

# 江苏科技大学

## 课程实践报告

2013 年 12 月

### 任务一

#### 一、实践任务

2. 试建立一个类 SP，求  $f(n,k) = 1^k + 2^k + 3^k + \dots + n^k$ ，另有辅助函数 power(m,n)用于求  $m^n$ 。

#### 二、详细设计

- 1、类的描述与定义
  - (1) 私有数据成员

- int n, k: 存放公式中 n 和 k 的值;

(2) 公有成员函数

- SP(int n1,int k1): 构造函数, 初始化成员数据 n 和 k。
- int power(int m, int n): 求  $m^n$ 。
- int fun(): 求公式的累加和。
- void show(): 输出求得的结果。

2、主要函数设计

在主程序中定义对象 s, 对该类进行测试。

### 三、源程序清单

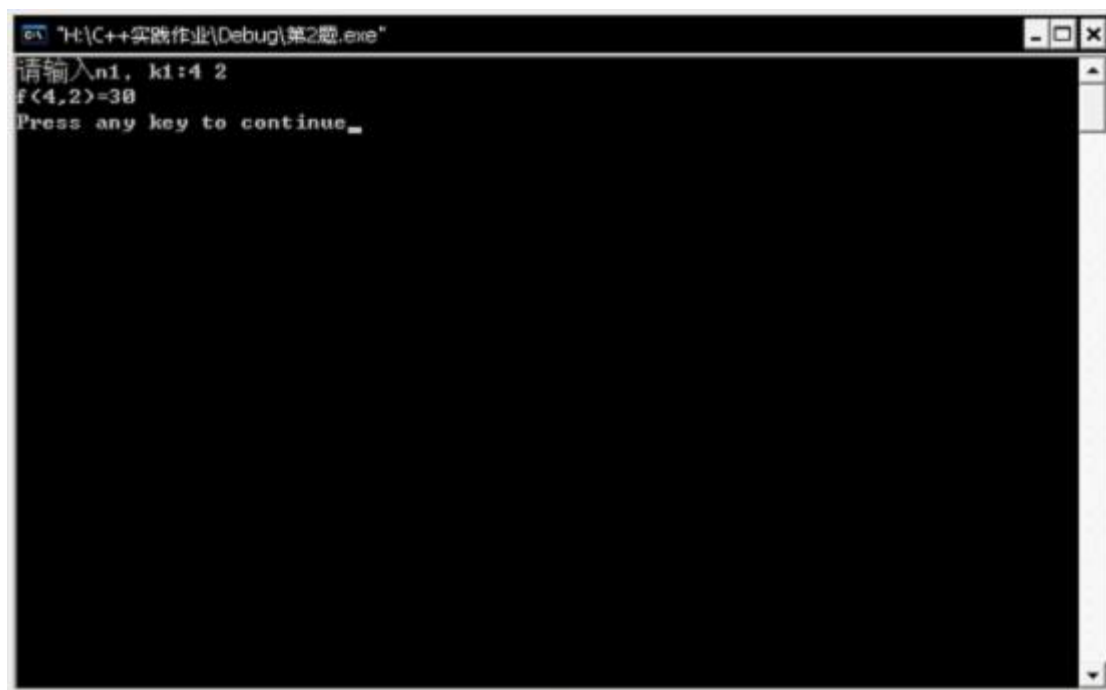
```
#include <iostream.h>
class SP
{
    int n,k;
public:
    SP(int n1,int k1)
    {
        n=n1;k=k1;
    }
    int power(int m,int n)
    {
        int p=1;
        for (int i=1;i<n+1;i++)
            p*=m;
        return p;
    }
    int fun()
    {
        int s=0;
        for (int i=1;i<n+1;i++)
            s+=power(i,k);
        return s;
    }
    void A()
    {
        cout<<"n="<<n<<" ,k="<<k<<" ,f(n,k)="<<fun()<<endl;
    }
};

void main()
{
    SP a(3,3);
    a.power(3,3);
    a.fun();
    a.A();
}
```

#### 四、实践小结

掌握用循环语句求  $m^n$ ，和  $m!$ ，熟练掌握函数的调用。

#### 五、运行结果



```
"H:\C++实践作业\Debug\第2题.exe"
请输入n1, k1:4 2
f(4,2)=30
Press any key to continue_
```

### 任务三

#### 一、实践任务

3. 建立一个类 MOVE，不进行排序，将数组中小于平均值的元素放到数组的左边，大于平均值的元素放到数组的右边。

#### 二、详细设计

##### 1、类的描述与定义

###### (1) 私有数据成员

- float array[20]: 一维整型数组。
- int n: 数组中元素的个数。

###### (2) 公有成员函数

- MOVE(float b[],int m): 构造函数，初始化成员数据。
- void average(): 输出平均值，并将数组中的元素按要求重新放置。
- void print(): 输出一维数组。

##### 2、主要函数设计

在主程序中用数据{1.3,6.2,3,9.1,4.8,7.4,5.6,9.2,2.3}对该类进行测试。

#### 三、源程序清单

#### 四、实践小结

应熟练掌握数组与指针的应用。

## 五、运行结果



```
H:\C++实践作业\Debug\第3题_1.exe
5.43333
1.3 3 4.8 2.3 9.2 5.6 7.4 9.1 6.2 Press any key to continue
```

## 任务四

### 一、实践任务

4. 建立一个类 MOVE，将数组中最大元素的值与最小元素的值互换。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int *array`: 一维整型数组。
- `int n`: 数组中元素的个数。

##### (2) 公有成员函数

- `MOVE(int b[],int m)`: 构造函数，初始化成员数据。
- `void exchange()`: 输出平均值，并将数组中的元素按要求重新放置。
- `void print()`: 输出一维数组。

- ~MOVE(): 析构函数。

## 2、主要函数设计

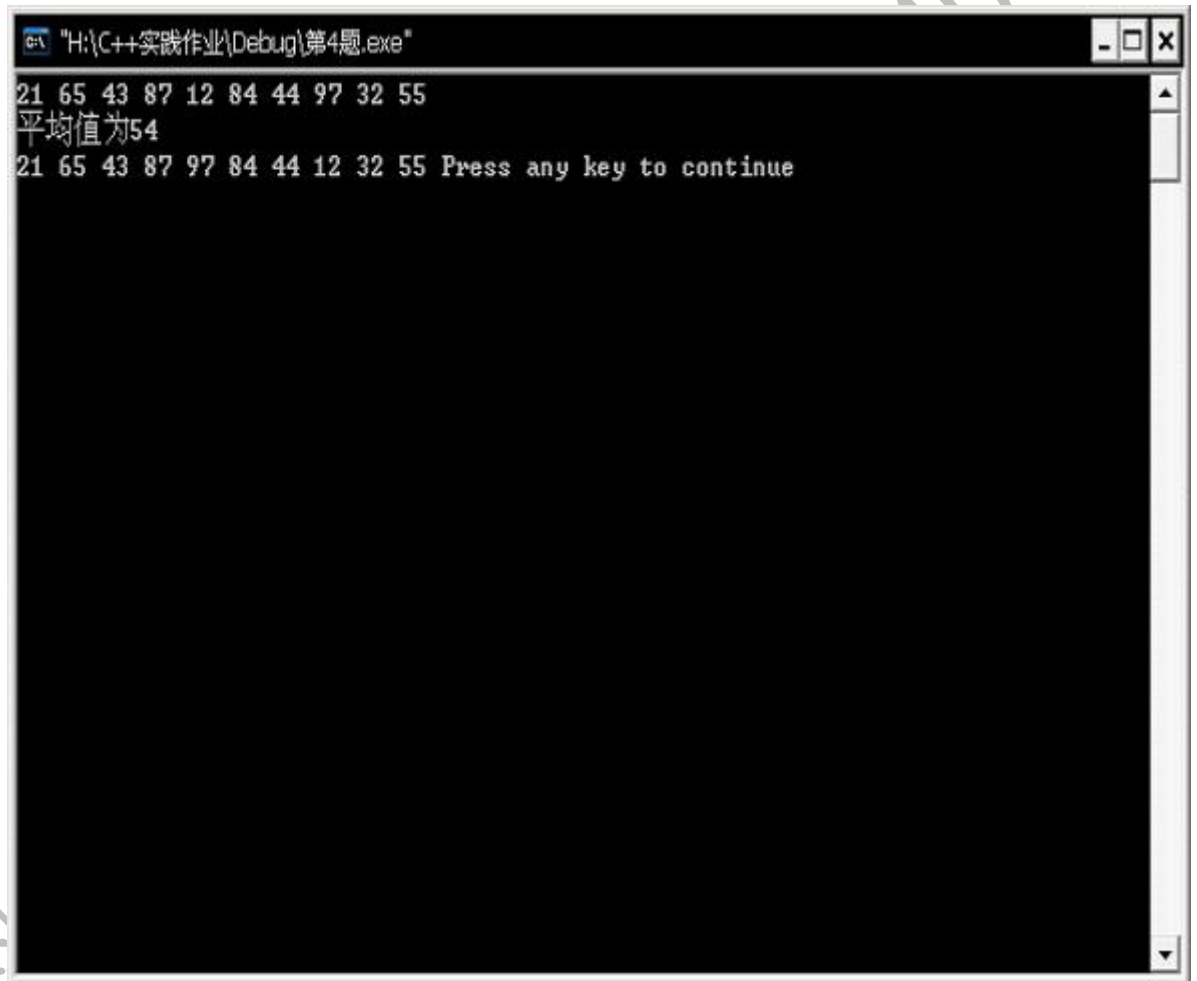
在主程序中用数据{21,65,43,87,12,84,44,97,32,55}对该类进行测试。

## 三、源程序清单

## 四、实践小结

学会求数组中最大元素与最小元素方法，并实现交换。

## 五、运行结果



```
"H:\C++实践作业\Debug\第4题.exe"
21 65 43 87 12 84 44 97 32 55
平均值为54
21 65 43 87 97 84 44 12 32 55 Press any key to continue
```

## 任务六

### 一、实践任务

6. 定义一个字符串类 String，实现判断该字符串是否为回文字符串。所谓回文字符串，是指该字符串左右对称。例如字符串“123321”是回文字符串。

### 二、详细设计

## 1、类的描述与定义

### (1) 私有数据成员

- `char *str;`
- `int y`: 标记是否为回文字符串。

### (2) 公有成员函数

- `String(char *s)` : 构造函数，用给定的参数 `s` 初始化数据成员 `str`。`y` 初始化为 0。
- `void huiwen()` : 判断 `str` 所指向的字符串是否为回文字符串。
- `void show()` : 在屏幕上显示字符串。

## 2、主要函数设计

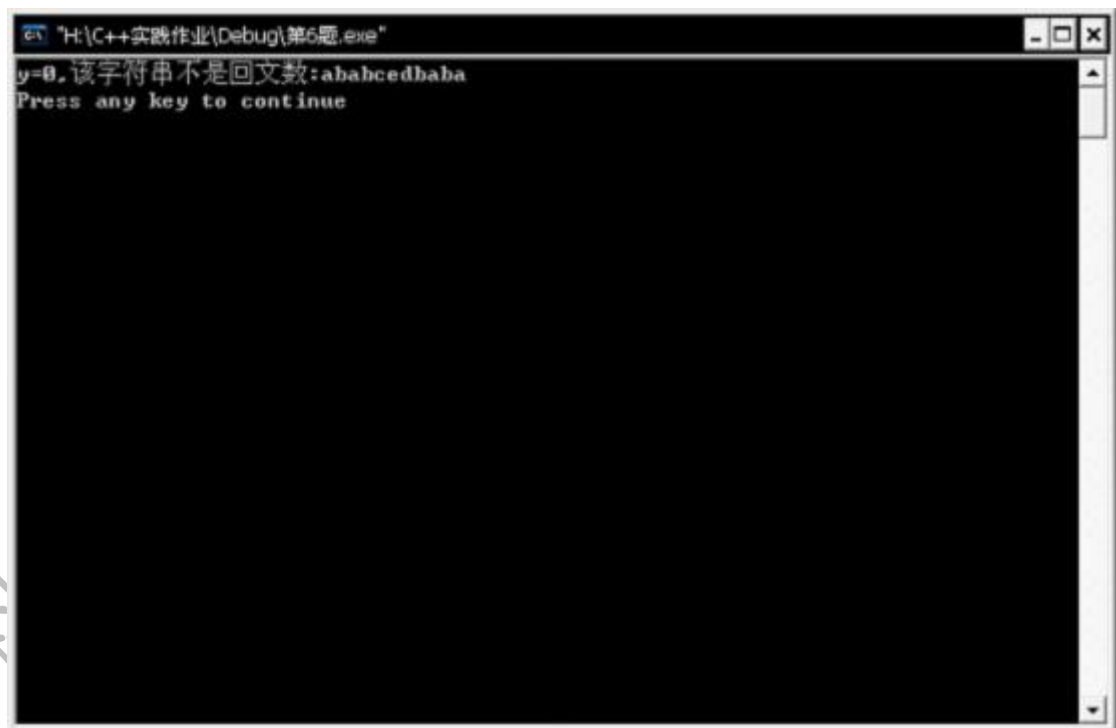
在主程序中定义字符串 `char s[]="ababcedbaba"` 作为原始字符串。定义一个 `String` 类对象 `test`，用 `s` 初始化 `test`，完成对该类的测试。

## 三、源程序清单

## 四、实践小结

掌握判断回文字符串的一般形式。

## 五、运行结果



## 任务十一

### 一、实践任务

11. 建立一个 STRING，将一个字符串交叉插入到另一个字符串中（假定两字符串等长）。

例如将字符串“abcde”交叉插入字符串“ABCDE”的结果为“aAbBcCdDeE”或“AaBbCcDdEe”。

## 二、详细设计

### 1、类的描述与定义

#### (1) 私有数据成员

- char str1[80]：存放被插入的字符串。
- char str2[40]：存放待插入的字符串。

#### (2) 公有成员函数

- STRING(char \*s1, char \*s2)：构造函数，用 s1 和 s2 初始化 str1 和 str2。
- void process()：将 str2 中的字符串插入到 str1 中。
- void print()：输出插入后的字符串。

### 2、主要函数设计

在主程序中定义 STRING 类的对象 test 对该类进行测试。

## 三、源程序清单

```
#include<iostream.h>
#include<string.h>
class STRING{
private:
char str1[80]; //存放被插入的字符串;
char str2[40]; //存放待插入的字符串;
public:
STRING(char*s1,char*s2)
{
strcpy(str1,s1);
strcpy(str2,s2);
}
void process();
void print();
};
void STRING::process()
{
int i,j;
int n=strlen(str1);
if(strlen(str2)>strlen(str1))
{//当待插入的字符串 ABCDEFG 比被插入的字符串 abcde 长或相等时，逻辑算法：
abcde->a b c d e->空格处依次插入 ABCDEFG->aAbBcCdDeEFG;
for(i=n-1;i>0;i--)
{
str1[i+i]=str1[i]; //被插入的字符串由最后一位开始各位向后移动 i 位;
}
for(i=1,j=0;i<2*n;i+=2,j++)
```

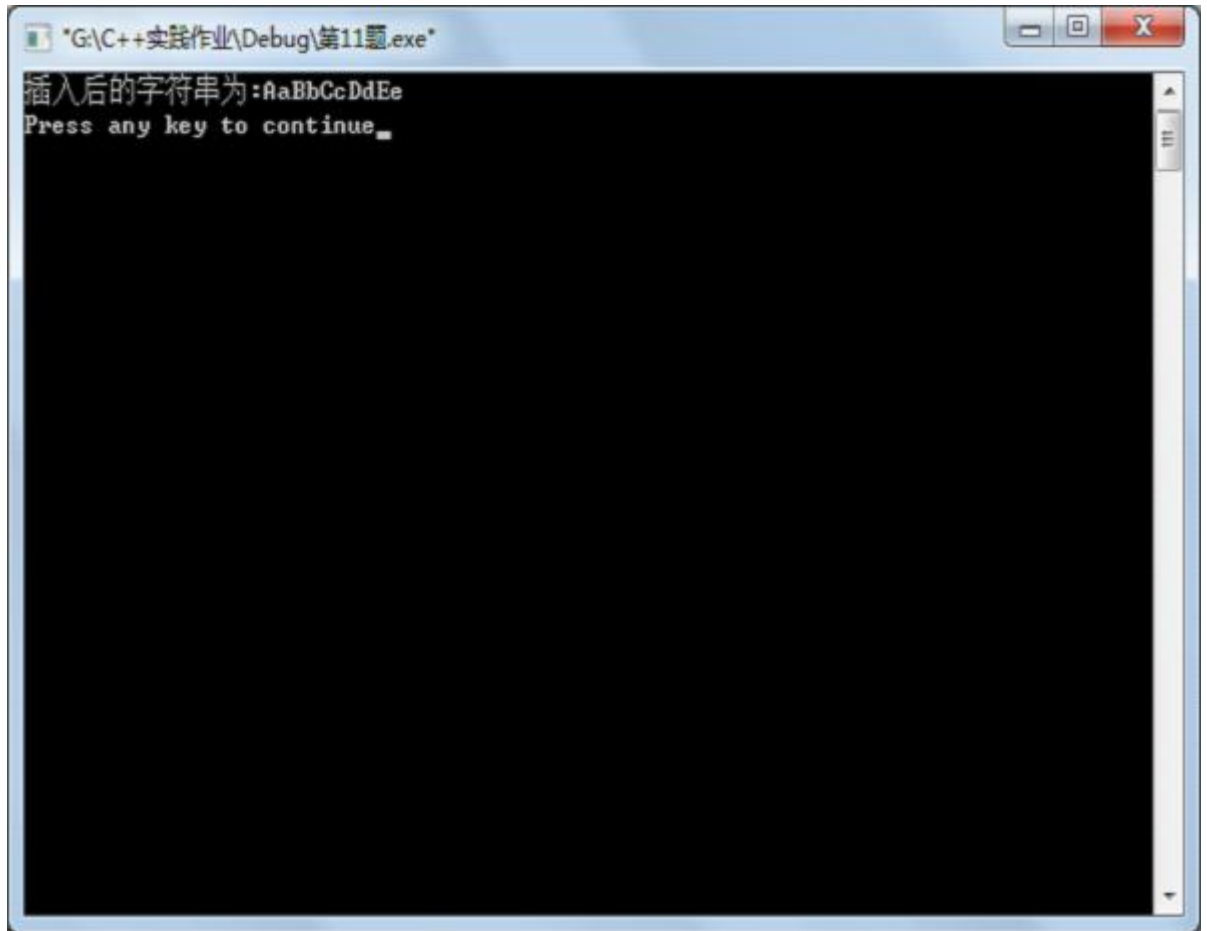
```
{
    str1[i]=str2[j]; //在空出的位置处依次插入字符串;
}
i--;
for(;j<=strlen(str2);j++,i++)
{
    str1[i]=str2[j]; //将过长额字符串放入被插入的字符串尾部,完成插入;
}
}
else//当待插入的字符串 abcde 比被插入的字符串 ABCDEFG 短时, 逻辑算法:
ABCDEFG->A B C D E FG->空格处插入 abcde->AaBbCcDdEeFG;
{
    for(i=n;i>strlen(str2)-1;i--)
    {
        str1[i+strlen(str2)]=str1[i]; //比待插入的字符串长的部分均向后移
strlen(str2)位;
    }
    for(i=strlen(str2)-1;i>0;i--)
    {
        str1[i+i]=str1[i]; //之前的部分均向后移 i 位;
    }
    for(i=1,j=0;i<2*strlen(str2);i+=2,j++)
    {
        str1[i]=str2[j]; //将待插入的字符串插入空格处,完成插入;
    }
}
}
void STRING::print()//输出插入后的字符串
{
    cout<<"插入后的字符串为:"<<str1<<endl;
}
void main()//测试
{
    STRING test("ABCDE","abcde");
    test.process();
    test.print();
}
```

#### 四、实践小结

发现字符插入的规律, 再依次放入相应字符位置。

#### 五、运行结果





## 任务十二

### 一、实践任务

12. 建立一个 STRING，将一个字符串交叉插入到另一个字符串中（假定两字符串不等长）。  
例如将字符串“abcde”交叉插入字符串“ABCDEFGH”的结果为“AaBbCcDdEeFG”或“AaBbCcDdEeFG”。

### 二、详细设计

#### 1. 类的描述与定义

##### (1) 私有数据成员

- char str1[60]：存放被插入的字符串。
- char str2[40]：存放待插入的字符串。
- char str3[100]：存放插入后的字符串。

##### (2) 公有成员函数

- STRING(char \*s1, char \*s2)：构造函数，用 s1 和 s2 初始化 str1 和 str2。
- void process()：将 str2 中的字符串插入到 str1 中，存放到 str3 中。
- void print()：输出插入后的字符串。

#### 2. 主要函数设计

在主程序中定义 STRING 类的对象 test 对该类进行测试。

### 三、源程序清单

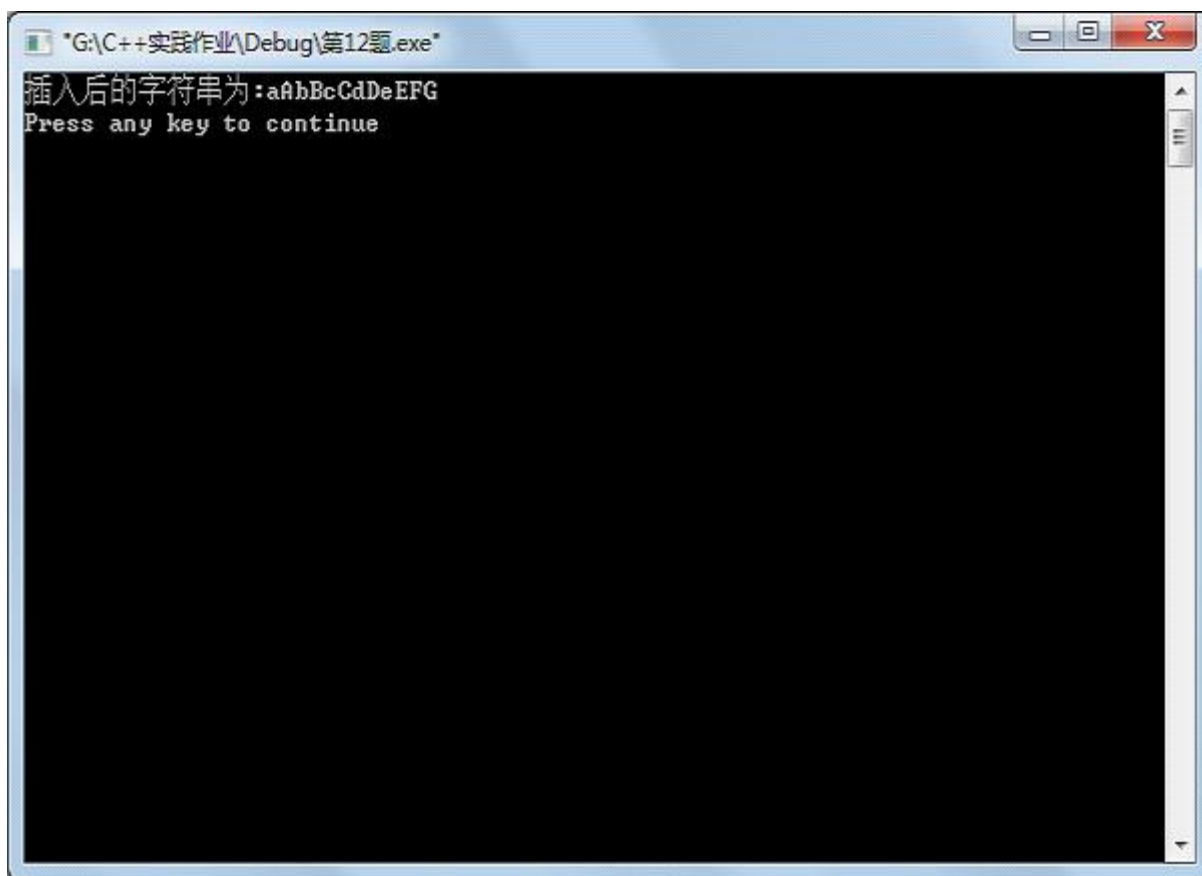
```
#include<iostream.h>
#include<string.h>
class STRING{
private:
char str1[60];
char str2[40];
char str3[100];
public:
STRING(char*s1,char*s2)
{
strcpy(str1,s1);
strcpy(str2,s2);
}
void process();
void print();
};
void STRING::process()
{
int i,j;
int n=strlen(str1);
if(strlen(str2)>strlen(str1))
{
//当待插入的字符串 ABCDEFG 比被插入的字符串 abcde 长或相等时，逻辑算法：
abcde->a b c d e->空格处依次插入 ABCDEFG->aAbBcCdDeEFG;
for(i=n-1;i>0;i--)
{
str1[i+i]=str1[i];//被插入的字符串由最后一位开始各位向后移动 i 位；
}
for(i=1,j=0;i<2*n;i+=2,j++)
{
str1[i]=str2[j];//在空出的位置处依次插入字符串；
}
i--;
for(;j<=strlen(str2);j++,i++)
{
str1[i]=str2[j];//将过金额字符串放入被插入的字符串尾部,完成插入；
}
}
else//当待插入的字符串 abcde 比被插入的字符串 ABCDEFG 短时，逻辑算法：
ABCDEFG->A B C D E FG->空格处插入 abcde->AaBbCcDdEeFG;
{
for(i=n;i>strlen(str2)-1;i--)
{
```

```
        str1[i+strlen(str2)]=str1[i]; //比待插入的字符串长的部分均向后移
        strlen(str2)位;
    }
    for(i=strlen(str2)-1;i>0;i--)
    {
        str1[i+i]=str1[i]; //之前的部分均向后移 i 位;
    }
    for(i=1,j=0;i<2*strlen(str2);i+=2,j++)
    {
        str1[i]=str2[j]; //将待插入的字符串插入空格处,完成插入;
    }
}
strcpy(str3, str1); //将 str2 中的字符串插入到 str1 中, 存放到 str3 中;
}
void STRING::print() //输出插入后的字符串
{
    cout<<"插入后的字符串为:"<<str3<<endl;
}
void main()
{
    STRING test("abcde","ABCDEFGH");
    test.process();
    test.print();
}
```

#### 四、实践小结

发现字符插入的规律，再依次放入相应字符位置。

#### 五、运行结果



## 任务十三

### 一、实践任务

13. 建立一个类 MOVE，对数组中元素进行循环换位，即每个元素后移三位，最后三个元素移到最前面。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int array[20]`: 一维整型数组。
- `int n`: 数组中元素的个数。

##### (2) 公有成员函数

- `MOVE(int b[],int m)`: 构造函数，初始化成员数据。
- `void change()`: 进行循环换位。
- `void print()`: 输出一维数组。

#### 2、主要函数设计

在主程序中用数据 {21,65,43,87,12,84,44,97,32,55} 对该类进行测试。

### 三、源程序清单

```
#include<iostream.h>
class MOVE{
```

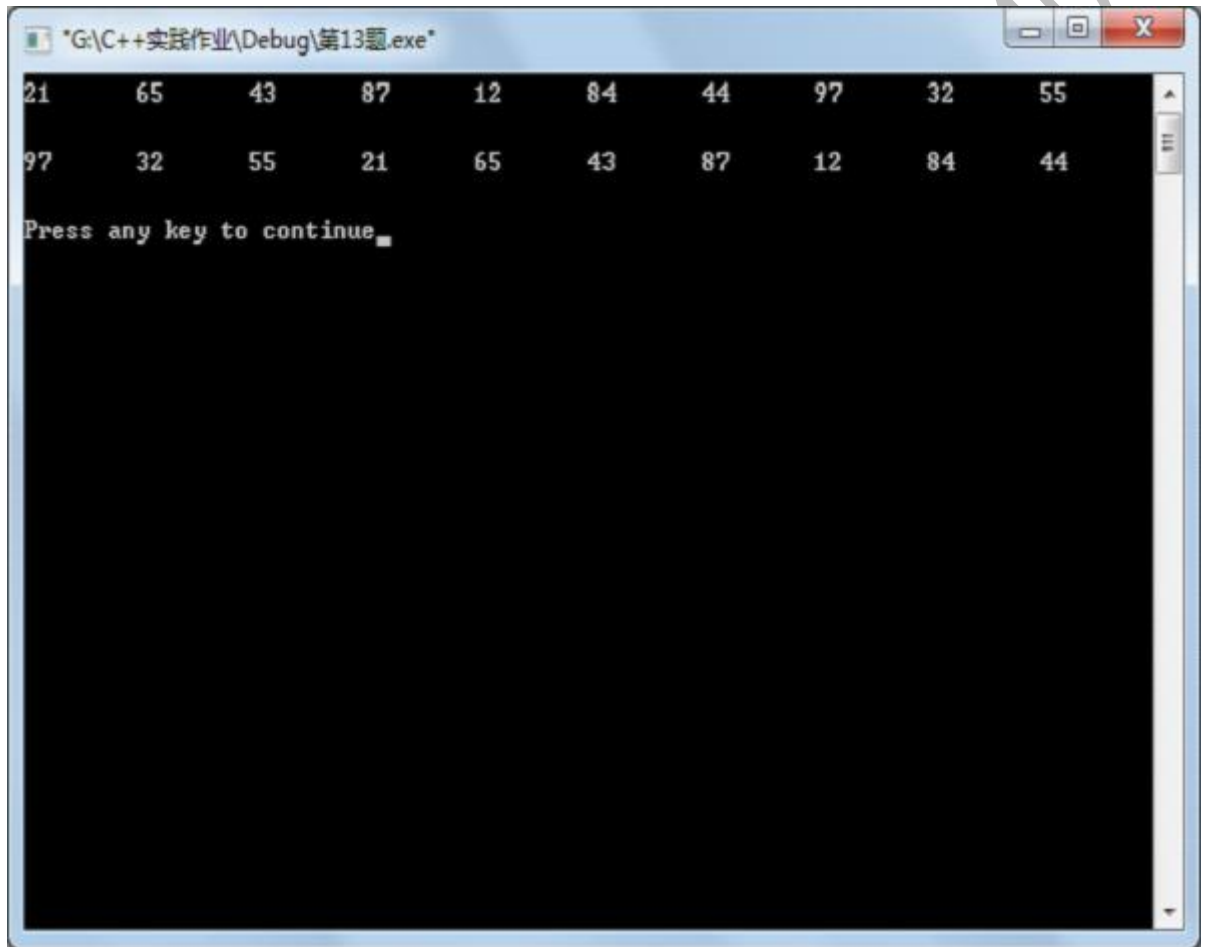
```
private:
int array[20]; //一维整型数组;
int n; //数组中的元素个数;
public:
MOVE(int b[],int m)
{
    for(int i=0;i<m;i++)
        array[i]=b[i];
    n=m;
}
void change();
void print();
};
void MOVE::change() //进行循环换位，即每个元素后移 3 位，最后 3 个元素移到最前面;
{
    int tem[3]; //建立临时数组，用于存放最后 3 个元素;
    int i=0;
    while(i<3)
    {
        tem[2-i]=array[n-1-i]; //将最后 3 个元素依次放入临时数组中;
        i++;
    }
    for(i=0;i<n-3;i++) //将原数组的最后一个至第 4 个元素依次后移 3 位;
    {
        array[n-1-i]=array[n-1-3-i];
    }
    for(i=0;i<3;i++) //将临时数组中存放的最后 3 个元素依次放入后移后的原数组中，完成循环换位;
    {
        array[i]=tem[i];
    }
}
void MOVE::print() //输出一维数组;
{
    for(int i=0;i<n;i++)
        cout<<array[i]<<"\t";
    cout<<endl;
}
void main()
{
    int s[]={21,65,43,87,12,84,44,97,32,55};
    int n=sizeof(s)/sizeof(int);
    MOVE test(s,n);
}
```

```
test.print();  
test.change();  
test.print();  
}
```

#### 四、实践小结

利用临时数组先保存后 3 位，再依次把数放入对应位。

#### 五、运行结果



### 任务十四

#### 一、实践任务

14. 建立一个类 MOVE，实现将数组中大写字母元素放在小写字母元素的左边。

#### 二、详细设计

##### 1、类的描述与定义

##### (1) 私有数据成员

- char \*array: 一维字符数组。
- int n: 数组中元素的个数。

(2) 公有成员函数

- MOVE(char b[],int m): 构造函数，初始化成员数据。
- void change(): 进行排序换位。
- void print(): 输出一维数组。
- ~MOVE(): 析构函数。

2、主要函数设计

在主程序中用数据"fdsUFfsTjfsKFEkWC"对该类进行测试。

三、源程序清单

```
#include <iostream.h>
#include<string.h>
class MOVE{
    char *array;
    int n;
public:
    MOVE(char b[],int m)
    { n=m;
      array=new char[n+1];
      strcpy(array,b);
    }
    void change()
    {
        char*p1=new char[strlen(array)+1];//用于存大写;
        char*p2=new char[strlen(array)+1];//存小写;
        int i,j=0,k=0;
        for(i=0;i<strlen(array);i++)//大写与小写分别存放;
        {
            if(array[i]>='A'&&array[i]<='Z')
            {
                p1[j]=array[i];
                j++;
            }
            else {p2[k]=array[i];k++;}
        }
        p1[j]=p2[k]='\0';
        strcat (p1, p2) ;//存放大写组与小写组拼接;
        strcpy (array, p1) ;//拷贝至 array;
        delete []p1;
        delete []p2;
    }
    void print()
    {
        cout<<array<<endl;
    }
}
```

```
~MOVE()
{ if (array)
    delete []array;
}

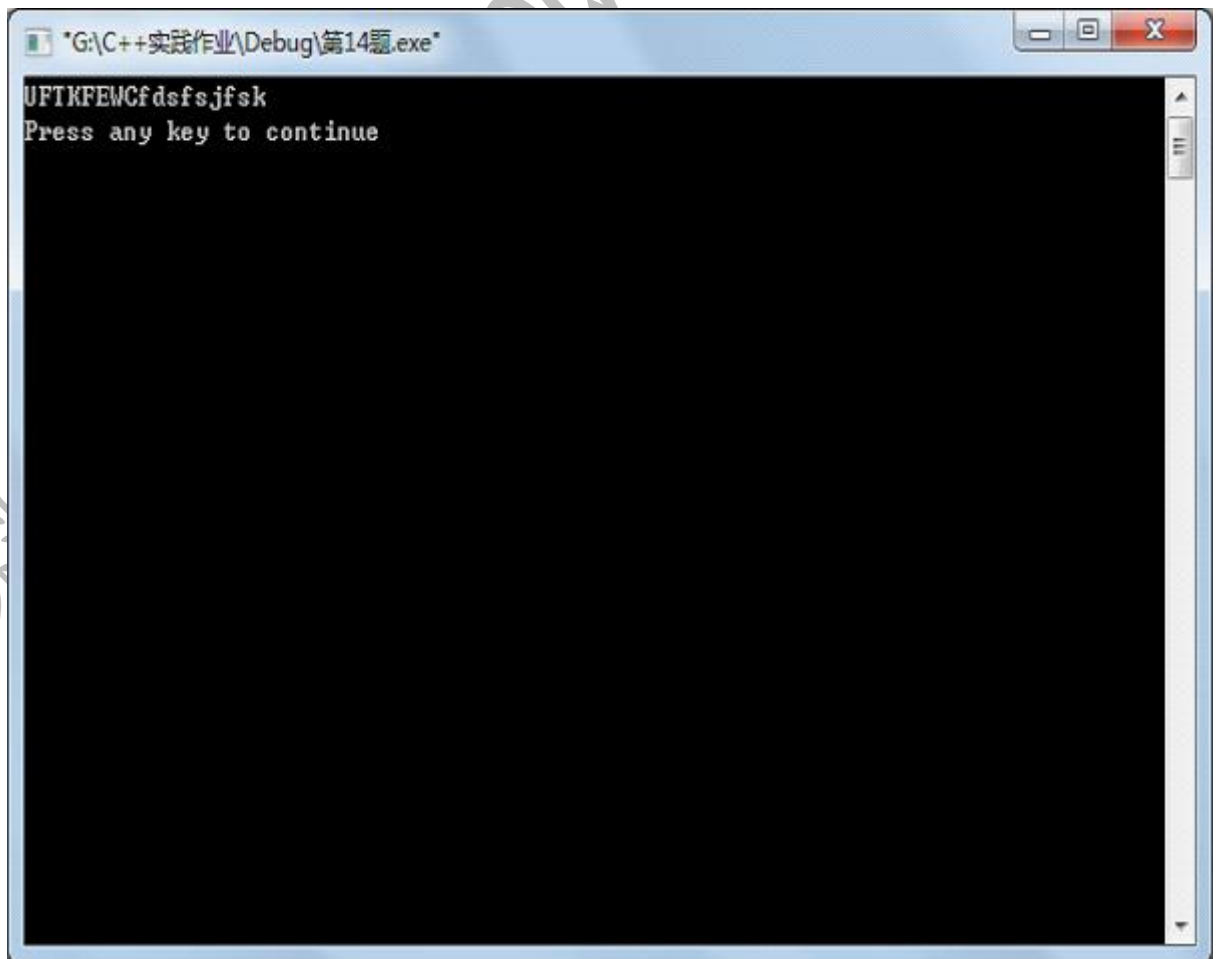
};

void main()
{
    char b[]="fdsUffsTjfsKFEkWC";
    int n;
    n=(sizeof(b)-1)/sizeof(char);
    MOVE test(b,n);
    test.change();
    test.print();
}
```

#### 四、实践小结

利用临时数组，分别保存大写与小写字母，再实现功能。

#### 五、运行结果





## 任务十五

### 一、实践任务

16. 定义一个方阵类 CMatrix，并根据给定算法实现方阵的线性变换。方阵的变换形式为：

$$F=W*f^T$$

f 为原始矩阵，f<sup>T</sup> 为原始矩阵的转置，w 为变换矩阵，这里设定为

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- int (\*a)[4]: a 指向方阵数组。
- int w[4][4]: w 为变换矩阵。
- int m: m 表示方阵的行和列数。

##### (2) 公有成员函数

- CMatrix (int a[][4],int m) : 用给定的参数 a 和 m 初始化数据成员 a 和 m; 对变换矩阵 w 进行初始化，要求必须用循环实现。
- void Transform () : 根据上述变换算法，求出变换后的数组形式，存放在原始数组内。
- void show () : 在屏幕上显示数组元素。
- ~CMatrix () : 释放动态分配的空间。

#### 2、主要函数设计

在主程序中定义数组 int arr[][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16} 作为原始数组。定义一个 CMatrix 类对象 test，用 arr 初始化 test，完成对该类的测试。

### 三、源程序清单

```
#include<iostream.h>
class CMatrix{
private:
int(*a)[4];
int w[4][4]; //变换矩阵
int m; //表示方阵的行和列数;
public:
CMatrix(int a[][4],int m)
{
```

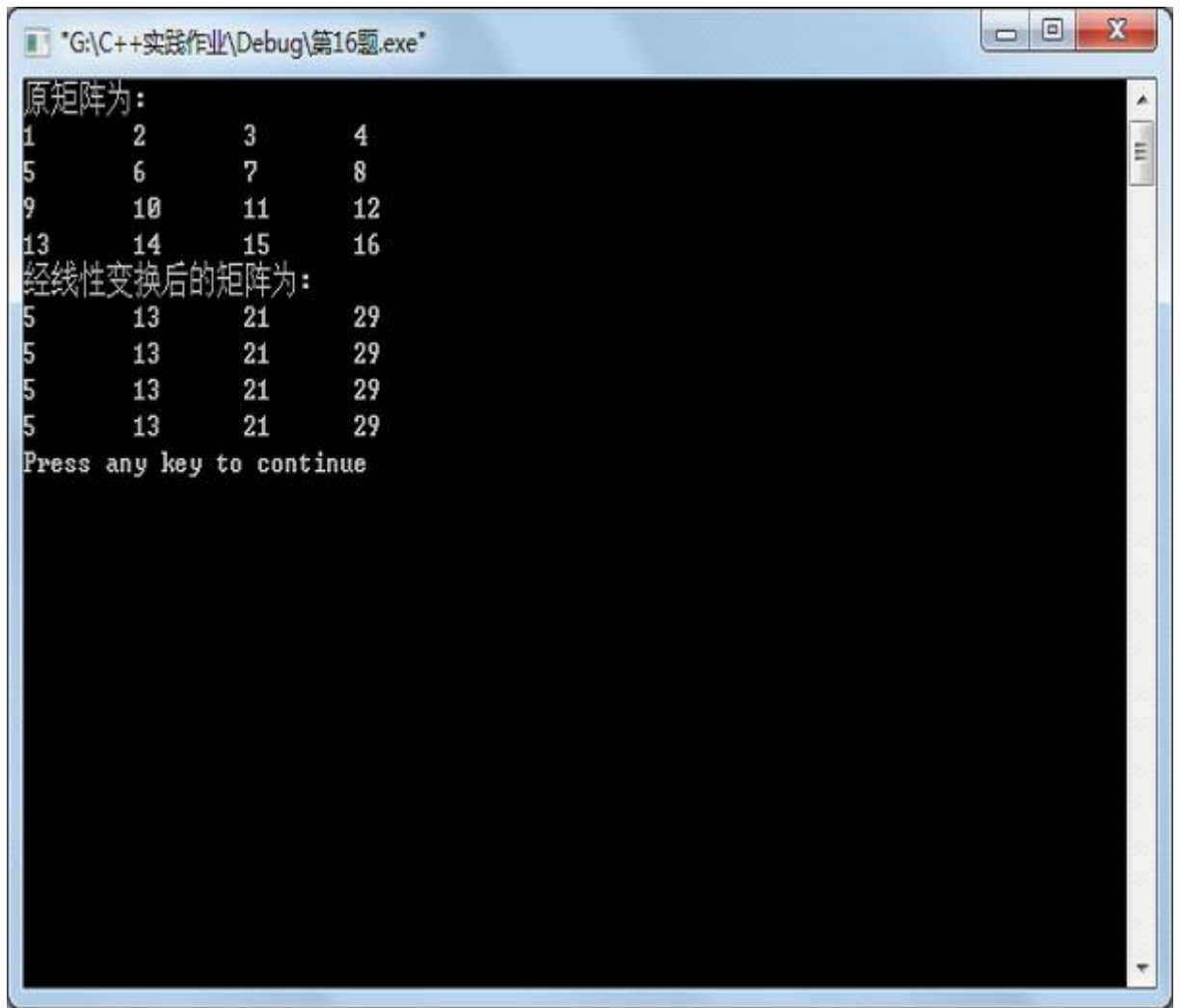
```
int i,j;
this->a=new int[m][4];
this->m=m;
for(i=0;i<4;i++)
{
    for(j=0;j<4;j++)
    {
        if(i==j||i+j==4-1)
            w[i][j]=1;
        else
            w[i][j]=0;
    }
}
for(i=0;i<m;i++)
{
    for(j=0;j<4;j++)
    {
        this->a[i][j]=a[i][j];
    }
}
}
void Transform();
void show();
~CMatrix()
{if(a) delete []a;}
};
void CMatrix::Transform()//根据变换算法，求出变换后的数组形式，存放在原始数组内；
{
    int i,j,k;
    for(i=0;i<m;i++)//求原始矩阵的转置，并存放在原始数组中；
    {
        for(j=i;j<4;j++)
        {
            k=a[i][j],a[i][j]=a[j][i],a[j][i]=k;
        }
    }
    int sum;//用来存放矩阵乘法中，行列中元素依次相乘的累加和；
    int turn[4][4];//临时数组，用来存放矩阵乘法所求得的值；
    for(i=0;i<4;i++)//实现矩阵的乘法；
    {
        for(j=0;j<4;j++)
        {
            sum=0;
```

```
        for(k=0;k<4;k++)
        {
            sum+=w[i][k]*a[k][j];
        }
        turn[i][j]=sum;
    }
}
for(i=0;i<4;i++)//将临时数组的值存放入原始数组中;
{
    for(j=0;j<4;j++)
    {
        a[i][j]=turn[i][j];
    }
}
}
void CMatrix::show()//在屏幕上显示数组元素;
{
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<4;j++)
        {
            cout<<a[i][j]<<"t";
        }
        cout<<endl;
    }
}
void main()
{
    int arr[][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
    CMatrix test(arr,4);
    cout<<"原矩阵为:"<<endl;
    test.show();
    test.Transform();
    cout<<"经线性变换后的矩阵为:"<<endl;
    test.show();
}
```

#### 四、实践小结

应熟练掌握矩阵的乘法方式。

#### 五、运行结果



```
"G:\C++实践作业\Debug\第16题.exe"
原矩阵为:
1      2      3      4
5      6      7      8
9      10     11     12
13     14     15     16
经线性变换后的矩阵为:
5      13     21     29
5      13     21     29
5      13     21     29
5      13     21     29
Press any key to continue
```

## 任务十六

### 一、实践任务

17. 定义一个类 SIN, 求  $\sin(x) = x/1 - x^3/3! + x^5/5! - x^7/7! + \dots + (-1)^{n+1} x^{(2n-1)}/(2n-1)!$

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- float x: 输入公式中 x 的值, 求 sin(x)。
- int n: 输入公式中 n 的值。

##### (2) 公有成员函数

- SIN(float x, int n): 构造函数, 用于初始化 x 和 n 的值。
- int power( int q): 求 q! 的值。
- float mi(float m, int n): 求  $m^n$  的值。
- float fun(): 用于求 SIN(X) 的值。

- void show(): 输出求得的结果。

## 2、主要函数设计

在主程序中定义对象 test，对该类进行测试（x 是弧度，弧度不可能大于 1）。

## 三、源程序清单

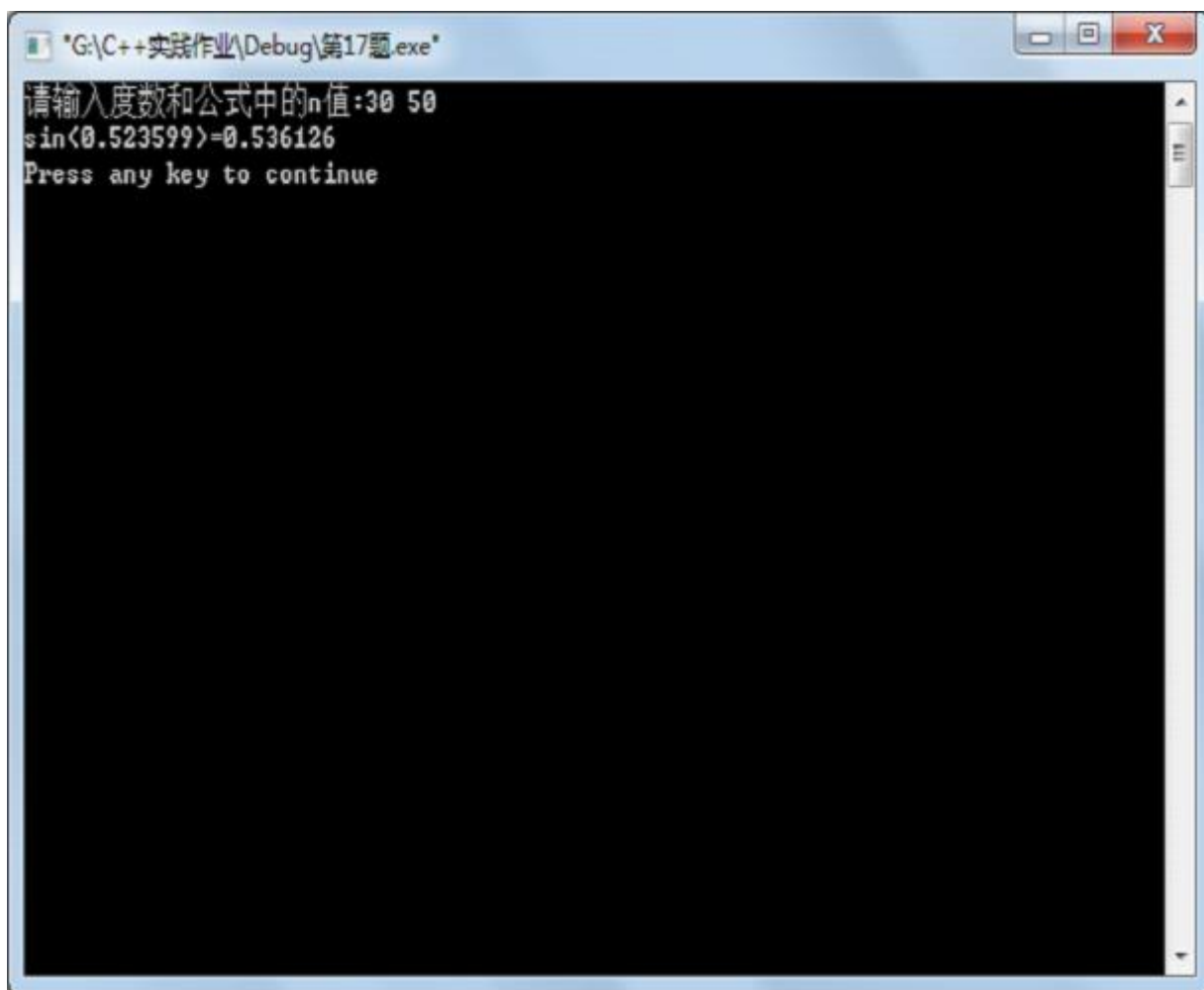
```
#include<iostream.h>
#include<math.h>
class SIN{
private:
double x;
int n;
public:
SIN(double x,int n)
{
    this->x=x;
    this->n=n;
}
double power(int q);
double mi(double m,int n);
double fun();
void show();
};
double SIN::power(int q)//求 q 的阶乘;
{
double s=1;
while(q!=1)
{
    s*=q;
    q--;
}
return s;
}
double SIN::mi(double m, int n)//求 m^n 的值;
{
while(n!=1)
{
    m*=m;
    n--;
}
return m;
}
double SIN::fun()//用于求 sin(x)的值; //注：当 n 较大时，阶乘和幂的运算可能超出变量的类型的字节大小！改用 double 类型可提高运算的范围！
{
```

```
double s=0;//记录每项相加的和
for(int i=1;i<=n;i++)
{
    s+=mi(x,2*i-1)*mi(-1,i+1)/power(2*i-1);//通项为(-1)^(i+1)*x^(2*i-1)/(2*i-1)!,
其中 n 为由 1 开始的奇数;
}
return s;
}
void SIN::show()//输出求得结果;
{
    cout<<"sin("<<x<<"")=<<fun()<<endl;
}
void main()
{
    int degree,n;
    double hudu;
    cout<<"请输入度数和公式中的 n 值:";
    cin>>degree>>n;
    hudu=degree%360*(3.1415926)/180.0;//度数转换为弧度;
    SIN test(hudu,n);
    test.show();
}
```

#### 四、实践小结

找到公式中的相关关系，再进行相应函数的组合。

#### 五、运行结果



## 任务十七

### 一、实践任务

18. 试建立一个类 VAR，用于求  $n$  ( $n \leq 100$ ) 个数的均方差。均方差的计算公式为

$$d = \sqrt{\sum_{i=0}^{n-1} \frac{(x_i - \bar{x})^2}{n}}, \text{ 其中平均值为 } \bar{x} = \frac{\sum_{i=0}^{n-1} x_i}{n}.$$

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- double a[100]: 用于存放输入的  $n$  个数。
- int n: 实际输入数的个数  $n$ 。

##### (2) 公有成员函数

- VAR(double x[], int n1): 构造函数，初始化成员数据  $a$  和个数  $n$ 。
- double average(double x[], int n): 求平均值，数组  $x$  具有  $n$  个元素。

- void variance(double x[],int n): 求均方差，数组 x 具有 n 个元素。
- void show(): 输出求得的均方差。

## 2、主要函数设计

在主程序中定义一个对象 test，对该类进行测试。

## 三、源程序清单

```
#include<iostream.h>
class VAR{
private:
double a[100]; //用于存放输入的 n 个数;
int n; //实际输入数的个数 n;
public:
VAR(double x[],int n1)
{
for(int i=0;i<n1;i++)
{
a[i]=x[i];
}
n=n1;
}
double average(double x[],int n);
double variance(double x[],int n);
void show();
};
double VAR::average(double x[],int n) //求平均值，数组 x 具有 n 个元素;
{
double ave=0;
for(int i=0;i<n;i++)
{
ave+=x[i];
}
return ave/n;
}
double VAR::variance(double x[],int n) //求均方差，数组 x 具有 n 个元素;
{
double d=0;
for(int i=0;i<n;i++)
{
d+=(x[i]-average(x,n))*(x[i]-average(x,n));
}
d/=n;
return d;
}
void VAR::show() //输出求得的均方差;
```

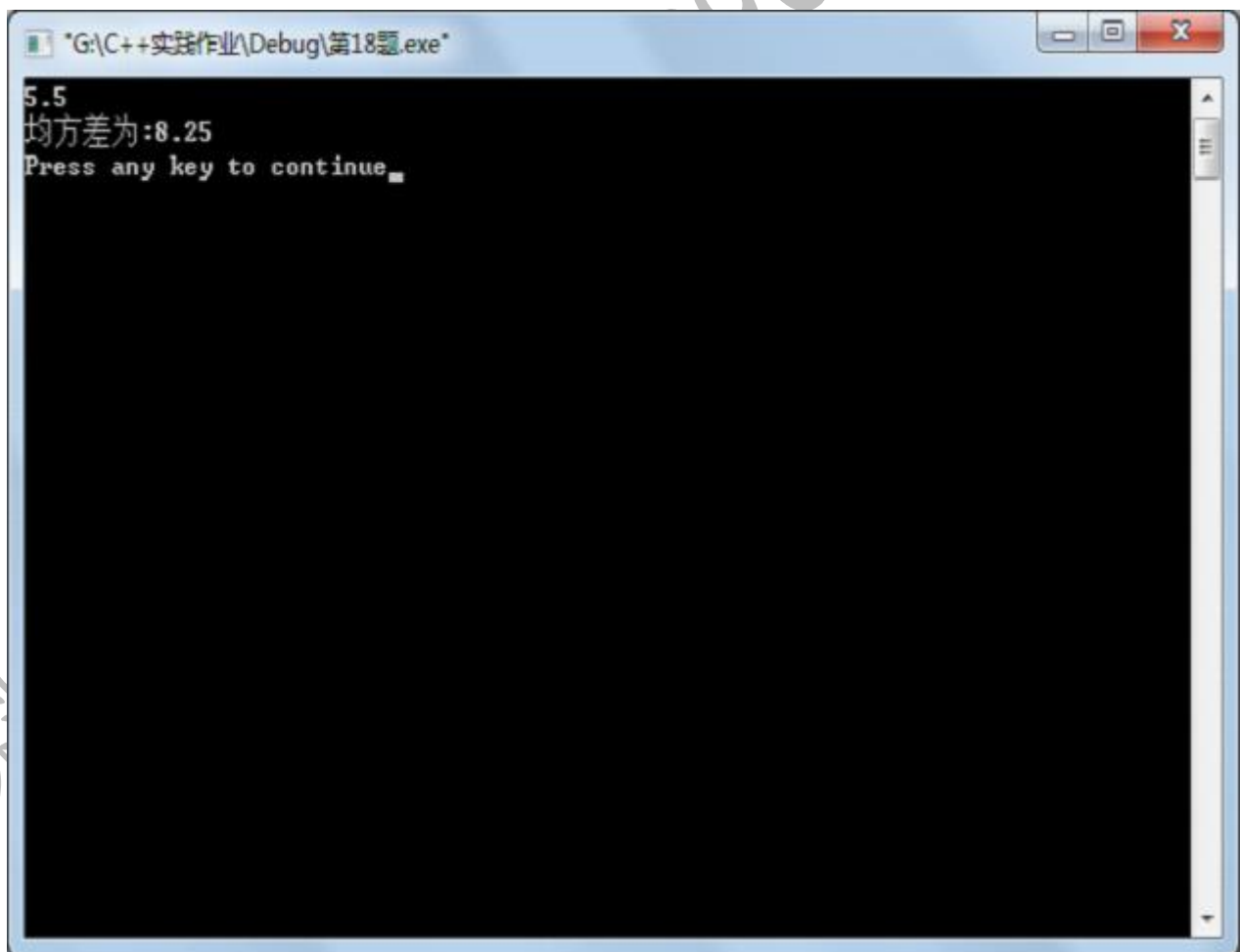


```
{
cout<<"均方差为:"<<variance(a,n)<<endl;
}
void main()//测试;
{
double s[]={1,2,3,4,5,6,7,8,9,10};
int n=sizeof(s)/sizeof(double);
VAR test(s,n);
cout<<test.average(s,n)<<endl;
test.show();
}
```

#### 四、实践小结

理解算数公式，根据题目线索，完成。

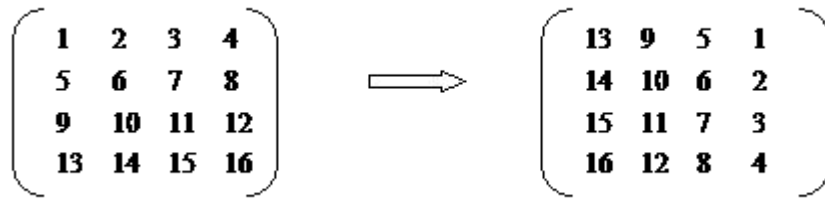
#### 五、运行结果



### 任务十八

#### 一、实践任务

19. 定义一个方阵类 Array，实现对方阵进行顺时针 90 度旋转。如图所示。



## 二、详细设计

### 1、类的描述与定义

#### (1) 私有数据成员

- `int a[4][4]`: 用于存放方阵。

#### (2) 公有成员函数

- `Array (int a1[][4],int n)`: 构造函数，用给定的参数 `a1` 初始化数据成员 `a`。
- `void xuanzhuan ()`: 实现对方阵 `a` 进行顺时针 90 度的旋转。
- `void show()`: 在屏幕上显示数组元素。

### 2、主要函数设计

在主程序中定义数组 `int b[ ][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}` 作为原始数组。定义一个 `Array` 类对象 `test`，用 `b` 初始化 `test`，完成对该类的测试。

## 三、源程序清单

```
#include<iostream.h>
class Array{
private:
int a[4][4];
public:
Array(int a1[][4],int n)
{
for(int i=0;i<n;i++)
{
for(int j=0;j<4;j++)
a[i][j]=a1[i][j];
}
}
void xuanzhuan();
void show();
};
void Array::xuanzhuan()//实现对方阵 a 进行顺时针 90 度的旋转;
{
int t[4][4]; //定义临时数组，用于保存旋转后，对应下标上的数值;
int i,j;
for(i=0;i<4;i++)
{
```

```
        for(j=0;j<4;j++)
        {
            t[j][i]=a[3-i][j]; //对方阵 a 进行顺时针 90 度旋转对应位所得值，赋予该
            //对应临时位上;
        }
    }
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            a[i][j]=t[i][j]; //将临时数组中的对应值保存到原数组中;
        }
    }
}

void Array::show() //输出数组元素;
{
    for(int i=0;i<4;i++)
    {
        for(int j=0;j<4;j++)
            cout<<a[i][j]<<"\t";
        cout<<endl;
    }
}

void main() //测试;
{
    int b[4][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
    Array test(b,4);
    cout<<"原始方阵为: "<<endl;
    test.show();
    test.xuanzhuan();
    cout<<"顺时针 90 度旋转后的方阵为: "<<endl;
    test.show();
}
```

#### 四、实践小结

理解方阵旋转的本质，利用临时数组，存放对应位上的数。

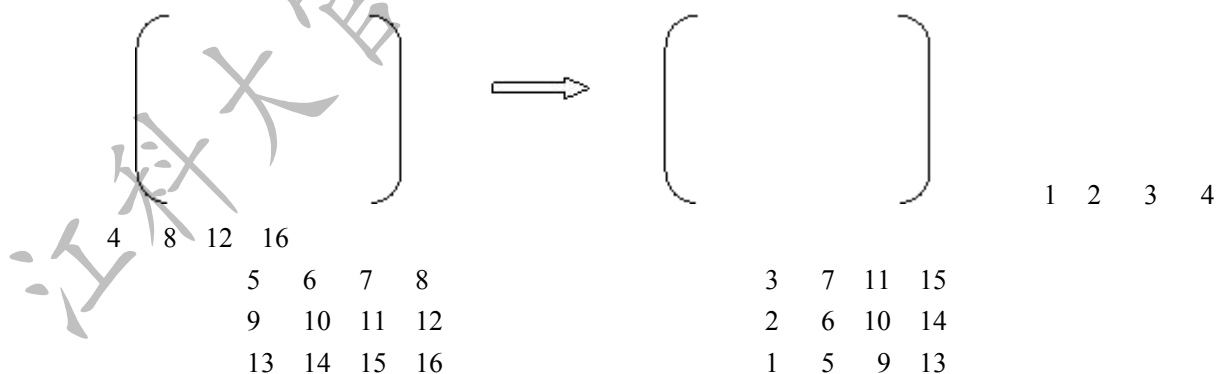
#### 五、运行结果

```
"G:\C++实践作业\Debug\第19题.exe"
原始方阵为：
1      2      3      4
5      6      7      8
9      10     11     12
13     14     15     16
顺时针90度旋转后的方阵为：
13     9      5      1
14     10     6      2
15     11     7      3
16     12     8      4
Press any key to continue.
```

## 任务十九

### 一、实践任务

20. 定义一个方阵类 Array，实现对方阵进行逆时针 90 度旋转。如图所示。



### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- int a[4][4]：用于存放方阵。

(2) 公有成员函数

- Array (int a1[][4],int n)：构造函数，用给定的参数 a1 初始化数据成员 a。
- void xuanzhuan ()：实现对方阵 a 进行逆时针 90 度的旋转。
- void show()：在屏幕上显示数组元素。

2、主要函数设计

在主程序中定义数组 int b[ ][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16} 作为原始数组。定义一个 Array 类对象 test，用 b 初始化 test，完成对该类的测试。

### 三、源程序清单

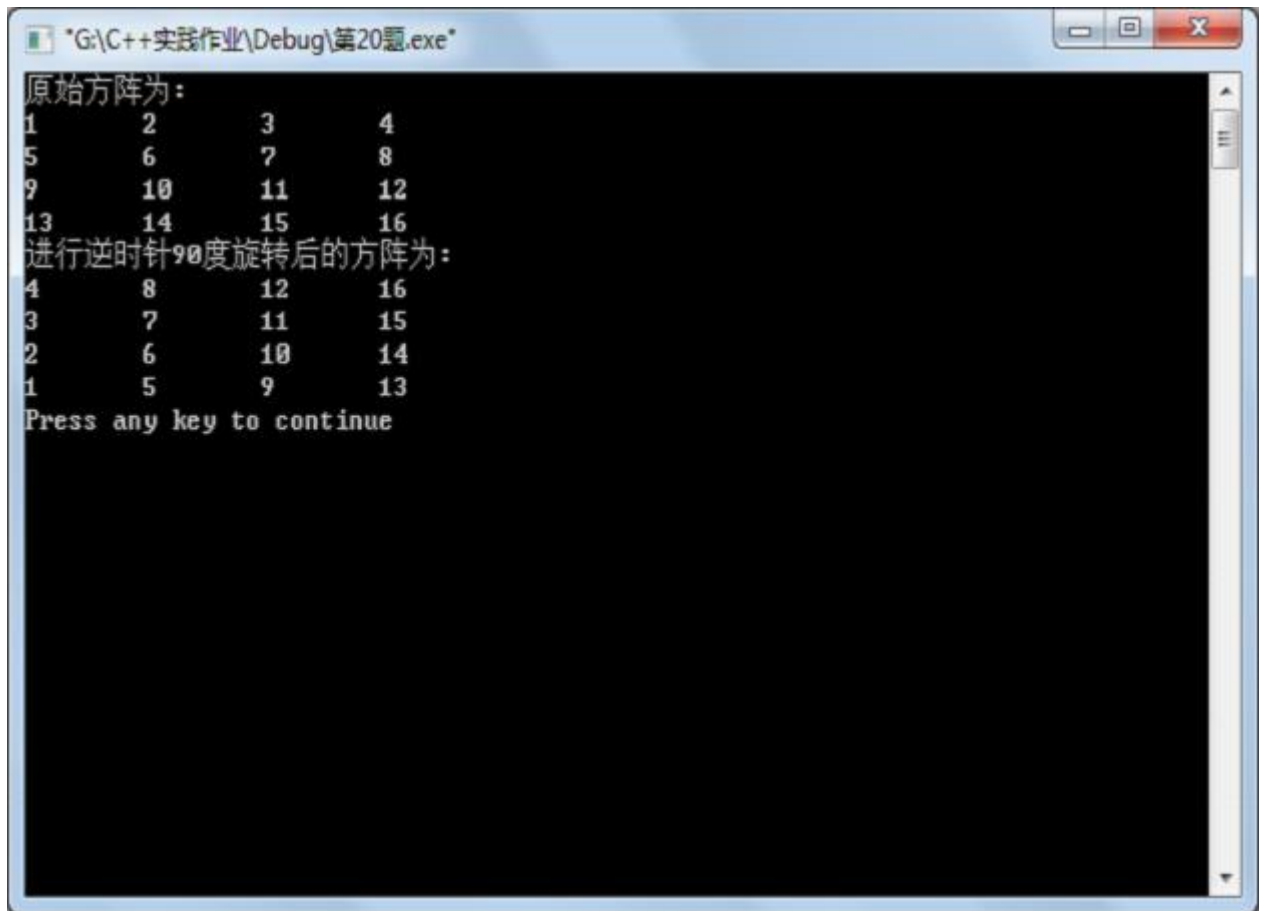
```
#include<iostream.h>
class Array{
private:
int a[4][4];
public:
Array(int a1[][4],int n)
{
for(int i=0;i<n;i++)
{
for(int j=0;j<4;j++)
a[i][j]=a1[i][j];
}
}
void xuanzhuan();
void show();
};
void Array::xuanzhuan()//实现对方阵 a 进行逆时针 90 度的旋转;
{
int i,j;
int t[4][4];//定义临时数组，用于保存原始方阵经逆时针 90 度旋转后对应下标上的
值;
for(i=0;i<4;i++)
{
for(j=0;j<4;j++)
t[3-j][i]=a[i][j];//将旋转后对应下标上的值赋给临时数组上的对应位置;
}
for(i=0;i<4;i++)
{
for(j=0;j<4;j++)
a[i][j]=t[i][j];//将保存在临时数组的值，赋给原始数组，从而原始数组完
成旋转;
}
}
void Array::show()//输出;
```

```
{
for(int i=0;i<4;i++)
{
    for(int j=0;j<4;j++)
        cout<<a[i][j]<<"\t";
    cout<<endl;
}
}
void main()//测试;
{
int b[][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
Array test(b,4);
cout<<"原始方阵为:"<<endl;
test.show();
test.xuanzhuan();
cout<<"进行逆时针 90 度旋转后的方阵为:"<<endl;
test.show();
}
```

#### 四、实践小结

理解方阵旋转的本质，利用临时数组，存放对应位上的数。

#### 五、运行结果



The screenshot shows a Windows-style application window titled "G:\C++实践作业\Debug\第20题.exe". The window contains a black console area with white text. The text displays a 4x4 matrix of numbers from 1 to 16, followed by the instruction "进行逆时针90度旋转后的方阵为:", and then the same 4x4 matrix rotated 90 degrees counter-clockwise. The original matrix is: 

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

. The rotated matrix is: 

4	8	12	16
3	7	11	15
2	6	10	14
1	5	9	13

. Below the matrices, the text "Press any key to continue" is visible.

## 任务二十

### 一、实践任务

21. 建立一个类 NUM, 求指定数据范围内的所有合数(非质数)。提示: 合数定义是“一个数, 除了 1 和它本身, 还有其它约数, 这样的数叫合数”。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int *data`: 动态存放在指定范围内求出的所有合数。
- `int span1, span2`: 存放指定范围的下限和上限。
- `int num`: 存放 `span1` 与 `span2` 之间的合数个数。

##### (2) 公有成员函数

- `NUM(int n1, int n2)`: 构造函数, 用参数 `n1` 和 `n2` 初始化 `span1` 和 `span2`, 同时初始化 `num`。
- `int isComposite(int x)`: 判断 `x` 是否为合数。若是合数, 返回 1, 否则, 返回 0。
- `void process()`: 求指定范围内的所有合数, 把它们依次存放在数组 `data` 中, 并将求出的合数个数赋给 `num`。
- `void print()`: 输出求出的素数个数及所有合数, 每行输出 8 个合数。
- `~NUM()`: 释放动态分配的存储空间。

## 2、主要函数设计

在主函数中完成对该类的测试。定义一个 NUM 类对象 test，指定查找范围为 100~200，即求 100 至 200 之间的所有合数。通过 test 调用成员函数完成求合数及输出合数的工作。

## 三、源程序清单

```
#include<iostream.h>
#include<math.h>
class NUM{
private:
    int*data;//动态存放在指定范围内求出的所有合数;
    int span1,span2;//存放指定范围的下限和上限;
    int num;//存放 span1 与 span2 之间的合数个数;
public:
    NUM(int n1,int n2)
    {
        data=new int[n2-n1+1];
        span1=n1;
        span2=n2;
        num=0;
    }
    int isComposite(int x);
    void process();
    void print();
    ~NUM()
    {if(data)delete data;}
};
int NUM::isComposite(int x)//判断 x 是否为合数。若是合数，返回 1，否则，返回 0;
{
    if(x>1)//比 1 大，但不是素数的数，称为合数;
    {
        if(x==2||x==3) return 0;
        for(int i=2;i<=sqrt(x);i++)
        {
            if(x%i==0)return 1;
        }
        if(i>sqrt(x)) return 0;
    }
    else return 0;
}
void NUM::process()//求指定范围内的所有合数，把它们依次存放在数组 data 中，并将求出的合数个数赋给 num;
{
    for(int i=span1,j=0;i<=span2;i++)
    {
```



```
        if(isComposite(i))
        {
            data[j]=i;
            j++;
            num++;
        }
    }
}

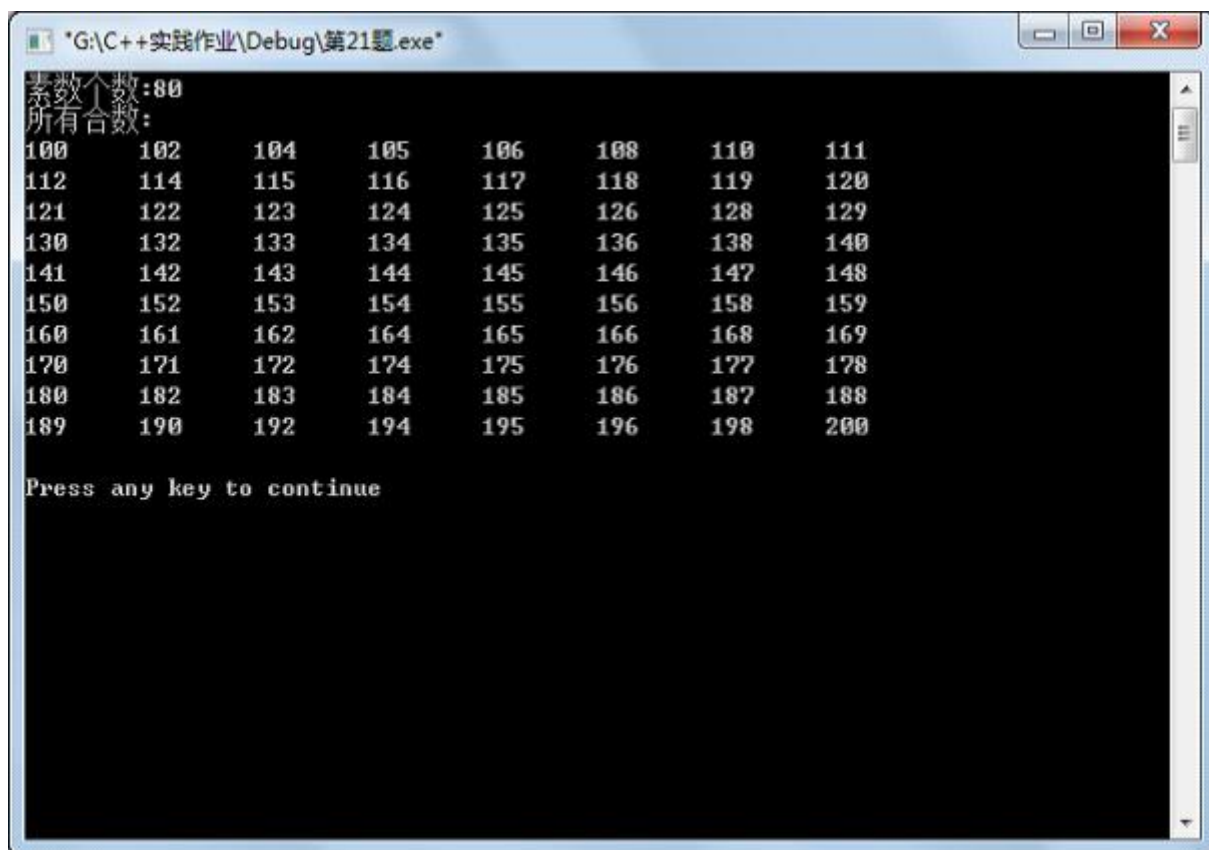
void NUM::print()//输出;
{
    cout<<"素数个数:"<<num<<endl;
    cout<<"所有合数:"<<endl;
    for(int i=0;i<num;i++)
    {
        cout<<data[i]<<"\t";
        if((i+1)%8==0) cout<<endl;//每行输出 8 个合数;
    }
    cout<<endl;
}

void main()//测试;
{
    NUM test(100,200);
    test.process();
    test.print();
}
```

#### 四、实践小结

理解素数的含义，熟练掌握判断素数的方法。

#### 五、运行结果



```
*G:\C++实践作业\Debug\第21题.exe*
素数个数:80
所有合数:
100    102    104    105    106    108    110    111
112    114    115    116    117    118    119    120
121    122    123    124    125    126    128    129
130    132    133    134    135    136    138    140
141    142    143    144    145    146    147    148
150    152    153    154    155    156    158    159
160    161    162    164    165    166    168    169
170    171    172    174    175    176    177    178
180    182    183    184    185    186    187    188
189    190    192    194    195    196    198    200

Press any key to continue
```

## 任务二十一

### 一、实践任务

22. 建立一个类 `Saddle_point`，求一个数组中的所有鞍点。提示：鞍点是这样的数组元素，其值在它所在行中为最大，在它所在列中为最小。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int a[4][4]`: 存放二维数组元素。
- `int b[4][4]`: 存放二维数组中的鞍点值。
- `int num`: 存放鞍点个数。

##### (2) 公有成员函数

- `Saddle_point(int data[][4])`: 构造函数，用参数 `int data[][4]` 初始化数组 `a`，同时初始化数组 `b` 与 `num` 的值均为 0。
- `void process()`: 求数组 `a` 所有鞍点（如果有鞍点），把它们行、列、及值相应存放在数组 `b` 中，并将求出的鞍点个数赋给 `num`。
- `void print()`: 输出数组 `a`、鞍点个数，与鞍点坐标及相应值。

#### 2、主要函数设计

在主程序中定义数组 `int b[ ][4]={2,6,3,4,5,6,5,5,5,7,6,7,1,9,2,7}` 作为原始数组。定义一个 `Saddle_point` 类对象 `fun`。通过 `fun` 调用成员函数完成求鞍点及输出工作。

### 三、源程序清单

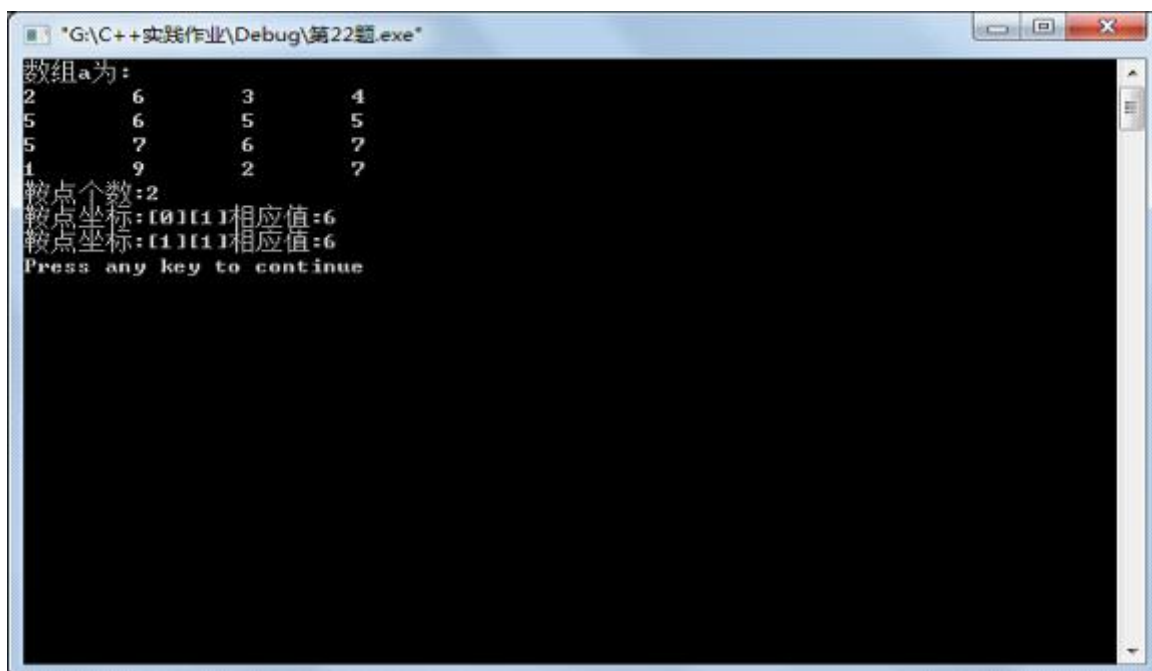
```
#include<iostream.h>
class Saddle_point{
private:
int a[4][4];
int b[4][4]; //存放二维数组中的鞍点值;
int num; //存放鞍点个数;
public:
Saddle_point(int data[][4])
{
for(int i=0;i<4;i++)
{
for(int j=0;j<4;j++)
{
a[i][j]=data[i][j];
b[i][j]=0;
}
}
num=0;
}
void process();
void print();
};
void Saddle_point::process() //求数组 a 所有鞍点,把它们行,列,及值相应存放在
数组 b 中,并将求出的鞍点个数赋给 num;
{
int i,j,k;
int m=0,n=0; //作为是否为鞍点的判断条件;
for(i=0;i<4;i++) //算法:依次判断数组中的每个元素是否为所在该行为最大,所在该
列为最小;
{
for(j=0;j<4;j++)
{
m=n=0;
for(k=0;k<4;k++)
{
if(a[i][j]>=a[i][k]) m++; //判断该元素是否为所在行最大;
else break;
if(a[i][j]<=a[k][j]) n++; //判断该元素是否为所在列最大;
else break;
}
if(m==4&& n==4) //满足条件即为鞍点;
{
```

```
        num++;
        b[i][j]=a[i][j];
    }
}
}
}
void Saddle_point::print()//输出数组 a，鞍点个数，与鞍点坐标及相应值；
{
    int i,j;
    cout<<"数组 a 为:"<<endl;
    for(i=0;i<4;i++)//输出数组 a;
    {
        for(j=0;j<4;j++)
            cout<<a[i][j]<<"\t";
        cout<<endl;
    }
    cout<<"鞍点个数:"<<num<<endl;
    if(num)
    {
        for(i=0;i<4;i++)
        {
            for(j=0;j<4;j++)
            {
                if(b[i][j])//当 b[i][j]不为 0 时，即其中此时对应下标保存有原数组 a
                中对应下标的鞍点值；
                {
                    cout<<" 鞍 点 坐 标 : "<<"["<<i<<"["<<j<<""]<<" 相 应
                    值:"<<b[i][j]<<endl;
                }
            }
        }
    }
    else
        cout<<"无鞍点"<<endl;
}
void main()
{
    int b[][4]={2,6,3,4,5,6,5,5,5,7,6,7,1,9,2,7};
    Saddle_point fun(b);
    fun.process();
    fun.print();
}
```

#### 四、实践小结

应理解鞍点的含义，再利用循环判断语句依次遍历寻找鞍点。

## 五、运行结果



```
*G:\C+++实践作业\Debug\第22题.exe*
数组a为:
2      6      3      4
5      6      5      5
5      7      6      7
1      9      2      7
鞍点个数:2
鞍点坐标:[0][1]相应值:6
鞍点坐标:[1][1]相应值:6
Press any key to continue
```

## 任务二十二

### 一、实践任务

23. 分数相加，两个分数分别是  $1/5$  和  $7/20$ ，它们相加后得  $11/20$ 。方法是先求出两个分数分母的最小公倍数，通分后，再求两个分子的和，最后约简结果分数的分子和分母（如果两个分数相加的结果是  $4/8$ ，则必须将其约简成最简分数的形式  $1/2$ ），即用分子分母的最大公约数分别除分子和分母。求  $m$ 、 $n$  最大公约数的一种方法为：将  $m$ 、 $n$  较小的一个数赋给变量  $k$ ，然后分别用  $\{k, k-1, k-2, \dots, 1\}$  中的数（递减）去除  $m$  和  $n$ ，第一个能把  $m$  和  $n$  同时除尽的数就是  $m$  和  $n$  的最大公约数。假定  $m$ 、 $n$  的最大公约数是  $v$ ，则它们的最小公倍数就是  $m*n/v$ 。试建立一个分数类 `Fract`，完成两个分数相加的功能。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int num, den` : `num` 为分子，`den` 为分母。

##### (2) 公有成员函数

- `Fract (int a=0,int b=1)` : 构造函数，用 `a` 和 `b` 分别初始化分子 `num`、分母 `den`。
- `int ged (int m, int n)` : 求 `m`、`n` 的最大公约数。此函数供成员 `add()` 函数调用。
- `Fract add (Fract f)` : 将参数分数 `f` 与对象自身相加，返回约简后的分数对象。
- `void show()` : 按照 `num/den` 的形式在屏幕上显示分数。

#### 2、主要函数设计

在主程序中定义两个分数对象 f1 和 f2，其初值分别是 1/5 和 7/20，通过 f1 调用成员函数 add 完成 f1 和 f2 的相加，将得到的分数赋给对象 f3，显示分数对象 f3。

### 三、源程序清单

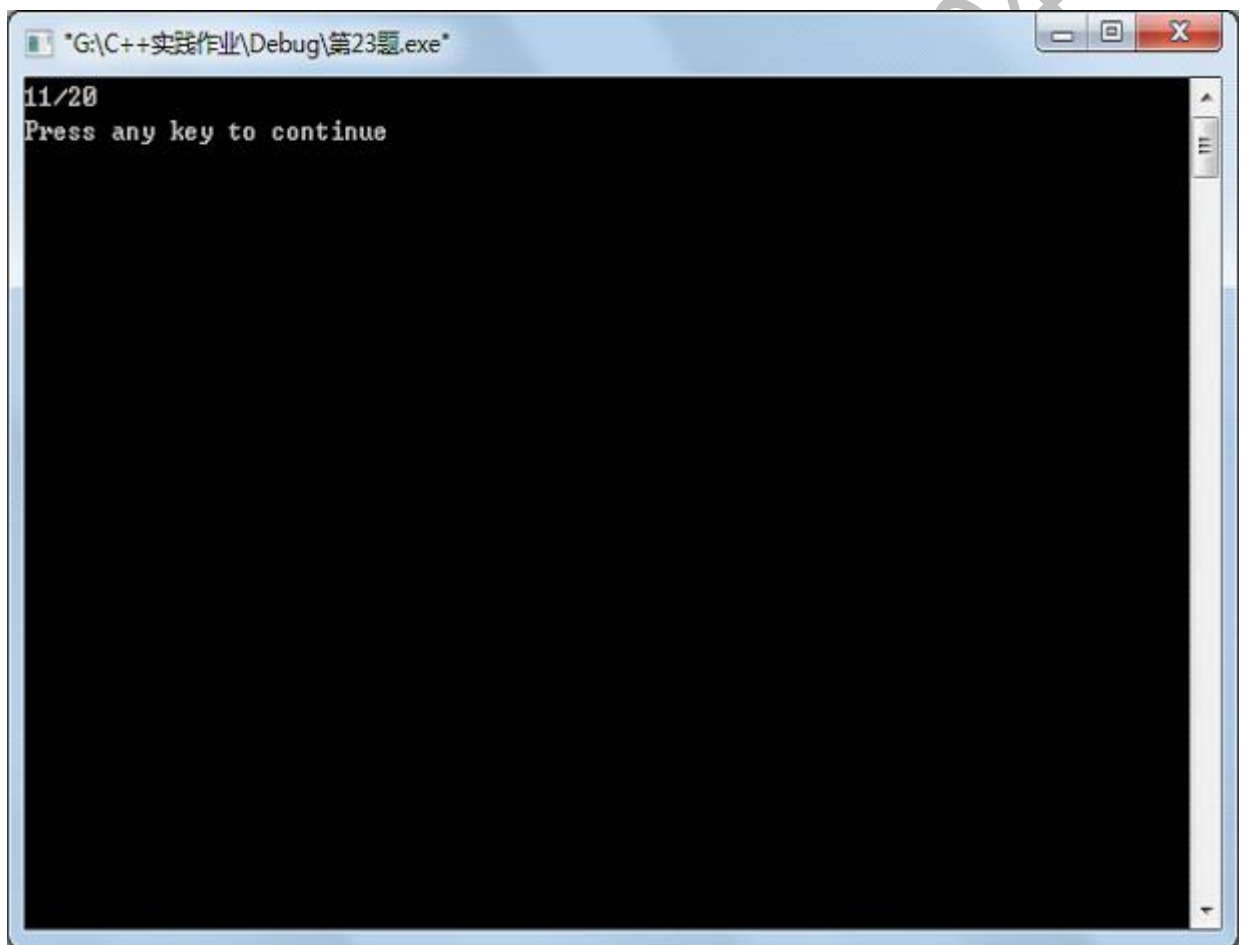
```
#include<iostream.h>
class Fract{
private:
int num,den;
public:
Fract(int a=0,int b=1)
{
    num=a;
    den=b;
}
int ged(int m,int n)//求 m, n 的最大公约数;
{
    int k;
    k=m<n?m:n;//将 m,n 中的较小的一个数赋给 k;
    while(1)//用 k 递去除 m 和 n，第一个能把 m 和 n 同时除尽的数就是 m 和 n 的
最大公约数;
    {
        if(m%k==0&& n%k==0)
            return k;
        else
            k--;
    }
}
Fract add(Fract f);
void show();
};
Fract Fract::add(Fract f)//将参数分数 f 与对象自身相加，返回约简后的分数对象;
{
    Fract s;
    s.den=this->den*f.den/ged(this->den,f.den);
    s.num=(this->num*f.den+this->den*f.num)/ged(this->den,f.den);
    s.den=s.den/ged(s.den,s.num);
    s.num=s.num/ged(s.den,s.num);
    return s;
}
void Fract::show()
{
    cout<<num<<"/"<<den<<endl;
}
void main()
```

```
{  
    Fract f1(1,5),f2(7,20),f3;  
    f3=f1.add(f2);  
    f3.show();  
}
```

#### 四、实践小结

理解函数中分数运算的实现方式，再依次完成。

#### 五、运行结果



### 任务二十三

#### 一、实践任务

24. 建立一个类 NUM，并统计特定序列中相同的数字的个数。

#### 二、详细设计

##### 1、类的描述与定义

##### (1) 私有数据成员

- `int data[25]`: 随机生成 25 个在 0-9 之间的数字。

- int num[10]: 储存每个数字出现的个数。

(2) 公有成员函数

- NUM(int data): 构造函数，初始化数组 data。
- void process(): 统计数组 data 中每个数字出现的个数，并保存到数组 num 中。
- void print(): 输出每个数字出现的个数，每行输出 5 个

2、主要函数设计

在主程序中定义一个对象，对该类进行测试。

三、源程序清单

```
#include<iostream.h>
#include<stdlib.h>
class NUM{
private:
int data[25];
int num[10];
public:
NUM(int data)
{
for(int i=0;i<25;i++)
{
this->data[i]=rand()%10; //随机生成 25 个在 0-9 之间的数字;
}
}
void process();
void print();
};
void NUM::process()
{
for(int i=0;i<10;i++)
{
int k=0;
for(int j=0;j<25;j++)
{
if(i==data[j])
k++; //记录值为 i 的数字出现的次数;
}
num[i]=k; //储存每个数字出现的个数;
}
}
void NUM::print() //输出每个数字出现的个数，每行输出 5 个;
{
for(int i=0;i<10;i++)
{
cout<<i<<"出现次数"<<num[i]<<"\t";
```

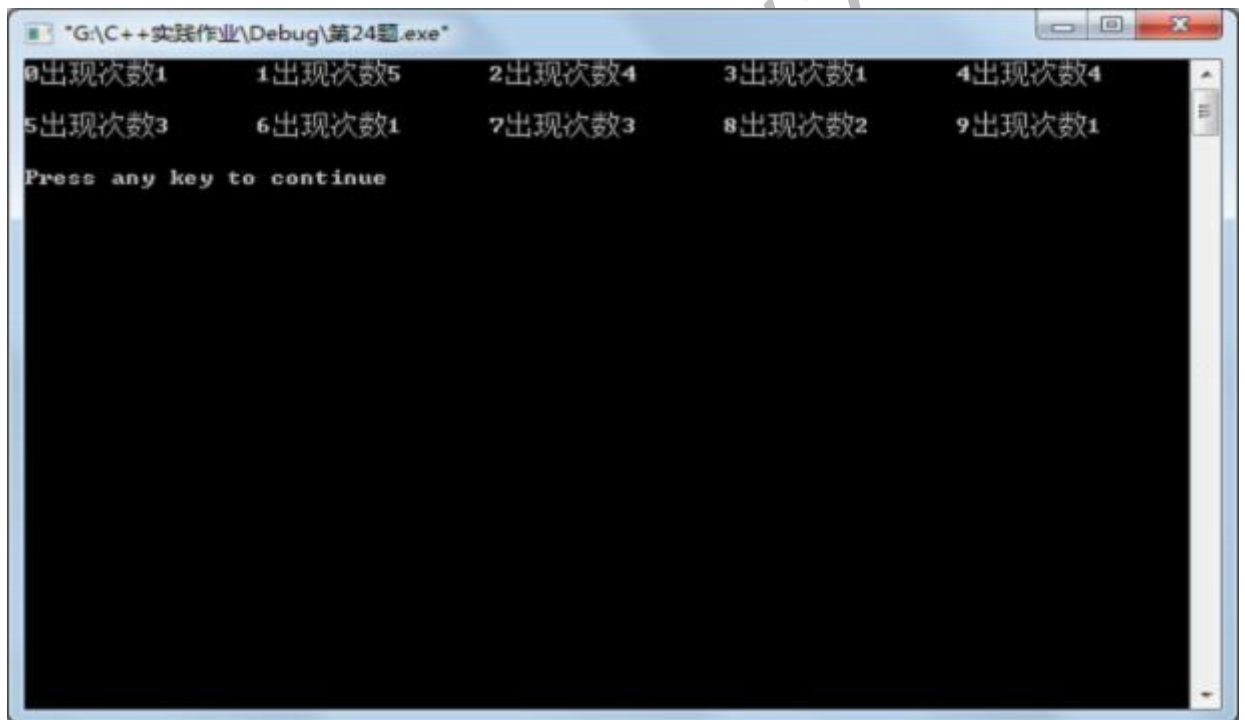


```
        if((i+1)%5==0) cout<<endl;
    }
}
void main()//测试;
{
    NUM test(10);
    test.process();
    test.print();
}
```

#### 四、实践小结

掌握数组，利用循环判断实现计数。

#### 五、运行结果



### 任务二十四

#### 一、实践任务

25. 建立一个类 NUM，并统计特定序列中相同的字符的个数。

#### 二、详细设计

##### 1、类的描述与定义

##### (1) 私有数据成员

- char data[25]: 随机生成 25 个字符。
- int num[128]: 储存每个字符出现的个数。

## (2) 公有成员函数

- NUM(int data): 构造函数，同时初始化数组 data。
- void process(): 统计数组 data 中每个字符出现的个数，并保存到数组 num 中。
- void print(): 输出每个出现过的字符及其出现的个数，每行输出 5 个，没有出现过的字符不显示。

## 2、主要函数设计

在主程序中定义一个对象，对该类进行测试。

## 三、源程序清单

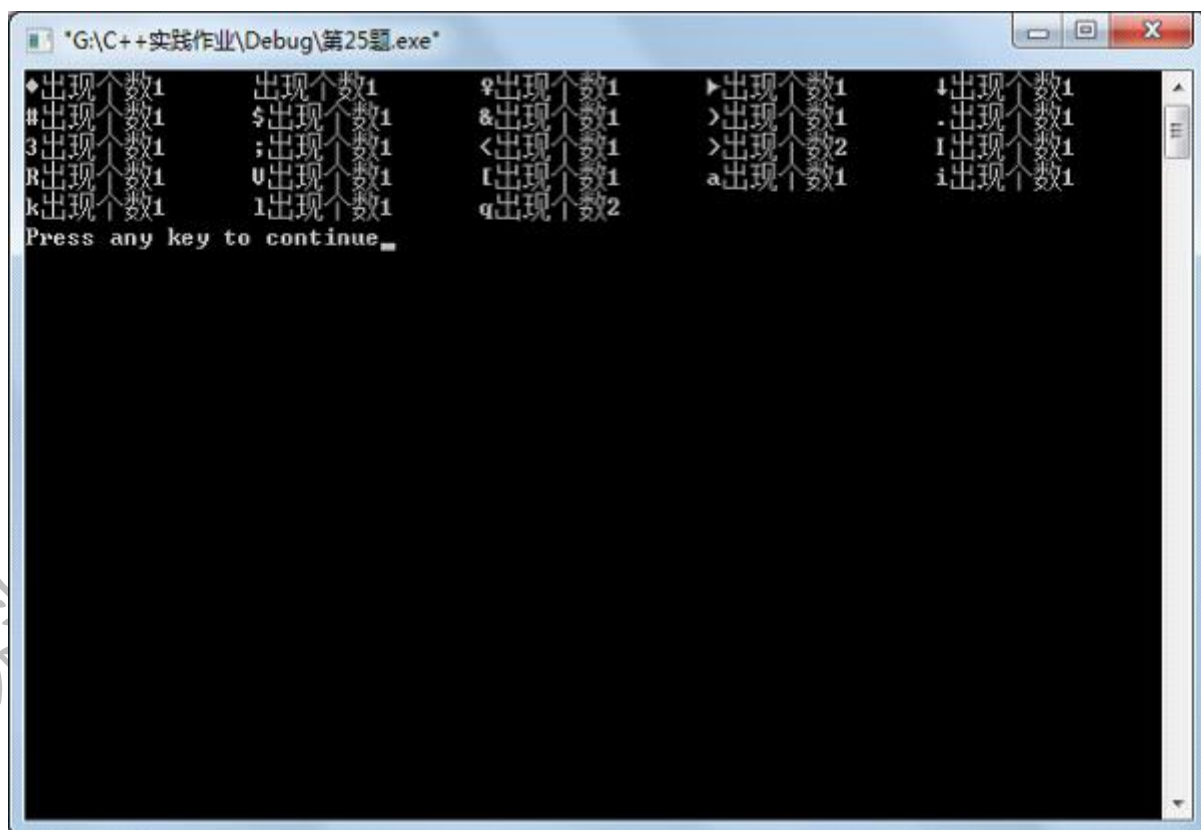
```
#include<iostream.h>
#include<stdlib.h>
class NUM{
private:
char data[25];
int num[128];
public:
NUM(int data)
{
for(int i=0;i<25;i++)
this->data[i]=rand()%data; //随机生成 25 个字符;
}
void process();
void print();
};
void NUM::process() //统计 data 中每个字符出现的个数，并保存到数组 num 中;
{
for(int i=0;i<128;i++)
{
int k=0;
for(int j=0;j<25;j++)
{
if(data[j]==i)
k++;
}
num[i]=k; //存储每个字符出现的个数;
}
}
void NUM::print() //输出每个出现过的字符及其出现的个数，每行输出 5 个，没有出现过的字符不显示;
{
for(int i=0;i<128;i++)
{
int k=0;
if(num[i])
```

```
{
    cout<<char(i)<<"出现个数"<<num[i]<<"\t";
    k++;
}
if((k+1)%5==0) cout<<endl;
}
cout<<endl;
}
void main()
{
    NUM test(128);
    test.process();
    test.print();
}
```

#### 四、实践小结

掌握数组，利用循环判断实现计数。

#### 五、运行结果



The screenshot shows a Windows command prompt window titled "G:\C++实践作业\Debug\第25题.exe". The output displays the frequency of characters in the string "1283Rk". The characters and their counts are as follows:

Character	Count
1	1
2	1
3	1
8	1
R	1
k	1

The output is formatted in a grid with 5 columns. The first column contains the characters, and the second column contains their counts. The text "Press any key to continue\_" is visible at the bottom of the window.

### 任务二十五

#### 一、实践任务

26. 建立一个类 NUM，随机生成 25 个字符序列，并为特定序列进行排序。

## 二、详细设计

### 1、类的描述与定义

#### (1) 私有数据成员

- `int data[25]`: 随机生成 25 个字符。

#### (2) 公有成员函数

- `NUM(int data[])`: 构造函数，初始化数组 `data`。
- `void process()`: 为数组 `data` 进行排序，要求按照 ASCII 码进行升序排列。
- `void print()`: 输出数组 `data`，每行输出 5 个字符。

### 2、主要函数设计

在主程序中定义一个对象，对该类进行测试。

## 三、源程序清单

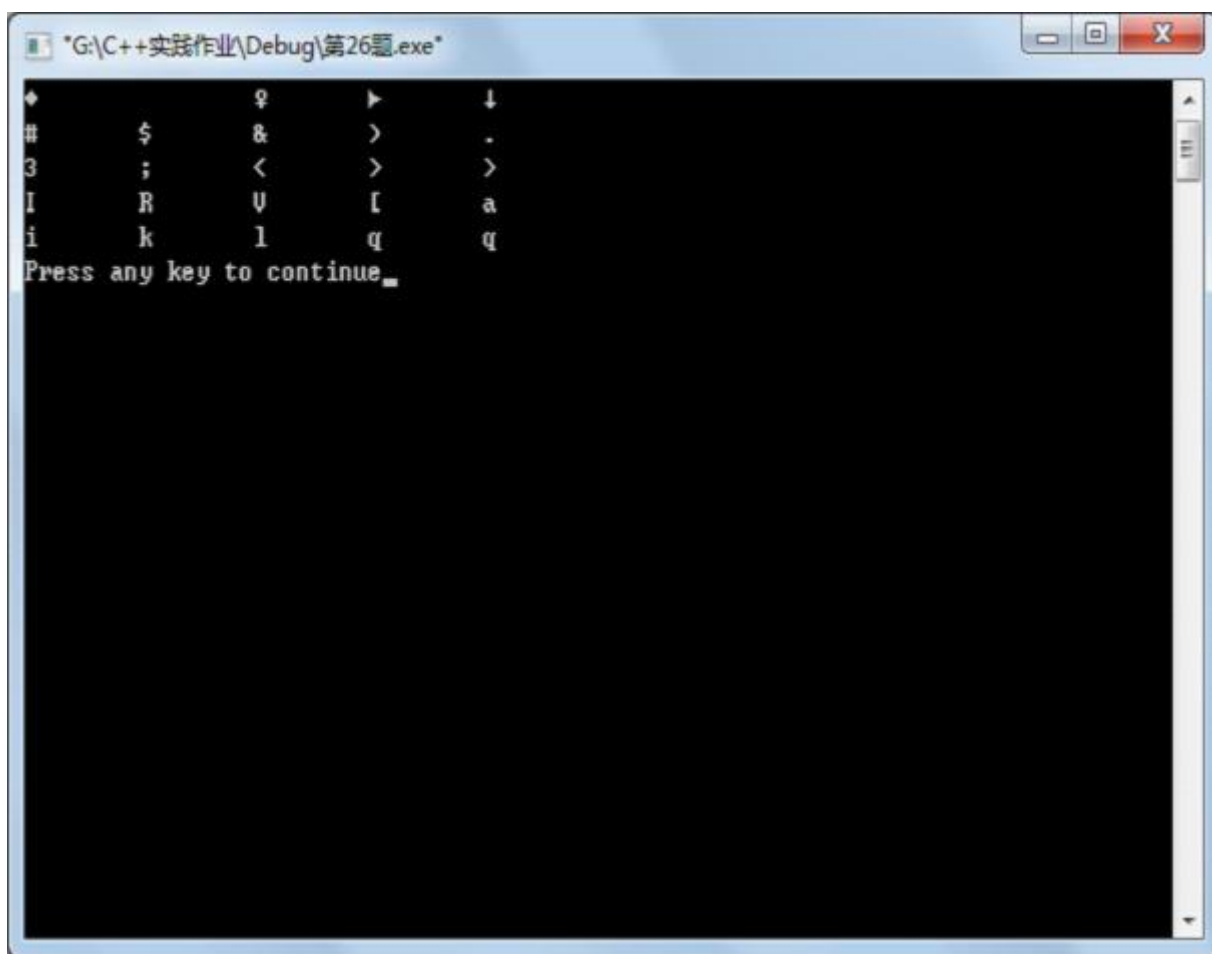
```
#include<iostream.h>
#include<stdlib.h>
class NUM{
private:
int data[25];
public:
NUM(int data)
{
for(int i=0;i<25;i++)
{
this->data[i]=rand()%data;
}
}
void process();
void print();
};
void NUM::process()//为数组 data 进行排序，按 ASCII 码升序排列;
{
for(int k,i=0;i<25-1;i++)
{
k=i;
for(int j=i+1;j<25;j++)
{
if(data[k]>data[j])
{
k=j;
}
}
}
if(k!=i)//优化程序的执行过程;
{
```

```
        j=data[k];
        data[k]=data[i];
        data[i]=j;
    }
}
}
void NUM::print()//输出数组 data,每行输出 5 个字符;
{
    for(int i=0;i<25;i++)
    {
        cout<<char(data[i])<<"\t";
        if((i+1)%5==0)cout<<endl;
    }
}
void main()//主函数中定义一个对象，对该类进行测试;
{
    NUM test(128);
    test.process();
    test.print();
}
```

#### 四、实践小结

掌握产生随机数的函数rand()及其所在头文件stdlib.h,再根据题目信息完成。

#### 五、运行结果



## 任务二十六

### 一、实践任务

27. 建立一个类 NUM，求指定数据范围内的所有素数（质数）。提示：素数定义是“只能被 1 和它本身整除的整数”，即质数。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- int data[10]: 依次存放原始数据。
- int prime[10]: 存放指定数据内所求出的所有素数。
- int num: 存放素数个数。

##### (2) 公有成员函数

- NUM(int n[]): 构造函数，用参数 n 初始化 data，同时初始化 num 为 0。
- int isprime(int x): 判断 x 是否为素数。若是素数，返回 1，否则，返回 0。
- void process(): 求指定 data 数组中的所有素数，把它们依次存放在数组 prime 中，并将求出的素数个数赋给 num。
- void print(): 输出求出的素数个数及所有素数，每行输出 4 个素数。

#### 2、主要函数设计

在主函数中完成对该类的测试。定义 NUM 类对象 test，通过 test 调用成员函数完成求素数及输出素数的工作。原始数据为{4,5,9,11,36,29,31,101,56,199}。

### 三、源程序清单

```
#include<iostream.h>
#include<math.h>
class NUM{
private:
int data[10];
int prime[10];
int num;
public:
NUM(int n[])
{
for(int i=0;i<10;i++)
data[i]=n[i]; //依次存放原始数据;
num=0;
}
int isprime(int x);
void process();
void print();
};
int NUM::isprime(int x) //判断 x 是否为素数。若是素数，返回 1，否则，返回 0;
{
if(x>1)
{
if(x==2||x==3) return 1;
else
{
for(int i=2;i<=sqrt(x);i++)
{
if(x%i==0) return 0;
}
if(i>sqrt(x)) return 1;
}
}
else return 0;
}
void NUM::process() //求指定 data 数组中的所有素数，把它们依次存放在数组 prime
中，并将求出的素数个数赋给 num;
{
for(int i=0,j=0;i<10;i++)
{
if(isprime(data[i]))
```

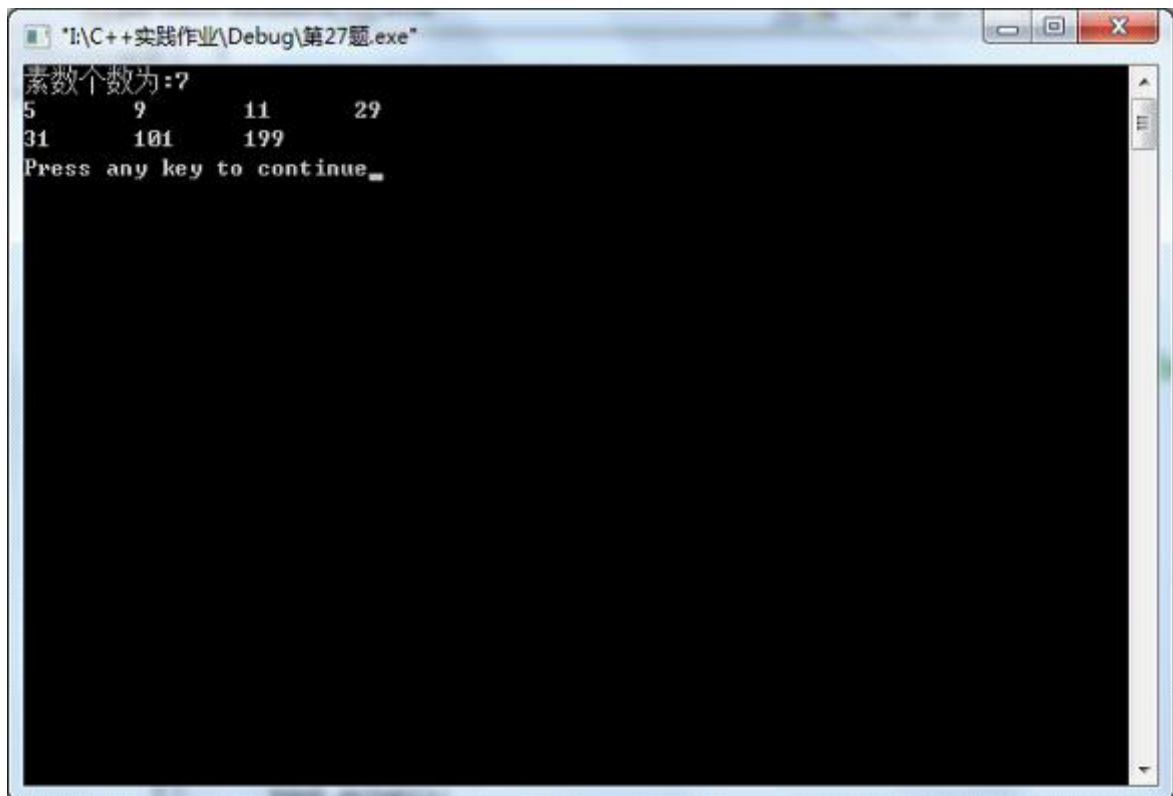
```
{
    prime[j]=data[i]; //存放指定数据内所求出的所有素数;
    j++;
    num++; //存放素数个数;
}
}
}
void NUM::print() //输出求出的素数个数及所有素数，把它们依次存放在数组 prime
中，并将求出的素数个数赋给 num;
{
    cout<<"素数个数为:"<<num<<endl;
    for(int i=0;i<num;i++)
    {
        cout<<prime[i]<<"\t";
        if((i+1)%4==0) cout<<endl;
    }
    cout<<endl;
}
void main()
{
    int n[]={4,5,9,11,36,29,31,101,56,199};
    NUM test(n);
    test.process();
    test.print();
}
```

#### 四、实践小结

应熟练掌握质数的判断方法。

#### 五、运行结果





## 任务二十七

### 一、实践任务

28. 编程实现对大于 1 的整数进行质因数分解，并求出其和。所谓整数的质因子分解是指将整数分解为其所有质数（素数）因数的积，例如， $60=2*2*3*5$ ，则整数 60 的质因数之和为 12。定义一个类 Decompose 实现上述功能。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int *a`: 指向存放对应整数的质因数之和的动态存储空间。
- `int *num`: 指向待分解质因数整数的动态存储空间。
- `int n`: 整数的个数。

##### (2) 公有成员函数

- `Decompose(int m, int b[ ])`: 用 `m` 初始化 `n`，并用 `n` 初始化为动态申请空间的指针 `a` 与 `num`。用参数 `b` 给数组 `a` 赋值。
- `void print()`: 输出数组 `a` 以及 `num` 所指向的存储空间中的内容。
- `void primenum()`: 求整数 `a[i]` 的所有质因数（保留重复部分，例如 60 的质因数为 2,2,3,5，之和为 12），并将这些质因数之和存放到指针 `num` 所指向的存储空间中。
- `~Decompose()`: 释放动态分配的存储空间。

#### 2、主要函数设计

在主函数中完成对该类的测试。从键盘输入一组大于 1 的整数，存放在 `number` 数组中，定义类 `Decompose` 的对象 `d`，并用 `number` 初始化 `d`，调用函数 `primenum()` 求 `number` 的所有质因数，最后输出测试结果。

### 三、源程序清单

```
#include<iostream.h>
#include<iomanip.h>
#include<math.h>
class Decompose{
private:
    int*a;//指向待分解质因数整数的动态存储空间;
    int*num;//指向存放对应整数的质因数之和的动态存储空间;
    int n;//整数的个数;
public:
    Decompose(int m,int b[])
    {
        n=m;
        a=new int[n];
        num=new int[n];
        for(int i=0;i<n;i++)
            a[i]=b[i];
        for(i=0;i<n;i++)
            num[i]=0;
    }
    void print();
    void primenum();
    ~Decompose()
    {
        if(a) delete []a;
        if(num) delete []num;
    }
};

void Decompose::print()//输出数组 a 以及 num 所指向的存储空间中的内容;
{
    cout<<"待分解整数及对应整数的质因数之和:"<<endl<<endl;
    for(int i=0;i<n;i++)
    {
        cout<<a[i]<<"质因数之和"<<num[i]<<"\t";
        if((i+1)%4==0)cout<<endl;
    }
    cout<<endl;
}

void Decompose::primenum()//求整数 a[i]的所有质因数，并将这些质因数之和存放到指针 num 所指向的存储空间中;
```

```
{
int i,j,k,turn;
for(i=0;i<n;i++)
{
    turn=a[i];
    if(turn==2||turn==3) num[i]=turn;
    else
    {
        for(j=2;j<=turn;j++)//依次在小于或等于 turn（即 a[i]）的整数中寻找其质
        因数;
        {
            if(turn%j==0)
            {
                if(j==2||j==3)
                {
                    num[i]+=j;
                    turn/=j;
                    j--;
                }
                else
                {
                    for(k=2;k<=sqrt(j);k++)//判断该整数是否为质数;
                    {
                        if(j%k==0) break;
                    }
                    if(k>sqrt(j))
                    {
                        num[i]+=j;
                        turn/=j;
                        j--;//自减，从而继续判断该质因数是否仍能被整除;
                    }
                }
            }
        }
    }
}

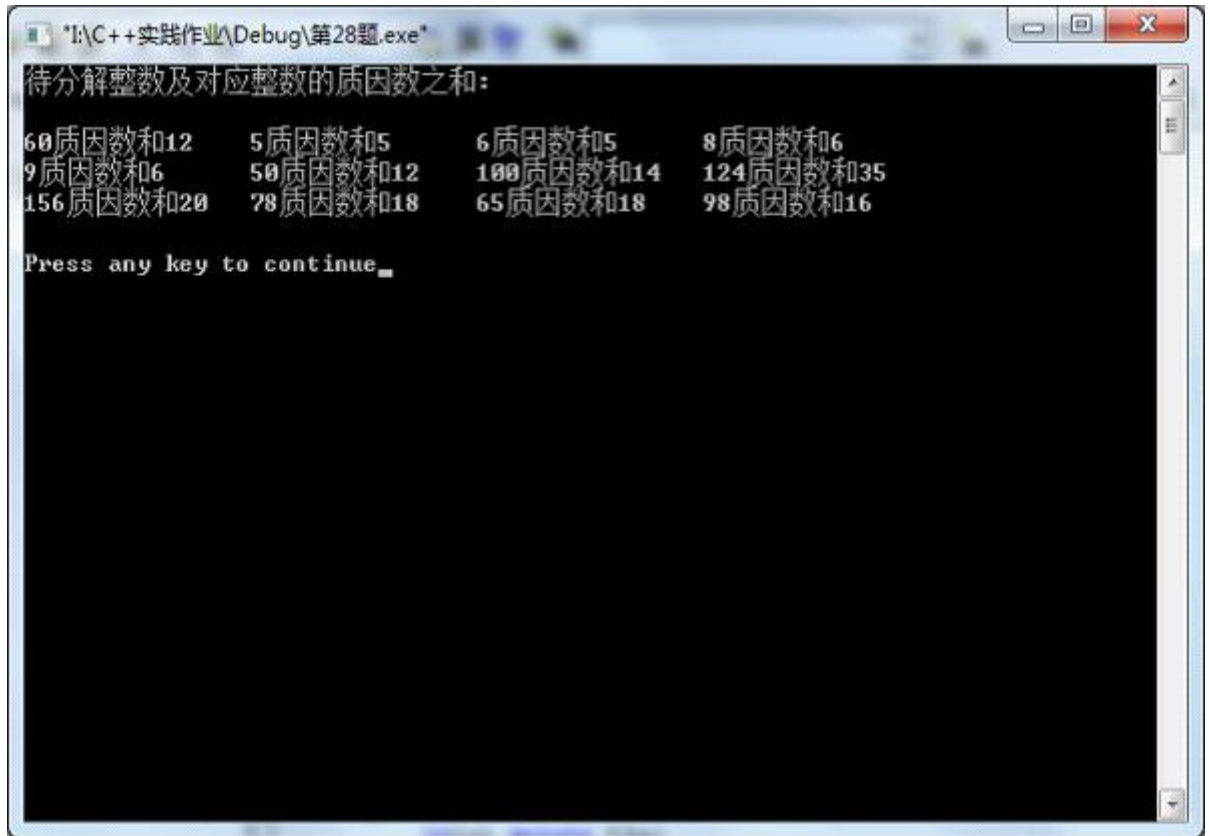
void main()
{
int number[]={60,5,6,8,9,50,100,124,156,78,65,98};
int n=sizeof(number)/sizeof(int);
Decompose d(n,number);
d.primenum();
d.print();
}
```

}

#### 四、实践小结

熟练掌握质数的判断方法。

#### 五、运行结果



### 任务二十八

#### 一、实践任务

29. 建立一个类 SUM，输入  $5 \times 5$  的二维数组，编写程序实现：求出两对角线上各元素的和，求出对角线上行、列下标均为偶数的各元素的积，找出对角线上其值最大的元素以及它在数组中的位置。

#### 二、详细设计

##### 1、类的描述与定义

###### (1) 私有数据成员

- `int array[5][5]`: 二维整型数组。
- `int s`: 数组 `array` 两对角线元素的和。
- `int a`: 数组 `array` 对角线上行、列下标均为偶数的各元素的积
- `int b,m,n`: 数组 `array` 对角线上其值最大的元素以及它在数组中的位置。

###### (2) 公有成员函数

- SUM(int d[5][5]): 构造函数，初始化成员数据。
- void process1(): 求二维数组两对角线元素的和。
- void process2(): 求二维数组两对角线上行、列下标均为偶数的各元素的积。
- void process3(): 求二维数组两对角线上其值最大的元素和它在数组中的位置。
- void print(): 输出二维数组（每行输出 5 个元素）及其它所求的值。

## 2、主要函数设计

在主程序中对该类进行测试。

## 三、源程序清单

```
#include<iostream.h>
class SUM{
private:
    int array[5][5];
    int s;//数组 array 两对角线元素的和;
    int a;//数组 array 对角线上行，列下标均为偶数的各元素的积;
    int b, m, n;//数组 array 对角线上其值最大的元素以及它在数组中的位置;
public:
    SUM(int d[5][5])
    {
        for(int i=0;i<5;i++)
        {
            for(int j=0;j<5;j++)
                array[i][j]=d[i][j];
        }
    }
    void process1();
    void process2();
    void process3();
    void print();
};
void SUM::process1()//求二维数组两对角线元素的和;
{
    int i,j;
    s=0;
    for(i=0;i<5;i++)
    {
        for(j=0;j<5;j++)
        {
            if(i==j||i+j==4)
                s+=array[i][j];
        }
    }
}
void SUM::process2()//求二维数组两对角线上行，列下标均为偶数的各元素的积;
```

```
{
int i,j;
a=1;
for(i=0;i<5;i++)
{
    for(j=0;j<5;j++)
    {
        if(i==j||i+j==4)
        {
            if(i%2==0&& j%2==0)
                a*=array[i][j];
        }
    }
}
}

void SUM::process3()//求二维数组两对角线上其值最大的元素和它在数组中的位置;
{
int i,j;
b=array[0][0];
m=n=0;
for(i=0;i<5;i++)
{
    for(j=0;j<5;j++)
    {
        if(i==j||i+j==4)
        {
            if(b<array[i][j])
            {
                b=array[i][j];
                m=i;
                n=j;
            }
        }
    }
}
}

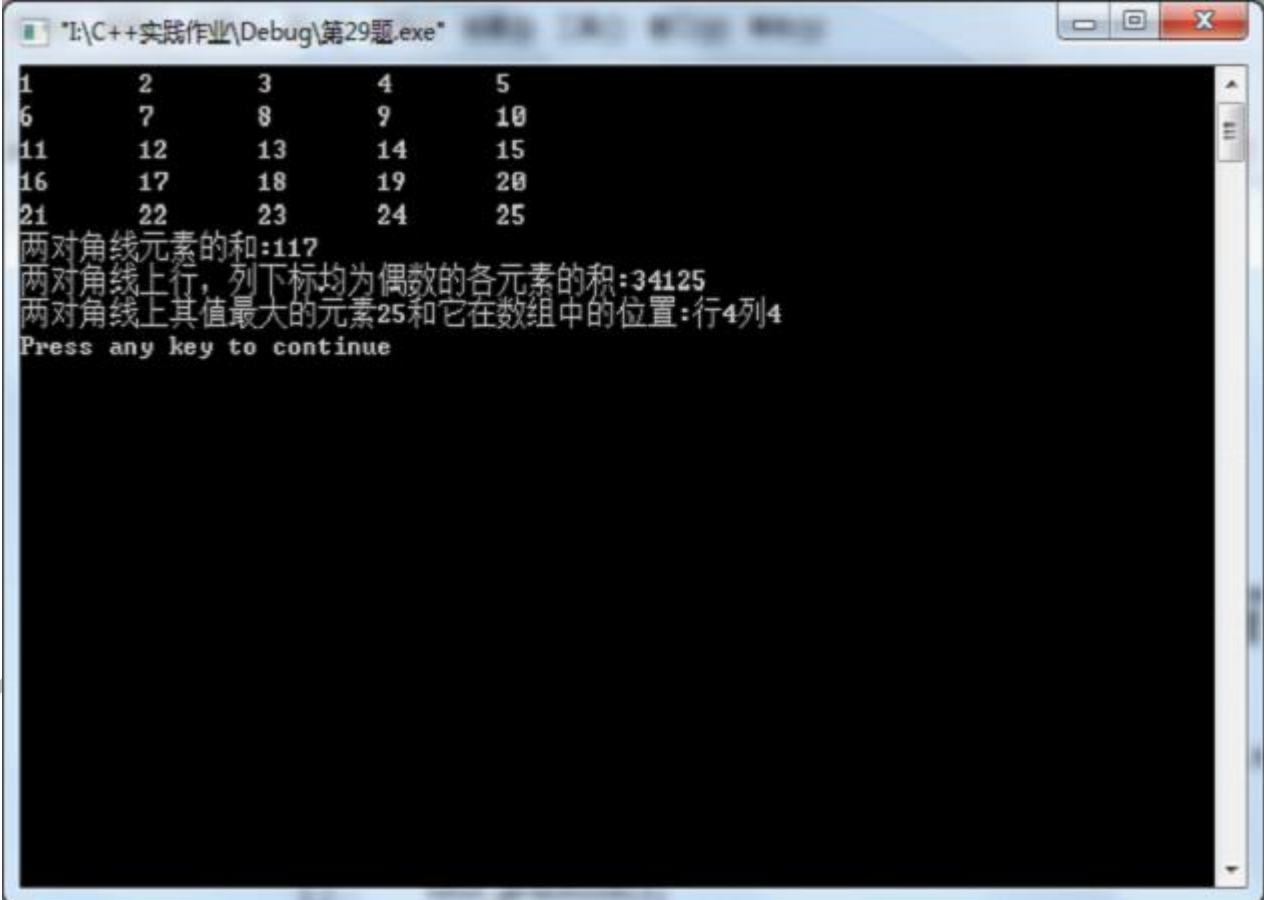
void SUM::print()//输出;
{
for(int i=0;i<5;i++)
{
    for(int j=0;j<5;j++)
        cout<<array[i][j]<<"\t";
    cout<<endl;
}
}
```

```
cout<<"两对角线元素的和:"<<s<<endl;
cout<<"两对角线上行，列下标均为偶数的各元素的积:"<<a<<endl;
cout<<"两对角线上其值最大的元素"<<b<<"和它在数组中的位置:"<<"行"<<m<<"列"
"<<n<<endl;
}
void main()//测试;
{
int d[5][5]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25};
SUM test(d);
test.process1();
test.process2();
test.process3();
test.print();
}
```

#### 四、实践小结

掌握数组中，对角线元素所在位置规律及如何寻找数组中相关条件的数。

#### 五、运行结果



The screenshot shows a Windows command prompt window titled "I:\C++\实践作业\Debug\第29题.exe". The output displays a 5x5 array of numbers from 1 to 25, followed by three lines of calculated results and a prompt to press any key to continue.

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

两对角线元素的和:117  
两对角线上行，列下标均为偶数的各元素的积:34125  
两对角线上其值最大的元素25和它在数组中的位置:行4列4  
Press any key to continue

## 一、实践任务

30. 建立一个矩阵类 `Array`，对二维数组中左下三角的全部元素（包括对角线上的元素）作如下变换：（1）若该数不是素数则保持不变；（2）若该数是素数，则用大于它的最小素数替换该数。并统计二维数组中左下三角的全部元素（包括对角线上的元素）中的素数个数。

## 二、详细设计

### 1、类的描述与定义

#### （1）私有数据成员

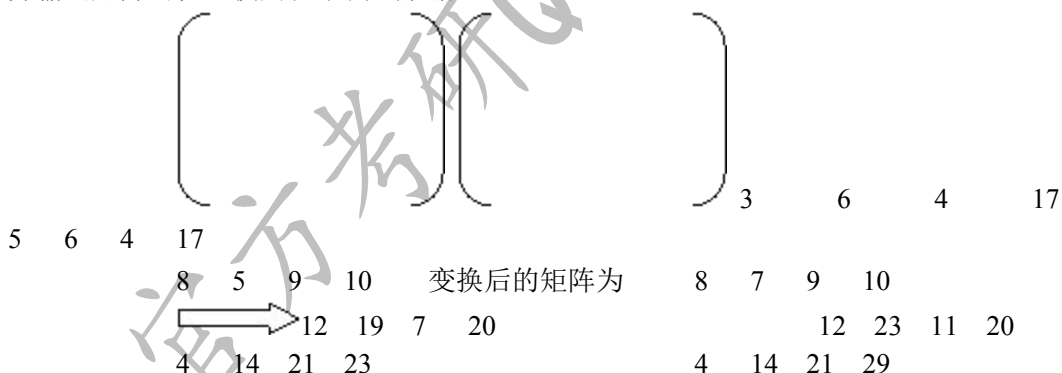
- `int x[4][4]`：存储需要处理的二维数组的各元素值。
- `int count`：存储左下三角元素中素数的个数。

#### （2）公有成员函数

- 构造函数：进行初始化 `x` 数组和 `count` 的值。
- `int fun(int)`：判断一个数是否为素数的函数。
- `int encode()`：对 `x` 数组中左下三角的全部元素（包括对角线上的元素）逐一进行判断，若该数不是素数则保持不变，若该数是素数，则用大于它的最小素数替换该数。
- `void print()`：按行输出矩阵的值。

### 2、主要函数设计

编写一个程序测试该类，说明（声明）`Array` 对象 `A`，将一个矩阵存入对象 `A` 中，并输出矩阵的值，使用以下测试数据：



## 三、源程序清单

```
#include<iostream.h>
#include<math.h>
class Array{
private:
int x[4][4];
int count;//存储左下三角元素中素数的个数;
public:
Array(int x[4][4])//初始化 x 数组和 count 的值;
{
int i,j;
```



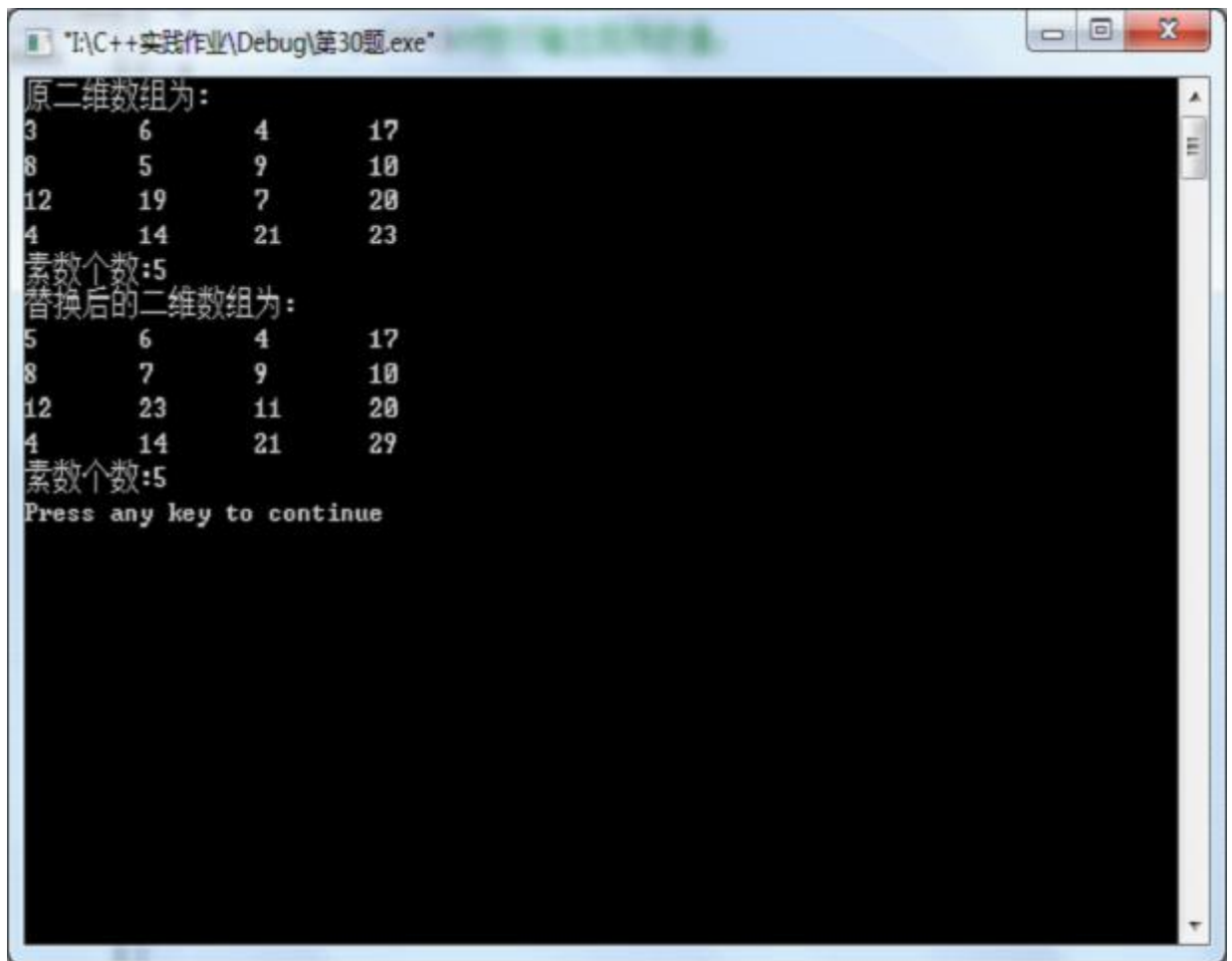
```
for(i=0;i<4;i++)
{
    for(j=0;j<4;j++)
        this->x[i][j]=x[i][j];
}
count=0;
for(i=0;i<4;i++)
{
    for(j=0;j<=i;j++)
    {
        if(fun(this->x[i][j]))
        {
            count++;
        }
    }
}
}
int fun(int);
int encode();
void print();
};
int Array::fun(int x)//判断一个数是否为素数，是返回 1，不是返回 0;
{
    if(x>1)
    {
        if(x==2||x==3) return 1;
        else
        {
            for(int i=2;i<=sqrt(x);i++)
            {
                if(x%i==0) break;
            }
            if(i>sqrt(x)) return 1;
        }
    }
    return 0;
}
int Array::encode()//对 x 数组中左下三角的全部元素逐一进行判断，若该数不是素数则保持不变，若是则用大于它的最小素数替代;
{
    int i,j;
    for(i=0;i<4;i++)
    {
        for(j=0;j<=i;j++)
```

```
{
    if(fun(x[i][j]))
    {
        while(1)
        {
            if(fun(++x[i][j]))//当概率是素数时，寻找大于它的最小素数替
代;
                break;
            }
        }
    }
}
return 0;
}
void Array::print()//按行输出矩阵的值;
{
    for(int i=0;i<4;i++)
    {
        for(int j=0;j<4;j++)
        {
            cout<<x[i][j]<<"\t";
        }
        cout<<endl;
    }
    cout<<"素数个数:"<<count<<endl;
}
void main()
{
    int x[4][4]={3,6,4,17,8,5,9,10,12,19,7,20,4,14,21,23};
    Array A(x);
    cout<<"原二维数组为:"<<endl;
    A.print();
    A.encode();
    cout<<"替换后的二维数组为:"<<endl;
    A.print();
}
```

#### 四、实践小结

熟练掌握判断素数的方法。

#### 五、运行结果



The screenshot shows a Windows command prompt window titled "I:\C++\实践作业\Debug\第30题.exe". The program displays the following output:

```
原二维数组为:
3      6      4      17
8      5      9      18
12     19     7      28
4      14     21     23
素数个数:5
替换后的二维数组为:
5      6      4      17
8      7      9      18
12     23     11     28
4      14     21     29
素数个数:5
Press any key to continue
```

## 任务三十

### 一、实践任务

31. 建立一个类 SUM，实现  $m$  行  $k$  列矩阵与  $k$  行  $n$  列矩阵的乘积。设  $A$  为  $m$  行  $k$  列的矩阵， $B$  为  $k$  行  $n$  列的矩阵，则  $C=A \times B$ 。

具体要求如下：

```
const int m=3;
```

```
const int k=4;
```

```
const int n=3;
```

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int A[m][k]`: 存放  $m$  行  $k$  列矩阵。
- `int B[k][n]`: 存放  $k$  行  $n$  列矩阵
- `int (*C)[n]`: 指向乘积矩阵

##### (2) 公有成员函数

- 构造函数：初始化成员数据。

- 析构函数：收回行指针。
- void process(): 求矩阵的乘积。
- void print(): 输出各二维数组（按行列形式）。

## 2、主要函数设计

在主程序中对该类进行测试。

## 三、源程序清单

```
#include<iostream.h>
const int m=3;
const int k=4;
const int n=3;
class SUM{
private:
int A[m][k];
int B[k][n];
int (*C) [n]; //指向乘积矩阵;
public:
SUM(int a[m][k],int b[k][n])
{
    int i,j;
    for(i=0;i<m;i++)
    {
        for(j=0;j<k;j++)
            A[i][j]=a[i][j];
    }
    for(i=0;i<k;i++)
    {
        for(j=0;j<n;j++)
            B[i][j]=b[i][j];
    }
    C=new int[m][n];
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
            C[i][j]=0;
    }
}
~SUM()
{if(C)delete []C;}
void process();
void print();
};
void SUM::process() //求矩阵的乘积;
{
```

```
int i,j,l;
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
    {
        for(l=0;l<k;l++)
            C[i][j]+=A[i][l]*B[l][j]; //算法：根据矩阵的乘法方式；
    }
}
}

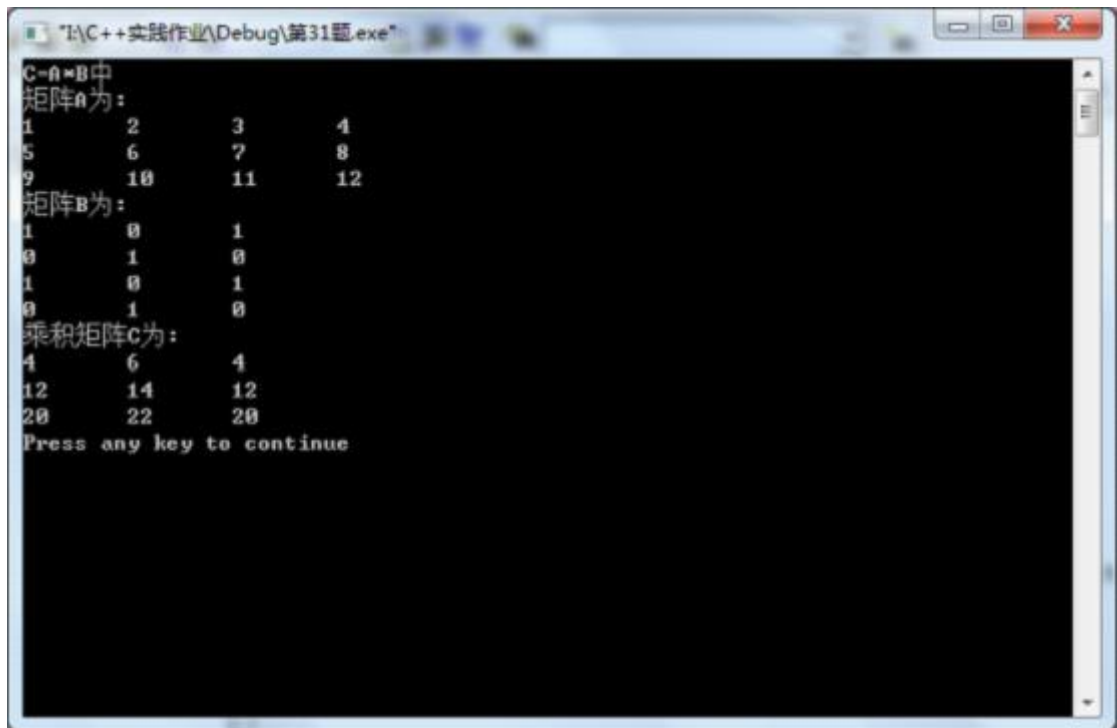
void SUM::print() //输出;
{
    int i,j;
    cout<<"C=A*B 中"<<endl;
    cout<<"矩阵 A 为:"<<endl;
    for(i=0;i<m;i++)
    {
        for(j=0;j<k;j++)
            cout<<A[i][j]<<"\t";
        cout<<endl;
    }
    cout<<"矩阵 B 为:"<<endl;
    for(i=0;i<k;i++)
    {
        for(j=0;j<n;j++)
            cout<<B[i][j]<<"\t";
        cout<<endl;
    }
    cout<<"乘积矩阵 C 为:"<<endl;
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
            cout<<C[i][j]<<"\t";
        cout<<endl;
    }
}

void main() //测试;
{
    int a[m][k]={1,2,3,4,5,6,7,8,9,10,11,12},b[k][n]={1,0,1,0,1,0,1,0,1,0};
    SUM test(a,b);
    test.process();
    test.print();
}
```

#### 四、实践小结

熟练掌握矩阵的乘法，利用 3 重循环实现。

## 五、运行结果



```
T:\C++\实践作业\Debug\第31题.exe
C=A*B中
矩阵A为:
1      2      3      4
5      6      7      8
9      10     11     12
矩阵B为:
1      0      1
0      1      0
1      0      1
0      1      0
乘积矩阵C为:
4      6      4
12     14     12
20     22     20
Press any key to continue
```

## 任务三十一

### 一、实践任务

32. 建立一个类 SUM，使用二维数组输入“Follow me”，“BASIC”，“Great wall”，“Fortran”，“Pascal”，将它们按从小到大的顺序排列后输出。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int *p[5]`: 存放二维数组每行的字符串的地址。

##### (2) 公有成员函数

- `SUM(int d[5][5])`: 构造函数，初始化成员数据。
- `void process1()`: 对二维数组中存放的字符串进行排序。
- `void print()`: 输出二维数组中排好序的字符串。

#### 2、主要函数设计

在主程序中对该类进行测试。

### 三、源程序清单

```
#include<iostream.h>
#include<string.h>
class SUM{
```

```
private:
    int*p[5]; //存放二维数组每行的字符串的地址;
public:
    SUM(int d[5][5])
    {
        for(int i=0;i<5;i++)
            p[i]=(int*)d[i][0]; //整型数组每行第0列保存字符串的地址(为int类型),
    }
    void process1();
    void print();
};
void SUM::process1() //对二维数组中存放的字符串进行排序;
{
    int*t;
    for(int i=0;i<5;i++)
    {
        for(int j=i+1;j<5;j++)
        {
            if(strcmp((char*)p[i], (char*)p[j])>0) //将int*类型数据转换为char*类型, 才可利用strcmp()函数对该地址值上的字符串比较, 排序;
            {
                t=p[i];
                p[i]=p[j];
                p[j]=t;
            }
        }
    }
}
void SUM::print()
{
    for(int i=0;i<5;i++)
    {
        cout<<(char*)p[i]<<endl;
    }
    cout<<endl;
}
void main()
{
    //每个字符串的值为该字符串的首元素地址,故将每个地址值转换为整型数值保存在整型数组中;
    Int          d[5][5]={ {int("Follow me"),{int("BASIC")},{int("Great wall")},{int("Fortran")},{int("Pascal")}}};
    SUM test(d);
    cout<<"原字符串为:"<<endl;
```

```
test.print();  
test.process1();  
cout<<"排好序的字符串为:"<<endl;  
test.print();  
}
```

#### 四、实践小结

掌握各数据类型之间的转化。

#### 五、运行结果



The screenshot shows a Windows command prompt window titled "I:\C++\实践作业\Debug\第32题.exe". The output of the program is as follows:

```
原字符串为:  
Follow me  
BASIC  
Great wall  
Fortran  
Pascal  
  
排好序的字符串为:  
BASIC  
Follow me  
Fortran  
Great wall  
Pascal  
  
Press any key to continue_
```

### 任务三十二

#### 一、实践任务

33. 建立一个类 Integer\_String，把一个正整数转换为字符串。

#### 二、详细设计

##### 1、类的描述与定义

##### (1) 私有数据成员

- int num: 要转换的正整数。



- char \*s: 用动态空间存储转换得到的字符串。

## (2) 公有成员函数

- Integer\_String(int n): 用参数 n 初始化数据成员 num。
- int f(): 求数据成员 num 的位数。
- void fun(): 把正整数 num 转换为字符串 s。
  - void show(): 输出数据成员 num 和 s;
  - ~Integer\_String(): 释放动态空间。

## 2、主要函数设计

在主函数中对定义类进行测试。用正整数 12345 初始化类 Integer\_String 的对象 test, 调用相关成员函数后输出转换结果。

## 三、源程序清单

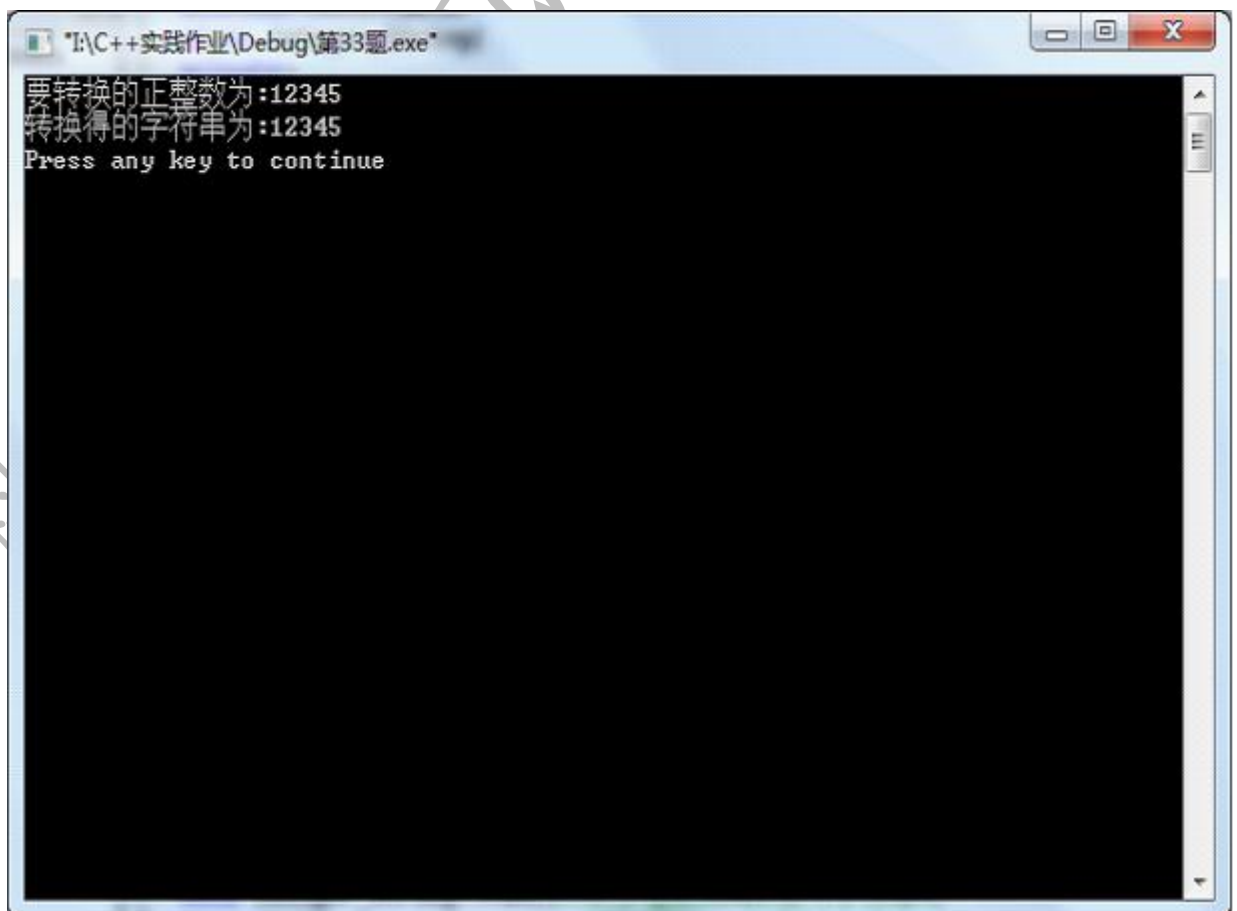
```
#include<iostream.h>
class Integer_String{
private:
    int num;//要转换的正整数;
    char*s;//用动态空间存储转换得到的字符串;
public:
    Integer_String(int n)
    {
        num=n;
    }
    int f();
    void fun();
    void show();
    ~Integer_String()
    {if(s)delete[]s;}
};
int Integer_String::f()//求数据成员 num 的位数;
{
    int n=num;
    int count=0;
    while(n)
    {
        count++;
        n/=10;
    }
    return count;
}
void Integer_String::fun()//把正整数 num 转换为字符串 s;
{
    int n=num;
    s=new char[f()+1];
    for(int i=0;i<f();i++)
```

```
{
    s[f()-1-i]=n%10+48;
    n/=10;
}
s[f()]=0;
}
void Integer_String::show()
{
    cout<<"要转换的正整数为:"<<num<<endl;
    cout<<"转换得的字符串为:"<<s<<endl;
}
void main()
{
    Integer_String test(12345);
    test.fun();
    test.show();
}
```

#### 四、实践小结

掌握字符的 ASCII 码值与整型值之间的转化。

#### 五、运行结果



## 任务三十三

### 一、实践任务

34. 建立一个类 `String_Integer`，把一个字符串中的数字字符转换为正整数。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `char *s`: 用动态空间存放字符串。

##### (2) 公有成员函数

- `String_Integer(char *str)`: 用参数 `str` 初始化数据成员 `s`。
- `operator int()`: 转换函数，数据成员 `s` 转换整数并返回该数。
- `void show()`: 输出数据成员 `s`。
- `~String_Integer()`: 释放动态空间。

#### 2、主要函数设计

在主函数中对定义的类进行测试。定义字符数组，把由键盘输入的字符串“ab123c00d45ef”存入数组，并用该数组初始化类 `String_Integer` 的对象 `test`，调用 `show` 函数输出 `test` 的数据成员 `s`，然后把对象 `test` 赋值给整型变量 `n` 并输出，转换结果如下所示（下划线部分是从键盘输入的内容）：

请输入字符串 ab12 3c00d45ef : ab12 3c00d45ef

字符串为: ab12 3c00d45ef

转换得到的整数为: 1230045

### 三、源程序清单

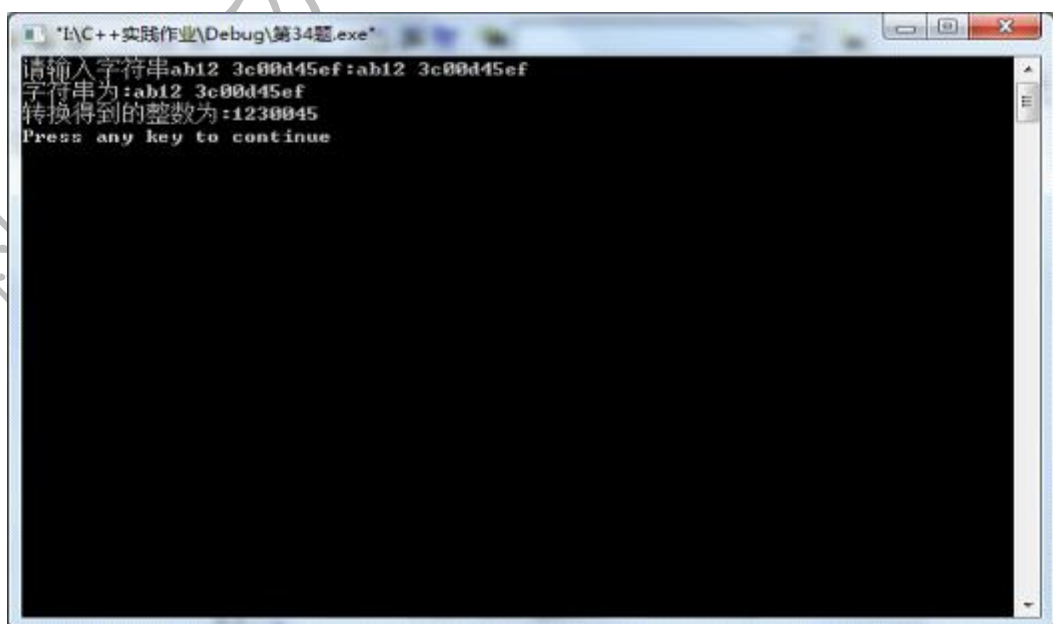
```
#include<iostream.h>
#include<string.h>
class String_Integer{
private:
char*s;
public:
String_Integer(char*str)
{
s=new char[strlen(str)+1];
strcpy(s,str);
}
operator int()//转换函数，数据成员 s 转换整数并返回该数;
{
int num=0;
for(int i=0;i<strlen(s);i++)
{
if(s[i]>='0' && s[i]<='9')//判断该字符是否为数字字符;
{
```

```
        num=num*10+s[i]-48;
    }
}
return num;
}
void show()
{
    cout<<"字符串为:"<<s<<endl;
}
~String_Integer()
{if(s)delete s;}
};
void main()
{
    char s[30];
    cout<<"请输入字符串 ab12 3c00d45ef:";
    cin.getline(s,30);
    String_Integer test(s);
    test.show();
    int n;
    n=test;
    cout<<"转换得到的整数为:"<<n<<endl;
}
```

#### 四、实践小结

掌握字符的 ASCII 码值与整型值之间的转化。

#### 五、运行结果



## 任务三十四

### 一、实践任务

35. 建立一个类 Union 求两个整数集合的并集。

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int *set1, len1`: 用动态空间 `set1` 存储集合 1, `len1` 表示其元素的个数。
- `int *set2, len2`: 用动态空间 `set2` 存储集合 2, `len2` 表示其元素的个数。
- `int set[20], len`: 用数组空间 `set` 存储并集, `len` 表示其元素的个数

##### (2) 公有成员函数

- `Union(int *s1, int l1, int *s2, int l2)`: 用变量 `s1` 和 `l1` 初始化集合 1 及其长度, 用变量 `s2` 和 `l2` 初始化集合 2 及其长度, 并把并集的长度置为 0;
- `int f(int num)`: 判断整数 `num` 是否属于集合 1, 是返回 1, 否则返回 0;
- `void fun()`: 求集合 1 和集合 2 的并集, 方法是先把集合 1 中的所有元素复制给并集, 然后调用 `f` 函数把集合 2 中不属于集合 1 的元素复制给并集;
- `void show()`: 输出集合 1、集合 2 和并集;
- `~Union()`: 释放动态空间。

#### 2、主要函数设计

在主函数中对定义的类进行测试。定义数组 `s1: {1,2,3,4,5,6,7,8}`、`s2: {1,3,5,7,9,11}`, 并用它们初始化类 Union 的对象 `obj`, 然后调用相关的成员函数, 求并集, 输出集合 1、集合 2 和并集。

### 三、源程序清单

```
#include<iostream.h>
class Union{
private:
    int*set1,len1;
    int*set2,len2;
    int set[20],len;
public:
    Union(int*s1,int l1,int*s2,int l2)
    {
        set1=new int[l1];
        int i;
        for(i=0;i<l1;i++)
            set1[i]=s1[i];
        len1=l1;
        set2=new int[l2];
        for(i=0;i<l2;i++)
            set2[i]=s2[i];
        len2=l2;
```

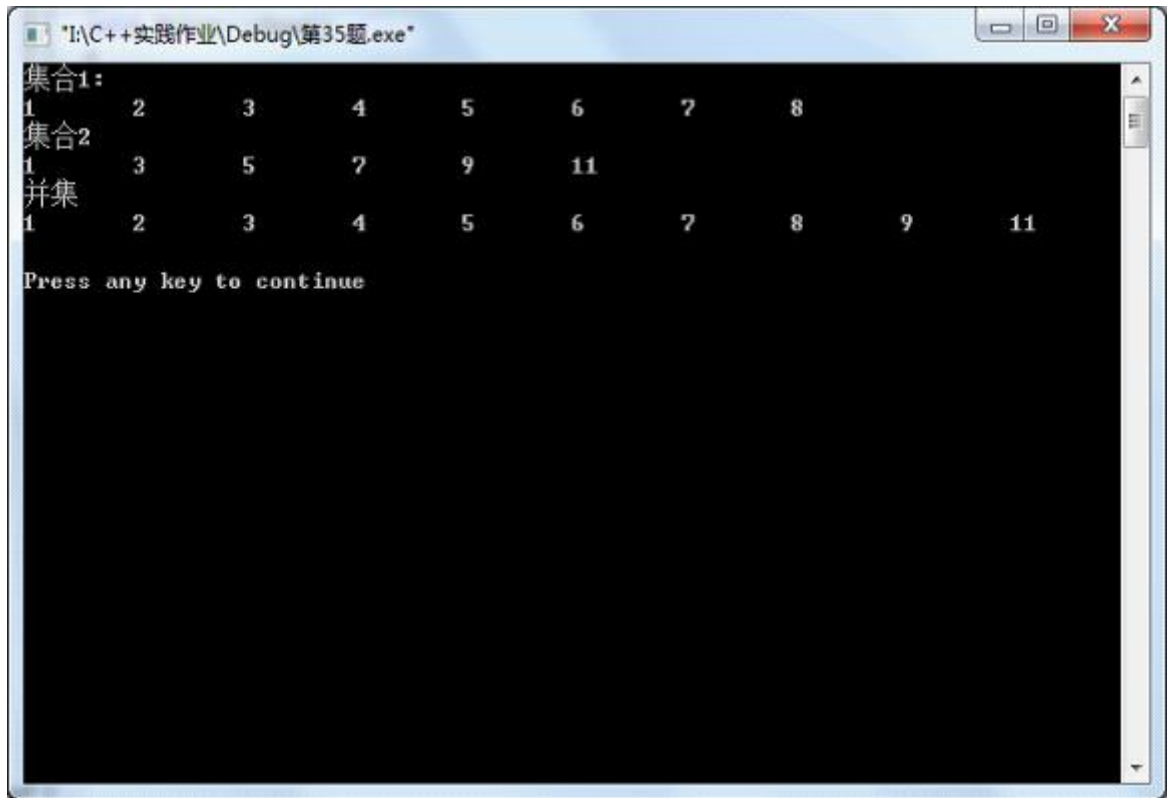
```
len=0;
}
int f(int num)//判断整数 num 是否属于集合 1，是返回 1，否则返回 0;
{
    for(int i=0;i<len1;i++)
    {
        if(set1[i]==num) return 1;
    }
    return 0;
}
void fun();
void show();
};
void Union::fun()//求集合 1 和集合 2 的并集;
{
    for(int i=0;i<len1;i++)
        set[len++]=set1[i]; //先把集合 1 中的所有元素复制给并集;
    for(i=0;i<len2;i++)
    {
        if(f(set2[i])==0)//然后调用 f 函数把集合 2 中不属于集合 1 的元素复制给并集;
            set[len++]=set2[i];
    }
}
void Union::show()
{
    cout<<"集合 1:"<<endl;
    for(int i=0;i<len1;i++)
        cout<<set1[i]<<"\t";
    cout<<endl;
    cout<<"集合 2:"<<endl;
    for(i=0;i<len2;i++)
        cout<<set2[i]<<"\t";
    cout<<endl;
    cout<<"并集"<<endl;
    for(i=0;i<len;i++)
        cout<<set[i]<<"\t";
    cout<<endl;
}
void main()
{
    int s1[]={1,2,3,4,5,6,7,8},s2[]={1,3,5,7,9,11};
    Union obj(s1,8,s2,6);
    obj.fun();
    obj.show();
}
```

```
}
```

#### 四、实践小结

应理解并集含义，并利用循环判断语句实现。

#### 五、运行结果



The screenshot shows a Windows command prompt window titled "I:\C++实践作业\Debug\第35题.exe". The program displays the following output:

```
集合1:
1      2      3      4      5      6      7      8
集合2:
1      3      5      7      9      11
并集
1      2      3      4      5      6      7      8      9      11
Press any key to continue
```

### 任务三十五

#### 一、实践任务

36. 建立一个类 Intersection 求两个整数集合的交集。

#### 二、详细设计

##### 1. 类的描述与定义

##### (1) 私有数据成员

- int set[20]: 用数组空间 set 存储集合。
- int len: 表示该集合中元素的个数

##### (2) 公有成员函数

- Intersection(int \*s,int l): 用 s 初始化集合，用变量 l 初始化其长度。
- Intersection(): 把 set 中各元素和长度初始化为 0。
- int f(int num): 判断整数 num 是否属于集合，是返回 1，否则返回 0；
- Intersection operator&&(Intersection t): 重载&&，求当前对象的集合和参数对象 t 的集合的交集，方法是用对象 t 的集合中的每个元素作为参数调用 f 函数，若该元素属于当前对象的集合，则把它复制给交集。

- void show(): 输出集合。

## 2、主要函数设计

在主函数中对定义的类进行测试。定义数组 s1: {1,3,4,5,7,8}、s2: {1,2,3,5,7,9,11}，并用它们初始化类 Intersection 的对象 obj1 和 obj2，然后调用相关的成员函数输出集合；定义对象 obj3，并用 obj1 和 obj2 的与运算符结果（交集）初始化该对象，并输出交集。

## 三、源程序清单

```
#include<iostream.h>
class Intersection{
private:
int set[20];
int len;
public:
Intersection(int*s,int l)
{
    for(int i=0;i<l;i++)
        set[i]=s[i];
    len=l;
}
Intersection()
{
    for(int i=0;i<20;i++)
        set[i]=0;
    len=0;
}
int f(int num)
{
    for(int i=0;i<len;i++)
    {
        if(num==set[i])return 1;
    }
    return 0;
}
Intersection operator&&(Intersection t)
{
    Intersection turn;
    int i;
    for(i=0;i<t.len;i++)
    {
        if(f(t.set[i]))turn.set[turn.len++]=t.set[i];
    }
    return turn;
}
void show()
```



```
{
    for(int i=0;i<len;i++)
        cout<<set[i]<<'\\t';
    cout<<endl;
}
};
void main()
{
    int s1[]={1,3,4,5,7,8},s2[]={1,2,3,5,7,9,11};
    Intersection obj1(s1,6),obj2(s2,7);
    cout<<"集合 1:"<<endl;
    obj1.show();
    cout<<"集合 2:"<<endl;
    obj2.show();
    Intersection obj3;
    obj3=obj1&&obj2;
    cout<<"交集:"<<endl;
    obj3.show();
}
```

#### 四、实践小结

理解交集含义，并利用循环判断语句实现，并需掌握运算符重载的方式。

#### 五、运行结果



## 任务三十六

### 一、实践任务

38. 建立一个类 `Sample`，对数组中元素用选择法进行升序排序。排序函数定义到 `Sample` 类的友元类 `Process` 中。

具体要求如下：

类 `Sample`

```
#define Max 100;
```

### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- `int A [MAX]`: 一维整型数组，存放需要排序的数。
- `int n`: 需要排序的数的个数。

##### (2) 公有成员函数

- `Sample ()`: 构造函数，初始化成员数据 `n`，初始值为 0。

友元类 `Process`

公有成员函数

- `void getdata(Sample &s)`: 从键盘输入数据，对数组 `A` 进行赋值。

- void selectsort(Sample &s): 对数组 A 中的元素进行升序排序。
- void disp(Sample &s): 输出数组中的元素。

## 2、主要函数设计

在主程序中定义对象对该类进行测试。

## 三、源程序清单

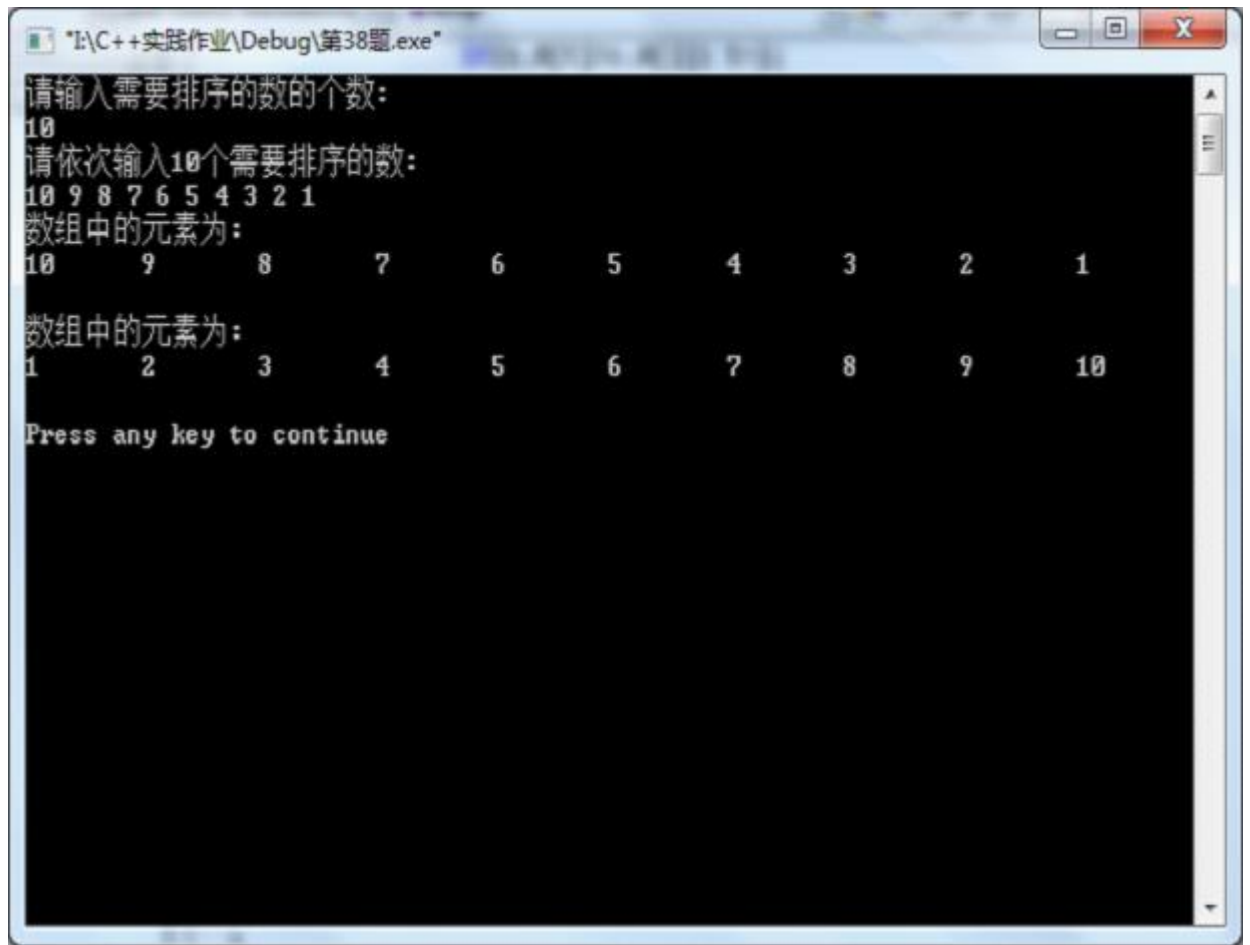
```
#include<iostream.h>
#define MAX 100
class Sample{
private:
    int A[MAX];
    int n;
public:
    Sample()
    {n=0;}
    friend class Process;//友元类的声明;
};
class Process{
public:
    void getdata(Sample &s)//从键盘输入数据，对数组 A 进行赋值;
    {
        cout<<"请输入需要排序的数的个数:"<<endl;
        cin>>s.n;
        cout<<"请依次输入"<<s.n<<"个需要排序的数:"<<endl;
        for(int i=0;i<s.n;i++)
            cin>>s.A[i];
    }
    void selectsort(Sample &s)//对数组 A 中的元素进行升序排序;
    {
        int t;
        for(int i=0;i<s.n-1;i++)
        {
            t=i;
            for(int j=i+1;j<s.n;j++)
            {
                if(s.A[t]>s.A[j]) t=j;
            }
            if(t!=i)
            {
                j=s.A[t];
                s.A[t]=s.A[i];
                s.A[i]=j;
            }
        }
    }
}
```

```
}  
void disp(Sample &s)//输出数组中的元素;  
{  
    cout<<"数组中的元素为:"<<endl;  
    for(int i=0;i<s.n;i++)  
        cout<<s.A[i]<<"\t";  
    cout<<endl;  
}  
};  
void main()  
{  
    Sample test1;  
    Process test2;  
    test2.getdata(test1);  
    test2.disp(test1);  
    test2.selectsort(test1);  
    test2.disp(test1);  
}
```

#### 四、实践小结

掌握宏定义，及数组中元素排序的方式。

#### 五、运行结果



The screenshot shows a Windows application window titled "I:\C++实践作业\Debug\第38题.exe". The program prompts the user to enter the number of elements to sort, which is 10. It then prompts for the elements, which are 10, 9, 8, 7, 6, 5, 4, 3, 2, 1. The program displays the array in descending order and then in ascending order after sorting. The window has standard Windows controls (minimize, maximize, close) in the top right corner.

```
"I:\C++实践作业\Debug\第38题.exe"
请输入需要排序的数的个数:
10
请依次输入10个需要排序的数:
10 9 8 7 6 5 4 3 2 1
数组中的元素为:
10      9      8      7      6      5      4      3      2      1
数组中的元素为:
1      2      3      4      5      6      7      8      9      10
Press any key to continue
```

淘宝店铺：江苏科技大学考研辅导

江科大官方考研QQ: 2962140400