5．定义一个类Palindrome，实现绝对回文数。设计一个算法实现对任意整型数字判断是否为绝对回文数。所谓绝对回文数，是指十进制数和二进制数均对称的数。

具体要求如下：

（1）私有数据成员

 int n：整型数字。

 int y： 标记是否为回文数。

（2）公有成员函数

 Palindrome (int x) ：构造函数，根据x参数初始化数据成员n，y初始化为0。

 void huiwen () ：判断数n是否为绝对回文数。

 void show( ) ：若该数为回文数，则在屏幕显示。

（3）在主程序中定义int a，由键盘输入数字。定义一个Palindrome类对象p，用a初始化p，完成对该类的测试。

1. 源代码：

#include<iostream>

using namespace std;

class palindrome{

int n;

int y;

public:

palindrome(int x);

void huiwen();

void show();

};

palindrome::palindrome(int x)

{

n=x;

y=0;

}

void palindrome::huiwen()

{

int b[20],c[50],m,i,p=0,t1=1,t2=1;

m=n;

for(i=0;m>0;m/=10)

{

p++;

b[i]=m%10;

i++;

}

for(i=0;i<p;i++)

if(b[i]!=b[p-i-1])

{

t1=0;

break;

}

for(i=0,m=n,p=0;m>0;m/=2)

{

p++;

c[i]=m%2;

i++;

}

for(i=0;i<p;i++)

if(c[i]!=c[p-i-1])

{

t2=0;

break;

}

if(t1&&t2)y=1;

}

void palindrome::show()

{

if(y==0)cout<<"该数不是回文数！"<<endl;

else cout<<"该回文数是："<<n<<endl;

}

int main()

{

int a;

cout<<"输入a的值"<<endl;

cin>>a;

palindrome p(a);

p.huiwen();

p.show();

system("pause");

return 0;

}

1. 测试结果：



10．定义一个字符串类CString，并设计一个算法实现，给定关键字str1在字符串str中出现时用关键字str2进行替换的功能。

具体要求如下：

（1）私有数据成员

 char \*str; 原始字符串。

 char \*str1; 目标关键字。

 char \*str2; 替换关键字。

 int flag; 标记替换是否完成替换。

（2）公有成员函数

 CString (char \*s,char s1[ ],char \*s2) ：用给定的参数s、s1和s2相对应的初始化数据成员str、str1和str2。flag设置缺省0。

 void Replace () ：判断str字符串中是否出现str1，若出现就用str2替换，否则什么都不做。若替换成功了标记flag为1，若替换不成功则标记flag为0。

 void Show( ) ：若替换成功，则在屏幕上显示目标关键字、替换关键字和替换后的原始字符串；若不成功则显示原始字符串。

 ~ CString() ：释放动态分配的空间。

1. 在主程序中定义字符串char s[]=”I am student, you are student too, we are all student.”作为原始字符串，定义char s1[]=” student”作为目标关键字，定义char s2[]=”teacher”作为替换关键字。定义一个CString类对象test，用s，s1和s2初始化test，完成对该类的测试。
2. 源代码：

#include<iostream>

#include<string>

using namespace std;

class cstring{

char \*str;

char \*str1;

char \*str2;

int flag;

public:

cstring(char\*s,char s1[],char \*s2);

void replace();

void show();

~cstring();

};

cstring::cstring(char\*s,char s1[],char \*s2)

{

str=new char[strlen(s)+1];

str1=new char[strlen(s1)+1];

str2=new char[strlen(s2)+1];

strcpy(str,s);

strcpy(str1,s1);

strcpy(str2,s2);

flag=0;

}

void cstring::replace()

{

int i,n1,n2,y=1;

for(i=0;str[i];i++)

{

if(str[i]==str1[0])

{

for(n1=i,n2=0;str1[n2];n1++,n2++)

if((str[n1]!=str1[n2])||(str[n1]=='\0'))

{

y=0;

break;

}

if(y==1)

{

char pp[100];

for(int x=0,p=i;x<(strlen(str1));p++,x++)

{

str[p]=str2[x];

}

flag=1;

}

}

}

}

void cstring::show()

{

if(flag==1) cout<<"改后"<<str<<endl;

else cout<<"未改"<<str<<endl;

}

cstring::~cstring()

{

delete []str;

delete []str1;

delete []str2;

}

int main()

{

char s[]="I am student,you are student too,we are all student.";

char s1[]="student";

char s2[]="teacher";

cstring test(s,s1,s2);

test.replace();

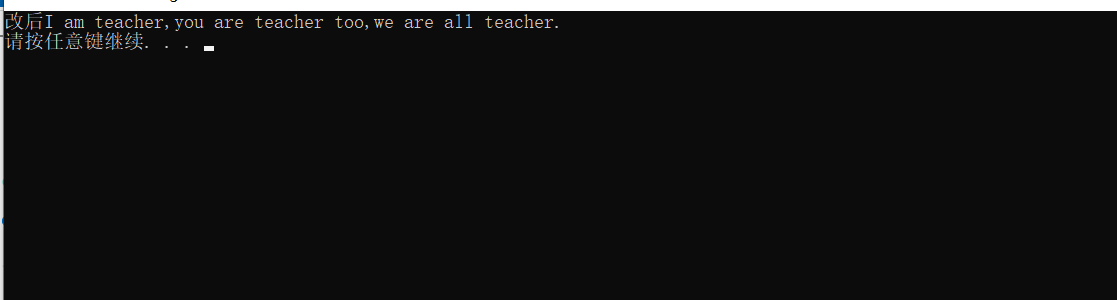
test.show();

system("pause");

return 0;

}

1. 测试结果：



15．定义一个一维数组类Carray，并根据给定算法实现对原始一维数组进行线性变换。这里给定的线性变换算法为：T(bx) = bT(x) + i; 其中，b为变换常量，x为变量，i为当前类中成员数组的下标值。根据该算法，原始数组在变化后，当前数组元素的值是由常量b和i下标来决定的。

具体要求如下：

（1）私有数据成员

 int \*a：指针a指向一个动态分配的原始数组。

 int n：n表示该数组的大小。

 int b：线性变换的常量。

（2）公有成员函数

 Carray (int a[ ],int n,int x) ：用给定的参数a、n和x初始化数据成员a、n和b。缺省都设置为0。

 void Transform () ：根据上述变化算法，求解数组变换。

 void Show( ) ：在屏幕上显示数组元素。

 ~ Carray () ：释放动态分配的空间。

（3）在主程序中定义数组int arr[ ]={1,2,3,4,5,6,7,8,9,10}作为原始数组，int b;由键盘输入，作为线性变换的常量。定义一个Carray类对象test，用arr初始化test，完成对该类的测试。

1. 源代码：

#include<iostream>

using namespace std;

class carray{

int \*a;

int n;

int b;

public:

carray(int a[],int n,int x);

void transform();

void show();

~carray();

};

carray::carray(int a[],int n=0,int x=0)

{

this->a=new int[n];

this->n=n;

b=x;

for(int i=0;i<n;i++)

{

this->a[i]=a[i];

}

}

void carray::transform()

{

for(int i=0;i<n;i++)

a[i]=b\*a[i]+i;

}

void carray::show()

{

cout<<"变换后的数组为："<<endl;

for(int i=0;i<n;i++)

cout<<a[i]<<'\t';

cout<<endl;

}

carray::~carray()

{

delete []a;

}

int main()

{

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

int m=10,b;

cout<<"输入线性变换常量"<<endl;

cin>>b;

carray test(arr,m,b);

test.transform();

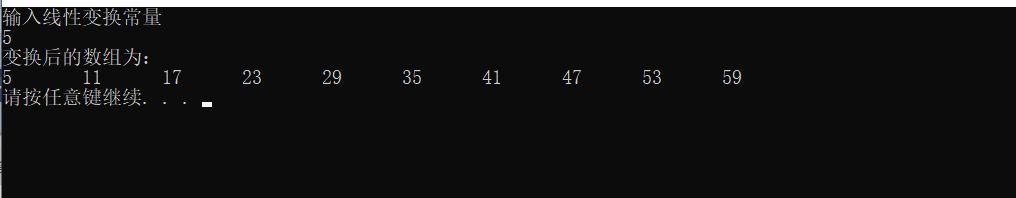
test.show();

system("pause");

return 0;

}

1. 测试结果



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。

1 2 3 4 4 8 12 16

5 6 7 8 3 7 11 15

9 10 11 12 2 6 10 14

13 14 15 16 1 5 9 13

具体要求如下：

（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。

1. 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。
2. 源代码：

#include<iostream>

#include<iomanip>

using namespace std;

class array{

int a[4][4];

public:

array(int a1[][4],int n);

void xuanzhuan();

void show();

};

array::array(int a1[][4],int n)

{

for(int i=0;i<n;i++)

for(int j=0;j<n;j++)

a[i][j]=a1[i][j];

}

void array::xuanzhuan()

{

int b[4][4],i,j;

for(i=0;i<4;i++)

for(j=0;j<4;j++)

b[i][j]=a[i][j];

for(i=0;i<4;i++)

for(j=0;j<4;j++)

a[3-j][i]=b[i][j];

}

void array::show()

{

cout<<"旋转后的矩阵为："<<endl;

for(int i=0;i<4;i++)

{

for(int j=0;j<4;j++)

cout<<setw(8)<<a[i][j];

cout<<'\n';

}

}

int main()

{

int b[][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};

int n=4;

array test(b,n);

test.xuanzhuan();

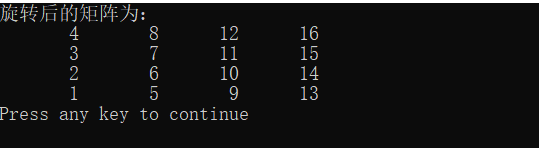
test.show();

system("pause");

return 0;

}

1. 测试结果



25．建立一个类NUM，并统计特定序列中相同的字符的个数。

具体要求如下：

（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）公有数据成员

 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。

1. 在主程序中定义一个对象，对该类进行测试。
2. 源代码

#include<iostream>

#include<stdlib.h>

using namespace std;

#include<time.h>

class Num{

char data[25];

int num[128];

public:

Num();

void process();

void print();

};

Num::Num()

{ int i;

srand(time(0));

for(i=0;i<25;i++)

data[i]=rand()%128;

}

void Num::process()

{

int i,x=0,n,m;

for(i=1;i<=128;i++)

{

for(m=0,n=0;m<25;m++)

if(data[m]==i) n++;

num[x++]=n;

}

}

void Num::print()

{ int i;

cout<<"输出随即字符串"<<endl;

cout<<data<<endl<<"字符出现的次数分别为："<<endl;

int x=0;

for(i=0;i<128;i++)

{

if(num[i]>0)

{

cout<<num[i]<<'\t';

x++;

if(x%5==0)cout<<endl;

}

}

}

int main()

{

Num test;

test.process();

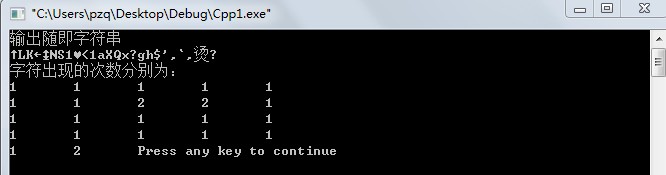
test.print();

system("pause");

return 0;

}

1. 测试结果



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。

具体要求如下：

（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。

（3）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：

3 6 4 17 5 6 4 17

8 5 9 10 变换后的矩阵为 8 7 9 10

12 19 7 20 12 23 11 20

4 14 21 23 4 14 21 29

1. 源代码：

#include<iostream>

using namespace std;

class array{

int x[4][4];

int count;

public:

array(int a[4][4]);

int fun(int );

void encode();

void print();

};

array::array(int a[4][4])

{

for(int i=0;i<4;i++)

for(int j=0;j<4;j++)

x[i][j]=a[i][j];

count=0;

}

int array::fun(int num)

{

for(int i=2;i<num;i++)

if(num%i==0)return 0;

return 1;

}

void array::encode()

{

int i,j,n;

for(i=0;i<4;i++)

for(j=0;j<4;j++)

{

if(i==0||j==3)

if(i!=j)continue;

if(fun(x[i][j]))

{

for(n=x[i][j]+1;;n++)

if(fun(n))

{

x[i][j]=n;

break;

}

count++;

}

}

}

void array::print()

{

int i,j;

cout<<"变换后的矩阵为："<<endl;

for(i=0;i<4;i++)

{

for(j=0;j<4;j++)

cout<<x[i][j]<<'\t';

cout<<endl;

}

}

int main()

{

int a[4][4]={3,6,4,17,8,5,9,10,12,19,7,20,4,14,21,23};

array test(a);

test.encode();

test.print();

system("pause");

return 0;

}

1. 测试结果：

