Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа
по курсу «Объектно-ориентированное программирование»
III Семестр

Задание 2
Вариант 4
Операторы, литералы

Студент:	Синявский А.В
Группа:	М80-208Б-18
Преподаватель:	Журавлёв А.А
Оценка:	
Дата:	

Москва 2019

1. Код программы на языке C++
1.1 main.cpp

```
#include <iostream>
#include "FazzyNumber.h"

int main()
{
    double tmp_l, tmp_r;

    std::cout << "Ввод левой и правой границ отрезка А:" << '\n';
    std::cin >> tmp_l;
    std::cin >> tmp_r;
    Fazzy A(tmp_l, tmp_r);

    std::cout << "Ввод левой и правой границ отрезка Б:" << '\n';
    std::cin >> tmp_l;
    std::cin >> tmp_r;
    Fazzy B(tmp_l, tmp_r);

    Fazzy res1;
    res1 = A + B;
    std::cout << "A+B: " << res1;

    Fazzy res2;
    res2 = A - B;
    std::cout << "A-B: " << res2;

    Fazzy res3;
    res3 = A * B;
    std::cout << "A*B: " << res3;

    Fazzy res4;
    res4 = A / B;
    std::cout << "A/B (если возможно): " << res4;

    Fazzy res5;
    res5 = ~A;
    std::cout << "A^-1 (если возможно): " << res5;

    bool res6;
    res6 = A > B;
    std::cout << std::boolalpha;
    std::cout << "A > B (?): " << res6 << '\n';

    bool res7;
    res7 = A < B;
    std::cout << std::boolalpha;
    std::cout << "A < B (?): " << res7 << '\n';

    bool res8;
    res8 = A == B;
```

```
    std::cout << std::boolalpha;
    std::cout << "A == B (?): " << res8 << '\n';

    Fazzy res9;
    res9 = "[-102.00195:-234664]"_seg;
    std::cout << res9 << '\n';

    return 0;
}
```

1.2 FazzyNumber.cpp

```
#include <iostream>
#include <cstring>
#include <math.h>
#include "FazzyNumber.h"


Fazzy::Fazzy(double l, double r)
{
    if (l <= r) {
        this->l = l;
        this->r = r;
    } else {
        this->l = r;
        this->r = l;
    }
}


Fazzy::Fazzy()
{
    this->l = 0;
    this->r = 0;
}

Fazzy operator ""_seg(const char *str, size_t size) //[12.5:16.785]
{
    double left = 0;
    double right = 0;
    double *num;
    num = &left;
    char c;
    int count = 0;
    bool flag = false, sign_flag = false;
    for (size_t i = 1; i < size-1; i++)
    {
        c = str[i];
        if (c == ':') {
            *num = *num / pow(10, count);
            if (sign_flag) *num = -*num;
            num = &right;
            count = 0;
```

```
            flag = false;
        } else if (c == '-') {
            sign_flag = true;
        } else if (c == '.') {
            flag = true;
        } else {
            if (flag) count++;
            *num = *num * 10 + (c - '0');
        }
    }
    *num = *num / pow(10, count);
    if (sign_flag) *num = -*num;
    return {left, right};
}


Fazzy operator+(const Fazzy& A, const Fazzy& B) //образец
{
    return {A.l + B.l, A.r + B.r};
}

Fazzy operator-(const Fazzy& A, const Fazzy& B)
{
    return {A.l - B.r, A.r - B.l};
}

Fazzy operator*(const Fazzy& A, const Fazzy& B)
{
    return {A.l * B.l, A.r * B.r};
}

Fazzy operator/(const Fazzy& A, const Fazzy& B)
{
    if ((B.l != 0) & (B.r != 0)) {
        return {A.l / B.r, A.r / B.l};
    } else {
        std::cout << "Can`t divide by zero\n";
        return {0, 0};
    }
}

Fazzy operator~(const Fazzy& A)
{
    if ((A.l != 0) & (A.r != 0))
    {
        return {1 / A.r, 1 / A.l};
    } else if ((A.l == 0) & (A.r != 0)) {
        return {A.r, 1 / A.l};
    } else if ((A.l != 0) & (A.r == 0)) {
        return {1 / A.r, A.l};
    } else return {0, 0};
}
```

```
bool operator>(const Fazzy& A, const Fazzy& B)
{
   double mid1, mid2;
   mid1 = A.l + (A.r - A.l) / 2;
   mid2 = B.l + (B.r - B.l) / 2;
   return mid1 > mid2;
}

bool operator<(const Fazzy& A, const Fazzy& B)
{
   double mid1, mid2;
   mid1 = A.l + (A.r - A.l) / 2;
   mid2 = B.l + (B.r - B.l) / 2;
   return mid1 < mid2;
}

bool operator==(const Fazzy& A, const Fazzy& B)
{
   double mid1, mid2;
   mid1 = A.l + (A.r - A.l) / 2;
   mid2 = B.l + (B.r - B.l) / 2;
   return mid1 == mid2;
}

std::ostream &operator<<(std::ostream &os, const Fazzy& A)
{
  os << "[" << A.l << "; " << A.r << "]" << '\n';
  return os;
}
```

2. Ссылка на репозиторий на GitHub

https://github.com/Siegmeyer1/oop_exercise_02

3. Набор тестов

1) 0
   2
   4
   8

2) 3
   4
   0
   9

3) 9.5
13
1.4
7.8

4) 2
1
6
8

4. Результат выполнения тестов

1)
A+B: [4; 10]
A-B: [-8; -2]
A*B: [0; 16]
A/B (если возможно): [0; 0.5]
A^-1 (если возможно): [0; 2]
A > B (?): false
[-234664; -102.002]

2)
A+B: [3; 13]
A-B: [-6; 4]
A*B: [0; 36]
A/B (если возможно): [3; 4]
A^-1 (если возможно): [0.25; 0.333333]
A > B (?): false
[-234664; -102.002]

3)
A+B: [10.9; 20.8]
A-B: [1.7; 11.6]
A*B: [13.3; 101.4]
A/B (если возможно): [1.21795; 9.28571]
A^-1 (если возможно): [0.0769231; 0.105263]
A > B (?): true
[-234664; -102.002]

4)
A+B: [7; 10]
A-B: [-7; -4]
A*B: [6; 16]
A/B (если возможно): [0.125; 0.333333]
A^-1 (если возможно): [0.5; 1]
A > B (?): false
[-234664; -102.002]

5. Объяснение работы программы

Данная программа состоит из двух cpp файлов.

Первый - FazzyNumber, в котором создаётся одноимённый класс для работы с отрезками. Отрезок представлен двумя полями типа double – левым и правым концами отрезка. При помощи методов класса реализованы: сложение, вычитание, умножение и деление (при возможности) отрезков, а также операция обращения, возвращающая отрезок, обратный данному и операция сравнения двух отрезков, определяющая, больше ли первый отрезок, чем второй. Все операции сделаны через перегрузку операторов. Так же добавлен пользовательский литерал, позволяющий удобно задавать константы типа объекта класса. Он посимвольно считывает строку и “прибавляет” новую цифру к числу. Переход от целой к дробной части, от левой к правой границе отрезка и учёт знака при переводе чисел из строки в тип double осуществляется при помощи флагов.

Второй файл – main, запрашивает ввод границ отрезков и проделывает с ними все заданные операции. Конструктор при вызове определяет, в правильном ли порядке были введены границы отрезка, и, в случае ошибки, меняет их местами. Вывод осуществляется при помощи метода write, определённого в первом файле.

Вывод

Проделав данную работу, я научился перегружать операторы, а также создавать пользовательские литералы. Благодаря этой и предыдущей работе я стал глубже понимать устройство и синтаксис языка C++.