第 4 章　数组、指针与字符串

【本章导读】本章在介绍数组的基础上，进一步介绍了指针的使用、字符串的处理和结构体的知识，考虑到有关专业数据结构课程的需要，本章还对链表的相关知识作了简要介绍。

通过本章的学习，要求读者熟练掌握数组与字符串的相关知识，并能利用它们解决实际问题；掌握指针的概念及其与数组、字符串的关系，能用指针解决一些简单问题；了解结构体与链表的关系，能够阅读一些关于结构体与链表知识的程序。本章的难点是指针与数组、字符串的关系；链表的基本操作；常用算法的理解与掌握。

4.1　数组的基本概念

前面已经学习了 C++语言的一些基本数据类型，如整型、实型和字符型等。但仅有这些基本类型，很难满足较复杂情况下的编程需要。

【问题】从键盘接收 10 个数，求平均数并输出小于平均数的数。

分析：从键盘接收 10 个数，求平均数很简单，可以采用简单变量和循环结构相结合的方法，边接收边求和，最后求平均数，程序段如下：

```
float aver=0;
for (i=0;i<10;i++)
{
 cout<<"输入第"<<i+1<<"个数：";
 cin>>a[i];
 aver+=a[i];}
aver=aver/10;            // 求 10 个数的平均数
```

但是输出小于平均数的数就比较麻烦了，因为从键盘接收的 10 个数在求和以后没有保存起来，等再比较比平均数小的数时就无法实现了。若要输出小于平均数的数，必须再重复输入这 10 个数。这样带来两个问题：

（1）输入数据的工作量成倍增加。

（2）若本次输入的数与上次不同，则输出的结果不正确。

要解决此问题，必须使用数组来解决，先将 10 个数保存到数组中去，等求过平均数后再从数组里取出 10 个数进行比较。用数组解决的完整程序如下：

```
//Ex4_0.cpp
#include "iostream"
using namespace std;
void main()
{
 int a[10],i;          //a[10]定义了有 10 个元素的数组
 float aver=0;
 for (i=0;i<10;i++)
```

```
 {
 cout<<"输入第"<<i+1<<"个数：";
 cin>>a[i];
 aver+=a[i];}
 aver=aver/10;                    //求 10 个数的平均数
 cout<<"10 个数是：";
 for(i=0;i<10;i++)                //本循环结构用于输出小于平均数的数
    cout<<a[i]<<"   ";
 cout<<endl<<"平均数是："<<aver<<"   小于平均数的数是：";
 for(i=0;i<10;i++)                //本循环结构用于输出小于平均数的数
  if (aver>a[i]) cout<<a[i]<<"   ";
 cout<<endl;
}
```

程序运行结果如图 4.1 所示。

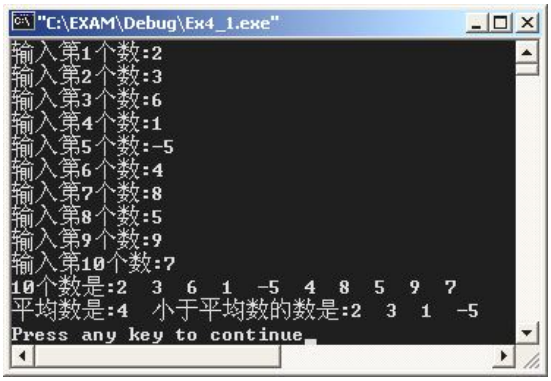


图 4.1　程序运行结果

在程序设计中，常把具有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型数据元素的集合称为数组。数组属于构造类型，在计算机中，一个数组在内存中占用一片连续的存储空间，在程序中用数组名来标识这一数组，而下标指明数组中各元素的序号，用下标变量来标识数组的每个元素。根据下标的个数不同可以把数组分为一维、二维和多维的，本章重点介绍一维数组和二维数组。

数组在使用前必须先定义（数组名、类型、大小、维数），后使用。

4.2　一维数组

4.2.1　一维数组的定义、初始化和引用

1. 一维数组的定义

形式：数据类型 数组名[整型常量表达式];

说明：

（1）数据类型是全体数组元素的数据类型，可以是基本类型或构造类型。

（2）数组名命名规则与变量名相同，遵循标识符表示原则。

（3）一维数组是用一对“[ ]”括起来的整型常量表达式，其中整型常量表达式的值指明了数组的长度（即数组中包含的元素个数），可以包括常量、符号常量，但不能是变量。

（4）数组的起始下标为 0，最大下标值比常量表达式的值小 1。

例如：

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

定义了一个有 10 个元素的 int 型数组 a，有 a[0]、a[1]、…、a[9]十个元素。

而 int a(5);、float k=3,a[k];、char b[3.4]; 等都是错误的。

2. 一维数组的初始化

在定义数组的同时，为数组元素赋初值，称为数组的初始化。

形式：数据类型 数组名[整型常量表达式]={常数 1,常数 2,…,常数 n};

例如：

```
int a[5]={1,3,5,7,9};      //a[0]～a[4] 元素依次为{ }内对应的值
int a[]={1,3,5,7,9};       //未指明数组长度，由初值个数决定为 5，与上述等价
float b[5]={1.0,3.5,50.8,8.1,10};
int c[10]={1,2};           //c[0]、c[1]元素的初值为 1 和 2，其余元素系统自动赋 0
```

注意：以下对数组赋值的语句都是错误的：

```
（1）int c[3]={1,2, 3,4,5};   //常量个数超过数组定义的长度
（2）int a[5];
       a={1,3,5,7,9};      //不允许对数组名赋值
（3）int a[5];
       a[5]={1,3,5};       //a[5]是下标为 5 的元素，不是代表数组，更何况下标也越界
```

3. 一维数组元素的引用

形式：数组名［下标］

说明：

（1）下标为整型常量或整型表达式，表示对应元素在数组中的顺序。

（2）元素可以像其他基本数据类型变量一样参与相应的各种操作。

（3）在 C++中，不对数组下标进行越界检查，即当下标越界时，编译器并不指出错误，程序还可运行，但结果不正确。

一般而言，数组除了作函数参数或对字符数组进行某些操作时可整体引用外（即以数组名的形式单独出现），其他情况下必须以元素的方式引用。

例如，已知有如下数组、变量定义和初始化：

```
int a[5]={1,3,5,7,9},i(2);
则：a[3]=a[0]+a[i];       //a[3]元素的值为 6
    cout<<a[2+i];         //输出 a[4]元素的值为 9
    cout<<a[a[i-1]];      //下标是数组元素，a[1]值为 3，输出 a[3]元素的值为 7
```

4.2.2　一维数组的应用

将数组元素的下标和循环语句的控制变量结合使用，可以极大地提高程序的灵活性，在程序设计中最常使用。

【例 4.1】输入 5 个数，按其逆序输出。

分析：将输入的 5 个数保存到数组 a[0]、a[1]、a[2]、a[3]、a[4]中，其逆序为 a[4]、a[3]、a[2]、a[1]、a[0]。因此只需要通过控制循环变量的增、减即可实现。

程序如下：

```
//Ex4_1.cpp
#include "iostream"
#include "iomanip"
using namespace std;
void main()
{
 int i;
 float a[5];
 for(i=0;i<5;i++) cin>>a[i];          //输入 5 个数
 for(i=0;i<5;i++)                     //原序输出
    cout<<setw(6)<<a[i];
 cout<<endl;
 for(i=4;i>=0;i--)                    //逆序输出
    cout<<setw(6)<<a[i];
 cout<<endl;
}
```

运行结果如图 4.2 所示。

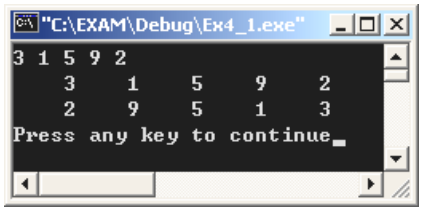


图 4.2　运行结果

【例 4.2】随机产生 10 个在 0～100 之间的数，求其大于平均数的个数、最大数及其位置。

分析：随机函数 rand()产生 0～32767 之间的整数，要产生[a,b]之间的数，可以利用 rand%(b-a+1)+a 来实现，即本题可通过 rand()%101 来达到生成 0～100 数的目的。由于该函数每次运行产生相同的一组数，为此，可利用函数 srand()与 time()配合初始化随机函数，表达式为：srand(time(NULL))。注意使用上述函数需要配合 cstdlib.h 和 ctime.h 头文件。

关于最大数，只需要假设第一个是最大的，然后让其余数与其比较来实现。

程序如下：

```
//Ex4_2.cpp
#include "iostream"
#include "cstdlib"
#include "ctime"
using namespace std;
void main()
{
 int f[10],i,count=0,max,row;
```

```
 float aver=0;
 srand(time(NULL));
 for(i=0;i<10;i++)   //产生 10 个数
      {f[i]=rand()%101; aver+=f[i];}
 aver/=10;
 max=f[0]; row=0;
 for(i=0;i<10;i++)   //输出 10 个数，统计并找最大数
      {
       if (f[i]>aver) count++;
       if (f[i]>max) {max=f[i];row=i;}
       cout<<f[i]<<"     ";
      }
 cout<<"\n 大于平均数("<<aver<<")的个数："<<count;
 cout<<",max="<<max<<",row="<<row+1<<endl;
}
```

运行结果如图 4.3 所示。

图 4.3　运行结果

【例 4.3】用数组方法计算 Fibonacci 数列前 20 项。

```
//Ex4_3.cpp
#include "iostream"
#include "iomanip"
using namespace std;
void main()
{
 int i;
 long f[20]={0,1};
 for(i=2;i<20;i++)     //产生 3 到 20 项
    f[i]=f[i-1]+f[i-2];
 for(i=0;i<20;i++)
 {
  if (i%5==0) cout<<endl;   //每行输出 5 个数
  cout<<setw(5)<<f[i];
 }
 cout<<endl;
}
```

【例 4.4】若有 N 个数的数组 a[N]，用冒泡法按从小到大排序。

算法思想：

（1）从第一个元素开始，对数组中两两相邻的元素比较，即 a[0]与 a[1]比较，若为逆序，则 a[0]与 a[1]交换；然后 a[1]与 a[2]比较，……，直到最后 a[N-2]与 a[N-1]比较，这时最大数

沉底成为数组中的最后一个元素，一些较小的数如同气泡一样上浮一个位置，这是第一趟排序。

（2）然后对 a[0]到 a[N-2]的 N-1 个数进行同（1）的操作，次最大数放入 a[N-2]元素内，完成第二趟排序；依此类推，进行 N-1 趟排序后，所有数均有序。

表 4.1 给出了 5 个数的冒泡法排序过程。

表 4.1　冒泡法排序示例

每一趟参加排序的个数	参加排序的元素					原始数据	4	8	3	1	5
5	a(1)	a(2)	a(3)	a(4)	a(5)	第 1 趟交换	4	3	1	5	8
4	a(1)	a(2)	a(3)	a(4)		第 2 趟交换	3	1	4	5	8
3	a(1)	a(2)	a(3)			第 3 趟交换	1	3	4	5	8
2	a(1)	a(2)				第 4 趟交换	1	3	4	5	8

程序如下：

```
//Ex4_4.cpp
#include "iostream"
using namespace std;
#define N 5
void main()
{
 int i,j,t,a[N]={4,8,3,1,5};
 cout<<"排序前数据：";
 for(i=0;i<N;i++)cout<<a[i]<<" ";
 for (i=1;i<=N-1;i++)                    //有 N 个数需要进行 N-1 趟比较
      {
         for (j=0;j<N-i;j++)
         if(a[j]>a[j+1])                 //进行两个相邻元素比较
             {t=a[j];a[j]=a[j+1];a[j+1]=t;}
       cout<<"\n 第 "<<i<<" 趟排序 ";
       for(j=0;j<N;j++)cout<<a[j]<<" ";  //显示第 i 趟排序结果
      }
 cout<<"\n 排序后结果：";
 for(j=0;j<N;j++)cout<<a[j]<<" ";
 cout<<endl;
}
```

运行结果如图 4.4 所示。

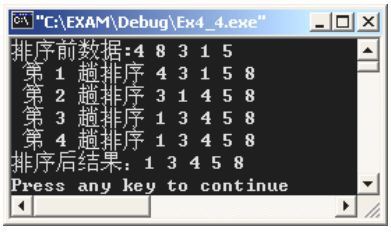


图 4.4　运行结果

【例 4.5】假定有 N 个数，要求用选择法按递增的次序排序。

算法思想：选择法是对冒泡法的改进，它每次在若干个无序数中找最小（大）数，并将该数与本轮第一个数交换。选择法排序共需要进行 n-1 轮，而每轮中最多交换一次。表 4.2 给出了 5 个数的选择法排序过程。

表 4.2　选择法排序示例

每一趟参加排序的个数	参加排序的元素					原始数据	4	8	3	1	5
5	a(1)	a(2)	a(3)	a(4)	a(5)	第 1 趟交换	1	8	3	4	5
4		a(2)	a(3)	a(4)	a(5)	第 2 趟交换	1	3	8	4	5
3			a(3)	a(4)	a(5)	第 3 趟交换	1	3	4	8	5
2				a(4)	a(5)	第 4 趟交换	1	3	4	5	8

注意：数据中有双下线的数表示每一趟找到的最小数的下标位置，与要排序序列中的最左边有单下线的数交换后的结果。

程序如下：

```
//Ex4_5.cpp
#include "iostream"
using namespace std;
#define N 5
void main()
{
 int i,j,t,min,a[N]={4,8,3,1,5};
 cout<<"排序前数据：";
 for(i=0;i<N;i++)cout<<a[i]<<" ";
 for (i=0;i<N-1;i++)                          //有 N 个数需要进行 N-1 趟
     {
      min=i;
      for (j=i+1;j<N;j++)
          if(a[j]>a[min]) min=j;              //找最小数下标
          if (min!=i) {t=a[i];a[i]=a[min];a[min]=t;}
      cout<<"\n 第 "<<i+1<<" 趟排序 ";
      for(j=0;j<N;j++)cout<<a[j]<<" ";        //显示第 i 趟排序结果
      }
 cout<<"\n 排序后结果：";
 for(j=0;j<N;j++)cout<<a[j]<<" ";
 cout<<endl;
}
```

运行结果如图 4.5 所示。

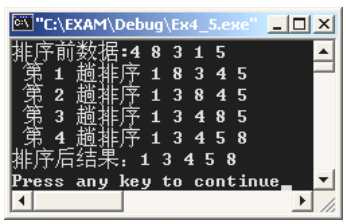


图 4.5　运行结果

4.3 二维数组

如果说一维数组对应一个线性表，那么二维数组就相当于一个矩阵，需要用行、列两个下标来描述。也即当数组元素具有两个下标时，该数组称为二维数组。二维数组可以看做具有行和列的平面数据结构，如矩阵。

4.3.1 二维数组的定义、引用与初始化

1. 二维数组的定义

形式：数据类型 数组名[常量表达式 1][常量表达式 2];

说明：

（1）常量表达式 1 代表了二维数组的行数，常量表达式 2 代表了二维数组的列数。

（2）行列下标都从零开始，其最大下标均比常量表达式的值小 1。

（3）二维数组在内存的排列是以“先行后列”的规则连续存放。

如：float a[2][3];

当定义了一个 a 数组后，它在逻辑上分为 2 行 3 列，各元素如图 4.6（a）所示，在内存中各元素存放形式如图 4.6（b）所示。每个元素在内存的排列序号可通过以下公式计算：

序号=当前行号*每行列数+当前列号

如 a[1][2]元素的序号为：1*3+2=5。

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

（a）逻辑表示

0	1	2	3	4	5
a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]

（b）内存排列顺序

图 4.6 二维数组

2. 二维数组元素的引用

形式：数组名［行下标］[列下标］

说明：为了直观地输入或显示二维数组的逻辑形式，可通过双重循环进行控制。

3. 二维数组的初始化

（1）按在内存中的排列顺序对所有元素赋初值，如：

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

也等价于

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

（2）按行给所有元素赋初值，每一行的数据放于一个花括号内，如：

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

（3）按行给部分元素赋初值，省略的元素初值自动为 0，如：

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

（4）按行赋初值还可以省略第一维的长度，但不能省略第二维的长度，如：

```
int c[ ][3]={{3},{ },{4}};
```

上面定义的二维数组 a、b、c 如图 4.7 所示。

$$a=\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}\qquad b=\begin{pmatrix}1&2&0&0\\0&2&3&0\\0&0&4&0\end{pmatrix}\qquad c=\begin{pmatrix}3&0&0&0\\0&0&0&0\\4&0&0&0\end{pmatrix}$$

图 4.7　二维数组示例

4.3.2　二维数组的应用

【例 4.6】不用输入自动生成并输出如下矩阵：

1　2　3　4
1　1　5　6
1　1　1　7
1　1　1　1

分析：由于不允许输入，所以要设法找到矩阵元素的分布规律，用循环的方式生成。矩阵下三角元素全为 1，而上三角的元素按行的顺序依次是 2、3、4、5、6、7。

程序如下：

```
//Ex4_6.cpp
#include "iostream"
#include "iomanip"
using namespace std;
#define N 4
void main()
{
 int a[N][N],i,j,k=2;
 for (i=0;i<N;i++)
    for(j=0;j<N;j++)
        if (j<=i) a[i][j]=1;
        else a[i][j]=k++;
 for (i=0;i<N;i++)
     {
        for(j=0;j<N;j++)
        cout<<setw(4)<<a[i][j];
        cout<<endl;
     }
}
```

运行结果如图 4.8 所示。

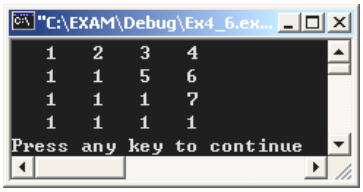


图 4.8　运行结果

【例 4.7】求 4×4 数组中反对角线元素的和、最大元素及下标。

3 6 4 8
9 46 13 7
5 11 76 24
1 2 0 10

分析：与一维数组求最大值的方式相同，区别在于利用两重循环来求得。先假定第一个元素为最大，同时保存其行、列的下标；然后利用两重循环逐一与最大值比较，一旦超过最大值就替换，同时也替换行、列下标。至于反对角线的和对行 i 列 j 满足 i+j=3 元素求和即可。

程序如下：

```
//Ex4_7.cpp
#include "iostream"
using namespace std;
void main()
{
 int a[4][4]={{3,6,4,8},{9,46,13,7},{5,11,76,24},{1,2,0,10}},i,j;
 int max,imax,jmax,sum=0;
 max=a[0][0]; imax=0; jmax=0;
 for (i=0;i<4;i++)
    for(j=0;j<4;j++)
          {
           if (i+j==3) sum+=a[i][j];
           if(a[i][j]>max) {max=a[i][j];imax=i;jmax=j;}
          }
 cout<<"反对角线和="<<sum<<endl;
 cout<<"max("<<imax+1<<","<<jmax+1<<")="<<max<<endl;
}
```

运行结果如图 4.9 所示。

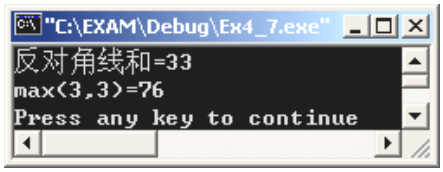


图 4.9 运行结果

【例 4.8】输入两个矩阵 A、B 的值，求 C=A+B，并显示结果。

$$A=\begin{pmatrix}3 & 5 & 7\\ 12 & 13 & 6\end{pmatrix} \quad B=\begin{pmatrix}4 & 8 & 10\\ 6 & 13 & 16\end{pmatrix}$$

分析：（1）数据的输入/输出问题。矩阵 A、B 各元素的输入可通过空格和回车符控制；输出内循环不换行，出了内循环输出 endl 换行控制符。

（2）A、B 矩阵相加，其实质是将两矩阵的对应元素相加。两个矩阵能相加的条件是有相同的行、列数。

程序如下：

```
//Ex4_8.cpp
#include "iostream"
#include "iomanip"
using namespace std;
void main()
{
 int a[2][3],b[2][3],c[2][3],i,j;
 cout<<"输入 A 矩阵："<<endl;
 for(i=0;i<2;i++)
    for(j=0;j<3;j++)
        cin>>a[i][j];
 cout<<"输入 B 矩阵："<<endl;
 for(i=0;i<2;i++)
    for(j=0;j<3;j++)
        cin>>b[i][j];
 for(i=0;i<2;i++)                //a+b 矩阵，每个对应元素相加
    for(j=0;j<3;j++)
       c[i][j]=a[i][j]+b[i][j];
 cout<<"输出 C 矩阵的结果："<<endl;
 for(i=0;i<2;i++)
   {
    for(j=0;j<3;j++)
    cout<<setw(4)<<c[i][j];
    cout<<endl;
   }
}
```

运行结果如图 4.10 所示。

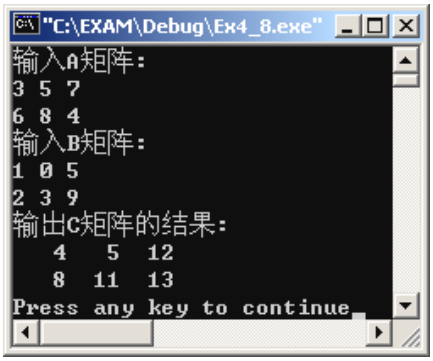


图 4.10　运行结果

4.4　字符数组

C++语言有字符常量和字符变量，有字符串常量，但没有专门的字符串变量，解决的方法

是用系统提供的字符数组、CString（string）类和字符指针 3 种形式处理字符串。本节主要介绍字符数组和字符指针方式，有关 CString（string）类将在第 7 章中介绍。

4.4.1 字符数组的定义

形式：char 数组名[整型常量表达式];

功能：最多可以存放整型常量表达式个字符或整型常量表达式-1 个字符组成的字符串。

字符串是由一对双引号作定界符的若干个有效字符的序列，存储时系统自动在字符序列最后加入一个结束标记符'\0'。字符串是计算机程序中经常处理的数据。字符数组中存放的是字符还是字符串，两者最大的区别是字符串有'\0'结束标记。

4.4.2 字符数组的初始化

1. 逐个字符方式

```
char s[6]={'C','h','i','n','a'};      //不等价 char s[6]={'C','h','i','n','a','\0'};
```

此时数组 s 中存储的是字符，存储形式：

C	h	i	n	a	

2. 字符串常量方式

```
char s[6]={"China"};    //char s[6]="China"; 或 char s[]="China";
```

此时数组 s 中存储的是字符串，存储形式：

C	h	i	n	a	'\0'

4.4.3 字符数组的引用

对字符数组不仅可以引用它的数组元素，也可以引用整个数组。例如：

```
char s[6];
s[0]= 'C'; s[1]= 'h'; s[2]= 'i'; s[3]= 'n'; s[4]= 'a'; s[5]= '\0';
for(int i=0;i<6;i++) cin>>s[i];   //存放字符
cin>>s;                 //存放字符串，以空格、Tab 或回车键结束输入
gets(s);                //存放字符串，以回车键结束输入，但需要使用#include cstring 命令
for(int i=0;i<6;i++) cout<<s[i];   //逐个字符输出
cout<<s;            //输出整个字符串
puts(s);            //输出整个字符串，但需要使用#include cstring 命令
for(int i=0;s[i]!= '\0';i++) cout<<s[i];   //输出整个字符串，不倡导使用
```

当要处理几个相关的字符数组时，可以使用二维数组。二维数组的每一行也可以看做一维字符数组。因此，二维数组也称字符串数组。

4.4.4 字符串处理函数

C++语言提供了一系列的字符串处理函数，这些函数都包含在头文件 cstring 中。常用字符串处理函数如表 4.3 所示。

表 4.3　常用字符串处理函数

函数形式	功能
gets(s)	从键盘接收字符串
put(s)	在显示器上显示字符串
strlen(s)	求字符串 s 的长度（不含结束符'\0'）
strcpy(s1,s2)	把字符串 s2 拷贝到字符串变量 s1 中，s1 的长度不应小于 s2 的长度
strcat(s1,s2)	把字符串 s2 连接到字符串 s1 内容的后面，注意 s1 要足够长
strcmp(s1,s2)	如果 s1 小于 s2，返回-1；如果 s1 等于 s2，返回 0；如果 s1 大于 s2，返回 1
strlwr(s)	把字符串 s 中的大写字母变成小写字母
strupr(s)	把字符串 s 中的小写字母变成大写字母

【例 4.9】输入一个字符串，统计其中的大写字母、小写字母、数字和其他字符的个数。程序如下：

```
//Ex4_9.cpp
#include "iostream"
#include "cstring"
using namespace std;
void main()
{
 char s[80];
 char name[4][10]={"Upper:","Lower:","Digit:","Other:"};
 int i,a[4]={0};
 cout<<"Input a String:";
 gets(s);
 for(i=0;s[i]!= '\0';i++)
      {
       if(s[i]>='A' && s[i]<='Z') a[0]++;
       else if(s[i]>='a' && s[i]<='z') a[1]++;
       else if(s[i]>='0' && s[i]<='9') a[2]++;
       else a[3]++;
      }
 for(i=0;i<4;i++) cout<<name[i]<<a[i]<<endl;
}
```

运行结果如图 4.11 所示。

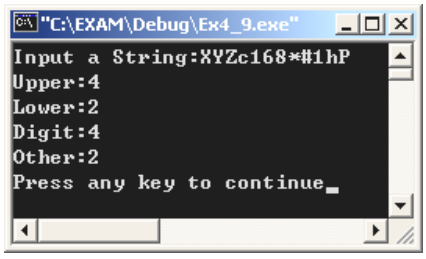


图 4.11　运行结果

【例 4.10】输入 5 个字符串，将其中最小的字符串输出。

```
//Ex4_10.cpp
#include "iostream"
#include "cstring"
using namespace std;
void main()
{
 char str[10],minstr[10];
 int i;
 gets(minstr);
 for(i=0;i<4;i++)
 {
  gets(str);
  if(strcmp(minstr,str)>0) strcpy(minstr,str);
 }
 cout<<"MinStr Is: "<<minstr<<endl;
}
```

运行结果如图 4.12 所示。

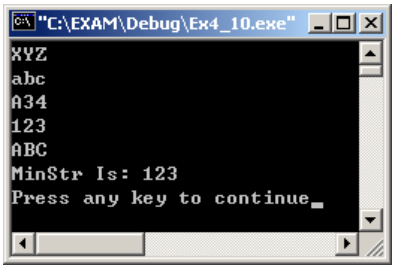


图 4.12　运行结果

4.5　指针与数组

在 C++中，指针是一种重要的数据类型，它是一个变量在内存中所对应存储单元的地址。指针是 C++语言的特色也是难点，指针变量就是存放地址的变量。合理地使用指针可以获得高质量的目标代码，提高程序的执行效率。

4.5.1　指针

1. 变量的地址

高级语言中的变量具有 3 个属性：变量名、变量的值和变量的地址。在编写程序时使用变量名；在程序运行时要将变量的值存入系统为该变量名分配的一定长度的内存单元中，该内存单元第一个字节的编号称为地址；进行运算时，要取出变量在对应内存单元中存放的值，参加各种运算，计算结果最后还要存入变量对应的内存单元中。

需要注意的是：在程序设计阶段一个变量的地址是无法确定的，但是在程序运行阶段可以通过运算符&求得。

（1）求地址运算符&。

形式：&变量

功能：返回一个变量的地址。&是一个单目运算符，如图 4.13 中变量 x 的地址是 1000，因此&x 的值为 1000。需要注意的是，&不能作用于常量或表达式，&(x*x)和&4 都是错误的。

（2）取内容运算符*。

形式：*地址

功能：*运算符的作用是根据地址取内容。

2. 指针变量的定义

指针变量与一般变量一样，使用之前必须定义。

形式：数据类型 *标识符;

说明：

（1）*后面的标识符是指针变量名，*本身不是变量名的组成部分。

（2）指针变量定义中的*与执行语句中的*的意义不同，执行语句中的*表示取内容。

（3）数据类型不是指针变量本身的类型，而是指针变量所指的对象的数据类型。

注意：指针就是地址，指针变量就是存放地址的变量，有时也把指针变量简称指针。

3. 指针变量的初始化

指针变量同其他类型的变量一样，也可以在定义的同时赋初值，称为指针变量的初始化。例如：

```
int x=3, *p=&x;     //把变量 x 的地址赋给 p
```

等价于：

```
int x=3, *p;
p=&x;
```

假设变量 x 的地址是 1000，指针变量 p 的地址是 2000，则变量 x 与指针变量 p 的关系如图 4.13 所示。由于指针变量 p 中存放的是变量 x 的地址，这时可称 p 指向变量 x。

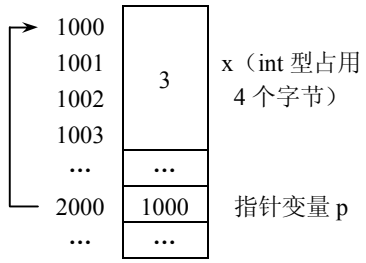


图 4.13　用指针访问变量示意图

当一个变量被一个指针变量指向后，则访问该变量有两种形式：

（1）通过变量名访问，被称为直接访问。

（2）通过指针访问，被称为间接访问。

例如，在图 4.13 中，指针 p 指向变量 x，因此访问变量 x 的两种形式为：x 和*p，即 x 与*p 等价，也就是说 x 和*p 的值都是 3。

4. 指针运算

（1）赋值运算。

为指针变量赋值，使其有明确的指向，是使用指针变量的前提。常见的对指针赋值的方式有：

1）把符号常量 NULL 赋给指针，例如：

int *p=NULL; //执行后 p 不指向任何对象

2）把一个变量的地址赋给指针，例如：

int x, *p=&x; //执行后 p 指向 x

3）把一个指针的值赋给同类型的另一个指针，例如：

int x, *p=&x, *q; q=p; //执行后 p 和 q 指向同一个对象

（2）++和--运算。

形式：++p　　p++　　--p　　p--

功能：指针变量 p 的值变成后一个变量或前一个变量的地址。

注意：

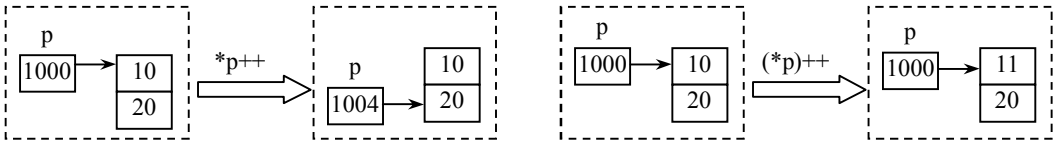
1）并不表示指针的值加 1 或减 1。例如：

int a=2,*p=&a;

假设变量 a 的地址为 1000，当执行 p++后，p 指向变量 a 后面的那个 int 类型的变量，因为 a 在内存中占 4 个字节，故 a 后面那个变量的地址为：1000+4=1004，所以执行 p++后 p 的值为 1004。

2）*(p++)、*(++p)与(*p)++的含义。

*(p++)表示先取*p，再使 p 加 1，p 变化，由于++运算符优先于*，因此*(p++)等价于*p++；*(++p) 表示先使*p 加 1，再取*p，p 变化；(*p)++表示*p+1，p 未变化。有关它们的区别如图 4.14 所示。



（a）执行*p++的++运算效果　　（b）执行(*p)++的++运算效果

图 4.14　*p++与(*p)++的区别

（3）p+n 和 p-n 运算（p 为指针，n 为整数）。

p+n 或 p-n 指向当前所指的那个变量的后面（或前面）第 n 个变量。例如：

int x,*p=&x;

假设 x 的地址为 1000，则 p 的值为 1000，p+5 的值为 1000+5*4=1000+5*4 =1020。

注意：计算 p+n 或 p-n 后 p 本身的值没有变。

（4）指针相减。

两个指向同一类型的指针可以相减，结果是这两个地址差之间能够存放所指类型的数据个数。例如：

int *p1,*p2;

假设 p1 指向 1000，p2 指向 1008，则 p2-p1 的值为(1008-1000)/4=2。

注意：一般来说，只有当两个指针类型相同且指向同一个数组中或用 new 申请的一块连

续存储空间时，它们之间的相减才有意义。

（5）关系运算。

两个指向同一类型的指针还可以进行>、>=、<、<=、==、!=关系运算。

【例 4.11】执行下列程序，分析运行结果。

```
//Ex4_11.cpp
#include "iostream"
using namespace std;
void main()
{
 int a,b,*p1=&a,*p2=&b,*p;
 cout<<"Input a    b:";
 cin>>a>>b;
 cout<<"a="<<a<<",b="<<b<<endl;
 //输出交换前 p1 和 p2 所指向的变量的值
 cout<<"*p1="<<*p1<<",*p2="<<*p2<<endl;
 p=p1;p1=p2;p2=p;     // 交换两个指针
 cout<<"a="<<a<<",b="<<b<<endl;
 //输出交换后 p1 和 p2 所指向的变量的值
 cout<<"*p1="<<*p1<<",*p2="<<*p2<<endl;
}
```

运行结果如图 4.15 所示。

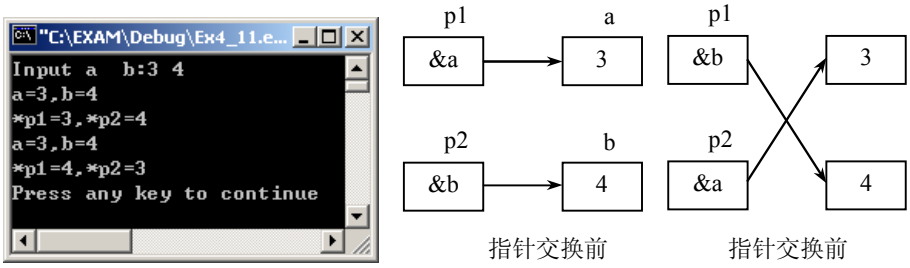


图 4.15　运行结果

4.5.2　动态存储空间

1. new 运算符

new 运算符用于向系统申请动态存储空间，并把首地址作为运算结果。其使用形式为：

指针=new 数据类型;　　// 申请一个指定类型的变量，并把首地址赋给指针

指针=new 数据类型(初值);　　// 申请变量的同时赋初值

例如，语句 int *p=new int; 向系统申请了一个 int 变量（4 个字节），并把首地址赋给 p，因而*p 代表这个动态变量。

2. delete 运算符

delete 用于释放用 new 申请的动态存储空间，其使用形式为：

delete 指针;　　// 释放指针所指的那个变量

或

delete []指针;　　// 释放指针所指的那个数组，不必指明数组长度

4.5.3 指针与一维数组

指针变量可以指向基本类型变量，还可以指向任何一种数据类型，最常用的是指向数组和数组元素。一个数组在内存占有一片连续的存储区域，数组名就是这块存储区域的首地址，当指针变量指向该数组后，可以通过指针变量来引用数组元素。

例如，有如下变量定义：

int a[5]={2,3,4,5,6},*p;

当执行了 p=a;或 p=&a[0]; 语句后，使得指针变量指向了 a 数组的首地址，p 与数组 a 的关系如图 4.16 所示。

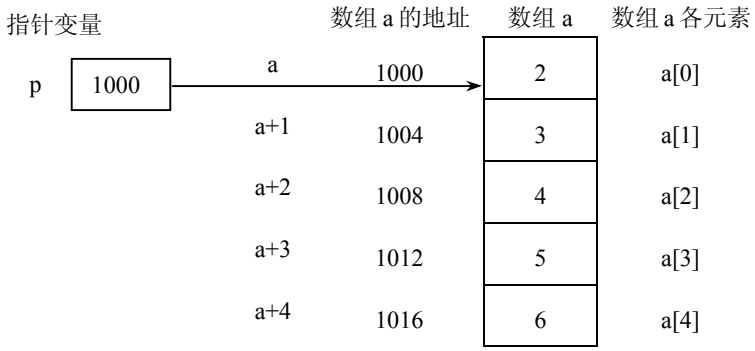


图 4.16 指针变量与数组的关系

由此可见，p 与数组 a 有如下的等价关系：

（1）p≡a≡&a[0]：表示数组元素的首地址。

（2）p+i≡a+i≡&a[i]：表示第 i+1 个数组元素的地址。

（3）*(p+i)≡*(a+i)≡a[i]：表示第 i+1 个数组元素。

通常将 a[i]、*(a+i)、*(p+i)表示的数组元素和&a[i]、a+i、p+i 表示的数组元素地址称为引用（或访问）数组元素的 3 种方式：

（1）下标方式：数组名[下标]。

（2）地址方式：*(地址)。

（3）指针方式：*指针变量。

上述 3 种方法中，下标方式和地址方式引用速度慢，指针方式速度最快，但不够直观。

【例 4.12】试编程用 3 种方式访问数组中的所有元素。

程序如下：

```
//Ex4_12.cpp
#include "iostream"
using namespace std;
void main( )
{
 int a[5]={2,3,4,5,6},*p,i;
 cout<<"下标方式：";
 for(i=0;i<5;i++)
     cout<<"a["<<i<<"]="<<a[i]<<" ";
```

```
 cout<<endl<<"地址方式：";
 for(i=0;i<5;i++)
     cout<<"*(a+"<<i<<")="<<*(a+i)<<" ";
 cout<<endl<<"指针方式：";
 for(p=a,i=0;i<5;i++)     //没有改变 p 的值
    cout<<"*(p+"<<i<<")="<<*(p+i)<<" ";     //*(p+i)也可改为*p++
 cout<<endl;
}
```

运行结果如图 4.17 所示。

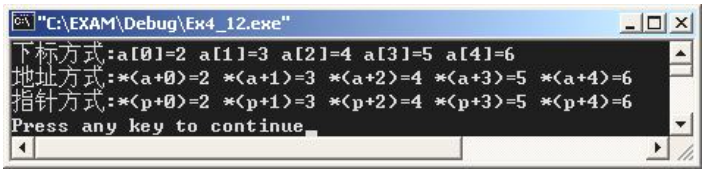


图 4.17　运行结果

注意：p 与 a 同样是表示数组 a 的地址，在表现形式上可以互换，但它们的本质是不同的，p 是地址变量，而 a 是地址常量。所以可以对 p 进行 p++、p--等赋值运算，改变 p 的值；但不能对 a 进行 a++、a--等赋值运算。

4.5.4　指针与二维数组

用指针可以指向一维数组，也可以指向二维数组。但在概念和使用上，要比一维数组复杂。

例如：

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

a 可以看成是由两个元素 a[0]和 a[1]构成的一维数组，a[0] 和 a[1]又可以分别看成是由 a[0][0]、a[0][1]、a[0][2]和 a[1][0]、a[1][1]、a[1][2]三个元素组成的一维数组，每个元素指向本行的首地址。这样，a[0] 和 a[1]可以理解为一个指向 int 类型的一级指针常量，而 a 就是一个指向 int 类型的二级指针常量。

1. 指针变量引用数组元素

根据指针变量引用一维数组元素的原则，要定义指针变量引用二维数组，就必须让指针变量指向二维数组的开始。

假设有定义：

```
int a[2][3],*p=&a[0][0];     //*p= a[0]
```

则由于二维数组在内存中也是线性连续存放，因此从 p 的角度来说，数组 a 是由 6 个元素组成的数组，通过 p 指针依次来引用二维数组元素，如图 4.18 所示。

例如要显示二维数组的各元素，可用如下语句：

```
for (i=0;i<6;i++)
{
 cout<<*p++<<"   ";
 if(i%3==2) cout<<endl;
}
```

```
for(i=0;i<6;i++)
{
   cout<<p[i]<<"   ";
   if(i%3==2) cout<<endl;
}
```

```
for(i=0;i<2;i++)
{
   for(j=0;j<3;j++)
     cout<<*(p+i+j)<<"   ";
     cout<<endl;
}
```

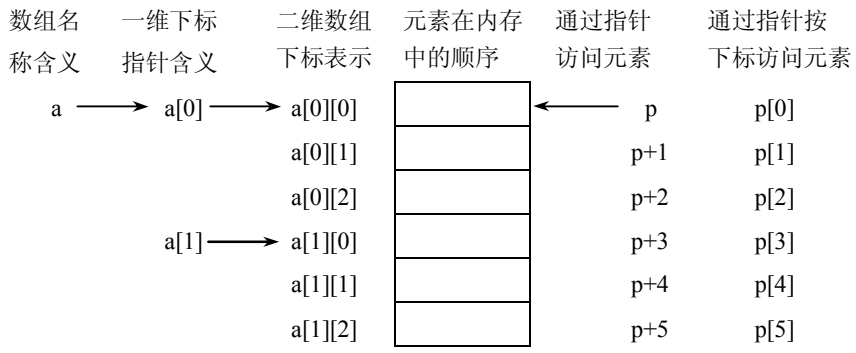


图 4.18　指针变量与二维数组

注意：

（1）由于 a 是二维数组，经过两次下标运算之后才能访问到数组元素，所以*p=*a、*p=a[0]与*p=&a[0][0]等价，而*p=a 是错误的。

（2）如果 p 是一个指向二维数组的指针变量，则还可以通过 p 的下标形式访问数组元素。

2. 指针数组引用数组元素

指针数组就是数组中的每个元素是指针。由于 a[0]、a[1]是二维数组每行的首地址，是一级指针数组，所以也可以定义指针数组来引用数组元素。

例如：

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

要引用 a[i][j]元素，可用指针数组表示：

指针数组方式	直接地址方式
*(p[i]+j)	*(a[i]+j)
((p+i)+j)	*(*(a+i)+j)

注意：（1）a[i]是地址常量，p[i]是地址变量。

（2）C++把重点放在面向对象的概念和处理上，对于指针的概念可以淡化一些。

【例 4.13】用指针数组方式显示 a 数组的所有元素。

程序如下：

```
//Ex4_13.cpp
#include "iostream"
using namespace std;
void main( )
{
 int a[2][3]={1,2,3,4,5,6},*p[2]={a[0],a[1]},i,j;
 for(i=0;i<2;i++)
      {
       for(j=0;j<3;j++)
          cout<<*(*(p+i)+j)<<" ";
       cout<<endl;
      }
}
```

4.5.5　指针与字符串

在第 3 节里已经学习了字符数组与字符串，字符指针也可以指向字符串，并可以用字符串常量对字符指针进行初始化。

1. 字符指针与字符数组

例如：

char s [6]= "China",*p="China";

s 是字符数组，最多可存放 6 个字符或 5 个字符组成的字符串；p 是指向字符类型的指针变量，存放的是字符串的地址。字符数组 s 和字符指针 p 都可以处理字符串，它们的区别是：p 是一个变量，可以改变 p 使它指向不同的字符串，但不能改变 p 所指向的字符串常量；s 是一个数组名，是一个常量，可以改变数组中保存的内容，具体如表 4.4 所示。

表 4.4　字符数组与字符指针的区别

	字符数组	字符指针
定义	char s [6];	char *p;
初始化	char s[]= "China";	char *p="China"; 或 char s[]= "China",*p=s;
赋值	s[0]='C'; s[1]='h';	p="China";
输入	cin>>s;或 gets(s);	p=new char[6]; cin>>s;或 gets(s);
运算	s 是一个常量，不能进行++、--运算	p 是一个变量，可以进行++、--运算

【例 4.14】输入一串字符存储在字符数组中，用指针方式逐一显示字符，并求其长度。

程序如下：

```
//Ex4_14.cpp
#include "iostream"
using namespace std;
void main()
{
 char s[80],*p;
 cout<<"请输入字符串：";
 gets(s);
 p=s;    //p 指向数组的第一个元素
 cout<<"输出字符串：";
 while(*p!='\0')
     cout<<*p++;   //指针下移，直到 p 指向字符串结束符'\0'
 cout<<endl<<"字符串长度："<<p-s<<endl;   //利用 p-s 指针相减求得字符串长度
}
```

运行结果如图 4.19 所示。

图 4.19　运行结果

2. 字符指针数组

字符指针数组主要用在多字符串的处理上。用二维字符数组处理多字符串问题要求各行的列数相等，比较浪费存储空间，用指针数组就没有这个问题。

（1）字符型指针数组可以通过初始化取得一批常量字符串的首地址，例如：

char *ps[4] ={"China","Japan","Korea","Australia"};

由于指针数组的每一个元素指向一个字符串常量，这样就大大节约了存储空间，如图 4.20 所示。

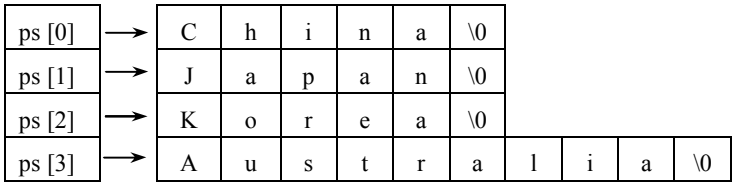


图 4.20　指针数组的元素存储字符串首地址

（2）用指针数组元素访问字符串，可以利用循环，大大提高了程序的效率。例如：

for(k=0; k<4;k++) puts(ps[k]);

（3）通过调整指针数组元素的指向，也可以对字符串进行排序。

【例 4.15】利用指针数组对多字符串进行字典排序。

程序如下：

```
//Ex4_15.cpp
#include "iostream"
#include "cstring"
using namespace std;
void main( )
{
  int i, j;
  char *ps[4]={"China","Japan","Korea","Australia"},*p;
  cout<<"order strings:"<<endl;
  for(i=0;i<4;i++) cout<<ps[i]<<endl;
  for(i=0;i<3;i++)       //对字符串进行排序
     for(j=i+1; j<4; j++)
        if(strcmp(ps[i],ps[j])>0){p=ps[i];ps[i]=ps[j];ps[j]=p;}
  cout<<"reordered strings:"<<endl;
  for(i=0;i<4;i++) cout<<ps[i]<<endl;
}
```

程序运行结果如图 4.21 所示。

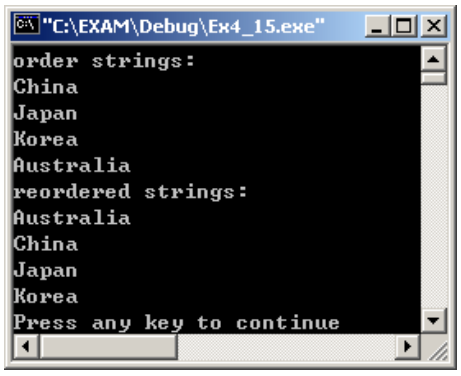


图 4.21　运行结果

注意：用指针数组对字符串排序只是改变了指针数组各元素的指向，并没有改变原来各字符串的存储顺序。

4.6　结构与链表

数组是由相同类型的数据元素组成，而结构是由一组不同的数据类型构成。结构类型为处理复杂的数据提供了便利的手段。如一个学生的信息就是由姓名、性别、出生日期、籍贯、成绩等不同类型的数据共同组成，它如同数据库表的结构。结构与数组都是构造类型，也十分相似，都是由若干分量组成，结构中的分量称为结构成员。

4.6.1　结构

在程序中使用结构之前，首先要对结构的组成进行描述，称为结构类型的定义。

1. 结构类型定义

形式：

```
struct  结构类型标识符
{
   结构成员 1;
   结构成员 2;
      ⋮
   结构成员 n;
};
```

说明：

（1）说明结构类型的关键字是 struct，其后的结构类型标识符是所定义的结构类型名。

（2）结构类型是一种混合的数据类型，其成员可为任意类型，只有自身的实例除外。

例如，描述一个人的出生日期可以说明下面的结构类型：

```
struct date
{
   int month;
```

```
    int day;
    int year;
};
```

该结构类型的变量又可以作为其他结构类型的成员。例如：

```
struct man
{
    char name[15];
    char sex;
    int age;
    struct date birthday;
};
```

以上说明了一个结构类型 struct man，其中有一个成员 birthday 又是 struct date 类型的变量，其结构如图 4.22 所示。

name	sex	age	birthday		
			month	day	year

图 4.22　结构类型 struct man

2. 结构类型变量的定义

从语法上来说，用户自定义的结构类型的地位等同于 char、int、float、double 等标准类型。结构类型本身并不占用存储空间，只有结构变量才需要分配存储空间。结构变量的成员在内存中占用一片连续存储区域，其大小为结构中每个成员的长度之和。C++语言中提供了 3 种利用已说明的结构类型定义结构变量的方法。

（1）说明结构类型的同时定义结构变量，如：

```
struct man
{
    char name[15];
    char sex;
    int age;
    struct date birthday;
}m1;
```

以上在说明结构类型 man 的同时定义了一个该类型的变量 m1。

（2）省略类型名，直接定义结构变量，如：

```
struct
{
    char name[15];
    char sex;
    int age;
    struct date birthday;
}m[3];
```

此处定义了一个包含 3 个元素的结构数组，其中每个元素都是一个包含若干个成员的结构类型数据。C++语言中允许定义这种无类型名的变量。

（3）说明结构类型之后，再定义该类型的变量，如：

struct man m2,*p;

该语句定义了一个 struct man 类型的变量 m2 和指向结构类型的指针变量 p。

3．结构类型变量的引用

（1）普通结构变量成员的访问。

形式：结构变量名.成员名

其中，“.”是成员访问运算符，是优先级最高的运算符之一。如果结构变量的成员本身也是结构类型，则需要使用多个“.”运算符来实现。

例如，对前面定义过的 m1 变量进行赋值：

strcpy (m1.name, "Wang Min");
m1.sex = 'F';
m1.age = 24;
m1.birthday.month = 8;
m1.birthday.day = 10;
m1.birthday.year = 1985;

（2）通过指针访问结构变量的成员。

形式：(*指向结构的指针).成员名

或

指向结构的指针->成员名

注意：不能省略圆括号。运算符“->”与“.”运算符相同，具有所有运算符中最高的优先级。假设有定义：

struct man1 m,*p=&m;

则可以这样对结构变量 m 的成员赋值：strcpy(p->name, "Wang Min");。

【例 4.16】用结构数组保存学生数据并显示。

程序如下：

```
//Ex4_16.cpp
#include "iostream"
#include "cstring"
using namespace std;
struct person
      {
       char name[10];          //姓名
       bool sex;               //性别
       int age;                //年龄
       float score;            //成绩
      }a[10];
void main()
{
 int n,i,k;
```

```
    person x;
    cout<<"请输入一个正整数（1～n）：";
    cin>>n;
    cout<<"从键盘输入具有 person 结构的"<<n<<"个记录："<<endl;
    for(i=0;i<n;i++)
        {
         cin>>x.name;
         cin>>k;
         if (k==1) x.sex=true; else x.sex=false;   //1 为男，非 1 为女
            cin>>x.age>>x.score ;
         a[i]=x;     //结构赋值
         }
    cout<<"显示具有 person 结构的"<<n<<"个记录："<<endl;
    for(i=0;i<n;i++)
       {
        cout<<a[i].name<<' ';
        if (a[i].sex==true) cout<<"man"<<' ';
        else cout<<"woman"<<' ';
        cout<<a[i].age<<' '<<a[i].score<<endl;
       }
}
```

运行结果如图 4.23 所示。

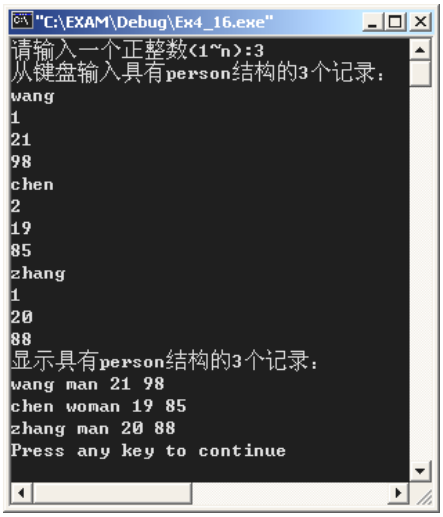


图 4.23　运行结果

4.6.2　链表基础

结构类型的重要应用之一就是链表。现实生活中存在大量需要动态存储和表示的数据，如排队、数据排序等。链表是一种动态数据结构，所谓“动态数据结构”是指在程序运行过程中，数据结构的规模可以根据实际的需要动态变化，从而可以克服数组在使用前必须确定大小的问题。

1. 链表的概念

在结构类型中有一种特殊类型，它除了包含有一般的数据域以外，还包含一个指向自身结构的指针。这种类型的对象又称节点，每个节点的指针域用来指向下一个节点，由此形成一个链表，如图 4.24 所示。

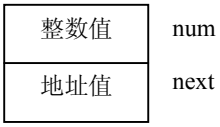


图 4.24　struct item

例如：

```
struct item
{
   int num;
   struct item *next;     //链指针
};
```

2. 线性链表

线性链表是用链指针链在一起的自引用结构的线性集合，如图 4.25 所示。

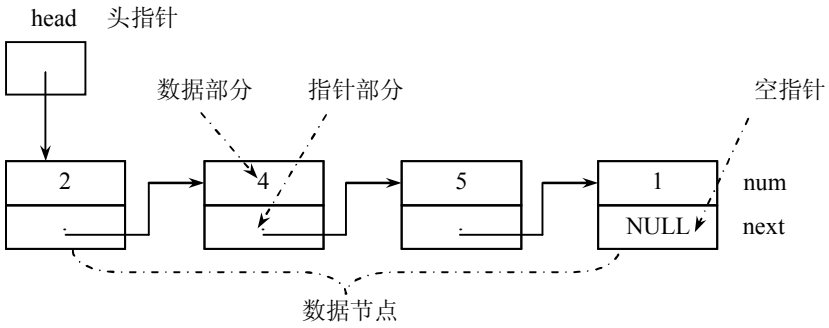


图 4.25　线性链表

在一个链表中，指向第一个节点的指针称为表头指针，第一个节点又称为表头节点，每个节点的指针域所指向的节点称为该节点的后继节点，而该节点又称为后继节点的前趋节点，链表中的第一个节点无前趋节点，最后一个节点（又称为表尾节点）无后继节点。

链表的常见形式有：

（1）单向链表（以正向链表为例），如图 4.25 所示。

（2）双向链表，如图 4.26 所示。

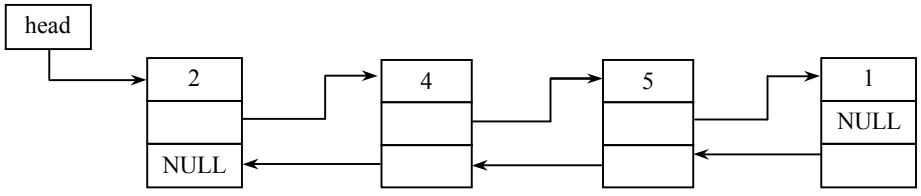


图 4.26　双向链表

（3）循环链表（以单向循环链表为例），如图 4.27 所示。

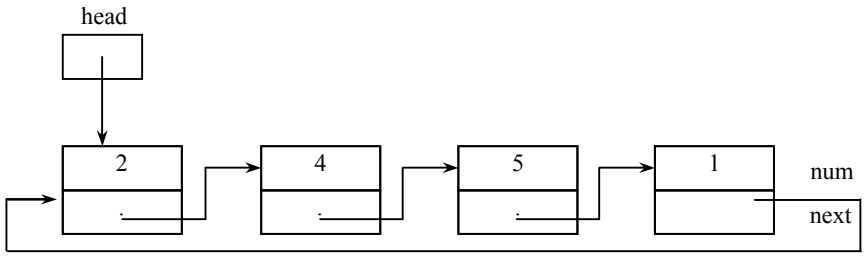


图 4.27　单向循环链表

对链表的主要操作有以下几种：

（1）建立或生成一个链表。

（2）查找链表元素。

（3）插入一个新元素。

- 插表头：将每一新节点固定插入至当前已生成链表的表头位置。
- 插表尾：将每一新节点固定插入至当前已生成链表的表尾位置。
- 有序表：将每一新节点插入至链表中的适当位置（表头、表中或表尾），以保持链表的有序性。

（4）删除一个链表节点。

（5）遍历链表，输出链表长度及节点的有关信息。

【例 4.17】输入简化的学生数据，把它们组成线性链表，并输出各学生信息。要求每一新节点都插入至已生成链表的表头位置。

分析：（1）生成链表：依次键入学号（假设互不相同），创建新的节点，并插入至表头位置。遇结束标志即止，此时已生成所需链表。

（2）遍历链表：从表头开始，“顺藤摸瓜”遍历所有节点并输出各节点中的有关信息——学号，直至表尾。

程序如下：

```
//Ex4_17.cpp
#include "iostream"
#include "cstring"
#include "iomanip"
using namespace std;
void main( )
{
 struct item
    {
      int num;
      item *next;
    };
 item *head=NULL,*temp;   //初始表头为 NULL
 int stno;
 cout<<"Enter students' numbers,-1 to stop:"<<endl;
```

```
 cin>>stno;
 while(stno!=-1)
     {
      temp=new item;        //申请新的节点
      temp->num=stno;       //向节点保存数据
      temp->next=head;      //初始表尾为 NULL，以后用于节点的连接
      head=temp;            //更新表头
      cin>>stno;            //输入新数据
     }
 cout<<"The output will be:"<<endl;
 temp=head;                          //非空表可略
 while(temp!=NULL)
     {cout<<setw(6)<<temp->num;
      temp=temp->next;}
 cout<<endl;
}
```

运行结果如图 4.28 所示。

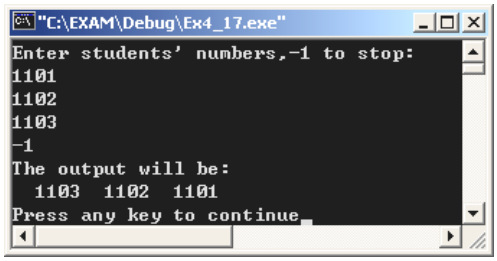


图 4.28　运行结果

思考：如要求每一新表元素都插入至已生成链表的表尾位置，该如何修改程序？

4.7　常用算法

1. 统计

【例 4.18】有一个协会在换届选举中由全体人员以无记名投票方式直选主席，共有 5 名候选人，每个人的代号分别用 1、2、3、4、5 表示。每名会员填写一张选票，若同意某名候选人则在其姓名后打上钩。试编写一计票程序。

分析：由于需要统计 5 位候选人的票数，可使用数组记录，为了方便处理，建立一个有 6 个元素的数组 a，使下标与候选人代号对应（下标是 0 未用），这样输入 2 就让 a[2]加 1。由于不知选票数目，而编号又没有-1，可以考虑用-1 作为终止标志。根据分析，编写程序如下：

```
//Ex4_18.cpp
#include "iostream"
using namespace std;
void main()
{
 int a[6]={0},i;
```

```
 cout<<"请依次输入每张选票所投候选人的代码："<<endl;
 cin>>i;
 while (i!=-1)
      {
       if (i>=1 && i<=5) a[i]++;
       cin>>i;
      }
 for(i=1;i<=5;i++)
    cout<<i<<": "<<a[i]<<endl;
}
```

运行结果如图 4.29 所示。

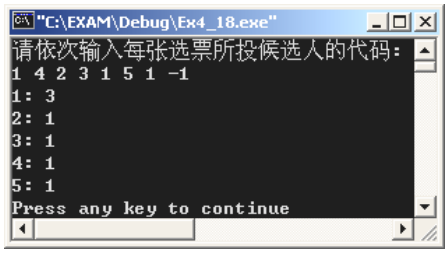


图 4.29　运行结果

【例 4.19】输入一行字符，统计其中有多少个单词（规定单词之间用空格隔开）。

分析：单词的数目可以由空格出现的次数决定（连续的若干空格算一个，一行开头的空格不统计）。根据分析，编写程序如下：

```
//Ex4_19.cpp
#include "iostream"
using namespace std;
void main()
{
 char str[80],c;
 int i,num=0,word=0;       //确保输入的 m 为正整数，r 在 2～16 之间
 gets(str);
 cout<<"原字符串："<<str<<endl;
 for(i=0;(c=str[i])!='\0';i++)
      if(c==' ') word=0;
      else if(word==0) {word=1;num++;}
 cout<<"单词数为："<<num<<endl;
}
```

运行结果如图 4.30 所示。

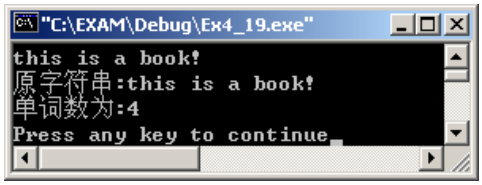


图 4.30　运行结果

2. 数值计算

【例 4.20】编程输出以下的杨辉三角形（输出前 10 行）。

```
1
1    1
1    2    1
1    3    3    1
1    4    6    4    1
1    5    10   10   5    1
1    6    15   20   15   6    1
……
```

分析：观察上面的杨辉三角形，可找到以下规律：

（1）第一列及对角线元素均为 1。

（2）其他元素为其所在位置的上一行对应列和上一行前一列元素之和，即：a[i][j]=a[i-1][j-1]+a[i-1][j]。

程序如下：

```
//Ex4_20.cpp
#include "iostream"
#include "iomanip"
using namespace std;
#define N 6
void main()
{
 int i,j,a[N][N];
 for(i=0;i<N;i++)
 {
  a[i][i]=1;a[i][0]=1;}          //对 a 数组的第一列和对角线元素赋值为 1
  for(i=2;i<N;i++)               //对除第一列和对角线之外的元素赋值
    for(j=1;j<=i-1;j++)
        a[i][j]=a[i-1][j-1]+a[i-1][j];
  for(i=0;i<N;i++)
     {
      for(j=0;j<=i;j++)       //注意条件 j<=i 表示只输出 a 数组的左下三角形部分
      cout<<setw(5)<<a[i][j];
      cout<<endl;
   }
}
```

3. 删除（插入）数据

【例 4.21】随机生成 10 个互不相同的 1～100 之间的整数放在一维数组中，找出值最大的元素，并从数组中删除该值。

分析：该题涉及两个问题：

（1）随机产生若干个互不相同的数，这通过随机数每产生一个数 x，然后 x 和数组中已有的数比较，若有相同，x 数作废，重新产生，否则该数放入数组中。重复上述过程直到产生

满 10 个数。

（2）删除操作首先从 10 个数中找最大值的下标位置 k ，然后从 k+1 到最后逐一向前移动。

根据分析，编写程序如下：

```
//Ex4_21.cpp
#include "iostream"
using namespace std;
void main()
{int a[10],i, k,maxi,x;
 a[0]=rand()%100+1;          //产生 1～100 之间的随机数，作为第 0 个元素
 for(i=1;i<10;i++)           //通过程序自动形成有 9 个元素、有规律的数组
     {
      re: x=rand()%100+1;
      for(k=0;k<i;k++)
         if(x==a[k]) goto re;      //找到有相同的数，重新产生数
      a[i]=x;   //无相同值元素，产生的数放入数组对应的元素位置中
     }
 cout<<"原始数据:";
 for(i=0;i<10;i++)           //产生的 10 个互不相同的元素
    cout<<a[i]<<" ";
 maxi=0;
 for(i=1;i<10;i++)           //查找最大值的位置 maxi
    if(a[maxi]<a[i]) maxi=i;
 cout<<endl<<"删除最大值："<<a[maxi];
 for(i=maxi;i<9;i++)   //从最大值后面的元素往前移一位
    a[i]=a[i+1];
 cout<<"后的数据：";
 for(i=0;i<9;i++)        //输出前 9 个元素
    cout<<a[i]<<" ";
 cout<<endl;
}
```

运行结果如图 4.31 所示。

图 4.31　运行结果

如果在一组有序数据中插入一个数，使这组数据仍旧有序。插入的基本思想是：

（1）查找待插入数据在数组中的位置 k。

（2）从最后一个元素开始往前直到下标为 k 的元素依次往后移动一个位置。

（3）第 k 个元素的位置空出，将要插入的数据插入。

4. 数制转换

【例 4.22】编写一个程序，实现将一个十进制正整数 m 转换成 2～16 的 r 进制的字符串。

分析：将一个十进制正整数 m 转换成 r 进制数的思路是：将 m 不断除 r 取余数（若余数超过 9，还要进行相应的变换，例如 10 变换成 A、11 变换成 B 等），直到商为 0，以反序得到结果，即最后得到的余数在最高位。根据分析，编写程序如下：

```
//Ex4_22.cpp
#include "iostream"
using namespace std;
void main()
{
 int i=0,r,n,c[10];
 long int m;
 char b[17]="0123456789abcdef";
  do
     {
      cout<<"输入十进制数 m 和 r 进制基数：";
      cin>>m>>r;
     }while (m<0||r<2||r>16 );     //确保输入的 m 为正整数，r 在 2～16 之间
 cout<<"十进制数"<<m<<"转换为"<<r<<"进制数,";
 do
     {c[i++]=m%r;}     //取余数
 while((m=m/r)!=0);
 cout<<"结果为：";
 for(--i;i>=0;--i)
     {n=c[i];cout<<b[n];}     //将余数转换成对应的字符
 cout<<endl;
}
```

运行结果如图 4.32 所示。

图 4.32　运行结果

知识要点

1. 数组

数组是具有相同类型的数据的集合。根据下标的个数可将数组分为一维数组、二维数组、三维数组等。数组的定义语句为：

类型说明符　数组名[常数表达式][…][…];

引用数组就是引用数组的各元素，可以通过下标的变化引用任意一个数组元素。需要注意的是，不要进行下标越界的引用，那样会带来意外的副作用。

同一个数组的所有元素在存储器中占用一片连续的存储单元。

C++可以使用字符型数组存放字符串数据并实现有关字符串的操作。字符串输出是从指定

的地址开始输出，直到遇到字符串结束符‘\0’为止，因此，定义字符数值时一定要留一个数组元素存放‘\0’。此外，C++还提供了许多字符串处理函数。使用这些函数，可以提高字符串处理的效率。

2. 指针

指针是 C++语言中的一种数据类型，是专门用来处理地址的。我们把用来存放地址的变量称为指针变量。定义指针变量的一般格式是：

类型说明符 *指针变量名表;

指针在定义后必须初始化才能使用；否则结果不正确。此外，还要注意指针的引用、基本运算及指针与变量的关系。

任何能由数组下标完成的操作都可以由指针来实现，用指针来处理数组及元素是最快捷的方式。要搞清楚数组名所代表的地址常量和指向数组元素的指针变量之间的本质区别和操作异同。

3. 动态存储空间

使用动态存储分配机制可以使程序在运行时根据具体情况灵活调整存储分配情况，它是使用指针、运算符 new 和 delete 来完成的。

4. 结构体与链表

结构体是由一组不同的数据类型构成的。学习时要注意掌握结构体类型的定义、结构体成员的引用及结构体与数组的区别。

链表是一种动态数据结构，学习时要掌握链表的概念、基本操作和简单应用，以便为今后进一步学习数据结构奠定基础。

习题四

一、选择题

1．下列数组定义语句，正确的是_____。

A．int a[3,4];　　B．int n=3,m=4;int a[n][m];

C．int a[3][4];　　D．int a(3)(4);

2．阅读下列初始化数组程序段：

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

则下面叙述正确的是_____。

A．a 和 b 完全相同　　B．a 和 b 只是长度相等

C．a 和 b 不相同，a 是指针数组　　D．a 数组长度比 b 数组长

3．设有数组定义：char array[]="China";，则数组 array 所占的空间为_____。

A．4 个字节　　B．5 个字节　　C．6 个字节　　D．7 个字节

4．下列关于指针运算的描述错误的是_____。

A．在一定条件下，两个指针可以相加

B．在一定条件下，两个指针可以进行逻辑判断

C．在一定条件下，指针可以为空值

D．在一定条件下，两个指针可以相互赋值

5．已知 int *p,a;，则语句 p=&a;中的运算符“&”的含义是_____。

A．逻辑与运算　　B．位与运算　　C．取指针内容　　D．取变量地址

6．若有说明语句 int a[10], *p=a;，则对数组元素的正确引用是_____。

A．a[p]　　B．p[a]　　C．*(p+2)　　D．p+2

7．定义如下一维数组：int a[5],*p=a;，则下列描述错误的是_____。

A．表达式 p=p+1 是合法的　　B．表达式 a=a+1 是合法的

C．表达式 p-a 是合法的　　D．表达式 a+2 是合法的

8．下列对变量的引用中错误的是_____。

A．int a;int &p = a;　　B．char a;char &p = a;

C．int a;int &p;p = a;　　D．float a;float &p = a;

9．设有如下程序：

```
# include < iostream>
using namespace std;
void main ()
{
   int **x,*y,z = 10;
   y = &z;
   x = &y;
   cout<<**x+1<< end1;
}
```

上述程序的输出结果是_____。

A．y 的地址　　B．z 的地址　　C．11　　D．运行错误

10．下列语句错误的是_____。

A．char *p="John"; p[2]='a';

B．char name[5]="John";name[2]='a';

C．char name[5]="John",*p=name; p[2]='a';

D．char name[5]="John",*p=&name[2];*p='a';

11．已知有 int a[3][4]={{1,2,5,3},{2,4,7,9},{3,6,5,8}},*p=&a[0][0];，则表达式(*p+2)+*(p+2)的值应为_____。

A．10　　B．5　　C．6　　D．8

12．已知有 int a[5]={1,3,5,7,9},*p=&a[3];，则表达式 p[-1]的值为_____。

A．表示形式不合法　　B．值不确定

C．5　　D．6

13．若有以下定义，则值为 3 的表达式是_____。

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

A．p+=2,*(p++)　　B．p+=2,*++p　　C．p+=3,*p++　　D．p+=2, ++*p

14．已知如下变量的定义 char s[10],*p;，下列_____语句正确。

A．s= "asdfghj";　　B．p="asdfgh";　　C．cin<<s;　　D．p=s;cin<<p;

15．下列程序的输出结果是_____。

```
# include < iostream>
using namespace std;
void main ()
{
  int x [6]={1,3,5,7,9,11},*k,**s;
  k = x;
  s = &k;
  cout << *(k ++) << "," << **s<< end1;
}
```

A．3，3　　B．1，1　　C．3，5　　D．1，3

16．已知：

```
struct
{
  int  i;
  char  c;
  float  a;
}ex;
```

则 sizeof(ex);的值是_____。

A．6　　B．7　　C．8　　D．9

17．设有以下说明语句：

```
struct ex
{
  int  x;
  float  y;
  char  z;
}example;
```

则下面的叙述中不正确的是_____。

A．struct 是结构类型的关键字　　B．example 是结构类型名

C．x、y、z 都是结构成员名　　D．struct ex 是结构体类型

18．若有以下结构说明和变量的定义，且如图 4.33 所示指针 p 指向变量 a，指针 q 指向变量 b。

```
struct node
  {
   char data;
   struct node   *next;
  }a,b,*p=&a,*q=&b;
```

图 4.33　指针

则不能把节点 b 连接到节点 a 之后的语句是_____。

A．a.next=q;　　B．p.next=&b;　　C．p->next=&b;　　D．(*p).next=q;

二、填空题

1．在 C++程序中，一个数组的名字实际上是指向该数组_____元素的指针，并且在任何时候都不允许_____。

2．若变量 y 是变量 x 的引用，则对变量 y 的操作就是对变量_____的操作。

3．假定有如下定义：int x;，若要将整型变量 y 定义为变量 x 的引用，则应使用的定义语句是_____。

4．执行_____操作将动态分配 p 所指向的数据空间，执行_____操作将释放由 p 所指向的动态分配的数据空间。

5．完善程序，使 5×5 数组的主对角线元素为 1，其他为 0。

```
#include "iostream"
using namespace std;
void main()
{
int j,k,a[5][5];
for(j=0;j<5;j++)
   for(k=0;k<5;k++)
      if(_____)     _____ ;
      else    _____ ;
   for (j=0;j<5;j++)
   {
       for(k=0;k<5;k++)
       cout<<a[j][k]<<"    ";
       cout<<endl;
   }
 }
```

6．随机产生 6 位学生的分数（分数范围为 0～100），存放在数组 a 中，以每 2 分一个“*”显示，如图 4.34 所示。

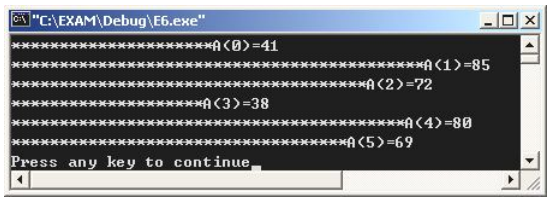


图 4.34　显示样式

```
#include "cstdlib"
#include "iostream"
using namespace std;
void main()
{
  int a[6],i,j;
```

```
 for(i=0;i<6;i++)
     {
      a[i]=_________;
      for (j=0;_________;j++)
          cout<<'*';
      cout<<_________<<a[i]<<endl;
     }
}
```

7．下列程序将数组 a 中的每 4 个相邻元素的平均值存放于数组 b 中。

```
#include "iostream"
using namespace std;
void main()
{
 int a[10],m,n;
 float b[7];
 for(m=0;m<10;m++) cin>>a[m];
    for(m=0;m<7;m++)
    {
     _________;
     for(n=m;_________; n++)
         b[m]=b[m]+a[n];
         _________;
    }
 for(m=0;m<7;m++) cout<<b[m]<<"    ";
}
```

8．已知数组 a 和 b 都是按由小到大顺序排列的有序数组，试将其合并后放入数组 c 中，使 c 也按由小到大顺序排列。

```
#include "iostream"
using namespace std;
#define M 5
#define N 6
void main()
{
 int a[M],b[N],c[M+N],i,j,k;
 for(j=0;j<M;j++) cin>>a[j];
 for(j=0;j<N;j++) cin>>b[j];
_________;
 while(k<M+N && i<M && j<N)
     {
          if(_________)
          {
            c[k]=a[i];
            i++;k++;
          }
      else {c[k]=b[j];j++;k++;}
     }
 while(k<M+N &&_________) c[k++]=b[j++];
```

```
 while(k<M+N &&_________) c[k++]=a[j++];
 for(k=0;k<M+N;k++) cout<<c[k]<<"    ";
}
```

9. 下面程序的功能是在结构体数组中查找分数最高和最低学生的姓名和成绩，请填空。

```
#include "iostream"
using namespace std;
void main()
{
 int max,min,i;
 struct
    {
     char   name[8];
     int    score;
    }stud[5]={"李海兵", 96, "王晓萍", 82, "钟山", 81, "孙求", 60, "徐虎", 86};
 max=min=0;
 for(i=1;i<5;i++)
    if(stud[i].score>stud[max].score)_________;
    else if(stud[i].score<stud[min].score)_________;
 cout<<"最高分:"<<stud[max].name<<","<<stud[max].score<<endl;
 cout<<"最低分:"<<stud[min].name<<","<<stud[min].score<<endl;
}
```

10. 下面程序段的功能是统计链表中节点的个数，其中 head 为指向头节点的指针，请填空。

```
struct   node
{
 int   data;
 stuct   node   *next;
};
struct   node   *p, *head;
int   count=0;
…
p=head;
while( _________ )
   {
    _________;
    p= _________;
   }
```

三、阅读程序，写出运行结果

1.

```
#include "iostream"
using namespace std;
void main()
{
 int j;
 int m[3][2]={10,20,30,40,50,60};
 for(j=0;j<2;j++)
```

```
        cout<<m[2-j][j]<<endl;
    }
```

2.

```
#include "iostream"
using namespace std;
void main()
{
  int i=0;
  char s[]="1234567890", *p;
  for(p=s+5;*p!='\0';p++) cout<<*p;
  p=s+4;
  while(i++<4) cout<<p[-i];
}
```

3.

```
#include "iostream"
using namespace std;
void main()
{
  int a[]={-2,3,0,-5,-4,6,9},*p=a,m,n;
  m=n=*p;
  for(p=a; p<a+7; p++)
    {
       if (*p>m) m=*p;
       if(*p<n) n=*p;
    }
  cout<<"m-n="<<m-n<<endl;
}
```

4.

```
#include "iostream"
using namespace std;
void main()
{
  int x;
  int &p=x;
  x=10;
  p=x+10;
  cout<<x<<","<<p<<endl;
}
```

5.

```
#include "iostream"
using namespace std;
void main ()
{
  int x[]={5,4,3,2,1},i,*p,m=0;
  for(p=x,i=1;p+i<=x+4;i++)
     {
```

```
        cout<<*(p+i);
        for(i=0;i<4;i++)
              {m+=p[i];cout<<"\t"<<m<<endl;}
       }
}
```

6．写出输入字符串为 abcdefg 时程序的运行结果。

```
#include "iostream"
using namespace std;
void main()
{
  char str[100],*p;
  cout<<"Please input a string:";
  cin>>str;
  p=str;
  for(int i=0;*p!='\0';p++,i++);
  cout<<"i="<<i<<endl;
}
```

四、编程题

1．随机生成两个元素值在[1,9]之间的 3×3 矩阵 A 和 B。要求：

（1）将 A 矩阵转置。

（2）以下三角形式显示 A 矩阵、上三角形式显示 B 矩阵。

（3）将 A 矩阵第一行与第三行对应元素交换位置，即第一行元素放到第三行，第三行元素放到第一行。

（4）求 A 矩阵主对角线元素之和。

（5）将两个矩阵相乘，结果放入 C 矩阵中。

提示：$C=A\times B=\sum_{k=0}^{p-1} a_{ik} b_{kj}$（$i=0,1,...,m-1$，$j=0,1,...,n-1$）

2．某社区对所属的 N 户居民进行用电量统计，每隔 50 度用电量为一个统计区间，但当大于等于 500 度时为一个统计区间，试编程统计每个区间内的居民户数。

3．有 10 个按从大到小顺序的数依次存放在数组 a 中，输入一个数，要求用折半查找法找出该数在数组中的位置。如果不在该数组中，则打印出“无此数!”。

提示：折半查找法的思路是要查找的关键值同数组的中间项元素比较，若相同则查找成功，结束；否则判别关键值落在数组的哪半部分，然后保留该一半，舍弃另一半。如此重复上述查找，直到查找到或数组中没有这样的元素。二分法查找每进行一次，就把查找区域数据的个数减少一半。

4．不使用函数 strlen，用字符指针的形式求字符串长度。

5．编写一个程序，判断一串字符是否是回文（即顺读与倒读一样）。

6．编写程序，将某一指定字符从一个已知的字符串中删除。

7．建立一个结构体，其中包括学生的姓名、性别、年龄和一门课程的成绩。建立的结构体数组通过输入存放全班（最多 45 人）学生信息，输出考分最高的学生的姓名、性别、年龄和课程的成绩。