ФГАОУВО «Южно-Уральский государственный университет (НИУ)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине «Объектно-ориентированное программирование»

 Автор работы

 студент группы ЕТ-212

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А.Александрова

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

 Работа зачтена с оценкой

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.К.Демидов

 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г.

Челябинск, 2017

# 1 Постановка задачи

I. Реализовать класс

1. Рациональные числа (дроби) Rationalional
это числа вида a/b, где a и b не имеют общих делителей, b>0, |a|,|b|<2^31

 Конструктор: Rationalional(a,b)

 Операции:

 x+y, x-y, x\*y, x/y,

 x+=y, x-=y, x\*=y, x/=y,

 x>y, x<y, x>=y, x<=y, x==y, x!=y

 !x (x равно нулю)

 где x,y - рациональные числа или long

 (определить преобразование long->Rationalional)

 вывод, ввод в виде 3/5

 Методы:

 long getp(); (числитель)

 long getq(); (знаменатель)

Операции (если есть в задании) =, [], +=, -=, \*=, /=, префиксные ++, -- определять как методы.
Для ввода переопределить >> , для вывода - << . Формат ввода-вывода объектов делать так, как указано в задании.
Запись [текст] означает, что текст может отсутствовать, например, конструктор ИмяКласса(a[,b[,c]]) может быть вызван с 1, 2 или 3 аргументами.

II. Реализовать main c тестами (создание объектов и выполнение действий с ними).

# 2 Описание интерфейса класса

class Rational {

 long a,b;//числитель и знаменатель.

public:

 Rational(long=0, long=1);//конструктор

 Rational &operator+=(const Rational&);//перегруз операции +=

 Rational &operator-=(const Rational&);//перегруз операции -=

 Rational &operator\*=(const Rational&);//перегруз операции \*=

 Rational &operator/=(const Rational&);//перегруз операции /=

 friend bool operator<(const Rational &,const Rational &);//перегруз операции <

 friend bool operator==(const Rational &, const Rational &);//перегруз операции ==

 bool operator!()const{return (a==0);}//перегруз операции !

 friend istream &operator>>(istream &, Rational &);//перегруз операции >>

 friend ostream &operator<<(ostream &, const Rational &);// перегруз операции <<

 long getp() const { return a; }

 long getq() const { return b; }

};

# 3 Описание тестов для проверки классов

int main() {

 Rational a(1, 3), b(1,2), c(15);

 cout<<"Вывод\n";

 cout<<"a="<<a<<" "<<"b="<<b<<" "<<"c="<<c<<endl;

 cout<<"Ввод a\n";

 cin>>a;

 cout<<"a="<<a <<endl;

 cout<<"Проверка операции +\n";

 cout<<a<<" + "<<b<<" = "<<(a+b)<<endl;

 cout<<a<<" + "<<2<<" = "<<(a+2)<<endl;

 cout<<2<<" + "<<a<<" = "<<(2+a)<<endl;

 cout<<a+b<<endl;

…

 cout<<"Проверка операции +=\n";

 a+=b;

 cout<<"a="<<a<<endl;

…

 cout<<"Проверка операции >\n";

 cout<<a<<" > "<<b<<" = "<<(a>b)<<endl;

 cout<<a<<" > "<<2<<" = "<<(a>2)<<endl;

 cout<<2<<" > "<<a<<" = "<<(2>a)<<endl;

…

 return 0;

}

**Полученные результаты.**

Вывод

a=1/3 b=1/2 c=15/1

Ввод a

2/7

a=2/7

Проверка операции +

2/7 + 1/2 = 11/14

2/7 + 2 = 16/7

2 + 2/7 = 16/7

Проверка операции –

…

Проверка операции \*

…

Проверка операции /

…

Проверка операции +=

a=11/14

Проверка операции -=

a=2/7

Проверка операции >

11/14 > 1/2 = 1

11/14 > 2 = 0

2 > 11/14 = 1

Проверка операции <

…

Проверка операции >=

…

Проверка операции <=

…

Проверка операции ==

…

Проверка операции !=

…

# 4 Листинг реализации класса

long NOD(long a, long b){

 long c;

 if(a<0) a=-a;

 while(b!=0){

 c=a;

 a=b;

 b=c%b;

 }

 return a;

}

Rational::Rational(long a, long b){

 long tmp=NOD(a,b);

 this->a=a/tmp;

 this->b=b/tmp;

}

Rational &Rational::operator+=(const Rational&p) {

 a=a\*p.b+p.a\*b;

 b=b\*p.b;

 long tmp=NOD(a,b);

 a/=tmp;

 b/=tmp;

 return \*this;

}

Rational operator+(Rational x, Rational y) {

 return x+=y;

}

Rational operator/(Rational x, Rational y) {

 return x/=y;

}

bool operator<(const Rational &x, const Rational &y) {

 if ((x.a\*y.b-y.a\*x.b)<0) return 1;

 return 0;

}

bool operator>(const Rational &x, const Rational &y) {

 return y<x;

}

bool operator>=(const Rational &x, const Rational &y) {

 return !(x<y);

}

bool operator<=(const Rational &x, const Rational &y) {

 return !(y<x);

}

bool operator==(const Rational &x, const Rational &y) {

 return ((x.a==x.b&&y.a==y.b);

}

bool operator!=(const Rational &x, const Rational &y) {

 return !(x==y);

}

istream &operator>>(istream &s, Rational &p) {

 char c;

 return s>>p.a>>c>>p.b;

}

ostream &operator<<(ostream &s, const Rational &p) {

 return s<<p.a <<"/"<<p.b;

}