

# 江苏科技大学

## 课程实践报告

### 任务一

#### 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

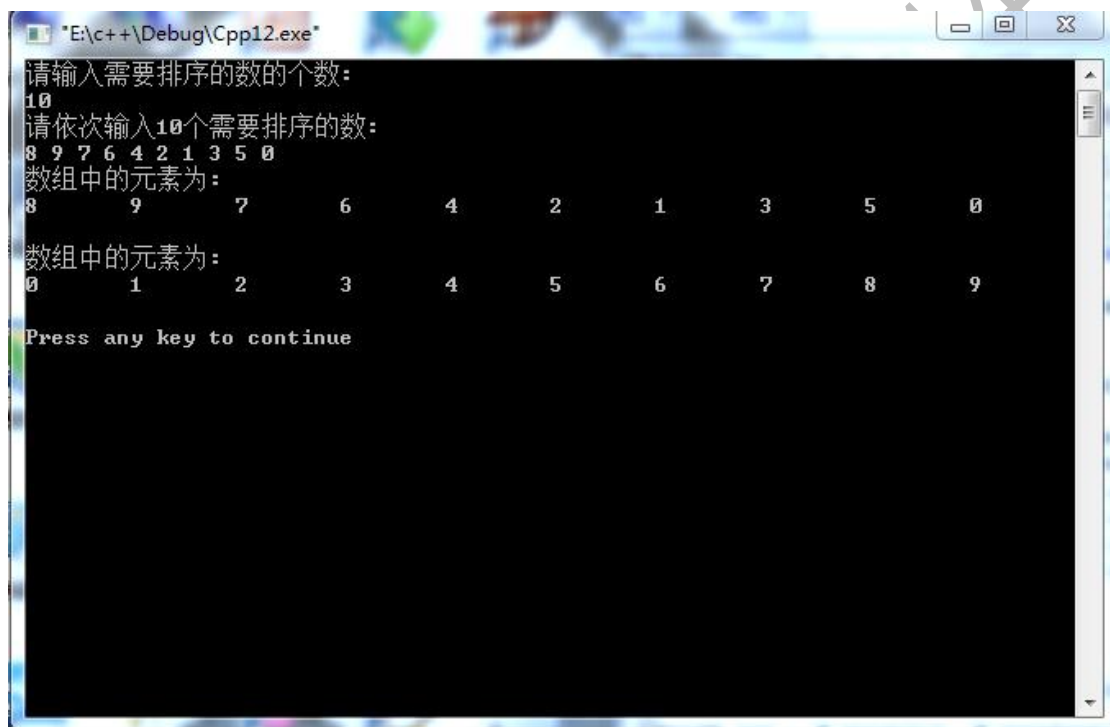
```
#include<iostream.h>
```

```
#define MAX 100
```

```
class Sample{
private:
    int A[MAX];
    int n;
public:
    Sample()
    {n=0;}
    friend class Process;
};
class Process{
public:
    void getdata(Sample &s)
    {
        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)
    {
        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)
    {
        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、运行结果



## 任务二

### 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

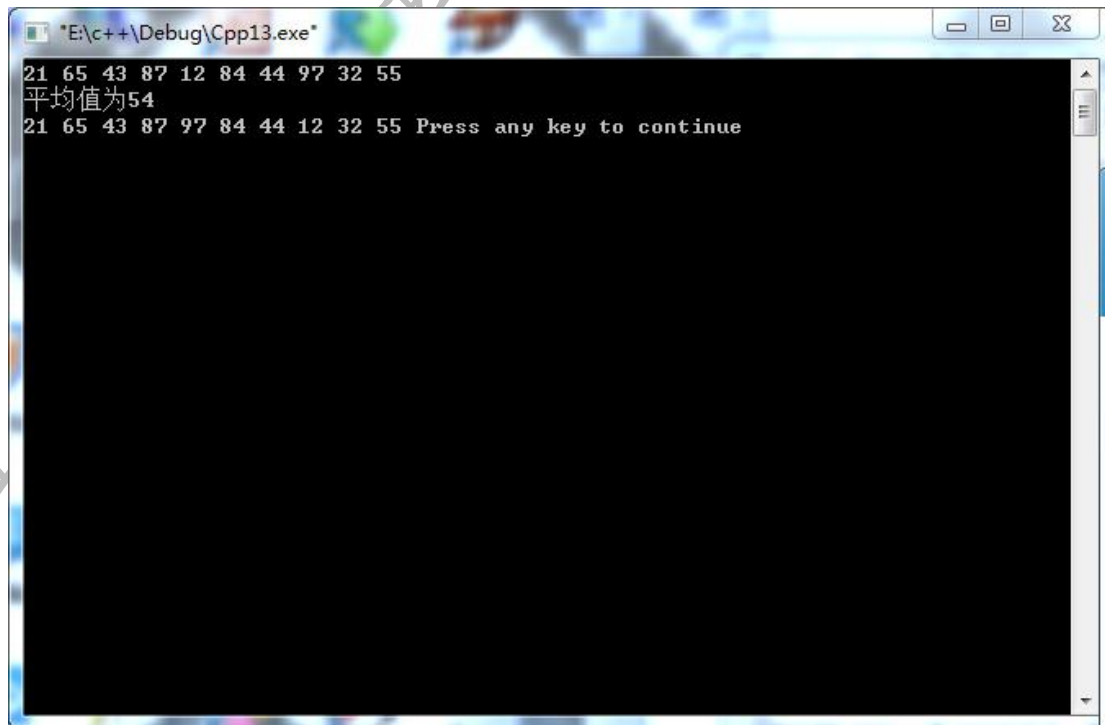
在主程序中用数据 {21,65,43,87,12,84,44,97,32,55} 对该类进行测试。

## 三、源程序清单

```
#include<iostream.h>
class MOVE{
private:
    int *array;
    int n;
public:
    MOVE(int b[],int m)
    {
        array=new int[m];
        n=m;
        for(int i=0;i<m;i++)
        {
            array[i]=b[i];
        }
    }
    void exchange();
    void print();
    ~MOVE()
    {
        if(array) delete array;
    }
};
void MOVE::exchange()
{
    float ave=0;
    for(int i=0;i<n;i++)
    {
        ave+=array[i];
    }
    ave/=n;
    cout<<"平均值为"<<ave<<endl;
    int max,min;
    max=min=array[0];
    int m;
    for(i=0,m=0;i<n;i++)
    {
        if(array[i]>=max) m=i,max=array[i];
    }
}
```

```
int l;  
for(i=0,l=0;i<n;i++)  
{  
    if(array[i]<max) l=i,max=array[i];  
}  
i=array[m],array[m]=array[l],array[l]=i;  
}  
void MOVE::print()  
{  
    for(int i=0;i<n;i++)  
        cout<<array[i]<<' '  
}  
void main()  
{  
    int text[]={21,65,43,87,12,84,44,97,32,55};  
    int num;  
    num=sizeof(text)/sizeof(int);  
    MOVE s(text,num);  
    s.print();  
    cout<<endl;  
    s.exchange();  
    s.print();  
}
```

#### 四、运行结果



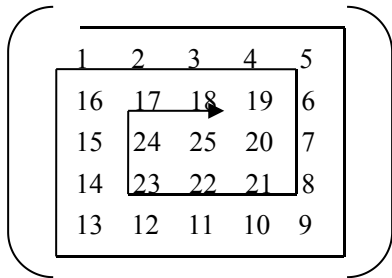
The screenshot shows a Windows command prompt window titled "E:\c++\Debug\Cpp13.exe". The output of the program is as follows:

```
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
```

## 任务三

### 一、实践任务

8. 建立一个 MATRIX，生成并显示一个螺旋方阵。螺旋方阵如下图所示，起始数置于方阵的左上角，然后从起始数开始依次递增，按顺时针方向从外向里旋转填数而成。



### 二、详细设计

#### 1、类的描述与定义

##### (1) 私有数据成员

- int a[20][20]: 二维整型数组存放螺旋方阵。
- int startnum: 螺旋方阵的起始数。
- int n: 存放方阵的层数。

##### (2) 公有成员函数

- MATRIX (int s, int m ): 构造函数，初始化成员数据 startnum 和 n。
- void process(): 生成起始数为 startnum 的 n 行螺旋方阵。
- void print(): 输出螺旋方阵。

#### 2、主要函数设计

在主程序中定义 MATRIX 类的对象 t 对该类进行测试。

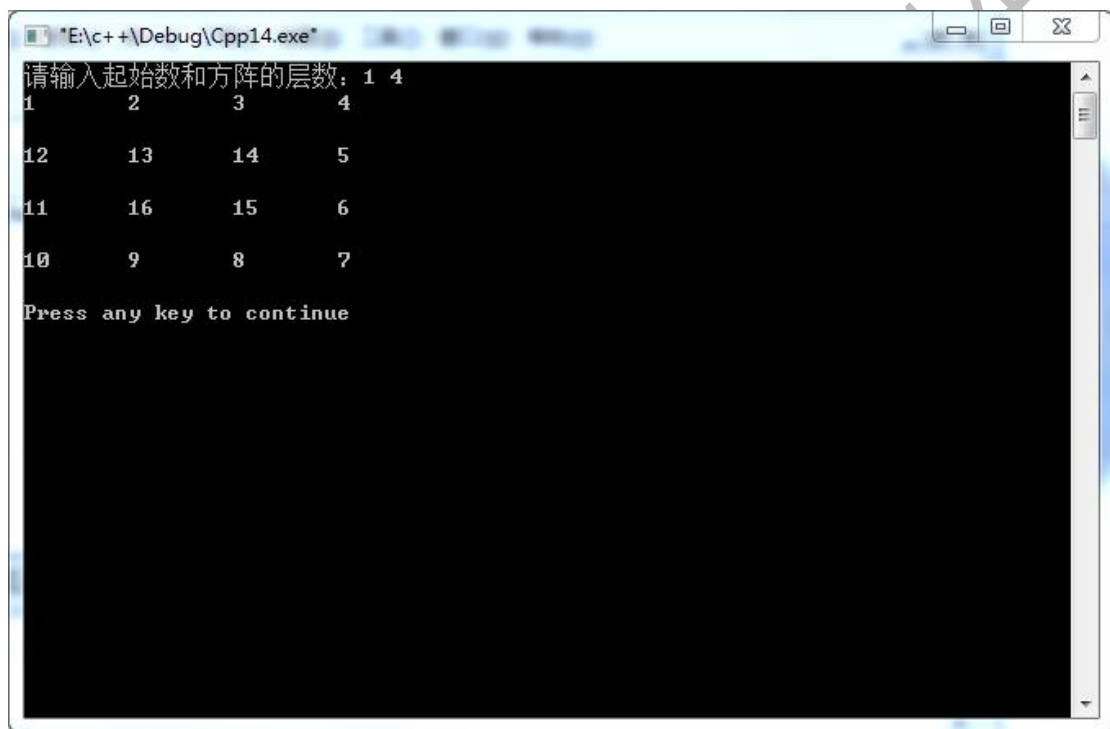
### 三、源程序清单

```
#include<iostream.h>
class MATRIX{
private:
    int a[20][20];
    int startnum;
    int n;
public:
    MATRIX(int s,int m)
    {
        startnum=s;
        n=m;
    }
    void process();
    void print();
};
```

```
};  
void MATRIX::process()  
{  
    int i,j;  
    int k;  
    int turn=startnum;  
    for(k=1;k<=n/2;k++)  
    {  
        for(i=k-1,j=k-1;j<n-(k-1);j++)  
        {  
            a[i][j]=turn++;  
        }  
        j--;  
        turn--;  
        for(;i<n-k+1;i++)  
        {  
            a[i][j]=turn++;  
        }  
        i--;  
        turn--;  
        for(;j>=k-1;j--)  
        {  
            a[i][j]=turn++;  
        }  
        j++;  
        turn--;  
        for(;i>=k;i--)  
        {  
            a[i][j]=turn++;  
        }  
    }  
    a[n/2][(n-1)/2]=n*n-1+startnum;  
}  
void MATRIX::print()  
{  
    int i,j;  
    for(i=0;i<n;i++)  
    {  
        for(j=0;j<n;j++)  
        {  
            cout<<a[i][j]<<"\t";  
        }  
        cout<<endl<<endl;  
    }  
}
```

```
}  
void main()//测试;  
{  
    int s,m;  
    cout<<"请输入起始数和方阵的层数: ";  
    cin>>s>>m;  
    MATRIX t(s,m);  
    t.process();  
    t.print();  
}
```

#### 四、运行结果



#### 任务四

##### 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

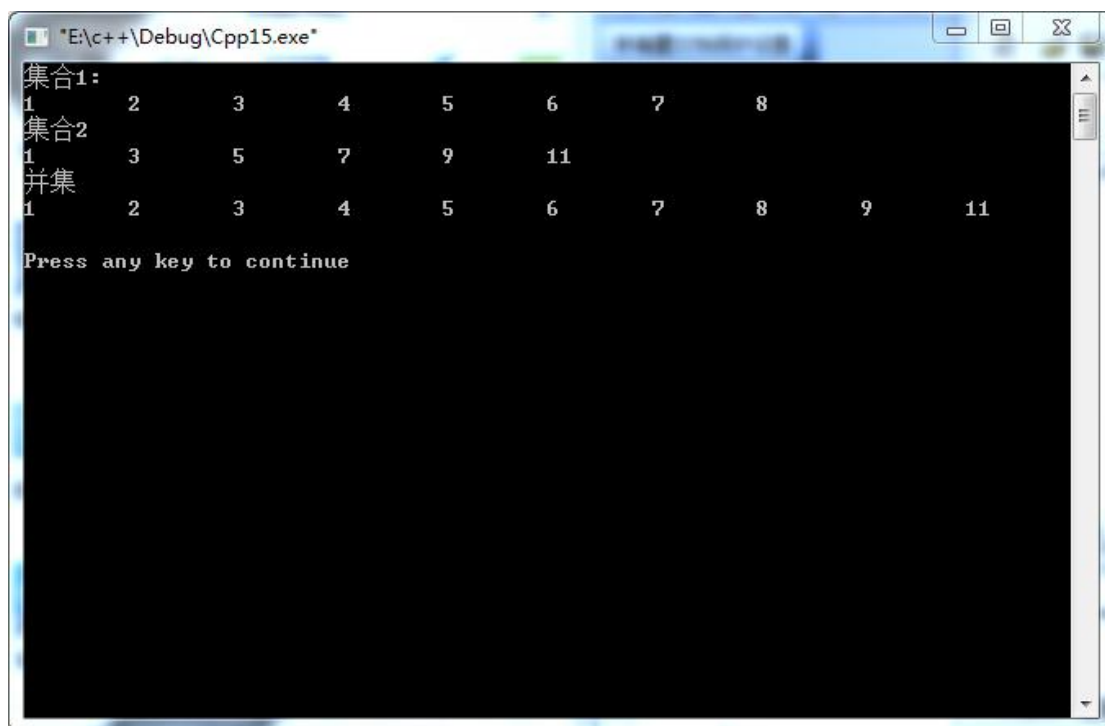
在主函数中对定义的类进行测试。定义数组 s1: {1,2,3,4,5,6,7,8}、s2: {1,3,5,7,9,11}，并用它们初始化类 Union 的对象 obj，然后调用相关的成员函数，求并集，输出集合 1、集合 2 和并集。

## 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)
{
    for(int i=0;i<len1;i++)
    {
        if(set1[i]==num) return 1;
    }
    return 0;
}
void fun();
void show();
};
void Union::fun()
```

```
{
for(int i=0;i<len1;i++)
    set[len++]=set1[i];
for(i=0;i<len2;i++)
{
    if(f(set2[i])==0)
        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、运行结果



## 任务五

### 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];
int num;
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()
{
    int i,j,k;
    int m=0,n=0;
    for(i=0;i<4;i++)
    {
        for(j=0;j<4;j++)
        {
            m=n=0;
            for(k=0;k<4;k++)
            {
                if(a[i][j]>=a[i][k]) m++;
                else break;
                if(a[i][j]<=a[k][j]) n++;
                else break;
            }
            if(m==4&& n==4)
            {
                num++;
                b[i][j]=a[i][j];
            }
        }
    }
}
```

```
void Saddle_point::print()
{
    int i,j;
    cout<<"数组 a 为:"<<endl;
    for(i=0;i<4;i++)
    {
        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、运行结果

```

E:\c++\Debug\Cpp17.exe
数组a为:
2      6      3      4
5      6      5      5
5      7      6      7
1      9      2      7
按点个数:2
按点坐标:[0][1]相应值:6
按点坐标:[1][1]相应值:6
Press any key to continue
    
```

## 任务六

### 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， $f^T$  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

#### 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， $f^T$  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

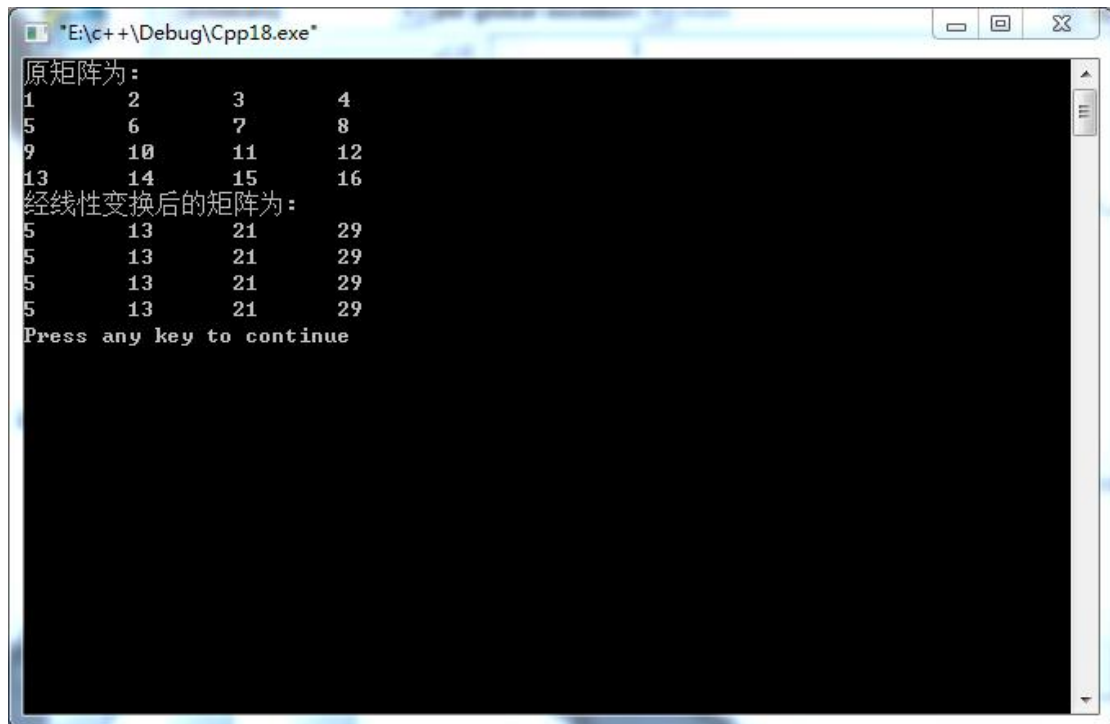
```
#include<iostream.h>
class CMatrix{
private:
    int(*a)[4];
    int w[4][4],m;
public:
    CMatrix(int a[][4],int m)
    {
        int i,j;
        this->a=new int[m][4];
        this->m=m;
        for(i=0;i<4;i++)
```

```
{
    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()
{
    int i,j,k;
    for(i=0;i<m;i++)
    {
        for(j=i;j<4;j++)
        {
            k=a[i][j],a[i][j]=a[j][i],a[j][i]=k;
        }
    }
    int sum,turn[4][4];
    for(i=0;i<4;i++)
    {
        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++)
{
    for(j=0;j<4;j++)
    {
        a[i][j]=turn[i][j];
    }
}
}
void CMatrix::show()
{
    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、运行结果



```
"E:\c++\Debug\Cpp18.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
```

## 任务七

### 一、实践任务

1. 以下程序中函数 `fun(int a[], int N)` 的功能是删除数组中 `a` 的前 `N` 个元素中重复的元素，并返回所删除元素的总数。请改正其中的错误。

程序正确的运行结果为

处理前的数组为：4 1 3 3 1 2 4 3 4 4

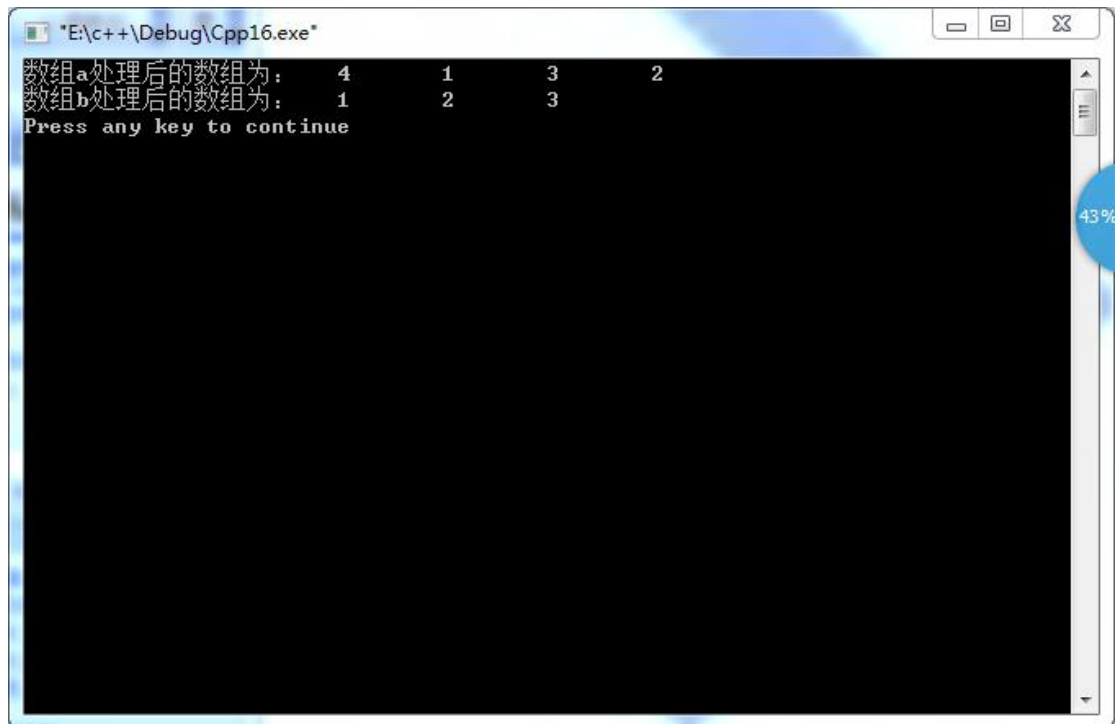
处理后的数组为：4 1 3 2

### 二、改正后正确的源程序如下：

```
#include <iostream>
using namespace std;
int fun(int *a, int N)
{
    int n=0, m;
    for(int i=0; i<N-n-1; i++)
    {
        for(int j=i+1; j<N-n; j++){
            if(a[j]==a[i]){
                m=1;
                for(int k=j+1; k<N-n; k++){
                    if(a[k]==a[i]){
                        m++;
                    }
                }
                else{
```

```
        a[j]=a[j+m];
        j++;
    }
}
n+=m;
}
}
return N-n;
}
void print(int a[],int n)
{
    for(int i=0;i<n;i++)
        cout<<'t'<<a[i];
    cout<<endl;
}
int main()
{
    int a[10]={4,1,3,3,1,2,4,3,4,4},b[6]={1,2,1,3,2,1};
    int n=fun(a,10);
    cout<<"数组 a 处理后的数组为： ";
    print(a,n);
    n=fun(b,6);
    cout<<"数组 b 处理后的数组为： ";
    print(b,n);
    return 0;
}
```

### 三、运行结果



## 任务八

### 一、实践任务

7. 以下程序统计一个字符串中包含某个字符的单词所出现的次数。这里假设单词之间由一个或多个空格分隔，且在判断字符是否相等时不区分大小写字母。请改正其中的错误。

程序正确的运行结果为

字符串 "I am a student..My name is Tony.I am twenty." 中包含字符 'T' 的单词有 4 个

### 二、改正后正确的源程序如下：

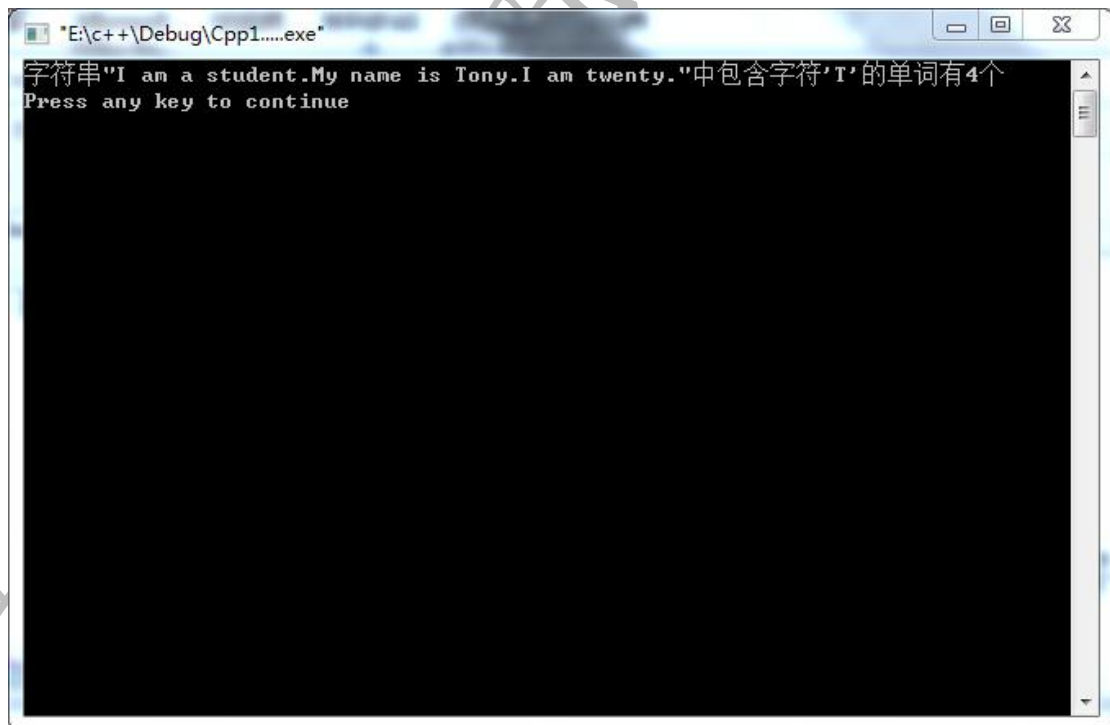
```
#include<iostream.h>
char convert(char c)
{
    if(c>='A'&&c<='Z')
        return c+'a'-'A';
    return c;
}

int search(char *str,char ch)
{
    int count=0;
    char *str1;
    for(;*str;str++)
    {
        for(str1=str;*str1!=' '&&*str1;str1++)
        {
```

```
        if(convert(*str1)==convert(ch))
        {
            count++;
            break;
        }
        while(*(str++)!=' ');
        str--;
    }
    return count;
}

void main()
{
    char str[50]="I am a student.My name is Tony.I am twenty.",ch='T';
    cout<<"字符串\"<str<<\"中包含字符\"<ch<<\"的单词有\"<search(str,ch)<<\"
    个\"<<endl;
}
```

### 三、运行结果



### C++程序设计实践课程心得

通过这次实验我从中学到了不少知识，并且熟练了在电脑编写程序。

我感受最深的一点是：在编写一个程序之前，要明确所需的数据结构，存储结构，简洁的

## 淘宝店铺：江苏科技大学考研辅导

算法。了解了典型数据结构的性质。选取合适的参数形式有益于实现函数之间的数据传输。面对对象设计时，要使程序效率高，可读性高，定义合理的类，在类中合理的安排数据和对数据的处理。

这次实验过程中，也有一些程序出现错误，但在和同学探讨中能解决问题，让我明白这门课程需要探索和互助。在自己薄弱的地方可以让他人指导，比如递归调用，静态联编，二维指针等方面都有了明显的提高。

江科大官方考研QQ: 2962140400