Министерство образования и науки Российской Федерации

ФГАОУ ВПО «УрФУ имени первого Президента России Б. Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РтФ

Кафедра радиоэлектроники и информационных систем

**Отчет по лабораторной работе №5**

Преподаватель Бабич Михаил Владимирович

Студенты гр. РИ-240002 Овчинникова К. С. Трефилов А.А.

Екатеринбург

2015

Задача 2

Задача: Для класса дроби, реализованного ранее, создайте пререгрузки операций +,-,\*,/, так чтобы компилятор верно трактовал записи вида fract1 = fract2 + fract3 и т.д. Создайте перегрузки операций, позволяющие прибавлять к дробям целые числа, так чтобы компилятор верно трактовал записи вида fract1 = fract2 + 5 и fract1 = 5 + fract2 и т.д. Для реализации второй перегрузки потребуется создать дружественную функцию. Реализуйте метод упрощения дополнительно при помощи дружественной функции.

Код:

#include <iostream>

#include <iomanip>

#include <clocale>

using namespace std;

class frac {

private:

friend frac operator +(frac f, frac a);

friend frac operator -(frac f, frac a);

friend frac operator \*(frac f, frac a);

friend frac operator /(frac f, frac a);

friend void upr();

public:

long numerator, denominator;

frac(long num) : numerator(num), denominator(1) {}

frac(long num, long den) : numerator(num), denominator(den) {}

void output() const {

cout << numerator << '/'<< denominator<< endl;

}

};

frac operator +(frac f, frac a)

{

frac fracf(0, 1);

fracf.numerator = f.numerator\*a.denominator + a.numerator\* f.denominator;

fracf.denominator = f.denominator\*a.denominator;

return fracf;

}

frac operator - (frac f, frac a)

{

frac fracf(0, 1);

fracf.numerator = f.numerator\*a.denominator - a.numerator\* f.denominator;

fracf.denominator = f.denominator\*a.denominator;

return fracf;

}

frac operator \* (frac f, frac a)

{

frac fracf(0, 1);

fracf.numerator = a.numerator\* f.numerator;

fracf.denominator = f.denominator\*a.denominator;

return fracf;

}

frac operator / (frac f, frac a)

{

frac fracf(0, 1);

fracf.numerator = f.numerator\*a.denominator;

fracf.denominator = f.denominator\*a.numerator;

return fracf;

}

frac upr(frac f1)

{

int a = f1.numerator, b = f1.denominator;

int t;

if (a<b) { t = a; a = b; b = t; }

while (b != 0) {

t = b;

b = a%b;

a = t;

}

int nod = a;

f1.numerator /= nod;

f1.denominator /= nod;

return f1;

}

void main() {

frac f1(9, 5.07);

frac f2 = 154 + f1;

}

Задача 3

Задача: Создайте базовый класс и два производных класса. Создайте функцию DisplayName (Base \*), которая принимает указатель на объект базового класса и выводит на консоль имя класса, который был передан по указателю. Создайте метод main, который позволит убедиться в работоспособности программы.

Код:

#include <iostream>

#include <iomanip>

#include <clocale>

#include <typeinfo>

using namespace std;

class base {

public: virtual void Name()=0;

};

class Class1 : public base {

void Name(){cout << typeid(Class1).name() << endl; }

};

class Class2 : public base {

void Name() { cout << typeid(Class2).name() << endl; }

};

void displayName(base\* clas) {

clas->Name();

}

void main() {

base\* clas1 = new Class1;

base\* clas2 = new Class2;

displayName(clas1);

displayName(clas2);

system("pause");

}

**Задача 4**

Задача: Создайте функцию swaps, обменивающую значения двух аргументов, посылаемых ей. Сделайте из функции шаблон, чтобы она могла использоваться с любыми числовыми типами данных (int, char, float и т. д.). Напишите main() для тестирования функции.

Код:

#include "iostream"

using namespace std;

template <typename T>

void swaps(T& a, T& b)

{

T c;

c = a;

a = b;

b = c;

}

void main()

{

int i1=5, i2=6;

char c1='w', c2='s';

swaps(i1, i2);

swaps(c1, c2);

}

**Задача 9**

Задача: Напишите программу, выводящую на консоль самое большое число Фибоначчи, которое может хранится на целевой машине и влазит в тип int.

Код:

#include <iostream>

using namespace std;

void main()

{

long a = 0, b = 1;

int c = 1;

try {

while (b <= INT\_MAX)

{

c = a + b;

a = b;

b = c;

}

}

catch(int){

cout << c << endl;

system("PAUSE");

}

**Вывод**

В данной лабораторной работе мы научились работать перезагружать операторы и создавать дружественные функции; ознакомились с функцией typeid; научились создавать и работать с шаблонами; узнали некоторые свойства простого типа int