МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №1 по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Зинин Владислав Владимирович, 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 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

Описание программы

Исходный код лежит в 3 файлах:

1. main.cpp - исполняемый код.

2. Modulo.h - специальный файл .h, содержащий прототипы используемых мною функций.

3. Modulo.cpp - реализация функций для моего задания.

4. CMakeLists.txt - специальный дополнительный файл типа CMakeLists.

**Дневник отладки**

Во время выполнения лабораторной работы программа не нуждалась в отладке, все ошибки компиляции были исправлены с первой попытки. После их исправления программа работала так, как было задумано изначально.

**Недочёты**  
Недочётов не было обнаружено.

**Выводы**

Данная лабораторная работа помогла мне использовать полученные на лекциях теоретические знания на практике, и я написал простенький полностью работающий класс.

**Исходный код**

**Modulo.cpp**

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include "Modulo.h" |
|  |

|  |
| --- |
| #include <cmath> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Modulo::Modulo(){ |
|  |

|  |
| --- |
| value = 1; |
|  |

|  |
| --- |
| N = 1; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Modulo::Modulo(std::istream &is){ |
|  |

|  |
| --- |
| is >> value; |
|  |

|  |
| --- |
| is >> N; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Modulo::Modulo(int first, int second){ |
|  |

|  |
| --- |
| value = first; |
|  |

|  |
| --- |
| N = second; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int Modulo::operator +(Modulo& a){ |
|  |

|  |
| --- |
| return this->value%this->N + a.value%a.N; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int Modulo::operator -(Modulo& a){ |
|  |

|  |
| --- |
| return this->value%this->N - a.value%a.N; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int Modulo::operator \*(Modulo& a){ |
|  |

|  |
| --- |
| return (this->value%this->N) \* (a.value%a.N); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int Modulo::operator /(Modulo& a){ |
|  |

|  |
| --- |
| return (this->value%this->N) / (a.value%a.N); |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Modulo Modulo::operator ++(){ |
|  |

|  |
| --- |
| this->N++; |
|  |

|  |
| --- |
| this->value++; |
|  |

|  |
| --- |
| return \*this; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Modulo Modulo::operator --(){ |
|  |

|  |
| --- |
| this->N--; |
|  |

|  |
| --- |
| this->value--; |
|  |

|  |
| --- |
| return \*this; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| std::ostream& operator<<(std::ostream& os,const Modulo& a){ |
|  |

|  |
| --- |
| os << a.value << " " << a.N << std::endl; |
|  |

|  |
| --- |
| return os; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| bool Modulo::operator==(const Modulo& other){ |
|  |

|  |
| --- |
| return this->N == other.N && this->value == other.value; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| Modulo::~Modulo(){ |
|  |

|  |
| --- |
| std::cout << "Modulo has deleted" << std::endl; |
|  |

}

**Modulo.h**

|  |
| --- |
| #ifndef MODULO\_H |
|  |

|  |
| --- |
| #define MODULO\_H |
|  |

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| class Modulo { |
|  |

|  |
| --- |
| public: |
|  |

|  |
| --- |
| Modulo(); |
|  |

|  |
| --- |
| Modulo(std::istream &is); |
|  |

|  |
| --- |
| Modulo(int value, int N); |
|  |

|  |
| --- |
| int operator +(Modulo& a); |
|  |

|  |
| --- |
| int operator -(Modulo& a); |
|  |

|  |
| --- |
| int operator \*(Modulo& a); |
|  |

|  |
| --- |
| int operator /(Modulo& a); |
|  |

|  |
| --- |
| Modulo operator ++(); |
|  |

|  |
| --- |
| Modulo operator --(); |
|  |

|  |
| --- |
| bool operator ==(const Modulo& other); |
|  |

|  |
| --- |
| friend std::ostream& operator<<(std::ostream& os,const Modulo& a); |
|  |

|  |
| --- |
| ~Modulo(); |
|  |

|  |
| --- |
| private: |
|  |

|  |
| --- |
| int value; |
|  |

|  |
| --- |
| int N; |
|  |

|  |
| --- |
| }; |
|  |

|  |
| --- |
|  |
|  |

#endif // MODULO\_H

**Main.cpp**

|  |
| --- |
| #include <iostream> |
|  |

|  |
| --- |
| #include "Modulo.h" |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| int main(){ |
|  |

|  |
| --- |
| Modulo c(std::cin); |
|  |

|  |
| --- |
| Modulo a(10, 6); |
|  |

|  |
| --- |
| Modulo b(12, 5); |
|  |

|  |
| --- |
| std::cout << "Modulo objects"<< a << b << c << std::endl; |
|  |

|  |
| --- |
| std::cout << "Sum: " << a+b << std::endl; |
|  |

|  |
| --- |
| std::cout << "Division of residues " << a/b << std::endl; |
|  |

|  |
| --- |
| std::cout << "Multiplication of residuals " << a\*b << std::endl; |
|  |

|  |
| --- |
| std::cout << "Sum " << c+b << std::endl; |
|  |

|  |
| --- |
| std::cout << "Operator -- : " << --a; |
|  |

|  |
| --- |
| std::cout << "Operator ++ : " << ++a; |
|  |

}