Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної

техніки Кафедра інформатики та програмної

інженерії

Звіт

з лабораторної роботи № 3 з дисципліни
«Основи програмування-2. Методології програмування»
«Перевантаження операторів»

Варіант 06

Виконав студент ІП-13 Вдовиченко Станіслав Юрійович
(шифр, прізвище, ім'я, по батькові)

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота №3

Перевантаження операторів

Мета роботи: вивчити механізми створення класів з використанням перевантажених операторів (операцій)

Завдання: Визначити клас «Відрізок», який задається координатами початку та кінця відрізка. Реалізувати для нього декілька конструкторві, теггери, метод перевірки приналежності заданої точки відрідзку. Перевантажити оператори: «+» — для додавання відрізків згідно правил додавання веторів, «постфіксний ++» — для збільшення координат кінця відрізка на 1. Створити три відрізка (V1, V2, V3), використовуючи різні конструктори. Визначити відрізок V3 як суму відрізків V1 та V2. Збільшити координати кінця відрізка V3 на 1. Перевірити, чи належить задана точка відрізку V3.

Код програми на C++

Line.h

```
#ifndef LAB4_LINE_H
#define LAB4_LINE_H
#include "header.h"
#include "string"
using namespace std;

class Line{
private:
    double start_x;
    double start_y;
    double end_x;
    double end_y;
public:
    Line();
    Line(const string&, const string&);
    Line(Line&);
    Line(double, double, double, double);
    double get_start_x() const;
    double get_start_y() const;
    double get_end_x() const;
    double get_end_y() const;
    void show_line();
    friend Line operator+(Line& line1, Line& line2);
    Line operator++(int);
    bool doesBelong(double x, double y);
};
#endif //LAB4_LINE_H
```

Line.cpp

```
#include "Line.h"
Line::Line(){
    start_x = 0.0;
    start_y = 0.0;
    end_x = 0.0;
    end_y = 0.0;
}
Line::Line(const string& line1, const string& line2) {
    start_x = stod(split(line1, ' ')[0]);
    start_y = stod(split(line1, ' ')[1]);
    end_x = stod(split(line2, ' ')[0]);
    end_y = stod(split(line2, ' ')[1]);
}
Line::Line(Line &line) {
    this->start_x = line.start_x;
    this->start_y = line.start_y;
    this->end_x = line.end_x;
    this->end_y = line.end_y;
}

Line::Line(double x1, double y1, double x2, double y2) {
    start_x = x1;
    start_y = y1;
    end_x = x2;
    end_y = y2;
}

double Line::get_start_x() const {
```

```
    return start_x;
}

double Line::get_start_y() const {
    return start_y;
}

double Line::get_end_x() const {
    return end_x;
}

double Line::get_end_y() const {
    return end_y;
}

Line operator+(Line &line1, Line &line2) {
    Line line(line1.start_x + line2.start_x, line1.start_y +
line2.start_y,line1.end_x + line2.end_x,line1.end_y + line2.end_y);
   return line;
}

Line Line::operator++(int) {
    Line line(*this);
    end_x += 1;
    end_y += 1;
    return line;
}
void Line::show_line() {
    cout << "start(" << start_x << ", " << start_y << ")" << " end(" <<
end_x << ", " << end_y << ")" << endl;
}

bool Line::doesBelong(double x, double y) {
    return ((((x - start_x) * (end_y - start_y)) - ((y - start_y) * (end_x
- start_x)) == 0) && ((x >= start_x && x <= end_x) || (x>= end_x && x <=
start_x)));
}
```

Header.h

```
#ifndef LAB4_HEADER_H
#define LAB4_HEADER_H
#include <iostream>
#include "vector"
#include "string"
#include "Line.h"
using namespace std;

vector<string> split(string, char);
#endif
```

Header.cpp

```
#include "header.h"
```

```
vector<string> split(string line, char sep) {
    vector<string> words;
    string temp_word;
    line += sep;
    for (int i = 0; i < line.length(); i++) {
        if (line[i] == sep) {
            if (temp_word.length() > 0) {
                words.push_back(temp_word);
            }
            temp_word = "";
        }
        else {
            temp_word += line[i];
        }
    }
    return words;
}
```

Main.cpp

```
#include "header.h"

int main() {
    double x1, y1, x2, y2;
    string point1;
    string point2;
    cout << "Enter first line. " << endl;
    cout << "First point: " << endl;
    cout << "X: "; cin >> x1;
    cout << "Y: "; cin >> y1; cout << endl;
    cout << "Second point: " << endl;
    cout << "X: "; cin >> x2;
    cout << "Y: "; cin >> y2; cout << endl;
    cout << "Enter second line. " << endl;
    cin.ignore();
    cout << "Enter first point in format[x y]: ";
    getline(cin, point1);
    cout << "Enter second point in format[x y]: ";
    getline(cin, point2); cout << endl;

    Line V1(x1, y1, x2, y2);
    Line V2(point1, point2);
    Line V3(V1);
    V3 = V1 + V2;
    cout <<"First line: ";  V1.show_line();
    cout << "Second line:"; V2.show_line();
    cout << "Third line: "; V3.show_line();
    V3++;
    cout << "Third line after increment: " ; V3.show_line(); cout << endl;

    cout << "Enter point to check: " << endl;
    cout << "X: "; cin >> x1;
    cout << "Y: "; cin >> y1;
    V3.doesBelong(x1, y1) ? cout << "Point belongs to the line." : cout <<
"Point doesnt belong to the line.";
    return 0;
}
```

Робота програми

```
C:\Users\Stas\CLionProjects\Lab4\cmake-build-debug\Lab4.exe
Enter first line.
First point:
X:4
 Y:5

Second point:
X:6
 Y:8

Enter second line.
Enter first point in format[x y]:10 1
 Enter second point in format[x y]:-4 6

First line: start(4, 5) end(6, 8)
Second line:start(10, 1) end(-4, 6)
Third line: start(14, 6) end(2, 14)
Third line after increment: start(14, 6) end(3, 15)

Enter point to check:
X:14
 Y:6
 Point belongs to the line.
Process finished with exit code 0
```

Висновок: під час виконання даної лабораторної роботи я вивчив механізми створення класів з використанням перевантажених операторів у мові С++.