

第六章 数组

在程序设计中，有时需要保存大量同种类型数据，如 100 名同学的成绩需要 100 个不同的变量来存放。如果这些变量名称各不相同会大大增加程序的复杂度，为了处理方便，可以把相同类型的若干变量按有序的形式组织起来。这些按序排列的同类数据元素的集合称为数组。

数组中各数据元素具有相同的变量名称，通过不同的下标来惟一地标识它们，在内存中占用连续的存储空间，通过循环语句可以方便地输入、处理、输出数组中各元素的值。使用数组不仅可以免除为大量变量命名，而且使得程序代码的编写更加简洁高效，某些问题不使用数组难以解决。

在 C 语言中，数组属于构造数据类型。一个数组可以分解为多个数组元素，这些数组元素可以是基本数据类型或是构造类型。按数组元素的类型划分，数组又可分为数值数组、字符数组、指针数组、结构数组等类别。本章介绍数值数组和字符数组。

6.1 一维数组

一维数组是只有一个下标的数组。

6.1.1 一维数组的定义

数组同变量一样需要先定义后使用。一维数组是只有一个下标的数组，其定义格式为：

类型说明符 数组名[常量表达式]；

其中，类型说明符可以是任意一种基本数据类型或构造数据类型；数组名是用户定义的数组标识符；方括号中的常量表达式表示数据元素的个数，也称为数组的长度。

例如：

```
int a[10];           定义整型数组 a，有 10 个元素
float b[10],c[20];   定义有 10 个元素的实型数组 b 和有 20 个元素的实型数组 c
char ch[20];         定义字符数组 ch，有 20 个元素。
```

说明：

(1) 数组的类型实际上是数组元素的取值类型。对于同一个数组，其所有元素的数据类型都是相同的。

(2) 数组名的书写规则应符合标识符的命名规则。

(3) 方括号中的常量表达式表示数组元素的个数，下标从 0 开始。例如数组 a[5] 中有 5 个元素，分别为 a[0]，a[1]，a[2]，a[3]，a[4]。

(4) 数组名不能与其它变量名相同，例如下列写法是错误的。

```
void main()
{
    int a;
    float a[10];
    .....
}
```

(5) 编译程序为数组开辟连续的存储单元来顺序存放数组中各数组元素的值，用数组名表示该数组存储区的首地址。例如：

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

定义了 int 型数组 a，编译程序将为 a 数组在内存中开辟 5 个连续的存储单元（每个 int 存储单元占 4 个字节），用来存放 a 数组的 5 个数组元素。a[0] 代表这片存储区的第一个存储单元。而数组名 a 代表 a 数组的首地址，即 a[0] 存储单元的地址。如图 6-1 所示。

由于 a[i] 实际上代表这片存储区序号为 i 的存储单元，所以 a[i] 就是一个带下标的 int 型变量。a 数组是这些 int 型下标变量的集合。

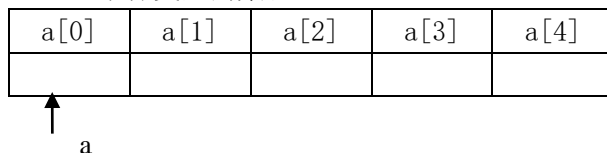


图 6-1 数组名代表数组的首地址

(6) 方括号中数组元素个数的表示可以是常量、符号常数或常量表达式，但不能是变量，C 语言不允许动态定义数组的大小。例如：

```
#define SD 6
void main()
{
    int a[3+2], b[7+SD];
    .....
}
```

是合法的。但是下述说明方式是错误的。

```
void main()
{
    int n=5;
```

```

        int a[n];
        .....
    }

```

(7) 允许在同一个类型说明中,说明多个数组和多个变量。例如:

```
int a,b,c,d,k1[10],k2[20];
```

6.1.2 一维数组元素的引用

数组的定义说明了该数组中有多少个数组元素,这些数组元素的名字以及数据类型。使用时,不能一次引用整个数组,只能逐个地使用数组元素。

数组元素是组成数组的基本单元。数组元素也是一种变量,其标识方法为数组名后跟一个下标,下标表示了元素在数组中的顺序号,所以数组元素通常也称为下标变量。

一维数组元素的表示形式为:

数组名[下标表达式]

其中,下标表达式只能为整型常量或整型表达式。

例如,有如下语句:

```
int a[5],i=1,j=2,k=4;
a[k]=a[i-1]+a[j];
```

则 $a[2]$, $a[k]$, $a[j-1]$, $a[j+i]$, $a[i++]$ 都是对 a 数组元素的合法引用;语句 $a[k]=a[i-1]+a[j]$; 表示 $a[0]$ 的值与 $a[2]$ 的值求和并赋给 $a[4]$ 。

实际应用中,一维数组元素的引用可以配合循环语句来实现,例如:

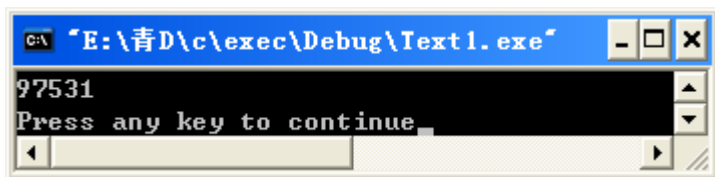
```
for(i=0;i<5;i++)scanf("%d",&a[i]);    表示依次为 a 数组的 5 个元素输入数据
for(i=0;i<5;i++)printf("%d",a[i]);    表示依次输出 a 数组的 5 个元素值
```

【例 6.1】运行如下程序,观察比较输出结果。

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

```
}
```

运行结果为:



分析: 本例中首先利用循环按下标从小到大给 a 数组各元素送入奇数值, 然后又用循环语句倒序输出各个奇数。在第一个 for 语句中省略了表达式 3, 通过在下标变量中使用表达式 `i++` 修改循环变量。

6.1.3 一维数组的初始化

数组定义后, 系统会为该数组在内存中开辟一片连续的存储单元, 但这些存储单元中还没有确定的值。为数组元素赋值可以采用如下方法:

- (1) 用赋值语句逐一对数组元素赋值。
- (2) 用循环语句配合 `scanf` 函数逐一对数组元素赋值。
- (3) 初始化赋值。

数组初始化赋值是指在数组定义时给数组元素赋予初值。数组初始化是在编译阶段进行的, 这样将减少运行时间, 提高程序效率。

初始化赋值的一般形式为:

类型说明符 数组名[常量表达式]={值 1, 值 2……值 n};

数组中各元素的初值列写在{ }中, 各值之间以逗号间隔。例如:

```
int a[10]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

相当于定义了一个整型的 a 数组, 并为其 10 个元素赋初值, 其中 `a[0]=0`, `a[1]=1`, `a[2]=2`, `a[3]=3`, `a[4]=4`, `a[5]=5`, `a[6]=6`, `a[7]=7`, `a[8]=8`, `a[9]=9`。

说明:

(1) 初始化时可以只给部分元素赋初值。当{ }中值的个数少于元素个数时, 只给前面部分元素赋值。例如: `int a[10]={0, 1, 2, 3, 4};` 表示只给 `a[0]~a[4]` 五个元素依次赋值, 而后五个元素自动赋 0 值。


(2) 只能给元素逐一赋值, 不能给数组整体赋值。例如给十个元素全部赋值 1, 只能写为: `int a[10]={1, 1, 1, 1, 1, 1, 1, 1, 1, 1};` 而不能写为: `int a[10]=1;`

(3) 数组元素的值全为 0 时, 可以简写为:

```
int a[100]={0};
```

相当于:

```
int a[100]={0, 0, 0, ....., 0};
```


以逗号隔开的 100 个 0

(4) 为数组全部元素赋值时, 可以不给出数组元素的个数, 但此时必须给出所有初值。

例如: `int a[5]={1, 2, 3, 4, 5};` 可写为 `int a[]={1, 2, 3, 4, 5};`。

6.1.4 一维数组的应用

在程序设计中经常需要使用数组存放数据, 许多算法是建立在数组和循环之上的。下面通过一些常用的基本算法说明如何使用数组解决实际问题。

【例 6.2】 输入 10 个数, 输出它们的平均值及这些数中所有大于平均值的数。

分析: 求 10 个数的平均值可以利用循环边输入、边累加, 再将得到的累加和除以 10 即可。因为要输出大于平均值的数, 应在输入数据的同时将其保存起来, 得到平均值后与其逐一比较才能得到结果。本例中, 10 个数的保存可以采用长度为 10 的一维数组, 利用循环语句来引用其各个元素。

程序如下:

```
#define N 10
#include <stdio.h>
void main()
{ int k;
  float num[N], aver, sum;
  sum=0;
  for(k=0; k<N; k++)                /* 输入 N 个数, 存放到 a 数组中, 并求和 */
  {   scanf("%f", &num[k]);
      sum=sum+num[k];
  }
  aver=sum/N;                        /* 求 N 个数的平均值并输出 */
  printf("average=%f\n", aver);
  for(k=0; k<N; k++)                /* 输出大于平均值的数 */
  {   if(num[k]>aver) printf("%f ", num[k]);
  }
}
```

程序运行结果如图 6-2 所示。

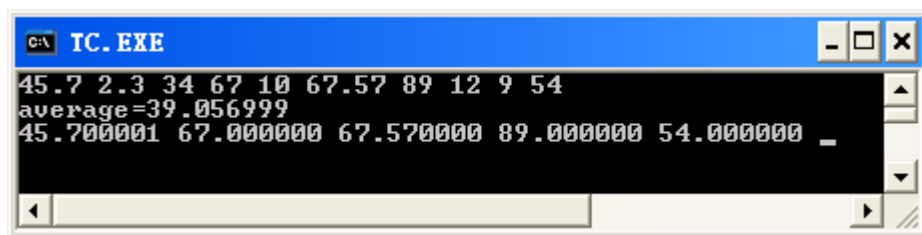


图 6-2 例 6.2 运行结果

思考：如果要计算 100 个数的平均值并输出大于平均值的数，上面程序应如何修改？

【例 6.3】编写程序，求 10 个数中的最大值、最小值以及它们的位置。

分析：求若干个数的最大值可以先将这些数存放在数组中。设置变量 max，使其初值为第一个数，然后从第二个数开始依次与 max 比较，每次比较都将较大者置于 max 中。这样，所有的数均比较一遍后 max 中存放的一定是最大值。最小值的求得方法亦然。

如果要输出最大值和最小值的位置，可再设两个变量 imax、imin，在比较的同时分别记录最大值、最小值元素的下标即可。

程序如下：

```
#define N 10
#include "stdio.h"
void main ( )
{   int a[N],k,max,min,imax,imin;
    for ( k=0; k<N; k++)                /* 将 N 个数输入到 a 数组中 */
        scanf("%d",&a[k]);
    max=a[0];                            /* 置初值为第一个数 */
    min=a[0];
    imax=0;                             /* 置初值为第一个数的小标 */
    imin=0;
    for ( k=1; k<N; k++)                /* 依次比较各数，找出最大值 max、最小值 min */
    {
        if ( max<a[k] ) {max=a[k];imax=k;}
        if ( min>a[k] ) {min=a[k];imin=k;}
    }
    printf("max=a[%d]=%d,min=a[%d]=%d\n", imax, max, imin, min);
}
```

运行结果为：

输入：20 30 -15 2 69 6 31 14 -23 11 54✓

输出：max=a[4]=69，min=a[8]=-23

以上程序也可以只设置变量max和min分别记录最大值和最小值的下标，即位置，则a[max]就是最大值，a[min]就是最小值。

主要程序段如下：

```
max=min=0;                                /*最大值、最小值初始位置为 0 */
for ( k=1;k<N;k++ )                        /*依次比较各数，找出最大值与最小值的下标 max、min
*/
{
    if(a[max]<a[k]) max=k;
    if(a[min]>a[k]) min=k;
}
printf("max=a[%d]=%d,min=a[%d]=%d\n",max,a[max],min,a[min]);
```

【例 6.4】 用冒泡法由大到小（降序）排序 n 个数。

分析：在有序数据基础上进行数据处理往往可以提高程序处理效率，因此排序算法是常见的典型算法。排序算法的核心是做数据的比较操作，而冒泡法降序排序的典型特点是做相邻两个数的比较，将小的数换到后面。

冒泡法降序排序的过程为：若要排序的数有 n 个，则需要 n-1 轮排序；第 j 轮排序中，从第一个数开始，依次进行相邻两个数的比较，即第 1 个与第 2 个数比较，第 2 个与第 3 个数比较，……，第 n-j 个与第 n+1-j 个数比较，共比较 n-j 次，若不符合所要求的顺序，则交换两个数的位置；直到第 n+1-j 个数为止，则第 n+1-j 个位置上的数按要求排好，不再参加以后的比较和交换操作。

例如，设 n=4，4 个数分别是 0，-2，13，9。按由大到小顺序排序，需要 4-1=3 轮排序。

第 1 轮：从第 1 个数开始相邻两数比较，小数互换到后面，比较 4-1=3 次。比较结果如表 6-1 所示。

表 6-1 第 1 轮排序比较结果表

比较结果	说明
第 1 次：0，-2，13，9	0 与-2 比较，较小数-2 已在后面，不需要交换
第 2 次：0，13，-2，9	-2 与 13 比较，两数交换位置，较小数-2 换到后面
第 3 次：0，13，9，-2	-2 与 9 比较，两数交换位置，较小数-2 换到后面

最小数-2 沉底，即第 4 个位置上的数按要求排好，不再参加下一轮排序。

第 2 轮：从第 1 个数开始相邻两数比较，小数互换到后面，比较 4-2=2 次。比较结果如表 6-2 所示。

表 6-2 第 2 轮排序比较结果表

比较结果	说明
第 1 次: 13, 0, 9, -2	0 与 13 比较, 两数交换位置, 较小数 0 换到后面
第 2 次: 13, 9, 0, -2	0 与 9 比较, 两数交换位置, 较小数 0 换到后面

本轮最小数 0 沉底, 即第 3 个位置上的数按要求排好, 不再参加下一轮排序。

第 3 轮: 从第 1 个数开始相邻两数比较, 小数互换到后面, 比较 4-3=1 次。比较结果如表 6-3 所示。

表 6-3 第 3 轮排序比较结果表

比较结果	说明
第 1 次: 13, 9, 0, -2	13 与 9 比较, 较小数 9 已在后面, 不需要交换

得到最终结果。

从以上排序过程可以看出, 较大的数像气泡一样向上冒, 而较小的数则往下沉。故称为冒泡法, 又称为气泡法。

编写程序时, 将待排序数据存入数组中, 用双重循环配合进行排序处理, 其中外层循环控制排序的轮次, 内层循环控制每轮中的各次比较。最终数组数据被按要求排好序。

程序如下:

```
#define N 4
#include <stdio.h>
void main()
{
    int i, j, m, a[N];
    for(i=0; i<N; i++)
        scanf("%d", &a[i]);
    for(j=1; j<=N-1; j++)
        for(i=0; i<N-j; i++)
            if(a[i]<a[i+1])
                { m=a[i]; a[i]=a[i+1]; a[i+1]=m; }
    for(i=0; i<N; i++)
        printf("%5d", a[i]);
}
```

运行结果为:

```
输入: 0 -2 13 9✓
输出: 13 9 0 -2
```

思考: (1) 如按由小到大顺序排序数据, 应如何修改程序?

(2) 本例程序是按由前至后顺序比较排序的, 如按由后至前顺序比较是否可以完成排序? 应如何编写程序?

除冒泡法排序外，还经常采用选择法排序。选择法排序也是每一轮确定一个数的位置，每轮从需排序数据中找出一个最大或最小数。

选择法排序的过程是：若有 n 个数要从小到大（升序）排序，则需要 $n-1$ 轮排序处理。第 1 轮排序，从第 1 个数至第 n 个数中找出最小的数，然后将其与第 1 个数交换位置，排好的第一个数不再参加后面的排序。依次类推，第 j 轮排序，从第 j 个数至第 n 个数中找出最小的数，然后与第 j 个数交换位置，确定第 j 个数顺序。

使用选择法对 N 个存放在 a 数组中的数据进行排序。主要程序代码如下：

```
for ( i=0; i<N-1; i++ )                                /* N-1 轮处
理 */
{   pos=i;                                              /* pos 记录本轮找出的最小值的下标
*/
    for ( j=i+1; j<N; j++ )                            /* 确定本轮最小值的下标
pos*/
        if(a[pos]>a[j]) pos=j;
    if ( pos!=i )                                       /* 最小值不是 a[i]时，a[i]与 a[pos]
互换*/
        { m=a[pos];a[pos]=a[i];a[i]=m;}
```

可以看到， n 个数进行选择法排序，要经过 $n-1$ 轮排序处理。第 j 轮比较 $n-j$ 次，至多交换一次（有时无需交换）。每轮中的每次比较记录最小（大）值的下标，经过若干次比较后，把确定的数一次性交换到目标位置。这样做减少了交换次数。

【例 6.5】 向一个有序数据系列中插入数据，并保持数据的有序性。

分析：假设有 5 个数构成的升序排列数据系列 12、17、25、34、56，将 20 插入其中后的数据系列为 12、17、**20**、25、34、56。设置一维数组 n 存放数据，其长度为 8（大于原始数据个数为插入数据预留存放位置），如图 6-3 所示。

n[0]	n[1]	n[2]	n[3]	n[4]	n[5]	n[6]	n[7]
12	17	25	34	56			

图 6-3 在长度为 8 的 n 数组中存放原始数据

插入数据操作主要有如下几个步骤：

（1）确定插入位置。将要插入数据与数据系列中各数依次比较，直至插入数据较小（如数据降序排列，则为插入数据较大）。要插入数据 20 与第 3 个数 25 比较，20 较小，确定 20 应插入在 25 前面。如图 6-4 所示。

n[0]	n[1]	n[2]	n[3]	n[4]	n[5]	n[6]	n[7]
------	------	------	------	------	------	------	------

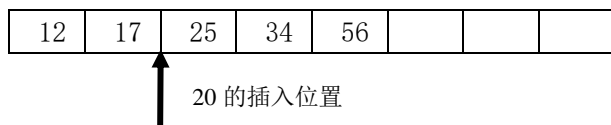


图 6-4 确定要插入数据 20 的插入位置

(2) 空出插入位置。将插入位置及其后数据依次向后移动一个位置。注意，应从最后一个数开始移动，否则会破坏原有数据。如图 6-5 所示。

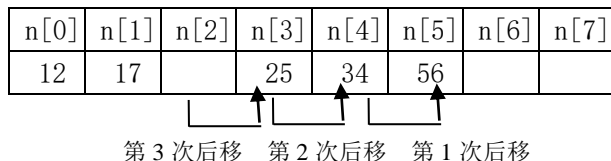


图 6-5 原数据从后开始依次后移空出插入位置

(3) 插入数据。在空出位置上存放插入数据。如图 6-6 所示。



图 6-6 确定要插入数据 20 的插入位置

本例完成在有序数据系列中插入任意 3 个数的操作。

程序如下：

```
#include "stdio.h"
void main()
{
    int i, j, k, num=0, m=5, n[8]={12, 17, 25, 34, 56};
    printf("original data:");
    for(i=0; i<m; i++)
        printf("%4d", n[i]);
    while(1)
    {
        printf("\nPlease input a insert data:");
        scanf("%d", &k);
        for(i=0; i<m; i++)                                /* 确定插入位置 i */
            if(k<n[i]) break;
        for(j=m; j>i; j--)                                  /* 后移数据，空出插入位置 n[i] */
            n[j]=n[j-1];
        n[i]=k;                                            /* 插入 k 到 n[i] */
    }
}
```

```

        m++;
        printf("The inserted data:");
        for(i=0;i<m;i++)
            printf("%4d",n[i]);
        printf("\n");
        num++;
        if(num==3)
            {printf("\nSorry!at most 3 data can be inserted!\n");break;}
    }
}

```

程序运行结果如图 6-7 所示。

```

C:\ TC.EXE
original data: 12 17 25 34 56
Please input a insert data:20
The inserted data: 12 17 20 25 34 56

Please input a insert data:33
The inserted data: 12 17 20 25 33 34 56

Please input a insert data:89
The inserted data: 12 17 20 25 33 34 56 89

Sorry!at most 3 data can be inserted!

```

图 6-7 例 6.5 运行结果

【例 6.6】 设有一有序数据系列，任意输入一个数，如在数据系列中则将其删除。

分析：本例在输入要删除数据后，先查询其是否在数据系列中，如不在，则提示用户“数据没找到！”；否则，将该数据从数据系列中删掉。

本例的查询操作是顺序查询，即将要删除数据与数据系列中数据依次比较，看是否相等。在有序数据系列中查找还可使用二分法查找，效率更高。删除数据实际上是做数据的前移，即从要删除数据后面的一个数开始至最后一个数依次前移一个位置。

程序如下：

```

#include "stdio.h"
void main()
{
    int i,j,k,m=5,n[5]={12,17,25,34,56};
    printf("original data:");

```

```

for(i=0;i<m;i++)
    printf("%4d",n[i]);
printf("\nPlease input a delete data:");
scanf("%d",&k);
for(i=0;i<m;i++)
    if(k==n[i])                                /* 确定要删除数据位置 i */
    {      for(j=i;j<m-1;j++)                    /*前移数据*/
            n[j]=n[j+1];
            break;}                            /*删除后跳出循环 */
if(i>=m)    /*如未找到删除数据，循环变量 i 的值一定大于循环终值*/
    printf("data not found!\n");
else
    { printf("The last data:" );                /*输出删除后结果*/
      for(i=0;i<m-1;i++)
          printf("%4d",n[i]);
      printf("\n"); }
}

```

程序运行结果如图 6-8 所示，这是程序两次运行后的结果。

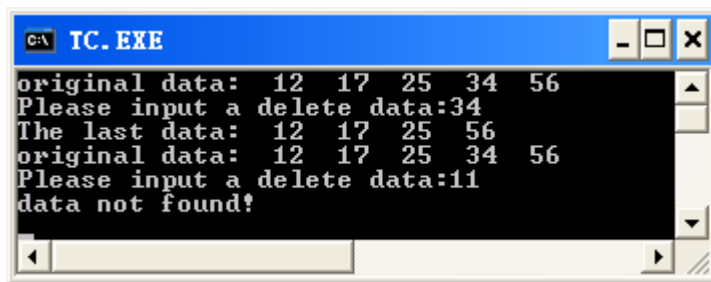


图 6-8 例 6.6 运行结果

6.2 二维数组

C 语言允许使用多维数组，最简单的多维数组是二维数组。当数组元素具有两个下标时，该数组称为二维数组。同样，多维数组具有多个下标。可将二维数组看作具有行和列的平面数据结构，例如矩阵。

6.2.1 二维数组的定义

二维数组的定义形式为：

类型说明符 数组名[常量表达式 1][常量表达式 2]；

其中“常量表达式 1”表示第一维下标的长度，即数组的行数；“常量表达式 2”表示第二维下标的长度，即数组的列数。下标从 0 开始。例如：

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

定义了一个三行四列的整型数组 a，该数组共有 $3 \times 4 = 12$ 个下标变量，即：

```
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]
```

说明：

(1) 在 C 语言中，二维数组在存储器中的存放是按行排列的。即先依次存放 0 行数组元素，然后再接着存放 1 行数组元素，直至最后一行。

(2) 数组名代表数组的首地址。

(3) 在 C 语言中，可将二维数组看作一种特殊的一维数组，每一行是其一个元素，而每一个元素又是一个一维数组。例如：

```
int a[2][3];
```

可将二维数组 a 看成一个一维数组，它有两个元素，我们用 a[i] 表示第 i 行构成的一维数组的数组名，即 a[0]、a[1]。而 a[0]、a[1] 均是包含 3 个元素的一维数组。一维数组 a[0] 的元素为 a[0][0]、a[0][1]、a[0][2]。须注意，a[0]、a[1] 不能当作下标变量使用，它们是数组名，不是一个单纯的下标变量。

6.2.2 二维数组的引用

同一维数组一样，二维数组的引用，也是引用其数组元素。

二维数组元素的表示形式为：

数组名[行下标表达式] [列下标表达式]

例如，有数组定义 `int a[2][3]`，`i=1`，`j=2`，`k=0`；则 `a[i][k]`，`a[j-1][i]`，`a[1][j+k]` 都是对 a 数组元素的合法引用。`a[i][k]=a[i-1][j]+a[1][j]`；表示 a[0][2] 的值与 a[1][2] 的值求和并赋给 a[1][0]。而表 6-4 所示各种形式都是错误的引用。

表 6-4 二维数组错误引用示例

引用	说明
----	----

a[2][3]	行下标可为 0、1，列下标可为 0、1、2，所以 a[2][3] 下标越界
a[i+j][2]	即 a[3][2]，行下标越界
a[1,0]	行、列下标应分别放在各自的中括号里，即 a[1][0]
a(1)(2)	下标应放在中括号中，而不是圆括号

实际应用中，二维数组元素的引用可以配合双重循环来实现。例如，依次为 a 数组的 6 个元素输入数据可表示为：

```
for(i=0;i<2;i++)
    for(j=0;j<3;j++)
        scanf("%d",&a[i][j]);
```

【例 6.7】某学习小组有 6 个人，每个人有三门课，各门课考试成绩如下。求全组各科平均成绩和每人平均成绩。

学号	数学	英语	C 语言
1001	80	75	92
1002	61	65	71
1009	59	63	70
1010	85	87	90
1016	76	77	85
1018	88	78	69

分析：设一个二维数组 score[7][4] 存放六个人的三门课成绩及平均成绩。其中数组的最后一行各元素存放对应各科的平均成绩，最后一列各元素存放对应各人的平均成绩。

程序中首先用一个双重循环，依次输入各人各科成绩，并求每人的平均成绩。其中，外层循环循环六次对应不同学生，内层循环控制输入某学生的各科成绩，并累加后求得每人的平均成绩，存放在数组的最后一列中。

第二个双重循环，外层循环循环四次对应某门课程及某人平均成绩，内层循环循环六次将各科成绩累加后求得各科平均成绩，存放在数组的最后一行中。

最后输出全部成绩及平均成绩。

程序如下：

```
#include "stdio.h"
void main()
{   int i,j;
    float score[7][4];
    printf("input score:\n");
```

```

    for(i=0;i<6;i++)
    {
        score[i][3]=0;
        for(j=0;j<3;j++)
        {
            scanf("%f",&score[i][j]);
            score[i][3]=score[i][3]+score[i][j];} /*求某人三科成绩之和*/
            score[i][3]=score[i][3]/3; /*求某人平均成绩*/
        }
        for(i=0;i<=3;i++) /*求各科平均成绩*/
        {
            score[6][i]=0;
            for(j=0;j<6;j++)
            score[6][i]=score[6][i]+score[j][i];
            score[6][i]=score[6][i]/6;
        }
        printf("output result:\n");
        for(i=0;i<=6;i++) /*输出成绩及平均成绩*/
        {
            for(j=0;j<=3;j++)
                printf("%.2f",score[i][j]);
            printf("\n");
        }
    }
}

```

程序运行结果如图 6-9 所示。

```

C:\ TC. EXE
input score:
80 75 92
61 65 71
59 63 70
85 87 90
76 77 85
88 78 69
output result:
80.00 75.00 92.00 82.33
61.00 65.00 71.00 65.67
59.00 63.00 70.00 64.00
85.00 87.00 90.00 87.33
76.00 77.00 85.00 79.33
88.00 78.00 69.00 78.33
74.83 74.17 79.50 76.17

```

图 6-9 例 6.7 运行结果

6.2.3 二维数组的初始化

二维数组初始化是指在数组定义时给数组元素赋予初值。二维数组初始化的表示形式为：

数据类型 数组名[整型常量表达式][整型常量表达式]={初始化数据};

将“{}”中的初值依次赋给各数组元素，各初值之间用逗号隔开。

二维数组可按行分段赋值，也可按行连续赋值。例如有数组 a[5][3]，则下面两种赋初值的结果是完全相同的。

(1) 按行分段赋值。将初值按行用 {} 分开，可写为：

```
int a[5][3]={ {80,75,92}, {61,65,71}, {59,63,70}, {85,87,90}, {76,77,85} };
```

(2) 按行连续赋值。可写为：

```
int a[5][3]={ 80,75,92,61,65,71,59,63,70,85,87,90,76,77,85 };
```

均完成如下赋值：

```
a[0][0]=80, a[0][1]=75, a[0][2]=92
```

```
a[1][0]=61, a[1][1]=65, a[1][2]=71
```

```
a[2][0]=59, a[2][1]=63, a[2][2]=70
```

```
a[3][0]=85, a[3][1]=87, a[3][2]=90
```

```
a[4][0]=76, a[4][1]=77, a[4][2]=85
```

说明：

(1) 初始化的数据个数不能超过数组元素的个数，否则出错。

(2) 可以只对部分元素赋初值，未赋初值的元素自动取 0 值。例如：

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

表示对每一行的第一列元素赋值，未赋值的元素取 0 值。赋值后 9 个元素的值分别为 1、0、0、2、0、0、3、0、0。

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

赋值后数组元素值为 0、1、0、0、0、2、3、0、0

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

只有两个初值，即 $a[0][0]=1$ ， $a[0][1]=2$ ，其余数组元素的初值均为 0。

(3) 当对全部元素赋初值时，第一维的长度可以不用给出，系统根据初始化的数据个数和第 2 维的长度确定第一维的长度。注意不能省略第二维的定义。例如：

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

可以写为：

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

a 数组的第一维的定义被省略，初始化数据共 9 个，第二维的长度为 3，即每行 3 个数，所以 a 数组的第一维是 3。

一般，第一维定义省略时，其大小的确定规则为：初值个数能被第二维整除，所得的商就是第一维的大小；若不能整除，则第一维的大小为商再加 1。例如：

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

等价于：

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

分行初始化时，也可以省略第一维的定义。下列数组定义中有两对 {}，表示 a 数组有两行。

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

6.2.4 二维数组的应用

本节通过几个例子说明二维数组的应用。利用数组来处理数据的关键是找出下标的规律。

【例 6.8】 完成一个矩阵的转置，即行、列元素互换。

分析：设要转置一个 3×4 的矩阵，则需定义二维数组 $a[3][4]$ 存放矩阵中的数值，二维数组 $b[4][3]$ 存放转置后的结果。由于数组的第一维代表行，第二维代表列，所以 a 数组中的元素 $a[i][j]$ 转置后对应 b 数组中元素 $b[j][i]$ 。

程序如下：

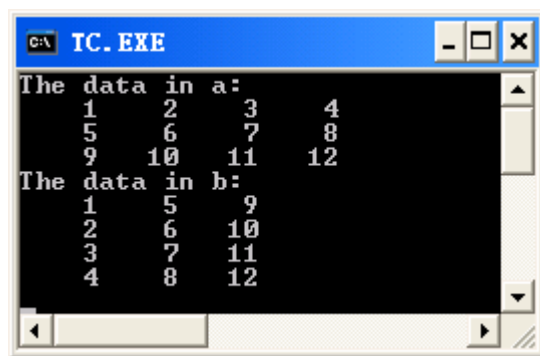
```
#include "stdio.h"
void main()
{   int i,j,b[4][3];
```

```

    int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
    printf("Thedata in a:\n");
    for(i=0;i<3;i++)                                /*输出 a 数组中元素值，同时完成转置*/
    {for(j=0;j<4;j++)
      {   printf("%5d",a[i][j]);
          b[j][i]=a[i][j];}
        printf("\n");
      }
    printf("Thedata in b:\n");
    for(i=0;i<4;i++)                                /*输出转置后结果*/
    {   for(j=0;j<3;j++)
        printf("%5d",b[i][j]);
        printf("\n");
      }
}

```

程序运行结果如图 6-10 所示。



```

C:\ TC. EXE
The data in a:
  1   2   3   4
  5   6   7   8
  9  10  11  12
The data in b:
  1   5   9
  2   6  10
  3   7  11
  4   8  12

```

图 6-10 例 6.8 运行结果

【例 6.9】输出杨辉三角的前 10 行。

分析：杨辉三角中数据的规律是：第 1 列与对角线上的元素值均为 1，其它位置上的元素值均为上一行同列元素与前一列元素值之和，见图 6.2.3。假设杨辉三角中某元素为 $a[i][j]$ ，则上一行同列元素为 $a[i-1][j]$ ，上一行前一列元素为 $a[i-1][j-1]$ 。

程序如下：

```

#include "stdio.h"
void main()
{   int i,j,a[10][10];

```

```

for(i=0;i<10;i++)                                /*将三角中所有元素均置 1*/
for(j=0;j<=i;j++)
    a[i][j]=1;
for(i=2;i<10;i++)                                /*求除第 1 列与对角线以外元素的值*/
for(j=1;j<=i-1;j++)
    a[i][j]=a[i-1][j]+a[i-1][j-1];
for(i=0;i<10;i++)                                /*输出杨辉三角*/
{
    for(j=0;j<=i;j++)
        printf("%4d",a[i][j]);
    printf("\n");
}

```

程序运行结果如图 6-11 所示。

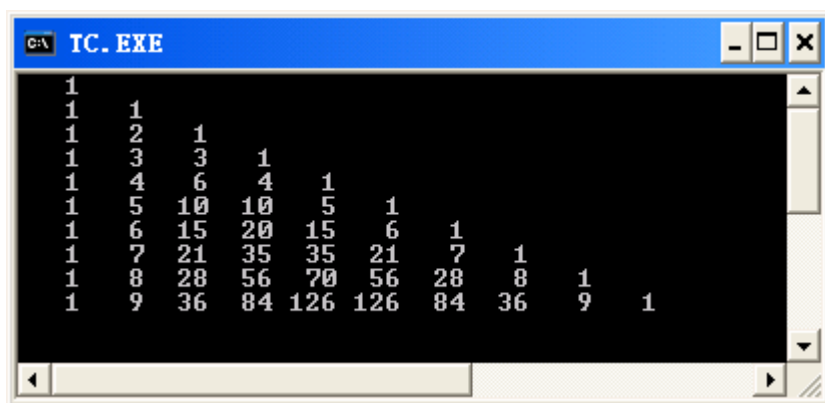


图 6-11 例 6.9 运行结果

6.3 字符数组

从前面的学习中我们知道，C 语言有表示单个字符的字符常量和字符变量，也有字符串常量，但没有字符串变量。字符串的存放可以使用字符数组，字符数组中各数组元素依次存放字符串的各个字符。字符数组的数组名代表该字符串的首地址。利用字符数组为处理字符串中字符和引用整个字符串带来了极大的方便。

6.3.1 字符数组的定义及结束标志

1. 字符数组的定义

用来存放字符型量的数组称为字符数组。字符数组的定义与数值数组相同。例如：

```
char c[10];
```

定义了一个具有 10 个元素的字符数组 c，其每一个元素都是字符型的，一个元素存放一个字符。由于字符型和整型通用，也可以定义为：

```
int c[10];
```

虽然合法，但由于每个整型量占 4 个字节，会浪费存储空间。

字符数组也可以是二维或多维数组，例如：

```
char c[5][10];
```

2. 字符串结束标志

当用字符数组处理字符串时，为了测定字符串的实际长度，C 语言规定了一个字符串结束标志，以字符 ‘\0’ 作为标志。也就是说，遇到字符 ‘\0’ 时，字符串结束，结束标志前面的字符组成字符串。在程序中往往依靠检测 ‘\0’ 字符的出现来判定字符串是否结束，而不是根据数组的长度来决定字符串长度。

系统对字符串常量也自动加一个 ‘\0’ 作为结束符。例如” I am a girl.” 共有 12 个字符，但在内存中占 13 个字节，最后一个字节 ‘\0’ 是系统自动加上的。

‘\0’ 代表 ASCII 码值为 0 的字符，是一个“空操作符”，输出时不显示字符。

6.3.2 字符数组的初始化

字符数组也允许在定义时作初始化赋值。

字符数组的初始化有如下方法：

1. 初始化时把字符逐一赋给数组元素。例如：

```
char c[11]={‘W’,’e’,’l’,’c’,’o’,’m’,’e’,’,’,’y’,’o’,’u’};
```

赋值后数组状态如图 6-12 所示。

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]	C[9]	C[10]
W	e	l	c	o	m	e		y	o	u

图 6-12 逐个字符对字符数组初始化

如果 {} 中提供的初值个数大于数组长度，则出错；如果初值个数小于数组长度，则剩余的数组元素被自动赋予 ‘\0’。例如：

```
char a[10]={‘H’,’e’,’l’,’l’,’o’};
```

赋值后数组状态如图 6-13 所示。

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]	C[9]
H	e	l	l	o	\0	\0	\0	\0	\0

图 6-13 初值个数小于数组长度时的初始化结果

当对全体元素赋初值时可以省去长度说明。例如：

```
char c[]={‘W’,‘e’,‘l’,‘c’,‘o’,‘m’,‘e’,’,’,‘y’,‘o’,‘u’};
```

这时 c 数组的长度自动定为 11。

2. 用字符串常量对字符数组的初始化。例如下面两个语句的功能是等价的：

```
char c[]={“Welcome you”};
```

```
char c[]="Welcome you";
```

注意：

(1) 字符串常量应使用双引号括起来。

(2) 数组 c 的长度是 12，而不是 11。因为字符串常量的最后由系统加上了一个 ‘\0’。

(3) 如果字符串的长度小于数组长度，则剩余的数组元素被自动赋予 ‘\0’。如：

```
char a[8]={ “Hello”};
```

则赋值后数组状态如图 6-14 所示。

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]
H	e	l	l	o	\0	\0	\0

图 6-14 字符串长度小于数组长度时的初始化结果

(4) 字符数组最后的 ‘\0’ 字符不是必须的，但为了处理判断方便最好加上。如：

```
char c[]={‘W’,‘e’,‘l’,‘c’,‘o’,‘m’,‘e’,’,’,‘y’,‘o’,‘u’,‘\0’};
```

在字符串的末尾人为地加了一个 ‘\0’，c 数组的长度自动定为 12。

6.3.3 字符数组的输入输出

字符数组的输入输出归纳起来有下面两种方法。

(1) 逐个字符输入输出

输入输出时使用格式符 “%c”。例如为字符数组 a 赋值并输出的程序段为：

```
#include "stdio.h"
void main()
{
    char a[5];
    int j;
    for(j=0;j<5;j++)
        scanf("%c",&a[j]);
    for(j=0;j<5;j++)
        printf ("%c",a[j]);
}
```

(2) 字符串整体或部分输入输出

输入输出时使用格式符 “%s”。例如为字符数组 b、c 赋值并输出的程序段为：

```
#include "stdio.h"
```

```

void main()
{
    char b[8],c[]="BASIC\ndBASE";
    scanf("%s",b);                /* 将键盘输入字符串存入 b 数组 */
    printf("%s\n",b);              /* 输出 b 数组中的字符串 */
    printf("%s\n",c);
    printf("%s",&c[2]);            /* 输出 c 数组中从 c[2]开始的字符串 */
}

```

运行结果为：

输入:123456✓

输出:123456

BASIC

dBASE

SIC

dBASE

注意：

(1) 用格式符 “%s” 输入输出字符串时：

① 如果是字符数组，其输入输出项必须是字符数组名，名前不能加地址符 “&”，即以字符串的地址形式出现，数组名代表数组的起始地址。例如：

```
printf("%s",b);
```

② 如果输入输出项是字符数组元素的地址，则表示输入输出内容是从该地址开始的字符串。例如：

```
printf("%s",&c[2]);
```

③ 如果输出项以字符串常量形式出现，表示该字符串的首地址。例如：

```
printf("%s","abcd");
```

(2) 用格式符 “%s” 输出的字符串是从输出项提供的地址开始直至碰到字符串结束标志 ‘\0’ 为止。例如程序段：

```
char a[10]="abcd\0123";
```

```
printf("a=%s\n", a);
```

输出：a=abcd。

(3) 用格式符 “%s” 不能输入带空格、回车或跳格的字符串。因为空格、回车或跳格是输入数据的结束标志。例如程序段：

```
Char a[10];
```

```
scanf("%s",a);
```

```
printf("%s\n",a);
```

运行时，输入：How are you✓

输出：How

因为空格、跳格和回车均是输入数据结束的标志，所以 How are you 被看作 3 个输入数据，只把 How 作为 a 数组的数据。为了避免这种情况，一方面可以多设几个字符数组分段存放含空格的字符串，另一方面可以使用 gets() 函数。

【例 6.10】求给定字符串的长度，并将其复制到另一字符串中。

分析：用两个字符数组分别存放源字符串和目标字符串。利用循环逐一读取源字符串中的字符，直至碰到字符串结束符，同时进行字符的计数及存入目标字符串的操作。

程序如下：

```
#include "stdio.h"
void main()
{   char s1[80], s2[80];
    int i, num=0;
    printf("input string s2:\n");
    gets(s2);
    for(i=0; s2[i]!='\0'; i++)                /* 逐个字符计数与复制 */
    {   num++; s1[i]=s2[i];
    }
    s1[i]='\0';                                /* 在目标字符串末尾加上字符串结束符 */
    printf("num=%d\n", num);
    puts(s1);
}
```

程序运行结果如图 6-15 所示。

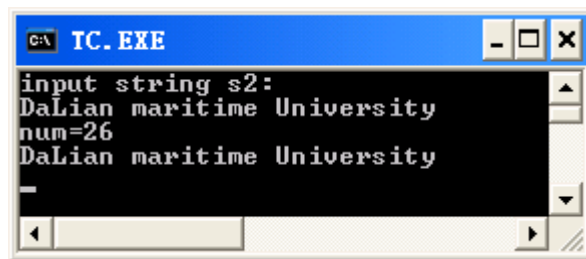


图 6-15 例 6.10 运行结果

6.3.4 字符串处理函数

C 语言提供了丰富的字符串处理函数，大致可分为字符串的输入、输出、合并、修改、比较、转换、复制、搜索等几类。利用这些函数可大大减轻编程负担。

在使用字符串输入输出函数时，应在程序中包含头文件“stdio.h”，使用其它字符串函数则应包含头文件“string.h”。最常用的字符串函数如表 6-5 所示。

表 6-5 常用字符串处理函数

函数名称	调用形式	作 用	说 明
puts()	puts(字符数组)	将一个字符串输出到终端设备	该函数输出的字符串中可以包含转义字符
gets()	gets(字符数组)	从终端输入一个字符串到字符数组中, 并且得到一个函数值	puts() 与 gets() 函数只能一次输入一个字符串, 即圆括号中只能有一个数组名
strcpy()	strcpy(字符数组 1, 字符数组 2)	将字符数组 2 所指字符串内容复制到字符数组 1 所指存储空间中。函数返回字符数组 1 的值, 即目的串的首地址	字符数组 1 必须有足够的空间存放字符数组 2 中的内容
strcat()	strcat(字符数组 1, 字符数组 2)	将字符数组 2 所指字符串内容连接到字符数组 1 所指字符串后, 并自动覆盖字符数组 1 串末尾的 ‘\0’。函数返回字符数组 1 的地址值	字符数组 1 必须有足够的空间存放两个字符串连接后的内容
strlen()	strlen(字符数组)	计算并返回字符数组的长度值	返回的长度值不包括结束标志 ‘\0’
strcmp()	strcmp(字符数组 1, 字符数组 2)	比较字符数组 1 和字符数组 2 所指字符串的大小。若字符数组 1> 字符数组 2, 函数值大于 0(正数); 若字符数组 1=字符数组 2, 函数值为 0; 若字符数组 1<字符数组 2, 函数值小于 0(负数)	两字符串比较时, 按 ASCII 码值从左至右逐一比较

1. 字符串输出函数 puts 与字符串输入函数 gets

字符串输出函数 puts 中可以使用转义字符, 其完全可以由 printf 函数代替。当需要按一定格式输出时, 通常使用 printf 函数。

字符串输入函数 gets 不以空格作为字符串输入结束的标志, 而只以回车作为输入结束。不同于 scanf 函数的是允许输入的字符串中含有空格。

【例 6.11】puts 与 gets 函数使用示例。

程序如下：

```
#include "stdio.h"
void main()
{ char st[15];
  static char c[]="BASIC\ndBASE";
  printf("input string:\n");
  gets(st);
  puts(c);
  puts(st);
}
```

程序运行结果如图 6-16 所示。

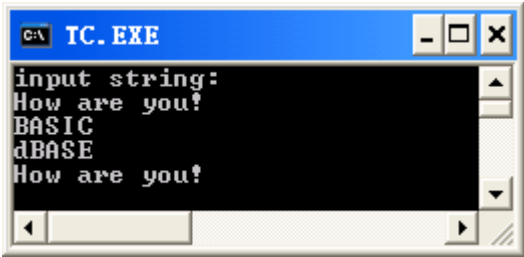


图 6-16 例 6.11 运行结果

2. 字符串连接函数 strcat 与字符串赋值函数 strcpy

两个函数中的“字符数组 1”应定义足够的长度，以容纳处理后的结果字符串。

字符串连接函数 strcat 在连接时，会将前一个字符串后的‘\0’去掉，只保留结果字符串最后的‘\0’。例如：

```
char st1[20]="My name is ";
char st2[]="Wang kai";
printf( "%s",strcat(st1,st2));
```

输出：

My name is Wang kai

连接前：

St1:	M	y		n	a	m	e		i	s		\0	\0	\0	\0	\0	\0	\0	\0
St2:	W	a	n	g		k	a	i	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0

连接后结果字符串：

M	y		n	a	m	e		i	s		W	a	n	g		k	a	i	\0
---	---	--	---	---	---	---	--	---	---	--	---	---	---	---	--	---	---	---	----

字符串赋值函数 `strcpy` 中的“字符数组 2”也可以是一个字符串常量，此时相当于把一个字符串赋予一个字符数组，完成一个字符串的整体赋值。用函数 `strcpy` 拷贝时，串结束标志“\0”也一同被赋值。例如：

```
strcpy(st1, st2);
strcpy(st1, "China");
```

3. 字符串比较函数 `strcmp`

该函数可以比较两个字符数组，也可比较两个字符串常量，或进行字符数组和字符串常量的比较。例如：

```
strcmp(st1, st2);
strcmp("computer", "compare");
strcmp(st1, "zhao");
```

比较两个字符串时，是自左至右逐个字符相比，直到出现不同的字符或碰到结束符为止。若全部字符相同，则相等；若出现不同字符，则以第一个不同字符的比较结果为准。

注意，两个字符串的比较不能使用以下形式：

```
if(st1>st2)
    printf("string1>string2");
```

正确形式为：

```
if(strcmp(st1, st2)>0)
    printf("string1>string2");
```

【例 6.12】输入五个国家的名称，使其按字母顺序排列输出。

分析：一个国家的名称是一个字符串，需要一个一维数组存放，五个国家的名称应存放在一个二维字符数组中。C 语言规定可以把一个二维数组当做若干个一维数组处理，因此本例定义一个二维数组 `c[5][18]`，将其按照五个一维数组 `c[0]`、`c[1]`、`c[2]`、`c[3]`、`c[4]` 处理，每个一维数组存放一个国家的名字。利用字符串比较函数比较各一维数组的大小，并排序输出结果。

程序如下：

```
#include "string.h"
#include "stdio.h"
void main()
{char st[18], c[5][18];
  int i, j, p;
  printf("input country's name:\n");
  for(i=0; i<5; i++)
```

```

        gets(c[i]);                                /*输入五个国家名字符串*/
printf("\n");
for(i=0;i<5;i++)                                  /*对五个国家名排序*/
{ p=i;strcpy(st,c[i]);
  for(j=i+1;j<5;j++)                               /*找出最小的字符串并拷贝到 st 中*/
    if(strcmp(c[j],st)<0)
        {p=j;strcpy(st,c[j]);}
  if(p!=i)                                          /*有比 c[i] 更小的字符串, 则交换 c[i] 与 st 内容*/
  {
      strcpy(st,c[i]);
      strcpy(c[i],c[p]);
      strcpy(c[p],st);
  }
  puts(c[i]);}
printf("\n");
}

```

程序运行结果如图 6-17 所示。

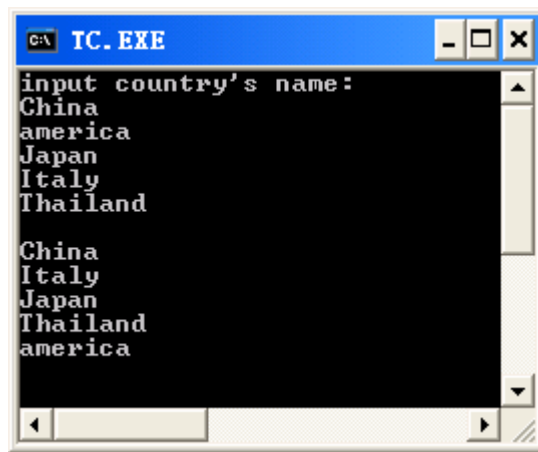


图 6-17 例 6.12 运行结果

【例 6.13】判断 s1 字符串中是否包含 s2 字符串。

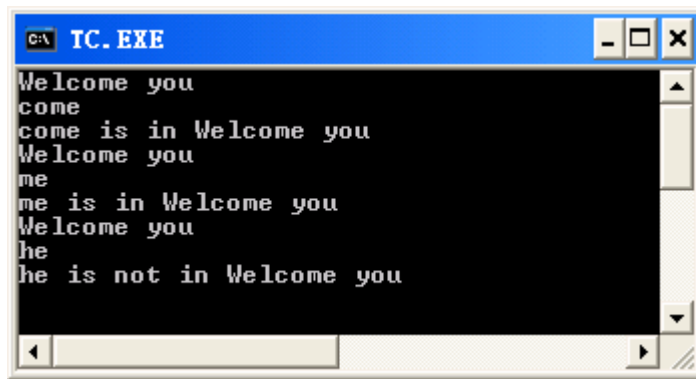
分析：本例的判断过程实际上是做比较处理，具体过程为：从 s1 字符串的第一个字符开始，依次与 s2 字符串的各字符比较，若均相同，则 s1 包含 s2。否则再从 s1 的下一个字符（第 2 个字符）开始，依次与 s2 字符串的各字符比较，……。设 k1、k2 分别表示 s1 串和 s2 串的

长度，则最后一次应从 s1 的第 k1-k2+1 个字符开始（即 s1[k1-k2]），依次与 s2 字符串的各字符比较。若存在不同字符，则 s1 中肯定不包含 s2。

程序如下：

```
#include <stdio.h>
#include <string.h>
void main()
{   char s1[80],s2[80];
    int i=0, j, k, k1, k2, f;
    gets(s1); gets(s2);
    k1=strlen(s1); k2=strlen(s2);
    f=0;                                /*标志 f=1, s2 包含在 s1 中; f=0, s2 不包含在 s1 中 */
    while (i<k1-k2+1&&!f)    /* 从 s1 串的 s1[i] 字符开始检测 s2 是否包含在 s1 中 */
    {   j=0;
        k=i;
        while(s2[j]&&s1[k]==s2[j])    /* 存在不同字符或 s2 包含在 s1 中时退出循环 */
        {   j++; k++;}
        if (s2[j]=='\0')    /* 若退出循环时, s2[j]=='\0', 则 s2 串包含在 s1
串中 */
        {   f=1;
            break;          /* 确认 s2 串包含在 s1 串中, f=1, 退出循环 */
        }
        i++;                /* 从 s1 的下一个字符开始继续检测 */
    }
    if ( f==1 ) printf ("%s is in %s\n",s2,s1);
    else printf ("%s is not in %s\n",s2,s1);
}
```

程序 3 次运行结果如图 6-18 所示。



```
C:\ TC.EXE
Welcome you
come
come is in Welcome you
Welcome you
me
me is in Welcome you
Welcome you
he
he is not in Welcome you
```

图 6-18 例 6.13 运行结果

习题六

一、单选题

1. 要说明一个有 10 个 int 元素的数组，应当选择的语句是：
A. `int a[10]` B. `int a[2,5]` C. `int a[]` D. `n=10;int a[n]`
2. 在下述对 C 语言字符数组的描述中错误的是：
A. 字符数组可以存放字符串
B. 字符数组中的字符串可以进行整体输入/输出
C. 可以在赋值语句中通过赋值运算符“=”对字符数组整体赋值
D. 字符数组的下标从 0 开始
3. 调用 `strlen("abcd\0ef\0g")` 的返回值为：
A. 4 B. 5 C. 8 D. 9
4. 若有以下语句，则正确的描述是：

```
char x[]="12345";  
char y[]={ '1','2','3','4','5' };
```


A. x 数组和 y 数组的长度相同 B. x 数组长度大于 y 数组长度
C. x 数组长度小于 y 数组长度 D. x 数组等价于 y 数组
5. 已知：`char str1[10],str2[10]={"books"};`，则在程序中能够将字符串“book”赋给数组 str1 的正确语句是：
A. `str1={"books"};` B. `strcpy(str1,str2);`
C. `str1=str2` D. `strcpy(str2,str1);`
6. 以下正确的数组定义语句是：
A. `int y[1][4] = {1,2,3,4,5};`
B. `float x[3][] = {{1},{2},{3}};`
C. `long s[2][3] = {{1},{1,2},{1,2,3}};`
D. `double t[][3] = {0};`
7. 若有说明：`int a[][4];`，则 `a[i][j]` 前有____个元素。
A. `j*4+i` B. `i*4+j` C. `i*4+j-1` D. `i*4+j+1`
8. 以下程序的输出结果是：

```
main()  
{  
    int i,x[3][3]={1,2,3,4,5,6,7,8,9};  
    for(i=0;i<3;i++) printf("%d,",x[i][2-i]);  
}
```


A. 1,5,9, B. 1,4,7, C. 3,5,7, D. 3,6,9,

9. 以下程序的输出结果是：

```
main()
{   int n[5]={ 10},i,k=2;
  for(i=0;i<k;i++) n[i]=n[i]+1;
  printf("%d\n",n[k]);}
```

A. 不确定的值 B. 30 C. 10 D. 0

10. 以下程序的输出结果是：

```
main()
{   int m[3][3]={ { 1},{ 2},{ 3}},i,j,t=1;
  for(i=0;i<3;i++)
    for(j=i;j<=i;j++)t=t+m[i][m[j][j]];
  printf("%d\n",t); }
```

A. 0 B. 2 C. 6 D. 1

11. 以下程序的输出结果是：

```
main( )
{   char a[2][5]={ "6937","8254"}; int i,j,s=0;
  for ( i = 0; i < 2; i++ )
    for ( j = 0; a[i][j]>'0' && a[i][j]<='9'; j+=2 )
      s=10*s+a[i][j]-'0';
  printf("s=%d\n",s); }
```

A. s=6385 B. s=69825 C. s=63825 D. s=693825

12. 下列程序____(每行前的数字表示行号)。

```
1 main()
2 {   float a[10]={ 1,2}; int i;
3   for(i=0;i<3;i++) scanf("%d",&a[i]);
4   for(i=1;i<10;i++) a[0]=a[0]+a[i];
5   printf("%f\n",a[0]);}
```

A. 没有错误 B. 第 3 行有错 C. 第 4 行有错 D. 第 5 行有错

13. 以下程序段的功能是：

```
main()
{   int j,k,e,t,a[ ]={ 4,0,6,2,64,1};
  for(j=0;j<5;j++)
  {   t=j;
    for(k=j;k<6;k++) if(a[k]>a[t]) t=k;
```

```
e=a[t];a[t]=a[j];a[j]=e; }
for(k=0;k<6;k++)
printf("%5d",a[k]);
}
```

- | | |
|---------------|---------------|
| A. 冒泡法对数组升序排序 | B. 冒泡法对数组降序排序 |
| C. 选择法对数组升序排序 | D. 选择法对数组降序排序 |

二、填空题

1. 在 C 语言中，一维数组的定义格式为：类型说明符_____。
2. 在 C 语言中，二维数组元素的存放顺序是_____。
3. 若有定义：int a[3][4]={ {1,2}, {0}, {4,6,8,10} };，则初始化后，a[1][2]得到的初值为_____，a[2][1]得到的初值为_____。

4. 以下程序的输出是_____。

```
main()
{   char st[]="abcdef";
    st[3]= '\0';
    printf("%s\n",st);
}
```

5. 以下程序的输出是_____。

```
main()
{   char a[3][4]={"abc","efg","hij"}; int k;
    for(k=1;k<3;k++) putchar(a[k][2]);
}
```

6. 以下程序的输出是_____。

```
main()
{   int i,f[10];
    f[0]=f[1]=1;
    for(i=2;i<10;i++)
        f[i]=f[i-2]+f[i-1];
    for(i=0;i<10;i++)
        {   if(i%4==0)printf("\n");
            printf("%3d",f[i]);
        }
```



```

    }
}

```

7. 以下程序的输出是_____。

```

#include "string.h"
main()
{ char a[80]="AB",b[80]="LMNP"; int j=0;
  strcat(a,b);
  while(a[j++]!='\0') b[j]=a[j];
  puts(b);
}

```

8. 以下程序从数组 a 的第 2 个元素开始，将后项减前项的差值存入数组 b，并按每行 3 个元素输出数组 b。请填空。

```

#include "stdio.h"
main( )
{ int a[10],b[10],i;
  for( i=0;_____; i++ )
      scanf("%d",&a[i] );
  for( i=0;_____; i++ )
      b[i+1]=a[i+1]-a[i];
  for( i=1;i<10; i++ )
      { printf("%5d",b[i]);
        if(_____)printf("\n");}
}

```

9. 以下程序的功能是：把两个按升序排列的数组合并成一个按升序排列的数组。请填空。

```

#include "stdio.h"
main()
{ int i=0,j=0,k=0,a[3]={5, 9, 19},b[5]={12,24,26,37,48},c[10];
  while(i<3 && j<5)
  if(____) { c[k]=b[j];k++;j++;}
  else { c[k]=a[i];k++;i++;}
  while(____) { c[k]=a[i];k++;i++;}
  while(____) { c[k]=b[j];k++;j++;}
  for(i=0;i<k;i++) printf("%3d",c[i]);
}

```

10. 以下程序求矩阵各行元素之和，并以矩阵形式输出原矩阵及相应行元素之和。请填空。

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

11. 以下程序的功能是：输入 10 个字符串，找出每个字符串的最大字符，并依次存入一维数组中，然后输出该一维数组。请填空。

```
#include "stdio.h"
main()
{   int j,k; char a[10][80],b[10];
    for(j=0;j<10;j++) gets(a[j]);
    for(j=0;j<10;j++)
    {   ____;
        for(k=1;a[j][k]!='\0';k++)
        if(b[j]<a[j][k]) ____;
    }
    for(j=0;j<10;j++)
    printf("%d %c\n",j,b[j]);
}
```

12. 以下程序的功能是：删除字符串中所有的'C'字符。请填空。

```
#include "stdio.h"
main()
{   int j,k; char a[80];
    gets(a);
    for(j=k=0;a[j]!='\0';j++)
```

```

if(a[j]!='c' && a[j]!='C') ____
a[k]='\0';
printf("%s\n",a);
}

```

三、编程题

1. 假设一个数组中元素的值依次为 12, 5, 7, 8, 4, 编写程序将其逆序存放, 变成 4, 8, 7, 5, 12。
2. 分别求一个 3×3 矩阵主对角线元素及下三角元素值之和, 并输出该矩阵及求得和。
3. 假设一个数组具有 10 个元素, 删除该数组中的最大值, 并输出删除前后的数组元素值。
4. 编写程序, 产生 30 个 $[1, 100]$ 间的随机整数到 5 行 6 列数组中, 求其中最大值和最小值, 并把最大值元素与右上角元素对调, 最小值元素与左下角元素对调。输出重排前后的数组元素值。
5. 编写程序, 输入任意 10 进制 4 位正整数, 将其化成二进制数。
6. 编写程序, 产生 30 个 50 以内的随机整数到 5 行 6 列数组中, 输出那些在行和列上均为最小的元素。
7. 编写程序, 实现 `gets()` 函数的功能。
8. 编写程序, 任意输入一个字符串, 将其中的字符按从小到大的顺序重排。
9. 编写程序, 任意输入一个字符串, 将其中的最大字符放在字符串的第 2 个字符位置, 将最小字符放在字符串的倒数第 2 个字符位置。
10. 编写程序, 任意输入 5 个字符串存放到 2 维数组中, 按字符串的长度从短到长顺序输出它们。
11. 请编写输入以下图案的程序, 图案的行数由输入的值确定。

```

A
BBB
CCCCC
DDDDDD
EEEEEEEEEE

```