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

ЛАБОРАТОРНАЯ РАБОТА №2 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 основ работы с классами в C++;
- Перегрузка операций и создание литералов

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.

Реализовать над объектами реализовать в виде перегрузки операторов.

Реализовать пользовательский литерал для работы с константами объектов созданного класса.

Описание программы

Исходный код лежит в 3 файлах:

1. main.cpp - исполняемый код.
2. TimePoint.h - специальный файл .h, содержащий прототипы используемых мною функций.
3. TimePoint.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

Лабораторная работа №2 - это, по сути, та же самая лабораторная №1, только предусматривающая возможность перегрузки операторов. Лабораторная была выполнена успешно, в ее процессе были еще раз осознаны основные принципы ООП и перегрузки операторов.

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"

Modulo operator "" _classmodulo(const char* str, size_t size){
int cnt = 0;
std::string s = "";
while (str[cnt] != ' '){
s += str[cnt++];
}
double r = 0, j = 0;
for (int i = 0; i < s.size(); ++i) {
r *= 10;
r += s[i] - '0';
}
s = "";
while (str[cnt++] != '\0') {
s += str[cnt];
}
for (int i = 0; i < s.size() - 1; ++i) {
j *= 10;
j += s[i] - '0';
}
Modulo g(r, j);
return g;
```

```
}

unsigned long long operator "" _minusone(unsigned long long a){
a -=1;
return a;
}

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;
std::cout << "Literal class Modulo+: " << "12 5"_classmodulo << std::endl;
std::cout << "Literal--: " << 12_minusone << std::endl;

}
```