**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра математического обеспечения и применения ЭВМ**

**Отчёты по лабораторным работам**

по курсу «Объектно-ориентированный анализ и проектирование»

**Вариант №2**

Выполнили: студенты группы КТбо2-1: Шутенко Остап Русланович

Проверил: доцент кафедры МОП ЭВМ Тарасов

ТАГАНРОГ 2020

**Лабораторная работа №1**

**Классы и объекты в С++**

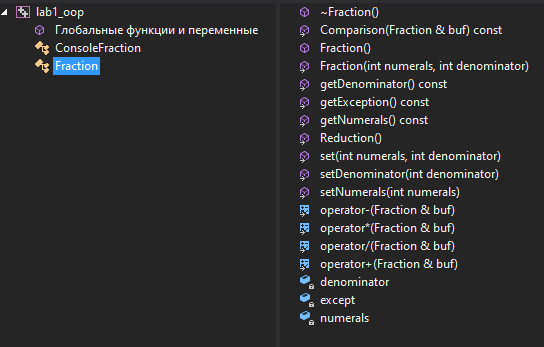
**Цель работы**

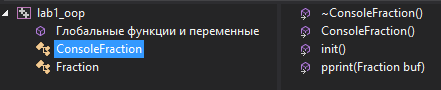
Ознакомление с технологией построения классов, разработкой методов, дружественных функций и перегрузкой операций.

**Формулировка задачи**

Определить класс Fraction, в котором производятся операции над обыкновенными дробями (сравнение, сложение, умножение, деление, сокращение).

**Диаграммы классов**





**Перечень сигнатур методов**

+ : сложение;

- : вычитание;

/ : деление;

\*: умножение;

Методы getNumerals(): выводит числитель.

Методы getDenominator(): выводит знаменатель.

Методы getException(): выводит флаг исключения.

Методы setNumerals(): меняет числитель.

Методы setDenominator(): меняет знаменатель.

Методы set(): меняет дробь.

Метод Comparison (): сравнивает две дроби, приводя их к общему знаменателю и сравнивая числители.

Метод Reduction (): функция сокращает дробь

**Листинг программы lab1\_oop.cpp на языке С++**

#include <iostream>

#include "ConsoleFraction.h"

int main()

{

ConsoleFraction CF;

CF.init();

return 0;

}

**Листинг программы ConsoleFraction.h на языке С++**

#pragma once

#include "Fraction.h"

#include <iostream>

class ConsoleFraction

{

public:

ConsoleFraction();

~ConsoleFraction();

void init();

void pprint(Fraction buf);

};

**Листинг программы Fraction.h на языке С++**

#pragma once

class Fraction

{

public:

// Конструктор

Fraction() {};

// Деструктор

~Fraction() {};

// Конструктор присваивания числителя и знаменателя, проверка знаменателя на равенство нулю

// В переменную а присваивается числитель, в переменную b - знаменатель, но только если он не равен нулю

Fraction(int numerals, int denominator);

// Сложение

Fraction operator+(Fraction& buf);

// разность

Fraction operator-(Fraction& buf);

// Умножение

Fraction operator\*(Fraction& buf);

// Деление

Fraction operator/(Fraction& buf);

// Сокращение

Fraction Reduction();

// Сравнение

int Comparison(Fraction& buf)const;

// Вывод

int getNumerals() const;

int getDenominator()const;

bool getException()const;

// Инициализация дроби(изменение) - числителя и знаменателя

void set(int numerals, int denominator);

void setNumerals(int numerals);

void setDenominator(int denominator);

private:

bool except = false;

int numerals = 0; //a-числитель

int denominator = 1; //b-знаменатель

};

**Листинг программы ConsoleFraction.cpp на языке С++**

#include "ConsoleFraction.h"

ConsoleFraction::ConsoleFraction() {};

ConsoleFraction::~ConsoleFraction() {};

void ConsoleFraction::init()

{

Fraction fa, fb;

while (true)

{

std::cout << "type first Fraction\n";

int a, b;

std::cin >> a >> b;

fa.set(a, b);

std::cout << "type last Fraction\n";

std::cin >> a >> b;

fb.set(a, b);

std::cout << "0 - test the assignment operator;\n1 - test operator +;\n2 - test operator-;\n3 - test operator \*;\n4 - test operator /;\n5 - to check the function of the comparison.\n";

while (true)

{

std::cin >> a;

if (a > 5 || a < -1)

{

std::cout << "command error\n";

}

else break;

}

switch (a)

{

case 0:fb = fa; fb.set(5, 6);

std::cout << "b=a b=";

pprint(fb);

std::cout << " a=";

pprint(fa);

std::cout << "\n";

break;

case 1:

fb = fb + fa;

std::cout << "b+a = ";

pprint(fb);

std::cout << "\n";

break;

case 2:

fb = fb - fa;

std::cout << "b-a =";

pprint(fb);

std::cout << "\n";

break;

case 3:

fb = fb \* fa;

std::cout << "b\*a =";

pprint(fb);

std::cout << "\n";

break;

case 4:

fb = fb / fa;

pprint(fb);

std::cout << "\n";

break;

case 5:

std::cout <<"-1=exc, 0= a==b, 1= a>b, 2= a<b:"<< fb.Comparison(fa) << "\n";

break;

}

if (a == -1)

break;

}

};

void ConsoleFraction::pprint(Fraction buf)

{

std::cout << buf.getNumerals() << "/" << buf.getDenominator() << " exc:" << buf.getException();

};

**Листинг программы Fraction.cpp на языке С++**

#include "Fraction.h"

#include <cstdlib>

Fraction::Fraction(int numerals, int denominator)

{

this->set(numerals, denominator);

/\*

try

{

if (denominator == 0) // Если знаменатель равен нулю, то мы предупреждаем о неправильном вводе данных

throw 1;

else

{

this->numerals = numerals;

this->denominator = denominator;

this->except = false;

}

}

catch (int thr)

{

if (thr == 1)

{

this->except = true;

}

}

\*/

}

// Сложение

Fraction Fraction::operator+(Fraction& buf)

{

Fraction Sum;

// Приводим обе дроби к общему знаменателю

Sum.numerals = numerals \* buf.denominator + denominator \* buf.numerals;

Sum.denominator = denominator \* buf.denominator;

this->except = false;

try

{

if (Sum.denominator == 0) // Если знаменатель равен нулю, то мы предупреждаем о неправильном вводе данных

throw 1;

}

catch (int thr)

{

if (thr == 1)

{

Sum.except = true;

}

}

// Возвращаем новую дробь, являющейся суммой первой и второй дроби

return Sum.Reduction();

}

// разность

Fraction Fraction::operator-(Fraction& buf)

{

Fraction Sum;

// Приводим обе дроби к общему знаменателю

Sum.numerals = numerals \* buf.denominator - denominator \* buf.numerals;

Sum.denominator = denominator \* buf.denominator;

Sum.except = false;

try

{

if (Sum.denominator == 0) // Если знаменатель равен нулю, то мы предупреждаем о неправильном вводе данных

throw 1;

}

catch (int thr)

{

if (thr == 1)

{

this->except = true;

}

}

// Возвращаем новую дробь, разностью суммой первой и второй дроби

return Sum.Reduction();

}

// Умножение

Fraction Fraction::operator\*(Fraction& buf)

{

Fraction Mult;

// Числитель конечной дроби равен произведению числителя первой и второй дроби

Mult.numerals = numerals \* buf.numerals;

Mult.denominator = denominator \* buf.denominator;

Mult.except = false;

// Возвращаем новую дробь, являющейся произведением первой и второй дроби

try

{

if (Mult.denominator == 0) // Если знаменатель равен нулю, то мы предупреждаем о неправильном вводе данных

throw 1;

}

catch (int thr)

{

if (thr == 1)

{

this->except = true;

}

}

return Mult.Reduction();

}

// Деление

Fraction Fraction::operator/(Fraction& buf)

{

Fraction Div;

// Числитель конечной дроби равен произведению числителя первой дроби и знаменателя второй

Div.numerals = numerals \* buf.numerals;

Div.denominator = denominator \* buf.numerals;

Div.except = false;

// Проверка на то что числитель второй дроби не равен 0

try

{

if (Div.denominator == 0) // Если знаменатель равен нулю, то мы предупреждаем о неправильном вводе данных

throw 1;

}

catch(int thr)

{

if (thr == 1)

{

Div.except = true;

}

}

// Возвращаем дробь поделенную на другую

return Div.Reduction();

}

// Сокращение

Fraction Fraction::Reduction()

{

Fraction Red;

Red.numerals = numerals;

Red.denominator = denominator;

int containers = numerals;

if (numerals > denominator)

containers = denominator;

// Ищем общий делитель для числителя и знаменателя, чтобы сократить дробь

for (int i=1;i<=containers; i++)

{

if ((numerals % i)==0 && (denominator % i)==0)

{

Red.numerals = numerals / i;

Red.denominator = denominator / i;

}

}

// Возвращаем сокращённую дробь

return Red;

}

// Сравнение

int Fraction::Comparison(Fraction& buf)const

{

Fraction d1, d2;

// Приводим обе дроби к общему знаменателю

d1.numerals = numerals \* buf.denominator;

d1.denominator = denominator \* buf.denominator;

d2.numerals = buf.numerals \* denominator;

d2.denominator = buf.denominator \* denominator;

if (this->except || buf.except)

return -1;

if (d1.numerals > d2.numerals)

return 1;

if (d1.numerals < d2.numerals)

return 2;

if (d1.numerals == d2.numerals)

return 0;

}

// Вывод

bool Fraction::getException()const

{

return this->except;

}

int Fraction::getNumerals()const

{

return this->numerals;

}

int Fraction::getDenominator()const

{

return this->denominator;

}

// Инициализация дроби(изменение) - числителя и знаменателя

void Fraction::set(int numerals, int denominator)

{

this->numerals = numerals;

this->denominator = denominator;

this->except = false;

try

{

if (denominator == 0) // Если знаменатель равен нулю, то мы предупреждаем о неправильном вводе данных

throw 1;

}

catch (int thr)

{

if (thr == 1)

{

this->except = true;

}

}

}

void Fraction::setNumerals(int numerals)

{

this->numerals = numerals;

}

void Fraction::setDenominator(int denominator)

{

this->denominator = denominator;

this->except = false;

try

{

if (denominator == 0)

throw 1;

}

catch (int thr)

{

if (thr == 1)

{

this->except = true;

}

}

}

**Заключение**

В процессе выполнения лабораторной работы №1, мы ознакомились с технологией построения классов и разработкой методов.

Автор Преподаватель: Тарасов

Шутенко Остап Русланович Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись \_\_\_\_\_\_\_ Дата: \_\_.\_\_.20\_\_