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

ЛАБОРАТОРНАЯ РАБОТА №1

по курсу “Объектно-ориентированное программирование”

I семестр, 2021/22 учебный год

Студент: *Москвин Артём Артурович, группа М8О-208Б-20*

Преподаватель: *Дорохов Евгений Павлович, каф. 806*

Задание:

Разработать программу на языке C++ согласно варианту задания. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Вариант №15:

Создать класс **TransNumber** для работы с трансцендентными числами. Трансцендентное число представлено парой (a, b), где a – действительная часть, b – трансцендентная часть. Трансцендентная часть представляет собой действительное число b, умноженное на константу. Реализовать арифметические операции (по аналогии с операциями над комплексными числами в алгебраической форме), и операции сравнения по значению (a + b ´).

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:

Программа в отладке не нуждалась, весь необходимый функционал был реализован без всяких заминок.

Вывод:

В процессе выполнения данной лабораторной работы я, можно сказать, познакомился с самим понятием ООП – прочувствовал, что такое классы, осознал отличие класса от структуры, познакомился с понятием “метод класса”, успешно реализовал необходимый функционал для работы. На примере данной лабораторной работы я столкнулся с одним из трех китов ООП – инкапсуляцией. Действительно, в public-зоне у меня лежат все необходимые методы, а в private-зоне, по правилу хорошего тона, лежат 3 переменные по заданию: часы, минуты и секунды.

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();
}
```

Пример работы:

```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();
}
```

```
PS D:\C C++\2 курс\ООП\Lab01> g++ main.cpp position.cpp
PS D:\C C++\2 курс\ООП\Lab01> .\a.exe
Enter 1st coordinates (latitude, then longitude):
1 2
Enter 2nd coordinates (latitude, then longitude):
3 4
Enter 3d coordinates (latitude, then longitude):
5 6
Second position is northern than first and second position is eastern than first
Second position is northern than first and second position is eastern than first
latitude, longitude: (6,8)
latitude, longitude: (-4,-4)
latitude, longitude: (5,12)
latitude, longitude: (0,0)
PS D:\C C++\2 курс\ООП\Lab01>
```