

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

**Кафедра ІІІ**

**Звіт**

з лабораторної роботи № 3 з дисципліни  
«Основи програмування 2.  
Модульне програмування»  
Варіант 25

**Виконав(ла)**

ІІ-15 Плугатирьов Дмитро Валерійович  
(шифр, прізвище, ім'я, по батькові)

**Перевірів**

Вєчерковська Анастасія Сергіївна  
(прізвище, ім'я, по батькові)

Київ 2022

## Лабораторна робота № 3

### ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ

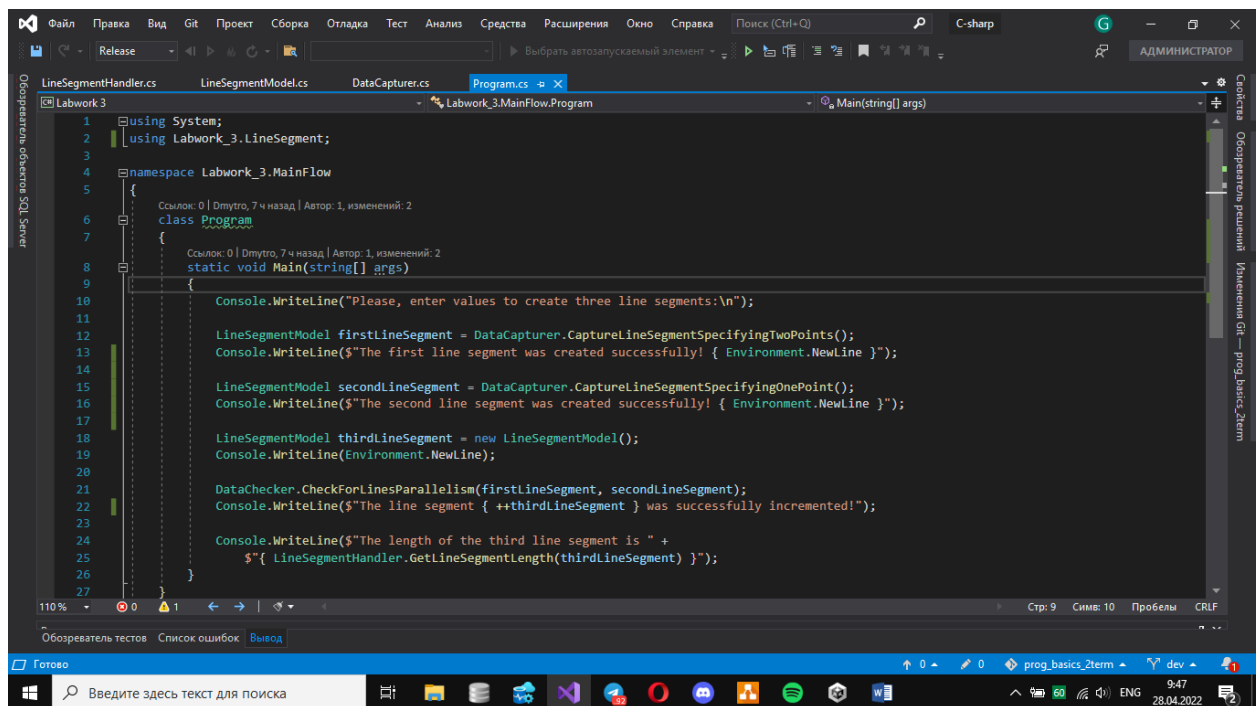
*Мета роботи* – вивчити механізми створення класів з використанням перевантажених операторів (операцій).

#### Завдання

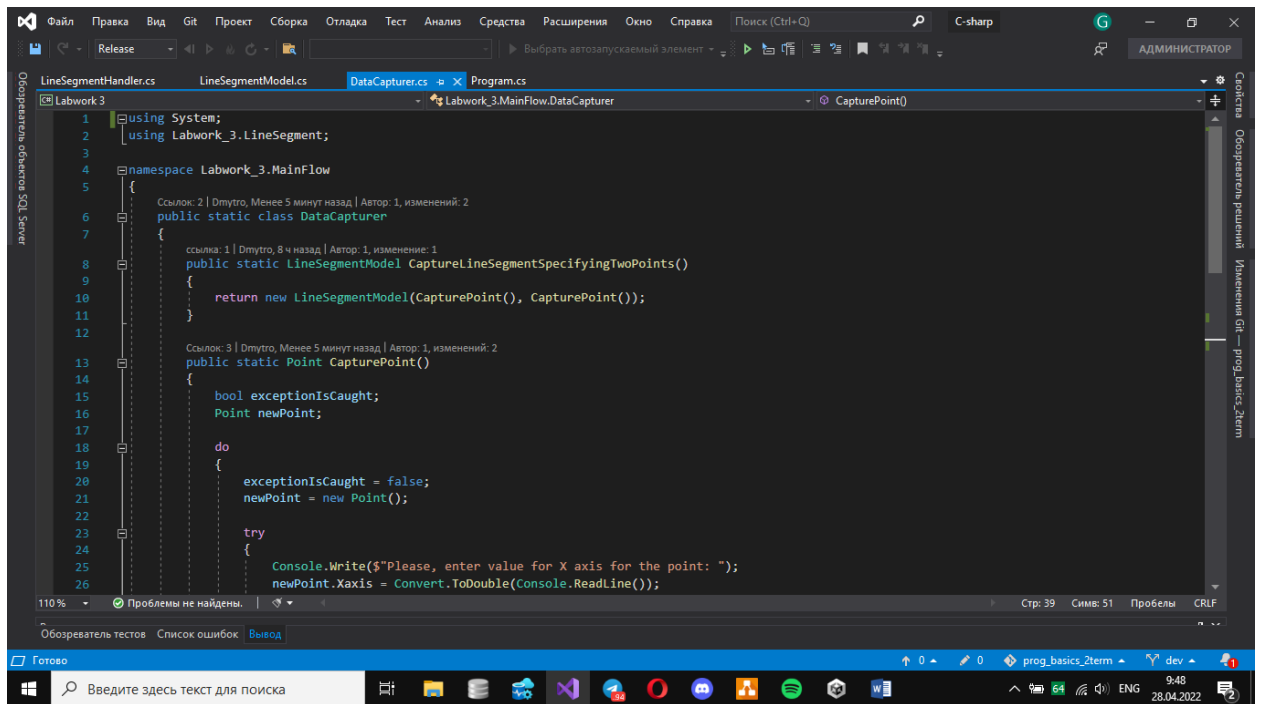
#### Варіант 25

Визначити клас «Відрізок», який задається координатами початку та кінця відрізка. Реалізувати для нього декілька конструкторів, геттери, метод обчислення довжини відрізка. Перевантажити оператори: префіксний «++» - для збільшення координат початку відрізка на 1, «||» - для перевірки паралельності відрізків. Створити три відрізки (V1, V2, V3), використовуючи різні конструктори. Перевірити, чи паралельні між собою відрізки V1 та V2. Збільшити координати початку відрізка V3 на 1. Знайти довжину отриманого відрізка V3.

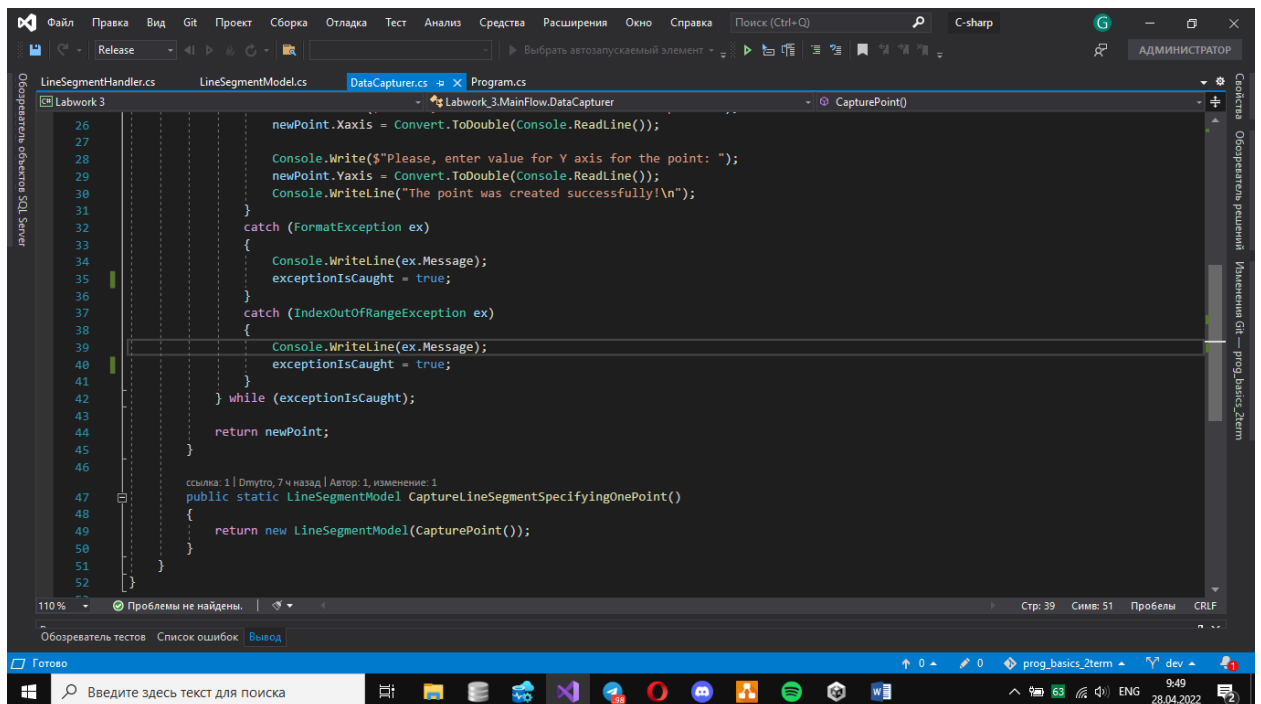
#### Код програми



```
1 using System;
2 using Labwork_3.LineSegment;
3
4 namespace Labwork_3.MainFlow
5 {
6     class Program
7     {
8         static void Main(string[] args)
9         {
10             Console.WriteLine("Please, enter values to create three line segments:\n");
11             LineSegmentModel firstLineSegment = DataCapturer.CaptureLineSegmentSpecifyingTwoPoints();
12             Console.WriteLine($"The first line segment was created successfully! { Environment.NewLine }");
13
14             LineSegmentModel secondLineSegment = DataCapturer.CaptureLineSegmentSpecifyingOnePoint();
15             Console.WriteLine($"The second line segment was created successfully! { Environment.NewLine }");
16
17             LineSegmentModel thirdLineSegment = new LineSegmentModel();
18             Console.WriteLine(Environment.NewLine);
19
20             DataChecker.CheckForLinesParallelism(firstLineSegment, secondLineSegment);
21             Console.WriteLine($"The line segment { ++thirdLineSegment } was successfully incremented!");
22
23             Console.WriteLine($"The length of the third line segment is " +
24                 $"{ LineSegmentHandler.GetLineSegmentLength(thirdLineSegment) }");
25         }
26     }
27 }
```



```
1 using System;
2 using Labwork_3.LineSegment;
3
4 namespace Labwork_3.MainFlow
5 {
6     Ссылка: 2 | 0 минут, Менее 5 минут назад | Автор: 1, изменений: 2
7     public static class DataCapturer
8     {
9         Ссылка: 1 | 0 минут, 8 ч назад | Автор: 1, изменение: 1
10        public static LineSegmentModel CaptureLineSegmentSpecifyingTwoPoints()
11        {
12            return new LineSegmentModel(CapturePoint(), CapturePoint());
13        }
14
15        Ссылка: 3 | 0 минут, Менее 5 минут назад | Автор: 1, изменений: 2
16        public static Point CapturePoint()
17        {
18            bool exceptionIsCaught;
19            Point newPoint;
20
21            do
22            {
23                exceptionIsCaught = false;
24                newPoint = new Point();
25
26                try
27                {
28                    Console.WriteLine($"Please, enter value for X axis for the point: ");
29                    newPoint.Xaxis = Convert.ToDouble(Console.ReadLine());
30                }
31            } while (exceptionIsCaught);
32
33            return newPoint;
34        }
35
36        Ссылка: 1 | 0 минут, 7 ч назад | Автор: 1, изменение: 1
37        public static LineSegmentModel CaptureLineSegmentSpecifyingOnePoint()
38        {
39            return new LineSegmentModel(CapturePoint());
40        }
41    }
42 }
```



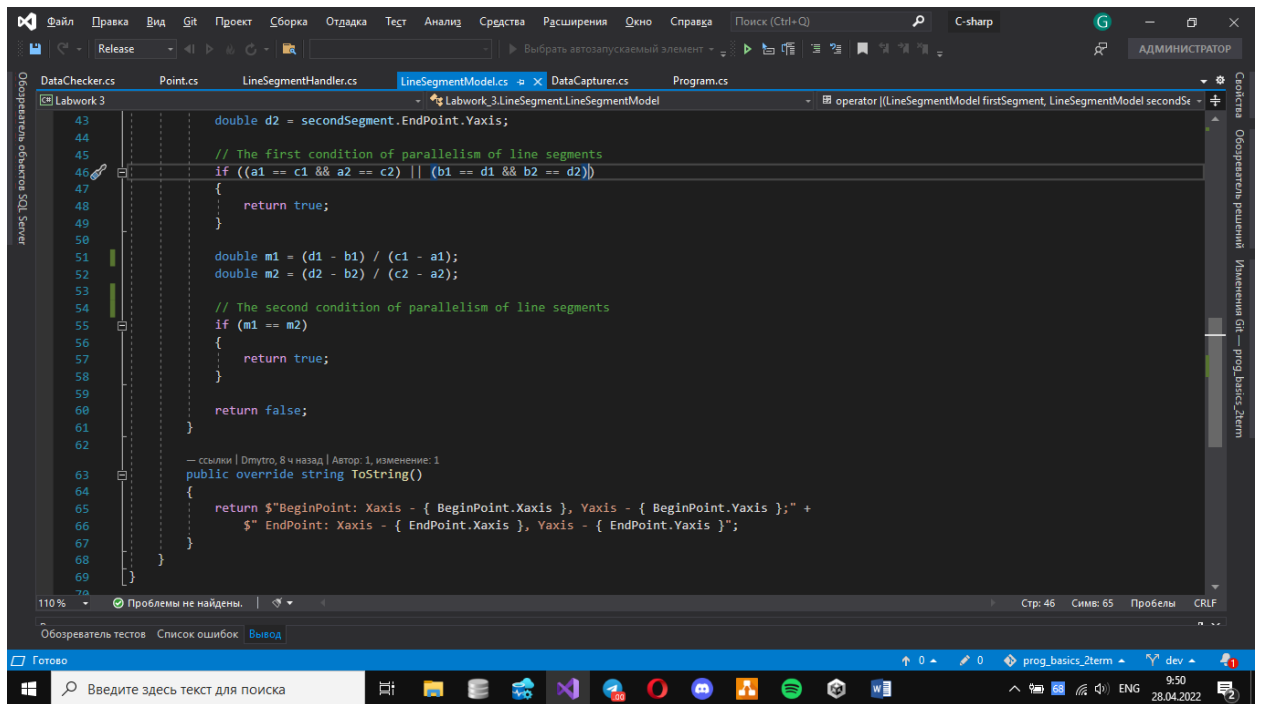
```
26 newPoint.Xaxis = Convert.ToDouble(Console.ReadLine());
27 Console.WriteLine($"Please, enter value for Y axis for the point: ");
28 newPoint.Yaxis = Convert.ToDouble(Console.ReadLine());
29 Console.WriteLine("The point was created successfully!\n");
30
31 }
32 catch (FormatException ex)
33 {
34     Console.WriteLine(ex.Message);
35     exceptionIsCaught = true;
36 }
37 catch (IndexOutOfRangeException ex)
38 {
39     Console.WriteLine(ex.Message);
40     exceptionIsCaught = true;
41 }
42 } while (exceptionIsCaught);
43
44 return newPoint;
45
46
47 Ссылка: 1 | 0 минут, 7 ч назад | Автор: 1, изменение: 1
48 public static LineSegmentModel CaptureLineSegmentSpecifyingOnePoint()
49 {
50     return new LineSegmentModel(CapturePoint());
51 }
52 }
```

This screenshot shows the Visual Studio IDE with the file `Labwork_3.LineSegmentModel.cs` open. The code defines a `LineSegmentModel` class within the `Labwork_3.LineSegment` namespace. The class has two public properties, `BeginPoint` and `EndPoint`, each of type `Point`, with getter and setter methods. It includes three constructors: a parameterized constructor taking `beginPoint` and `endPoint`, a constructor taking a shared `Point` (which creates separate `Point` instances for `BeginPoint` and `EndPoint`), and a default constructor that initializes both points to new `Point` objects. The status bar at the bottom indicates 110% zoom, no problems found, and the file is 46 lines long with 65 symbols and CRLF line endings.

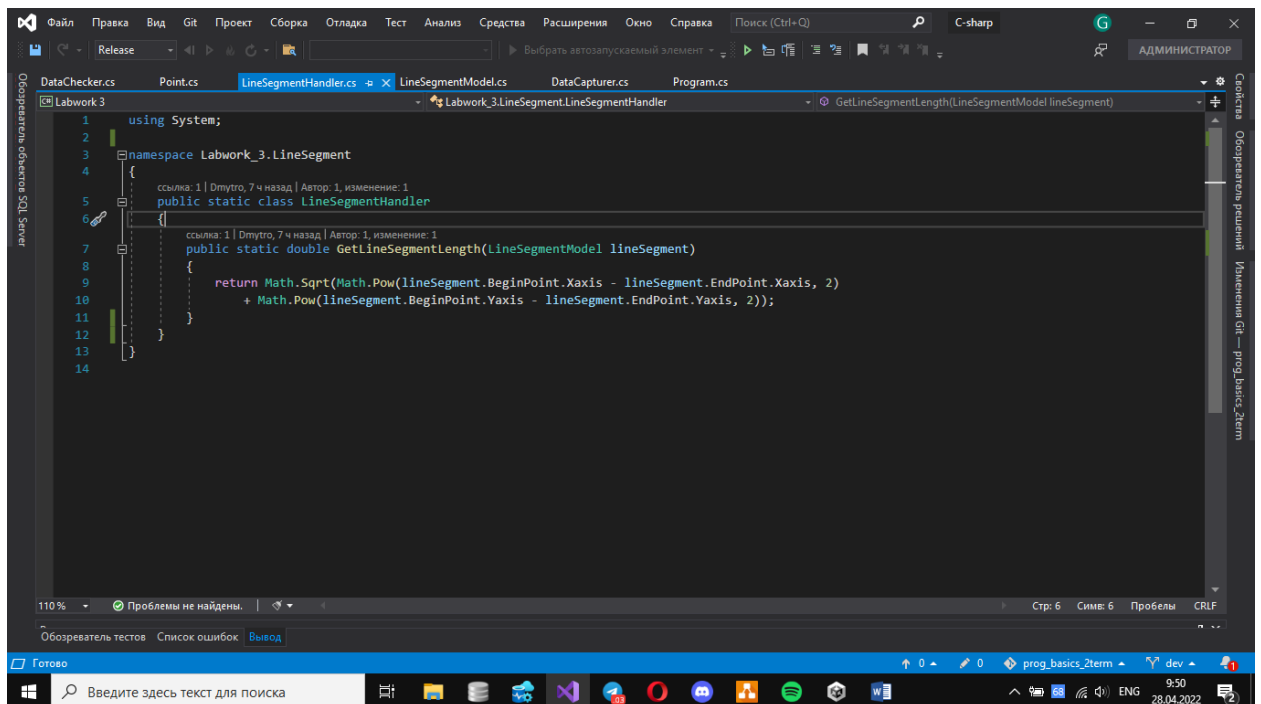
```
1 namespace Labwork_3.LineSegment
2 {
3     // Ссылка: 18 | Дмутьро, 5 мин назад | Автор: 1, изменений: 2
4     public class LineSegmentModel
5     {
6         // Ссылка: 12 | Дмутьро, 8 ч назад | Автор: 1, изменение: 1
7         public Point BeginPoint { get; set; }
8         // Ссылка: 11 | Дмутьро, 8 ч назад | Автор: 1, изменение: 1
9         public Point EndPoint { get; set; }
10
11         // ссылка: 1 | Дмутьро, 8 ч назад | Автор: 1, изменение: 1
12         public LineSegmentModel(Point beginPoint, Point endPoint)
13         {
14             BeginPoint = beginPoint;
15             EndPoint = endPoint;
16         }
17
18         // ссылка: 1 | Дмутьро, 8 ч назад | Автор: 1, изменение: 1
19         public LineSegmentModel(Point sharedPoint)
20         {
21             BeginPoint = new Point(sharedPoint.Xaxis);
22             EndPoint = new Point(sharedPoint.Yaxis);
23         }
24
25         // ссылка: 1 | Дмутьро, 8 ч назад | Автор: 1, изменение: 1
26         public LineSegmentModel()
27         {
28             BeginPoint = new Point();
29             EndPoint = new Point();
30         }
31     }
32 }
```

This screenshot shows the continuation of the `LineSegmentModel` class in `Labwork_3.LineSegmentModel.cs`. It defines two static operator methods. The first is an increment operator `++` that increments the `BeginPoint` property and returns the modified object. The second is a boolean operator `|` that checks if two line segments are parallel by comparing their endpoints' coordinates. The status bar at the bottom shows 110% zoom, no problems found, and the file is 46 lines long with 65 symbols and CRLF line endings.

```
23 BeginPoint = new Point();
24 EndPoint = new Point();
25
26
27 // ссылка: 1 | Дмутьро, 8 ч назад | Автор: 1, изменение: 1
28 public static LineSegmentModel operator ++(LineSegmentModel lineSegment)
29 {
30     ++lineSegment.BeginPoint;
31     return lineSegment;
32 }
33
34 // ссылка: 1 | Дмутьро, 5 мин назад | Автор: 1, изменений: 2
35 public static bool operator |(LineSegmentModel firstSegment, LineSegmentModel secondSegment)
36 {
37     // P1(a1, b1), P2(c1, d1), P3(a2, b2), P4(c2, d2)
38     double a1 = firstSegment.BeginPoint.Xaxis;
39     double b1 = firstSegment.BeginPoint.Yaxis;
40     double c1 = firstSegment.EndPoint.Xaxis;
41     double d1 = firstSegment.EndPoint.Yaxis;
42     double a2 = secondSegment.BeginPoint.Xaxis;
43     double b2 = secondSegment.BeginPoint.Yaxis;
44     double c2 = secondSegment.EndPoint.Xaxis;
45     double d2 = secondSegment.EndPoint.Yaxis;
46
47     // The first condition of parallelism of line segments
48     if ((a1 == c1 && a2 == c2) || (b1 == d1 && b2 == d2))
49     {
50         return true;
51     }
52 }
```



```
43         double d2 = secondSegment.EndPoint.Yaxis;
44
45         // The first condition of parallelism of line segments
46         if ((a1 == c1 && a2 == c2) || (b1 == d1 && b2 == d2))
47         {
48             return true;
49         }
50
51         double m1 = (d1 - b1) / (c1 - a1);
52         double m2 = (d2 - b2) / (c2 - a2);
53
54         // The second condition of parallelism of line segments
55         if (m1 == m2)
56         {
57             return true;
58         }
59
60         return false;
61     }
62
63     — ссылки | 10 минут, 8 ч назад | Автор: 1, изменение: 1
64     public override string ToString()
65     {
66         return $"BeginPoint: Xaxis - { BeginPoint.Xaxis }, Yaxis - { BeginPoint.Yaxis }; " +
67             $"EndPoint: Xaxis - { EndPoint.Xaxis }, Yaxis - { EndPoint.Yaxis }";
68     }
69 }
```



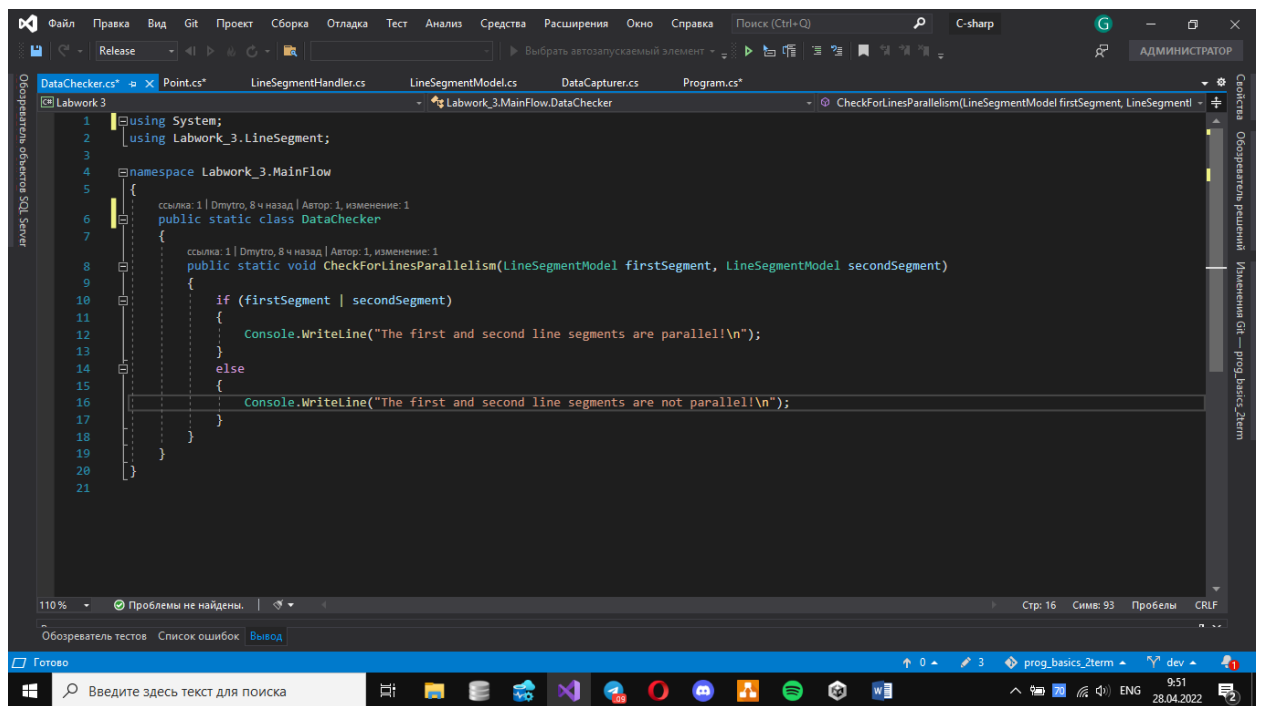
```
1 using System;
2
3 namespace Labwork_3.LineSegment
4 {
5     ссылка: 1 | 10 минут, 7 ч назад | Автор: 1, изменение: 1
6     public static class LineSegmentHandler
7     {
8         ссылка: 1 | 10 минут, 7 ч назад | Автор: 1, изменение: 1
9         public static double GetLineSegmentLength(LineSegmentModel lineSegment)
10         {
11             return Math.Sqrt(Math.Pow(lineSegment.BeginPoint.Xaxis - lineSegment.EndPoint.Xaxis, 2)
12                 + Math.Pow(lineSegment.BeginPoint.Yaxis - lineSegment.EndPoint.Yaxis, 2));
13         }
14     }
15 }
```

Visual Studio interface showing the initial implementation of the `Point` class in `Labwork_3.cs`. The class is defined within the `Labwork_3` namespace and includes properties `Xaxis` and `Yaxis` with `get` and `set` methods. It also features two constructors: one taking `double Xaxis` and `double Yaxis` as parameters, and another taking a `sharedAxis` parameter. A static operator `++` is also implemented.

```
1 namespace Labwork_3
2 {
3     // Ссылка: 17 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
4     public class Point
5     {
6         // Ссылка: 13 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
7         public double Xaxis { get; set; }
8         // Ссылка: 13 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
9         public double Yaxis { get; set; }
10
11         // Ссылка: 0 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
12         public Point(double Xaxis, double Yaxis)
13         {
14             this.Xaxis = Xaxis;
15             this.Yaxis = Yaxis;
16         }
17
18         // Ссылка: 2 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
19         public Point(double sharedAxis)
20         {
21             Xaxis = sharedAxis;
22             Yaxis = sharedAxis;
23         }
24
25         // Ссылка: 3 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
26         public Point() { }
27
28         // Ссылка: 1 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
29         public static Point operator ++(Point point)
30     }
```

Visual Studio interface showing the updated implementation of the `Point` class. The `++` operator is now implemented to increment both `Xaxis` and `Yaxis` properties of the `point` object and return the modified object.

```
13 }
14
15 // Ссылка: 2 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
16 public Point(double sharedAxis)
17 {
18     Xaxis = sharedAxis;
19     Yaxis = sharedAxis;
20 }
21
22 // Ссылка: 3 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
23 public Point() { }
24
25 // Ссылка: 1 | Dmitry, 8 ч назад | Автор: 1, изменение: 1
26 public static Point operator ++(Point point)
27 {
28     point.Xaxis++;
29     point.Yaxis++;
30     return point;
31 }
```



## Висновок

На цій лабораторній роботі я вивчив механізми створення класів з використанням перевантажених операторів. Як виявилося, це робить код наочним. Мені довелося використати бітовий оператор через специфіку мови C#: | замість ||.