

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа
по курсу «ООП»**

**Тема:
Операторы, литералы.**

Студент:	Черненко И.Д
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	4
Оценка:	
Дата:	

Москва
2019

1. Код программы на языке C++:

Fazzy_Number.hpp:

```
#ifndef LAB1_FAZZYNUMBER_HPP
#define LAB1_FAZZYNUMBER_HPP

class fn {
public:
    fn();
    fn(double a, double b);
    fn operator+(const fn &b);
    fn operator-(const fn &b);
    fn operator*(const fn &b);
    fn operator/(const fn &b);
    bool operator==(const fn &b);
    bool operator<(const fn &b);
    bool operator>(const fn &b);
    static fn inv(const fn &a);
    friend std::ostream &operator<<(std::ostream &out, fn const &a);
    friend std::istream &operator>>(std::istream &in, fn &a);
private:
    double array[2];
};
fn operator"" _fn(long double);
#endif //LAB1_FAZZYNUMBER_HPP
```

Fazzy_Number.cpp:

```
#include <iostream>
#include <cassert>
#include "FazzyNumber.hpp"
#include <algorithm>
fn::fn():array{0,0} {};
fn::fn(double a, double b): array{a, b} {};
fn fn::operator+(const fn &b) {
    fn result;
    result.array[0] = array[0] + b.array[0];
    result.array[1] = array[1] + b.array[1];
    return result;
};
fn fn::operator-(const fn &b) {
    fn result;
    result.array[0] = array[0] - b.array[1];
    result.array[1] = array[1] - b.array[0];
    return result;
};
fn fn::operator*(const fn &b) {
```

```

    fn result;
    double t, m, p, k;
    t = array[0] * b.array[0];
    m = array[0] * b.array[1];
    p = array[1] * b.array[0];
    k = array[1] * b.array[1];
    double maximum = std::max(std::max(std::max(t,m), p), k);
    double minimum = std::min(std::min(std::min(t,m), p), k);
    result.array[0] = minimum;
    result.array[1] = maximum;
    return result;
};

fn fn::operator/(const fn &b) {
    assert(b.array[0] * b.array[1] > 0);
    fn result;
    double t, m, p, k;
    t = array[0] / b.array[0];
    m = array[0] / b.array[1];
    p = array[1] / b.array[0];
    k = array[1] / b.array[1];
    double maximum = std::max(std::max(std::max(t,m), p), k);
    double minimum = std::min(std::min(std::min(t,m), p), k);
    result.array[0] = minimum;
    result.array[1] = maximum;
    return result;
    return result;
}

fn fn::inv(const fn &a) {
    fn result;
    assert(a.array[1] * a.array[0] > 0);
    result.array[0] = 1 / a.array[1];
    result.array[1] = 1 / a.array[0];
    if (result.array[0] > result.array[1]){
        std::swap(result.array[0], result.array[1]);
    }
    return result;
}

bool fn::operator==(const fn &b) {
    return ((array[0] + array[1])/2 == (b.array[0] + b.array[1])/2);
}

bool fn::operator<(const fn &b) {
    if ((array[0] + array[1])/2 < (b.array[0] + b.array[1])/2) {
        return true;
    }
}

```

```

    } else {
        return false;
    }
}

bool fn::operator>(fn const &b) {
    if ((array[0] + array[1])/2 > (b.array[0] + b.array[1])/2) {
        return true;
    } else {
        return false;
    }
}

std::ostream &operator<<(std::ostream &out, fn const &a) {
    out << a.array[0] << " " << a.array[1] << std::endl;
}

std::istream &operator>>(std::istream &in, fn &a) {
    in >> a.array[0] >> a.array[1];
}

fn operator"" _fn(long double op) {
    fn result{(double)op, (double)op};
    return result;
}

```

CmakeLists.txt:

```

project(lab2)
cmake_minimum_required(VERSION 3.2)
add_executable(lab2 lab2.cpp FuzzyNumber.cpp FuzzyNumber.hpp cmake-build-
debug/test_00.txt cmake-build-debug/test_01.txt cmake-build-debug/test_02.txt
cmake-build-debug/test_03.txt cmake-build-debug/test_04.txt)
set_property(TARGET lab2 PROPERTY CXX_STANDARD 11)

```

2. Ссылка на репозиторий на GitHub

https://github.com/IlCher/ooop_exercise_02

3. Набор testcases.

test_00:
-100.123 100.125
900.534 990.12333

test_01:
0.12976 99.654
883.3123 10.2123

test_02:
2 33.125

0.1242312321 90000.324234

test_03:

1200 330.125

0.1242312321 3453543.657765

test_04:

500 12

12 500

4. Результаты выполнения тестов.

test_00:

a:

Assertion failed: a.array[1] * a.array[0] > 0, file
D:\Study\Labs2course\oop_exercise_02\FazzyNumber.cpp, line 49

-100.123 100.125

b:

900.534 990.123

a + b:

800.411 1090.25

a - b

-1090.25 -800.409

a * b

-99134.1 99136.1

a / b

-0.111182 0.111184

test_01:

a:

0.12976 99.654

b:

883.312 10.2123

a + b:

883.442 109.866

a - b

-10.0825 -783.658

a * b

1.32515 88025.6

a / b

0.000146902 9.75823

1 / a

0.0100347 7.70654

a = b ? 0

$a < b ? 1$

$a > b ? 0$

c:

2.5 2.5

test_02:

a:

2 33.125

b:

0.124231 90000.3

a + b:

2.12423 90033.4

a - b

-89998.3 33.0008

a * b

0.248462 2.98126e+006

a / b

2.22221e-005 266.64

1 / a

0.0301887 0.5

$a = b ? 0$

$a < b ? 1$

$a > b ? 0$

c:

2.5 2.5

test_03:

a:

1200 330.125

b:

0.124231 3.45354e+006

a + b:

1200.12 3.45387e+006

a - b

-3.45234e+006 330.001

a * b

41.0118 4.14425e+009

a / b

9.55902e-005 9659.41

1 / a

0.000833333 0.00302916

$a = b ? 0$

$a < b ? 1$

$a > b ? 0$

c:

2.5 2.5

test_04:

a:

500 12

b:

12 500

a + b:

512 512

a - b

0 0

a * b

144 250000

a / b

0.024 41.6667

1 / a

0.002 0.0833333

a = b ? 1

a < b ? 0

a > b ? 0

c:

2.5 2.5

5. Объяснение результатов работы программы.

- 1) При запуске программы с аргументом test_?.txt объекты a, b в основной программе получают данные из файлов test_?.txt.
- 2) Считываются данные для объектов a и b через перегрузку оператора ввода «>>>».
- 3) Вывод данных объектов a, b в стандартный поток вывода через перегрузку оператора вывода «<<<» (кроме того, в дальнейшем любой вывод данных объектов будет проводиться с помощью этой перегрузки).
- 4) Перегружаются операторы «+», «-», «*», «/» и выводятся соответственно результаты сложения, вычитания, произведения и деления объектов a и b.
- 5) Объект a инвертируется и результат инвертирования выводится на стандартный поток вывода.
- 6) Перегружается оператор «==» и в случае равенства объектов a и b выводится «1», иначе «0».
- 7) Перегружается оператор «<» и в случае, если объект a меньше объекта b, то выводится «1», иначе «0».
- 8) Перегружается оператор «>» и в случае, если объект a больше объекта b, то выводится «1», иначе «0».
- 9) Выводится объект c, значения которого вводятся с помощью пользовательского литерала _fn.

6. Вывод.

Выполняя данную лабораторную я получил опыт работы с простыми классами, перегрузкой операторов и пользовательскими литералами. Создал класс, пользовательский литерал и перегрузил операторы, соответствующие варианту моего задания. Пользовательские литералы и перегрузка операторов являются весьма удобными инструментами, которые позволяют совершать с пользовательскими классами те же операции что и со стандартными типами данных.