Санкт-Петербургский государственный университет телекоммуникаций имени профессора М. А. Бонч-Бруевича

Лабораторная работа № 1.

Вариант №23.

Группа ИКПИ-72.

Сёмин Иван

## 1.Цель работы

Цель настоящей работы состоит в ознакомлении студента с правилами организации классов, принятыми при программировании на языке С++. В процессе выполнения настоящей работы каждый студент должен разработать два класса и написать тестовые программы для демонстрации их работоспособности.

## 2. Постановка задачи и варианты заданий

## Матрица

#### **Задача 11**

Необходимо разработать класс, инкапсулирующий динамическую матрицу. Память для такой матрицы должна выделяться во время выполнения программы.

В задаче № 11 следует создать матрицу, которую можно условно назвать матрицей в стиле языка Си. Конструктор такой матрицы должен определять количество строк и столбцов. Минимальное значение индекса для строк и столбцов должно быть принято равным нулю.

Разрабатываемые классы должны обеспечить выполнения следующих операций:

* сложение матриц,
* вычитание матриц,
* умножение матриц (факультативно),
* вывод матрицы на экран дисплея,
* ввод элементов матрицы с клавиатуры,
* вычисление суммы элементов каждой строки,
* вычисление суммы элементов каждого столбца.

### **Рациональные числа**

#### **Задача 14**

Рациональные числа – это множество частных вида P / Q, где P и Q – целые числа, причем Q != 0. Число P называется числителем, а Q – знаменателем.

Такое представление чисел не является однозначным. Например,

2 / 5 == 4 / 10 == 12 / 30.

Обычно результат вычислений с рациональными числами приводят к так называемой редуцированной форме, когда числитель и знаменатель не имеют общего знаменателя. Чтобы выполнить преобразование рационального числа к редуцированной форме, числитель и знаменатель необходимо разделить на их наибольший делитель(GCD, greatest common denominator).

Разработанный класс(Rational) должен обеспечить выполнение следующих операций:

* конструктор умолчания.
* конструктор с параметрами.
* сложение,
* вычитание,
* умножение,
* деление,
* сравнения на равенство и неравенство.
* отношения.
* вывод рационального числа на экран дисплея.

Замечание. После выполнения арифметических операций результат должен быть преобразован к редуцированной форме.

3.Код на языке С++

**MATRIX**

**//main.cpp//**

#include <iostream>  
#include <string>  
#include "matrix.h"  
  
using namespace std;  
  
int main()  
{  
 int button;  
 string mat\_name1 = "Matrix #1";  
 string mat\_name2 = "Matrix #2";  
  
 while (1 != 0)  
 {  
 cout << " [1] - Add matrix\n"  
 " [2] - Sub matrix\n"  
 " [3] - Mul matrix\n"  
 " [4] - Sum of matrix elements row and col\n"  
 " [0] - Exit\n";  
 cin >> button;  
 system("cls");  
 switch (button)  
 {  
 case 1://add  
 {  
 matrix M1 (mat\_name1);  
 matrix M2 (mat\_name2);  
 M1.output\_mat();  
 M2.output\_mat();  
 matrix M\_Res = M1 + M2;  
 M\_Res.output\_mat();  
 }  
 break;  
 case 2://sub  
 {  
 matrix M1 (mat\_name1);  
 matrix M2 (mat\_name2);  
 M1.output\_mat();  
 M2.output\_mat();  
 matrix M\_Res = M1 - M2;  
 M\_Res.output\_mat();  
 }  
 break;  
 case 3://mul  
 {  
 matrix M1 (mat\_name1);  
 matrix M2 (mat\_name2);  
 M1.output\_mat();  
 M2.output\_mat();  
 matrix M\_Res = M1 \* M2;  
 M\_Res.output\_mat();  
 }  
 break;  
 case 4://sum\_col\_and\_row  
 {  
 matrix M1(mat\_name1);  
 M1.output\_mat();  
 M1.sum\_row\_mat();  
 M1.sum\_col\_mat();  
 M1.output\_sum\_col\_and\_row();  
 }  
 break;  
 case 0://exit  
 {  
 system("pause");  
 }  
 return 0;  
 default:  
 printf("Такой кнопки нет(\n");  
 break;  
 }  
 }  
}

**//matrix.h**

#ifndef LAB2\_MATRIX\_H  
#define LAB2\_MATRIX\_H  
#include <string>  
using namespace std;  
  
class matrix  
{  
 int \*sum\_col;  
 int \*sum\_row;  
 int x, y;  
 int \*\*mat;  
 bool status\_sum=false;  
 string name;  
 public:  
 matrix(string mat\_name);  
 matrix();  
 matrix(int X,int Y,string mat\_name);  
 matrix(const matrix &Matrix);  
 ~matrix();  
 void get\_mat\_size();  
 void dynamic\_mat();  
 void delete\_dynamic\_mat();  
 void input\_mat();  
 void output\_mat()const;  
 matrix operator+(matrix Matrix)const;  
 matrix operator-(matrix Matrix)const;  
 matrix operator\*(matrix Matrix)const;  
 void sum\_row\_mat();  
 void sum\_col\_mat();  
 void output\_sum\_col\_and\_row()const;  
  
};  
  
#endif //LAB2\_MATRIX\_H

**//matrix.cpp**

#include "matrix.h"  
#include <iostream>  
#include <iomanip>  
#include <fstream>  
#include <sstream>  
#include <string.h>  
  
using namespace std;  
  
matrix::matrix(string mat\_name) : name(mat\_name)//keyboard data  
{  
 get\_mat\_size();  
 dynamic\_mat();  
 input\_mat();  
}  
  
matrix::matrix(int X,int Y,string mat\_name) : x(X), y(Y), name(mat\_name)//data  
{  
 dynamic\_mat();  
}  
  
matrix::matrix(const matrix &Matrix)//copy  
{  
 x=Matrix.x;  
 y=Matrix.y;  
 dynamic\_mat();  
 mat = Matrix.mat;  
 name = Matrix.name + " Copy";  
}  
matrix::matrix() : x(0),y(0) {}  
  
matrix::~matrix()  
{  
 delete\_dynamic\_mat();  
 if (status\_sum)  
 {  
 delete[] sum\_col;  
 delete[] sum\_row;  
 }  
  
}  
  
void matrix::get\_mat\_size()  
{  
 cout << "Enter the number of col of the matrix " << name << " : ";  
 cin >> x;  
 cout << "Enter the number of rows of the matrix " << name << " : ";  
 cin >> y;  
}  
  
void matrix::dynamic\_mat()  
{  
 mat = new int \*[y];  
 for (int i = 0; i < y; i++)  
 {  
 mat[i] = new int[x];  
 }  
}  
  
void matrix::input\_mat()  
{  
 cout << "Enter the elements of the matrix "<<name<<" : " << endl;  
 for (int i = 0; i < y; i++)  
 for (int j = 0; j < x; j++)  
 cin >> mat[i][j];  
}  
  
void matrix::output\_mat()const  
{  
 cout << name <<" : " << endl;  
 cout << setw(5);  
 for (int i = 0; i < y; i++)  
 {  
 for (int j = 0; j < x; j++)  
 {  
 cout << mat[i][j] << setw(5);  
 }  
 cout << endl;  
 }  
}  
  
matrix matrix::operator+(matrix Matrix)const  
{  
 string mat\_name = "Matrix-Result";  
 matrix result (x,y,mat\_name);  
 for (int i = 0; i < y; i++)  
 for (int j = 0; j < x; j++)  
 {  
 result.mat[i][j] = mat[i][j] + Matrix.mat[i][j];  
 }  
 return result;  
}  
  
matrix matrix::operator-(matrix Matrix)const  
{  
 char mat\_name[] = { "Matrix-Result" };  
 matrix result (x,y,mat\_name);  
 for (int i = 0; i < y; i++)  
 for (int j = 0; j < x; j++)  
 {  
 result.mat[i][j] = mat[i][j] - Matrix.mat[i][j];  
 }  
 return result;  
}  
  
matrix matrix::operator\*(matrix Matrix)const  
{  
 char mat\_name[] = { "Matrix-Result" };  
 int X=x;  
 if (x>y) X=y;  
 matrix result (X,X,mat\_name);  
 for(int i=0; i< y; i++)  
 for(int j=0; j< y; j++)  
 {  
 result.mat[i][j] = 0;  
 for (int v = 0; v < x; v++)  
 result.mat[i][j] += mat[i][v] \* Matrix.mat[v][j];  
 }  
 return result;  
}  
  
void matrix::sum\_row\_mat()  
{  
 sum\_row = new int[y];  
 for (int k = 0; k < y; ++k) sum\_row[k]=0;  
 for (int i = 0; i < y; i++)  
 for (int j = 0; j < x; j++)  
 sum\_row[i] += mat[i][j];  
  
 status\_sum=true;  
}  
  
void matrix::sum\_col\_mat()  
{  
 sum\_col = new int[x];  
 for (int k = 0; k < x; ++k) sum\_col[k]=0;  
  
 for (int i = 0; i < x; i++)  
 for (int j = 0; j < y; j++)  
 sum\_col[i] += mat[j][i];  
}  
void matrix::output\_sum\_col\_and\_row()const  
{  
 cout << "The sum of elements of each row of the matrix " << name << " :" << endl;  
 cout<<setw(5);  
 for (int i = 0; i < y; i++)  
 {  
 cout << " [" << i+1 << "] ";  
 cout << sum\_row[i] << setw(5) << endl;  
 }  
  
 cout << "The sum of elements of each col of the matrix " << name << " :" << endl;  
 cout<<setw(5);  
 for (int i = 0; i < x; i++)  
 {  
 cout << " [" << i << "] ";  
 cout << sum\_col[i] << setw(5) << endl;  
 }  
}  
  
void matrix::delete\_dynamic\_mat()  
{  
 for (int i = y - 1; 0 <= i; i--)  
 delete[] mat[i];  
}

**RATIONAL**

**//main.cpp**

#include <iostream>  
#include "rational.h"  
using namespace std;  
  
int main()  
{  
 int button;  
  
while(1!=0)  
{  
 cout << " [1] - Reduced form of R number\n"  
 " [2] - Operations with R numbers\n"  
 " [0] - Exit\n";  
 cin >> button;  
 system("cls");  
 switch (button)  
 {  
 case 1://Reduced form of R number  
 {  
 rational Num1;  
 cout<<"Reduced: ";  
 Num1.get\_rational();  
 cout << endl;  
 }  
 break;  
 case 2://Operations with R numbers  
 {  
 rational Num1;  
 rational Num2;  
 cout<<"Operations with R numbers: "<<endl;  
 rational Res1 = Num1 + Num2;  
 Res1.get\_operation(Num1,Num2, " + " );  
 rational Res2 = Num1 - Num2;  
 Res2.get\_operation(Num1,Num2, " - " );  
 rational Res3 = Num1 \* Num2;  
 Res3.get\_operation(Num1,Num2, " \* " );  
 rational Res4 = Num1 / Num2;  
 Res4.get\_operation(Num1,Num2, " / " );  
 if (Num1<Num2)  
 {  
 Num1.get\_rational();cout<<" < ";Num2.get\_rational();  
 }  
 else if (Num1==Num2)  
 {  
 Num1.get\_rational();cout<<" = ";Num2.get\_rational();  
 }  
 else  
 {  
 Num1.get\_rational();cout<<" > ";Num2.get\_rational();  
 }  
 cout<<endl;  
 }  
 break;  
 case 0://exit  
 {  
 system("pause");  
 }  
 return 0;  
 default:  
 printf("Такой кнопки нет(\n");  
 break;  
 }  
}  
}

**//rational.h**

#ifndef LAB1\_2\_RATIONAL\_H  
#define LAB1\_2\_RATIONAL\_H  
  
#include <string>  
using namespace std;  
  
class rational  
{  
 int x,y;  
 public:  
 rational();  
 rational(int X, int Y);  
 rational(const rational & Rational);  
 void reduced();  
 void get\_rational()const;  
 void get\_operation(rational Num1, rational Num2, string sign);  
 rational operator+(rational Num)const;  
 rational operator-(rational Num)const;  
 rational operator\*(rational Num)const;  
 rational operator/(rational Num)const;  
 bool operator<(rational Num)const;  
 bool operator==(rational Num)const;  
};  
  
#endif //LAB1\_2\_RATIONAL\_H

**//rational.cpp**

#include "rational.h"  
#include <string>  
#include <iostream>  
  
using namespace std;  
  
rational::rational(int X, int Y) : x(X),y(Y)  
{  
 reduced();  
}  
  
rational::rational()  
{  
 cout<<"Enter numerator: "<<endl;  
 cin>>x;  
 cout<<"Enter denominator: "<<endl;  
 cin>>y;  
 if (y==0)  
 {  
 cout << "Denominator not 0" << endl;  
 return;  
 }  
 reduced();  
}  
  
rational::rational(const rational & Rational)  
{  
 x=Rational.x;  
 y=Rational.y;  
}  
  
void rational::reduced()  
{  
 int a=x, b=y, c;  
 while (b)  
 {  
 c = a % b;  
 a = b;  
 b = c;  
 }  
 x=x/a;  
 y=y/a;  
}  
  
void rational::get\_rational()const  
{  
 if (y==1)  
 cout<<x;  
 else cout<<x<<"/"<<y;  
}  
void rational::get\_operation(rational Num1, rational Num2, string sign)  
{  
 Num1.get\_rational();  
 cout<< sign;  
 Num2.get\_rational();  
 cout<<" = ";  
 get\_rational();  
 cout<<endl;  
}  
rational rational::operator+(rational Num)const  
{  
 int X=x\*Num.y+Num.x\*y;  
 int Y=y\*Num.y;  
 return rational(X,Y);  
}  
  
rational rational::operator-(rational Num)const  
{  
 int X=x\*Num.y-Num.x\*y;  
 int Y=y\*Num.y;  
 return rational(X,Y);  
}  
  
rational rational::operator\*(rational Num)const  
{  
 int X=x\*Num.x;  
 int Y=y\*Num.y;  
 return rational(X,Y);  
}  
  
rational rational::operator/(rational Num)const  
{  
 int X=x\*Num.y;  
 int Y=y\*Num.x;  
 return rational(X,Y);  
}  
bool rational::operator<(rational Num)const  
{  
  
 int X1 = x\*Num.y;  
 int X2 = Num.x\*y;  
 return (X1 < X2);  
}  
  
bool rational::operator==(rational Num)const  
{  
  
 int X1 = x\*Num.y;  
 int X2 = Num.x\*y;  
 return (X1 == X2);  
}