入一个字符串，把该字符串中的字符按照由小到大的顺序重新排列。

（4）编程按顺序计算 500～1 中的整数之和，当和数超过 5000 时停止，输出结果时参与求和的整数的个数。

（5）将 1～200 中能被 7 整除的数删除，显示余下的数据。

本章实验实训

【实验目的】

（1）掌握数值型一维数组及二维数组的定义、初始化，以及输入/输出方法。

（2）掌握用一维数组及二维数组解决实际问题的算法。

（3）掌握字符型数组的定义、初始化，以及输入/输出的方法。

(4) 掌握用字符型数组解决字符串问题的方法。

(5) 掌握常用字符串处理函数的应用。

【实验内容及步骤】

设计一个学生成绩管理程序。为简化设计,学生人数、课程名称和门数可以事先确定。程序应具有以下主要功能。

(1) 能输入学生的姓名和课程的成绩。

(2) 可按成绩总分从高到低的次序排序,姓名同时做相应调整,输出排序后的学生成绩表。

(3) 按学生姓名查询成绩表,输出各门课的成绩和总分,若未找到要输出提示。