# **Лабораторная работа № 01**

**Студент**: Сазонов Вадим Кириллович , м8О-201Б-20

# **Тема: Простые классы**

**Цель:**

* Изучение системы сборки на языке C++, изучение систем контроля версии.
* Изучение основ работы с классами в С++;

## **Порядок выполнения работы**

1. Ознакомиться с теоретическим материалом.

2. Получить у преподавателя вариант задания.

3. Реализовать задание своего варианта в соответствии с поставленными требованиями.

4. Подготовить тестовые наборы данных.

5. Создать репозиторий на GitHub.

6. Отправить файлы лабораторной работы в репозиторий.

7. Отчитаться по выполненной работе путём демонстрации работающей программы на тестовых наборах данных (как подготовленных самостоятельно, так и предложенных преподавателем) и ответов на вопросы преподавателя (как из числа контрольных, так и по реализации программы).

## **Требования к программе**

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Необходимо настроить сборку лабораторной работы с помощью CMake. Собранная программа должна называться **oop\_exercise\_01** (в случае использования Windows **oop\_exercise\_01.exe)**

Необходимо зарегистрироваться на GitHub (если студент уже имеет регистрацию на GitHubто можно использовтаь ее) и создать репозитарий для задания лабораторной работы.

Преподавателю необходимо предъявить ссылку на публичный репозиторий на Github. Имя репозитория должно быть [https://github.com/*login*/oop\_exercise\_01](https://github.com/login/oop_exercise_01)

Где login – логин, выбранный студентом для своего репозитория на Github.

Репозиторий должен содержать файлы:

· main.cpp //файл с заданием работы

· CMakeLists.txt // файл с конфигураций CMake

· test\_xx.txt // файл с тестовыми данными. Где xx – номер тестового набора 01, 02 , … Тестовых наборов должно быть больше 1.

· report.doc // отчет о лабораторной работе

**Вариант 4:**

**Реализовать класс FazzyNumber** для работы с нечеткими числами, которые представляются тройками чисел (x – el, x, x + er). Для чисел A = (A – al, A, A + ar) и B = (B – bl, B, B + br) арифметические операции выполняются по следующим формулам:

- сложение *A* + *B* = (*A* + *B* – *a*l – *b*l, *A* + *B*, *A* + *B* + *ar* + *br*);

- вычитание *A* – *B* = (*A* – *B* – *a*l – *b*l, *A* – *B*, *A* – *B* + *ar* + *br*);

- умножение *A* · *B* = (*A* ´ *B* – *B* ´ *a*l – *A* ´ *b*l + *a*l ´ *b*l, *A* ´ *B*, *A* ´ *B* + *B* ´ *a*l + *A* ´ *b*l + *a*l ´ *b*l);

- обратное число *A* = (1 / (*A* + *ar*), 1 / *A*, 1 / (*A* – *a*l)), *A* > 0;

- деление *A* / *B* = ((*A* – *a*l) / (*B* + *br*), *A* / *B*, (*A* + *ar*) / (*B* – *b*l)), *B* > 0;

Считать el = er, то есть число представлено парой <*x*, *e*>.

Реализовать операции сравнения по х.

Код программы:

#include <iostream>

class FazzyNumber

{

private:

float xl;

float x;

float xr;

public:

FazzyNumber(float x, float e) : xl(x - e), x(x), xr(x + e){};

FazzyNumber() : xl(0), x(0), xr(0){};

~FazzyNumber(){};

friend FazzyNumber add(const FazzyNumber &a, const FazzyNumber &b)

{

FazzyNumber res;

res.xl = a.x + b.x - a.xl - b.xl;

res.x = a.x + b.x;

res.xr = a.x + b.x + a.xr + b.xr;

return res;

}

friend FazzyNumber subtract(const FazzyNumber &a, const FazzyNumber &b)

{

FazzyNumber res;

res.xl = a.x - b.x - a.xl - b.xl;

res.x = a.x - b.x;

res.xr = a.x - b.x + a.xr + b.xr;

return res;

}

friend FazzyNumber multiply(const FazzyNumber &a, const FazzyNumber &b)

{

FazzyNumber res;

res.xl = a.x \* b.x - b.x \* a.xl - a.x \* b.xl + a.xl \* b.xl;

res.x = a.x \* b.x;

res.xr = a.x \* b.x + b.x \* a.xl + a.x \* b.xl + a.xl \* b.xl;

return res;

}

friend FazzyNumber divide(const FazzyNumber &a, const FazzyNumber &b)

{

FazzyNumber res;

res.xl = (a.x - a.xl) / (b.x + b.xr);

res.x = a.x / b.x;

res.xr = (a.x + a.xr) / (b.x - b.xl);

return res;

}

friend FazzyNumber reverse(const FazzyNumber &a)

{

FazzyNumber res;

res.xl = 1 / (a.x + a.xr);

res.x = 1 / a.x;

res.xr = 1 / (a.x - a.xl);

return res;

}

friend bool operator>(const FazzyNumber &a, const FazzyNumber &b)

{

if (a.x > b.x)

return true;

return false;

}

friend bool operator<(const FazzyNumber &a, const FazzyNumber &b)

{

if (a.x > b.x)

return false;

return true;

}

friend bool operator==(const FazzyNumber &a, const FazzyNumber &b)

{

if (a.x == b.x)

return true;

return false;

}

friend std::ostream &operator<<(std::ostream &out, const FazzyNumber &a)

{

out << "( " << a.xl << " ), ( " << a.x << " ), (" << a.xr << ")\n";

return out;

}

};

int main()

{

int a,b,c,d;

std::cin>>a>>b>>c>>d;

FazzyNumber first(a, b);

FazzyNumber second(c, d);

FazzyNumber res;

std::cout << "First " <<first;

std::cout << "Second "<<second;

res = add(first, second);

std::cout << "Adding: " << res;

res = subtract(first, second);

std::cout << "Subtracting: " << res;

res = multiply(first, second);

std::cout << "Mupltiply: " << res;

res = divide(first, second);

std::cout << "Divide: " << res;

first=reverse(first);

std::cout<<"Reversed first "<<first;

return 0;

}

Тесты:

1. 3 2 1 7
2. 6 9 8 4

Результат тестов:

1. First ( 1 ), ( 3 ), (5)

Second ( -6 ), ( 1 ), (8)

Adding: ( 9 ), ( 4 ), (17)

Subtracting: ( 7 ), ( 2 ), (15)

Mupltiply: ( 14 ), ( 3 ), (-20)

Divide: ( 0.222222 ), ( 3 ), (1.14286)

Reversed first ( 0.125 ), ( 0.333333 ), (0.5)

2. First ( -3 ), ( 6 ), (15)

Second ( 4 ), ( 8 ), (12)

Adding: ( 13 ), ( 14 ), (41)

Subtracting: ( -3 ), ( -2 ), (25)

Mupltiply: ( 36 ), ( 48 ), (36)

Divide: ( 0.45 ), ( 0.75 ), (5.25)

Reversed first ( 0.047619 ), ( 0.166667 ), (0.111111)