Министерство образования Республики Беларусь Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей Кафедра Информатики

Дисциплина «Программирование»

## ОТЧЕТ

к лабораторной работе №7 на тему:

## «ПЕРЕГРУЗКА ОПЕРАТОРОВ»

БГУИР 6-05-0612-02 124

|  |
| --- |
| Выполнил студент группы 453504  ЯРЦЕВ Александр Александрович |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики РОМАНЮК Максим Валерьевич |
| (дата, подпись преподавателя) |

Минск 2025

# ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

**Задание 1. Вариант 5.** Класс – полином ax + by + cz. Поля – int(a, b, c). Перегрузить +, -, ++, --, \* на число, / на число. Сравнивать на == и !=. (d) если a=b=c=0 полином = false. Преобразовать в число (a) и назад в (ax) – в обоях случаях явно.

Для заданной задачи реализовать следующее:

* + спроектировать класс согласно варианту индивидуального задания.

Для класса использовать отдельный модуль;

* + спроектировать конструкторы и свойства с контролем корректности вводимых значений;
  + перегрузить метод toString();
  + перегрузить математические операции (имеющие смысл для объектов класса), инкремент и декремент (изменить поля на 1), отношения (==, !=, <,

>), true и false, преобразования типа;

* + создать несколько объектов класса. Продемонстрировать использование конструкторов и свойств;
  + продемонстрировать работу всех методов и операций.

# ВЫПОЛНЕНИЕ РАБОТЫ

Для выполнения задания был создан класс Polynomial, где реализована логика программы. В классе имеются приватные поля a, b, c, отвечающие за коэффициенты полинома ax + by + cz. Далее реализована перегрузка всех требуемых операций и методов, стоит акцентировать внимание на перегрузке преобразовании типов, где используется ключевое слово explicit, что позволяет явно определять преобразования типов.

public class Polynomial

{

private int a, b, c;

public Polynomial(int a, int b, int c)

{

this.a = a;

this.b = b;

this.c = c;

}

public Polynomial() : this(0, 0, 0) { }

public int A

{

get => a;

set => a = value;

}

public int B

{

get => b;

set => b = value;

}

public int C

{

get => c;

set => c = value;

}

public override string ToString() => $"Polynomial: {a}x + {b}y + {c}z";

public int this[int index]

{

get => index switch

{

0 => a,

1 => b,

2 => c,

\_ => throw new IndexOutOfRangeException()

};

set

{

switch (index)

{

case 0: a = value; break;

case 1: b = value; break;

case 2: c = value; break;

default: throw new IndexOutOfRangeException();

}

}

}

public static Polynomial operator +(Polynomial p1, Polynomial p2) =>

new Polynomial(p1.a + p2.a, p1.b + p2.b, p1.c + p2.c);

public static Polynomial operator -(Polynomial p1, Polynomial p2) =>

new Polynomial(p1.a - p2.a, p1.b - p2.b, p1.c - p2.c);

public static Polynomial operator \*(Polynomial p, int k) =>

new Polynomial(p.a \* k, p.b \* k, p.c \* k);

public static Polynomial operator /(Polynomial p, int k)

{

if (k == 0)

throw new DivideByZeroException("Division by zero is not allowed");

if (p.a % k != 0 || p.b % k != 0 || p.c % k != 0)

throw new ArgumentException("Division would result in non-integer coefficients");

return new Polynomial(p.a / k, p.b / k, p.c / k);

}

public static Polynomial operator ++(Polynomial p) =>

new Polynomial(p.a + 1, p.b + 1, p.c + 1);

public static Polynomial operator --(Polynomial p) =>

new Polynomial(p.a - 1, p.b - 1, p.c - 1);

public static bool operator ==(Polynomial p1, Polynomial p2)

{

if (ReferenceEquals(p1, p2)) return true;

if (p1 is null || p2 is null) return false;

return p1.a == p2.a && p1.b == p2.b && p1.c == p2.c;

}

public static bool operator !=(Polynomial p1, Polynomial p2) => !(p1 == p2);

public static bool operator true(Polynomial p) => p.a != 0 || p.b != 0 || p.c != 0;

public static bool operator false(Polynomial p) => p.a == 0 && p.b == 0 && p.c == 0;

public static explicit operator int(Polynomial p) => p.a;

public static explicit operator Polynomial(int a) => new Polynomial(a, 0, 0);

}

Продемонстрируем работу класса Polynomial в классе Program, используя все реализованные методы и операции. Создадим несколько объектов класса Polynomial и проведём над ними операции, для демонстрации работоспособности программы.

class Program

{

static void Main()

{

try

{

Console.WriteLine("=== Создание полиномов ===");

Polynomial p1 = new Polynomial(2, 3, 4);

Polynomial p2 = new Polynomial(1, 1, 1);

Polynomial p3 = new Polynomial();

Console.WriteLine($"p1: {p1}");

Console.WriteLine($"p2: {p2}");

Console.WriteLine($"p3: {p3}");

Console.WriteLine("\n=== Свойства и индексатор ===");

p1.A = 6;

Console.WriteLine($"p1 после p1.A = 6: {p1}");

p1[1] = 8;

Console.WriteLine($"p1 после p1[1] = 8: {p1}");

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

Polynomial sum = p1 + p2;

Console.WriteLine($"p1 + p2: {sum}");

Polynomial diff = p1 - p2;

Console.WriteLine($"p1 - p2: {diff}");

Polynomial mult = p1 \* 2;

Console.WriteLine($"p1 \* 2: {mult}");

Polynomial div = p1 / 2;

Console.WriteLine($"p1 / 2: {div}");

Console.WriteLine("\n=== Инкремент и декремент ===");

p3 = ++p3;

Console.WriteLine($"p3 после ++: {p3}");

p3 = --p3;

Console.WriteLine($"p3 после --: {p3}");

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

Console.WriteLine($"p1 == p2: {p1 == p2}");

Console.WriteLine($"p1 != p2: {p1 != p2}");

Console.WriteLine("\n=== True/False ===");

Console.WriteLine($"p1 is true: {(p1 ? "Yes" : "No")}");

Console.WriteLine($"p3 is true: {(p3 ? "Yes" : "No")}");

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

int value = (int)p1;

Console.WriteLine($"p1 -> int: {value}");

Polynomial fromInt = (Polynomial)10;

Console.WriteLine($"10 -> Polynomial: {fromInt}");

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

try

{

Polynomial error = p1 / 0;

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

try

{

Polynomial test = p1 / 3; // 6,8,4 не делится на 3 без остатка

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка деления: {ex.Message}");

}

}

catch (Exception ex)

{

Console.WriteLine($"Произошла ошибка: {ex.Message}");

}

}

}

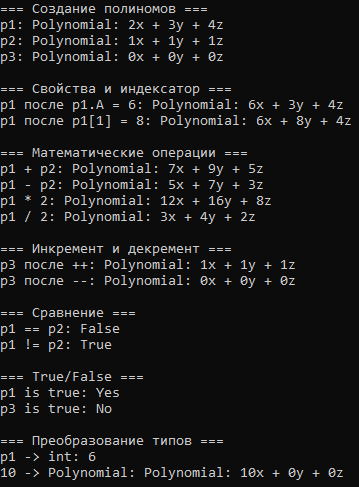
Результат работы программы продемонстрирован ниже (см. рисунок 1)

Рисунок 1 – Результат работы программы

# ВЫВОД

В ходе лабораторной работы были изучены принципы перегрузки операторов и методов, а также добавление индексирования для получения полей класса.