Московский Авиационный Институт
(Национальный Исследовательский Университет)

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

Лабораторная работа
Дисциплина: «Объектно-ориентированное программирование»
I I семестр
Задание 1: «Простые классы»

Группа:	М8О-108Б-18, №5
Студент:	Валов Вадим Вячеславович
Преподаватель:	Журавлёв Андрей Андреевич
Оценка:	
Дата:	

Москва, 2019

1. Задание

(*вариант № 12*): Создать класс Modulo для работы с целыми числами по модулю N. В классе должно быть два поля: число и N. Реализовать все арифметические операции. Реализовать операции сравнения.

2. Адрес репозитория на GitHub

https://github.com/vindosVP/oop_exercise_01

3. Код программы на C++

modulo.cpp

```
#include <iostream>
#include <cassert>

#include "modulo.hpp"

int ExtendedEuclid(int a, int b, int& x, int& y) {
    if(a == 0) {
        x = 0;
        y = 1;
        return b;
    }
    int x1, y1;
    int gcd = ExtendedEuclid(b % a, a, x1, y1);
    x = y1 - (b / a) * x1;
    y = x1;
    return gcd;
}

Modulo Modulo::Add(const Modulo& addend) const {
    assert(mod == addend.mod);
    Modulo result;
    result.number = (number + addend.number)%mod;
    result.mod = mod;
    return result;
}

Modulo Modulo::Multiply(const Modulo& multiplier) const {
```

```
    assert(mod == multiplier.mod);
    Modulo result;
    result.number = (number * multiplier.number)%mod;
    result.mod = mod;
    return result;
}

Modulo Modulo::Subtract(const Modulo& subtrahend) const {
    assert(mod == subtrahend.mod);
    Modulo result;
    result.number = (number % mod - subtrahend.number % mod + mod) % mod;
    result.mod = mod;
    return result;
}

Modulo Modulo::Divide(const Modulo& divisor) const {
    assert(mod == divisor.mod);
    int x, y;
    if(ExtendedEuclid(divisor.number, mod, x, y) != 1) {
        std::cerr << "Divisor and aren't coprime, therefore division can't be made" <<
std::endl;
        return {number, 0};
    }
    Modulo result;
    int ModInverse = (x % mod + mod) % mod;
    result.number = (number * ModInverse) % mod;
    result.mod = mod;
    return result;
}

void Modulo::Read(std::istream& is) {
    is >> number >> mod;
    if(number % mod >= 0) {
        number %= mod;
    } else {
        number = mod + (number % mod);
    }
}

void Modulo::Print(std::ostream& os) const {
    os << number << " mod " << mod << std::endl;
}

void Modulo::SetNumber(int number) {
    this->number = number;
```

```
}

void Modulo::SetMod(int mod) {
   this->mod = mod;
}

int Modulo::GetNumber() const {
   return number;
}

int Modulo::GetMod() const {
   return mod;
}

bool Modulo::IsEqual(const Modulo& to_compare) const {
   assert(mod == to_compare.mod);
   return number == to_compare.number;
}

bool Modulo::IsGreater(const Modulo& to_compare) const {
   assert(mod == to_compare.mod);
   return number > to_compare.number;
}

bool Modulo::IsLess(const Modulo& to_compare) const {
   assert(mod == to_compare.mod);
   return number < to_compare.number;
}
```

modulo.hpp

```
#ifndef _MODULO_H_
#define _MODULO_H_


#include <iostream>


class Modulo {
   public:
      Modulo() : number(0), mod(0) {}
      Modulo(int number, int mod) : number(number < 0 ? mod + (number %
mod) : number % mod), mod(mod) {}
      Modulo Add(const Modulo& addend) const;
      Modulo Multiply(const Modulo& multiplier) const;
```

```
        Modulo Subtract(const Modulo& subtracthend) const;
        Modulo Divide(const Modulo& divisor) const;
        void Read(std::istream& is);
        void Print(std::ostream& os) const;
        void SetNumber(int number);
        void SetMod(int mod);
        int GetNumber() const;
        int GetMod() const;
        bool IsEqual(const Modulo& to_compare) const;
        bool IsGreater(const Modulo& to_compare) const;
        bool IsLess(const Modulo& to_compare) const;
    private:
        int number;
        int mod;
};

#endif
```

main.cpp

```
#include <iostream>

#include "modulo.hpp"

int main() {
    Modulo a;
    Modulo b;
    Modulo c;

    a.Read(std::cin);
    b.Read(std::cin);

    std::cout << "Addition:" << std::endl;
    c = a.Add(b);
    c.Print(std::cout);

    std::cout << "Subtraction:" << std::endl;
    c = a.Subtract(b);
    c.Print(std::cout);

    std::cout << "Multiplication:" << std::endl;
    c = a.Multiply(b);
    c.Print(std::cout);

    std::cout << "Division:" << std::endl;
```

```
    c = a.Divide(b);
    if(c.GetMod()) {
        c.Print(std::cout);
    }

    if(a.IsEqual(b)) {
        std::cout << "Numbers are equal" << std::endl;
    }

    if(a.IsGreater(b)) {
        std::cout << "First number is greater" << std::endl;
    }

    if(a.IsLess(b)) {
        std::cout << "First number is less" << std::endl;
    }

    return 0;
}
```

CmakeLists.txt

```
cmake_minimum_required(VERSION 2.8) # Проверка версии CMake.
# Если версия установленой программы
# старее указаной, произайдёт аварийный выход.

project(hello_world) # Название проекта

set(SOURCE_EXE main.cpp) # Установка переменной со списком исходников для исполняемого файла

set(SOURCE_LIB modulo.cpp) # Тоже самое, но для библиотеки

add_library(modulo STATIC ${SOURCE_LIB}) # Создание статической библиотеки с именем foo

add_executable(main ${SOURCE_EXE}) # Создает исполняемый файл с именем main

target_link_libraries(main modulo)
```

Test01.txt

```
6
3
2
```

3

Test02.txt

9
4
3
4

Test03.txt

6
3
2
3

4. Результаты выполнения тестов

Result_test01.txt

Addition:
2 mod 3
Subtraction:
2 mod 3
Multiplication:
0 mod 3
Division:
0 mod 3
First number is less

Result_test02.txt

Addition:
0 mod 4
Subtraction:
2 mod 4
Multiplication:
3 mod 4
Division:
3 mod 4
First number is less

Result_test03.txt

Addition:
2 mod 3
Subtraction:
1 mod 3
Multiplication:
0 mod 3
Division:
0 mod 3
First number is less

Объяснение работы программы

Данная программа создает класс Modulo для работы с целыми числами по модулю N. В классе есть быть два поля: число и N.

В программе раелизованы следующие операции:
1. Сложение и вычитание по модулю N
2. Умножение и деление по модулю N
3. Сравнение чисел по модулю N.

5. Вывод

Проделав данную работу я изучил классы и научился их применять. Сделал вывод, что класс — очень удобная структура, позволяющая представлять свойства объекта и предоставляющая удобную работу с ними.