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 *Москвин Артём Артурович, 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. position.h - специальный файл .h, содержащий прототипы используемых мною функций.

3. position.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**

position.h

#ifndef LAB0\_1\_POSITION\_H

#define LAB0\_1\_POSITION\_H

#include "math.h"

#include <iostream>

class Position{

private:

   int latitude, longitude; // широта и долгота

public:

    Position(int latitude, int longitude);

    Position static Sum(Position a, Position b);

    Position static Diff(Position a, Position b);

    Position static Mult(Position a, Position b);

    Position static Substr(Position a, Position b);

    void static Compare(Position a, Position b);

    void print();

    friend bool operator ==(Position a, Position b);

};

#endif //LAB0\_1\_POSITION\_H

position.cpp   
#include "position.h"

Position::Position(int latitude, int longitude) {

    this->latitude = latitude;

    this->longitude = longitude;

}

Position Position::Sum(Position a, Position b) {

    int newLatitude = a.latitude + b.latitude;

    int newLongitude = a.longitude + b.longitude;

    if(newLatitude > 90 || newLatitude < -90){

        newLatitude = 180 - abs(newLatitude);

    }

    if(newLongitude < -180){

        newLongitude = 360 + newLongitude;

    }

    else if(newLongitude > 180){

        newLongitude = newLongitude - 360;

    }

    return Position(newLatitude, newLongitude);

}

Position Position::Diff(Position a, Position b) {

    int newLatitude = a.latitude - b.latitude;

    int newLongitude = a.longitude - b.longitude;

    if(newLatitude > 90 || newLatitude < -90){

        newLatitude = 180 - abs(newLatitude);

    }

    if(newLongitude < -180){

        newLongitude = 360 + newLongitude;

    }

    else if(newLongitude > 180){

        newLongitude = newLongitude - 360;

    }

    return Position(newLatitude, newLongitude);

}

Position Position::Mult(Position a, Position b) {

    int newLatitude = a.latitude \* b.latitude;

    int newLongitude = a.longitude \* b.longitude;

    if (newLatitude > 90){

        newLatitude %= 90;

    }

    else if (newLatitude < -90) {

        newLatitude \*= -1;

        newLatitude %= 90;

        newLatitude \*= -1;

    }

    if (newLongitude > 180) {

        newLongitude %= 180;

    }

    else if (newLongitude < -180) {

        newLongitude \*= -1;

        newLongitude %= 180;

        newLongitude \*= -1;

    }

    return Position(newLatitude, newLongitude);

}

Position Position::Substr(Position a, Position b) {

    int newLatitude = a.latitude / b.latitude;

    int newLongitude = a.longitude / b.longitude;

    return Position(newLatitude, newLongitude);

}

bool operator ==(Position a, Position b) {

    return a.latitude == b.latitude && a.longitude == b.longitude;

}

void Position::Compare(Position a, Position b) {

    if (a == b){

        std::cout << "Positions are equal\n";

    }

    else {

        if (a.latitude == b.latitude) {

            std::cout << "Positions have the same latitude and ";

        }

        else if(a.latitude > b.latitude) {

            std::cout << "First position is northern than second and ";

        }

        else {

            std::cout << "Second position is northern than first and ";

        }

        if (a.longitude == b.longitude) {

            std::cout << "positions have the same longitude\n";

        }

        else if(a.longitude > b.longitude) {

            std::cout << "first position is eastern than second\n";

        }

        else {

            std::cout << "second position is eastern than first\n";

        }

    }

}

void Position::print() {

    std::cout << "latitude, longitude: (" << latitude << "," << longitude << ")\n";

}

main.cpp  
  
#include "position.h"

int main() {

    std::cout << "Enter 1st coordinates (latitude, then longitude):\n";

    int a, b;

    std::cin >> a >> b;

    std::cout << "Enter 2nd coordinates (latitude, then longitude):\n";

    int c, d;

    std::cin >> c >> d;

    std::cout << "Enter 3d coordinates (latitude, then longitude):\n";

    int e, f;

    std::cin >> e >> f;

    Position pos1(a, b);

    Position pos2(c, d);

    Position pos3(e, f);

    Position::Compare(pos1, pos2);

    Position::Compare(pos1, pos3);

    Position sum = Position::Sum(pos1, pos3);

    Position diff = Position::Diff(pos1, pos3);

    Position mult = Position::Mult(pos1, pos3);

    Position substr = Position::Substr(pos1, pos3);

    sum.print();

    diff.print();

    mult.print();

    substr.print();

}