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-11 Головатюк Владислав Ігорович

(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.

1. **С#**
   1. **Код:**
      1. **Program.cs**

using System;

using System.Linq;

namespace Lab4

{

internal class Program

{

/\*

https://www.desmos.com/calculator/ex5j3xz9rh

\*/

static void Main(string[] args)

{

Line V1, V2, V3;

V1 = new Line((3, -5));

V2 = new Line(new Point((-4, 0)), new Point((8, -1)));

Console.WriteLine("Line1: \n" + V1.ToString());

Console.WriteLine("=================\n");

Console.WriteLine("Line2: \n" + V2.ToString());

Console.WriteLine("=================\n");

V3 = V1 + V2;

Console.WriteLine("Line3: \n" + V3.ToString());

Console.WriteLine("=================\n");

V3++;

Console.WriteLine("Line3 expanded: \n" + V3.ToString());

Console.WriteLine("=================\n");

Console.WriteLine("Enter point coords to check [11,11;22,22]");

string input = Console.ReadLine();

bool isValid = input.Split(';').All(p => double.TryParse(p, out \_));

while (isValid == false)

{

Console.Write("Invalid input. Try again: ");

input = Console.ReadLine();

isValid = input.Split(';').All(p => double.TryParse(p, out \_));

}

string[] splitted = input.Split(';');

Point toCheck = new Point(double.Parse(splitted[0]), double.Parse(splitted[1]));

Console.Write(V3.IsOnLine(toCheck) ? "Point is on line" : "Point isn't on line");

Console.ReadLine();

}

}

}

* + 1. **Line.cs**

namespace Lab4

{

internal class Line

{

public Point A { get; private set; }

public Point B { get; private set; }

public Line()

{

A = new Point();

B = new Point(x: 1);

}

public Line(Point pointEnd)

{

A = new Point();

B = pointEnd;

}

public Line(Point pointStart, Point pointEnd)

{

A = pointStart;

B = pointEnd;

}

public Line((double, double) pointEnd)

{

A = new Point();

B = new Point(pointEnd);

}

public Line((double, double) pointStart, (double, double) pointEnd)

{

A = new Point(pointStart);

B = new Point(pointEnd);

}

public static Line operator +(Line first, Line second)

{

Point vectorFirst = new Point(first.B.X - first.A.X, first.B.Y - first.A.Y);

Point vectorSecond = new Point(second.B.X - second.A.X, second.B.Y - second.A.Y);

Point resultVector = new Point(vectorFirst.X + vectorSecond.X, vectorFirst.Y + vectorSecond.Y);

return new Line((resultVector.X + first.A.X, resultVector.Y + first.A.Y));

}

public static Line operator ++(Line self)

{

return new Line(self.A, new Point(self.B.X + 1, self.B.Y + 1));

}

public bool IsOnLine(Point point)

{

var equation = this.GetEquationOfLine();

return point.X \* equation.xCoeff + equation.freeCoeff == point.Y &&

point.X.InRange(A.X, B.X) &&

point.Y.InRange(A.Y, B.Y);

}

public (double xCoeff, double freeCoeff) GetEquationOfLine()

{

double xCoeff = (double)(B.Y - A.Y) / (B.X - A.X);

double freeCoeff = -1.0D \* A.X \* xCoeff + A.Y;

return (xCoeff, freeCoeff);

}

public override string ToString()

{

var (xCoeff, freeCoeff) = GetEquationOfLine();

return $"[{A} -> {B}]\n" +

$"y = {xCoeff:F4}x {freeCoeff:+0.##}";

}

}

* + 1. **Point.cs**

namespace Lab4

{

internal class Point

{

public double X { get; private set; } = 0;

public double Y { get; private set; } = 0;

public Point(double x = 0, double y = 0)

{

X = x;

Y = y;

}

public Point((double x, double y) coord)

{

(X, Y) = coord;

}

public override string ToString()

{

return $"({X:0.##}, {Y:0.##})";

}

}

}

* + 1. **Extension.cs**

namespace Lab4

{

internal static class Extension

{

public static bool InRange(this double self, double minValue, double maxValue)

{

if (minValue > maxValue)

(minValue, maxValue) = (maxValue, minValue);

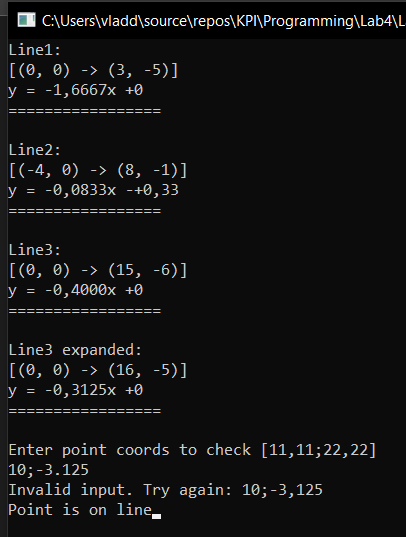
return self >= minValue && self <= maxValue;

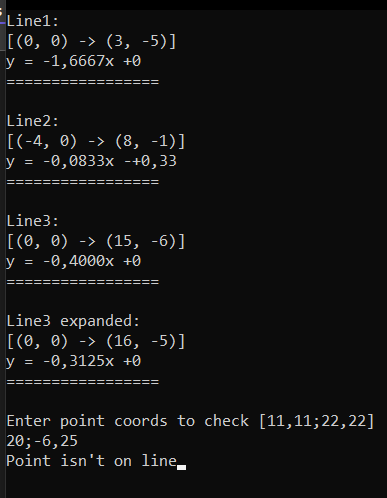
}

}

}

* 1. **Запуск програми**

****

****

***Висновок:*** В процесі виконання лабораторної роботи мною були опрацьовані навички у роботі із перегрузкою операторів, а саме 2 їх різновиди: унарні та бінарні.