**Домашнее задание**

**Встреча №7**

***Тема:*** *Перегрузка операторов.*

**Задание 1.**

Создайте класс Дробь (или используйте уже ранее созданный вами). Используя перегрузку операторов реалізуйте для него арифметические операции для работы с дробами (операции +, -, \*, /).

#include <iostream>

using namespace std;

class fraction {

public:

int num, den, whole\_part;

fraction(){}

fraction (int \_num, int \_den) : num(\_num), den(\_den) {}

int Getden() {

return den;

}

int Getnum() {

return num;

}

};

int nod(int a, int b) {

int e, f;

e = b / a;

if (b % a == 0)

return (a);

else

{

e = b % a;

f = a / e;

if (a % e == 0)

return (e);

else

{

f = a % e;

nod(f, e);

}

}

}

fraction whole(fraction frac) {

frac.whole\_part = frac.num / frac.den;

frac.num = frac.num % frac.den;

frac.den = frac.den;

return frac;

}

fraction operator+(fraction frac1, fraction frac2) {

fraction frac\_rez;

if (frac1.den == frac2.den) {

frac\_rez.num = frac1.num + frac2.num;

frac\_rez.den = frac2.den;

}

else

{

frac\_rez.num = frac1.num \* frac2.den + frac2.num \* frac1.den;

frac\_rez.den = frac1.den \* frac2.den;

}

return frac\_rez;

}

fraction operator-(fraction frac1, fraction frac2) {

fraction frac\_rez;

if (frac1.den == frac2.den) {

frac\_rez.num = frac1.num - frac2.num;

frac\_rez.den = frac2.den;

}

else

{

frac\_rez.num = frac1.num \* frac2.den - frac2.num \* frac1.den;

frac\_rez.den = frac1.den \* frac2.den;

}

return frac\_rez;

}

fraction operator\*(fraction frac1, fraction frac2) {

fraction frac\_rez;

frac\_rez.num = frac1.num \* frac2.num;

frac\_rez.den = frac1.den \* frac2.den;

return frac\_rez;

}

fraction operator/(fraction frac1, fraction frac2) {

fraction frac\_rez;

frac\_rez.num = frac1.num \* frac2.den;

frac\_rez.den = frac1.den \* frac2.num;

return frac\_rez;

}

fraction sokr (fraction frac\_rez, bool &w){

int e;

e = nod(frac\_rez.num, frac\_rez.den);

if (e > 1)

{

frac\_rez.num = frac\_rez.num / e;

frac\_rez.den = frac\_rez.den / e;

}

if (abs(frac\_rez.num) >= abs(frac\_rez.den))

{

w = true;

frac\_rez = whole(frac\_rez);

}

return frac\_rez;

}

void print(fraction frac, bool w) {

if (w == true)

{

if (frac.whole\_part < 0 && frac.num < 0)

cout << frac.whole\_part << " " << -frac.num << " / " << frac.den << endl;

else

cout << frac.whole\_part << " " << frac.num << " / " << frac.den << endl;

}

else

cout << frac.num << " / " << frac.den << endl;

}

void main() {

fraction frac1(1, 2);

fraction frac2(2, 3);

cout << "frac 1 = " << frac1.Getnum() << " / " << frac1.Getden() << endl;

cout << "frac 2 = " << frac2.Getnum() << " / " << frac2.Getden() << endl;

fraction frac\_rez;

bool e(false), w(false);

cout << "Sum frac1 & frac2 ";

frac\_rez = frac1 + frac2;

frac\_rez = sokr(frac\_rez, w);

print(frac\_rez, w);

w = false;

cout << "Sub dig & frac ";

frac\_rez = frac1 - frac2;

frac\_rez = sokr(frac\_rez, w);

print(frac\_rez, w);

w = false;

cout << "Mul dig & frac ";

frac\_rez = frac1 \* frac2;

frac\_rez = sokr(frac\_rez, w);

print(frac\_rez, w);

w = false;

cout << "Div dig & frac ";

frac\_rez = frac1 / frac2;

frac\_rez = sokr(frac\_rez, w);

print(frac\_rez, w);

system("pause");

}

**Задание 2.**

Создайте класс Complex (комплексное число) или используйте уже созданный вами класс. Создайте перегруженные операторы для реализации арифметических операций для по работе с комплексными числами (операции +, -, \*, /).

#include <iostream>

using namespace std;

class komplex {

public:

float a, b;

komplex() {}

komplex (int \_a, int \_b) : a(\_a), b(\_b) {}

float Geta() {

return a;

}

int Getb() {

return b;

}

};

komplex operator+(komplex numb1, komplex numb2) {

komplex buf;

buf.a = numb1.a + numb2.a;

buf.b = numb1.b + numb2.b;

return buf;

}

komplex operator-(komplex numb1, komplex numb2) {

komplex buf;

buf.a = numb1.a - numb2.a;

buf.b = numb1.b - numb2.b;

return buf;

}

komplex operator\*(komplex numb1, komplex numb2) {

komplex buf;

buf.a = numb1.a\*numb2.a - numb1.b\*numb2.b;

buf.b = numb1.a\*numb2.b + numb2.a\*numb1.b;

return buf;

}

komplex operator/(komplex numb1, komplex numb2) {

komplex buf;

buf.a = (numb1.a\*numb2.a + numb1.b\*numb2.b) / (numb2.a\*numb2.a + numb2.b\*numb2.b);

buf.b = (numb2.a\*numb1.b - numb1.a\*numb2.b) / (numb2.a\*numb2.a + numb2.b\*numb2.b);

return buf;

}

void main() {

komplex numb1(1, 2);

komplex numb2(2, 3);

cout << numb1.a << " " << numb1.b << "i" << endl;

cout << numb2.a << " " << numb2.b << "i" << endl;

komplex res;

cout << "the sum of the number 1 and the number 2" << endl;

res = numb1 + numb2;

cout << res.a << " " << res.b << "i" << endl;

cout << "the difference between the number 1 and the number 2" << endl;

res = numb1 - numb2;

cout << res.a << " " << res.b << "i" << endl;

cout << "the product of numbers 1 and 2" << endl;

res = numb1 \* numb2;

cout << res.a << " " << res.b << "i" << endl;

cout << "the quotient of numbers 1 and 2" << endl;

res = numb1 / numb2;

cout << res.a << " " << res.b << "i" << endl;

system("pause");

}