МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №0.1 по курсу

объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Васютинский Вадим Адександрович, группа М80-208Б-20*
Преподаватель *Дорохов Евгений Павлович*

Цель:

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

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

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

6. Создать класс Rational для работы с действительными числами.Рациональная (несократимая) дробь представляется парой целых чисел (a, b), где a — числитель, b — знаменатель.

Описание программы

Исходный код лежит файле:

main.cpp - исполняемый код.

Дневник отладки

Во время выполнения лабораторной работы программа не нуждалась в отладке, все ошибки компиляции были исправлены с первой попытки.

Недочёты

Недочётов не было обнаружено.

Выводы

В процессе выполнения работы я на практике познакомился с классами. Благодаря им, упрощается написание кода для различных объемных программ, использующих различные типы данных, содержащие сразу несколько различных полей. Например, при необходимости использовать тип данных, соответствующий адресу дома, вместо хранения трех различных полей в программе, можно создать структуру типа адреса и использовать ее.

Исходный код

main.cpp

```
#include <iostream>
#include <exception>

class Rational {
public:
```

```
    Rational(int a, int b): a(a), b(b) {
        if (b == 0) {
            b = 1;
        }
        reduce();}
    Rational(Rational& other): a(other.a), b(other.b) {reduce();}
    void reduce() {
        int c = a;
        int d = b;
        while (c != d) {
            if (c > d) {
                c -= d;
            } else {
                d -= c;
            }
        }
        a /= d;
        b /= d;
    }
    void add(Rational& r) {
        a = a * r.b + b * r.a;
        b *= r.b;
        reduce();
    }
    void sub(Rational& r) {
        a = a * r.b - b * r.a;
        b *= r.b;
        reduce();
    }
    void mul(Rational& r) {
        a *= r.a;
        b *= r.b;
        reduce();
    }
    void div(Rational& r) {
        a *= r.b;
        b *= r.a;
        reduce();
    }
    int a, b;
};

std::ostream& operator<<(std::ostream& os, const Rational& rat) {
    std::cout << rat.a << "/" << rat.b << std::endl;
    return os;
```

```
}


int main() {
    Rational a(1, 2);
    Rational b(1, 3);
    a.add(b);
    std::cout << a << std::endl;
    return 0;
}
```