Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 13

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Классы и объекты»

Выполнил: ст. гр. ТУУ-111

Грачева Н.С.

Вариант №5

Проверил: к.т.н., доц. Сафронов А.И.

Москва – 2024 г.

1. **Цель работы**

В среде *Microsoft* *Visual Studio* на языке *Visual C#* в консольном режиме составить программное обеспечение для решения типовых задач программирования по тематике «Классы и объекты» («*Classes and Objects*»), заданных по варианту.

1. **Формулировка задачи**

**Вариант №5**  
Создайте класс «Полином» (возможное имя «clsPolynom»).

**Поля (fields):**

* идентификатор (int);
* наименование переменной аргумента (string);
* наименование переменной функции (string);
* степень полинома n (int);
* коэффициенты полинома (double[n]).

**Конструктор (instance constructor):**

* по усмотрению разработчика.

**Методы (methods):**

1. Предусмотреть метод для смены наименования переменной аргумента (void).
2. Предусмотреть метод для смены наименования переменной функции (void).
3. Предусмотреть метод для вывода вектора коэффициентов полинома (void).
4. Предусмотреть метод переназначения всех коэффициентов полинома (void).
5. Предусмотреть метод переназначения коэффициента при указанной степени (void).

**Операторы (operators):**

* повышение степени полинома (++);
* понижение степени полинома (--);
* сложение полиномов (+);
* разность полиномов (–);
* произведение полиномов (\*).

**Хранение элементов:**

* в файле с возможностью чтения и записи.

**Сопоставление (Equals()):**

Полиномы считать равными по совпадении их степеней.

**Вывод (ToString()):**

G(x) = x^3 – 2x^2 + 7x + 1

R(z) = –z^5 + 9z^3 – z

**Хэширование (GetHashCode()):**

Сумма абсолютных величин коэффициентов полинома, а также порядковых номеров в алфавите наименований переменной аргумента и переменной функции, обозначающих полином.

1. **Блок-схема алгоритма**

НАЧАЛО

CreatePoly()

choice = -1

choice != 22

да

нет

DisplayMenu()

choice = GetUserChoice()

ProcessChoice(choice)

Вывод:  
пустая строка

КОНЕЦ

Рисунок 1 - Блок-схема Main файла Program.cs

НАЧАЛО

Ввод (прием параметра):  
choice

choice

PerformAddition()

PerformSubtraction()

PerformMultiplication()

IncreaseDegreeG()

DecreaseDegreeG()

IncreaseDegreeR()

DecreaseDegreeR()

1

2

1

2

3

4

5

6

7

Рисунок 2 - Блок-схема метода ProcessChoice файла Program.cs (часть-1)

1

2

ComparePolynomials()

PrintHashCodes()

PrintCoefficientsG()

PrintCoefficientsR()

ChangeVariableNameR()

ChangeVariableNameG()

ChangeFunctionNameG()

ChangeFunctionNameR()

ResetCoefficientsG()

ResetCoefficientsR()

3

4

8

9

10

11

12

13

14

15

16

17

Рисунок 3 - Блок-схема метода ProcessChoice файла Program.cs (часть-2)

3

4

SetCoefficientG()

SetCoefficientR()

SavePolynomialsToFile()

LoadPolynomialsFromFile()

Вывод:  
“Неверный выбор”

КОНЕЦ

18

19

20

21

22

Рисунок 4 - Блок-схема метода ProcessChoice файла Program.cs (часть-3)

НАЧАЛО

result = G.Add(R)

Вывод:  
result

КОНЕЦ

Рисунок 5 - Блок-схема метода PerformAddition файла Program.cs

НАЧАЛО

result = G.Subtract(R)

Вывод:  
result

КОНЕЦ

Рисунок 6 - Блок-схема метода PerformSubstraction файла Program.cs

НАЧАЛО

result = G.Multiply(R)

Вывод:  
result

КОНЕЦ

Рисунок 7 - Блок-схема метода PerformMultiplication, файла Program.cs

НАЧАЛО

Ввод (передача):  
p1, p2

int maxDegree = Math.Max(p1.coefficients.Count, p2.coefficients.Count);

List<int> resultCoefficients = new List<int>(new int[maxDegree]);

i = 0

i <= maxDegree

да

нет

int thisCoeff = i < p1.coefficients.Count ? p1.coefficients[i] : 0;

int otherCoeff = i < p2.coefficients.Count ? p2.coefficients[i] : 0;

resultCoefficients[i] = thisCoeff + otherCoeff;

i = i + 1

Вывод (возврат):  
resultCoefficients, p1.variable, p1.functionName

КОНЕЦ

Рисунок 8 - Блок-схема оператора (сложения полиномов) файла Polynomial.cs

НАЧАЛО

Ввод (передача):  
p1, p2

int maxDegree = Math.Max(p1.coefficients.Count, p2.coefficients.Count);

List<int> resultCoefficients = new List<int>(new int[maxDegree]);

i = 0

i < maxDegree

да

нет

int thisCoeff = i < p1.coefficients.Count ? p1.coefficients[i] : 0;

int otherCoeff = i < p2.coefficients.Count ? p2.coefficients[i] : 0;

resultCoefficients[i] = thisCoeff - otherCoeff;

i = i + 1

Вывод (возврат):  
resultCoefficients, p1.variable, p1.functionName

КОНЕЦ

Рисунок 9 - Блок-схема оператора вычитания полиномов файла Polynomial.cs

НАЧАЛО

Ввод (передача):  
p1, p2

int newDegree = p1.coefficients.Count + p2.coefficients.Count - 1;

List<int> resultCoefficients = new List<int>(new int[newDegree]);

i = 0

p1.coefficients.Count

да

нет

j = 0

p2.coefficients.Count

да

нет

resultCoefficients[i + j] += p1.coefficients[i] \* p2.coefficients[j];

j = j + 1

i = i + 1

Вывод (возврат): resultCoefficients, p1.variable, p1.functionName

КОНЕЦ

Рисунок 10 - Блок-схема умножения полиномов файла Polynomial.cs

НАЧАЛО

StreamWriter writer = new StreamWriter(fileName)

Вывод в файл:  
this.variable, this.functionName, this.coefficients

КОНЕЦ

Ввод:  
filename

Рисунок 11 - Блок-схема метода SaveToFile файла Polynomial.cs

НАЧАЛО

variable = lines[0];  
functionName = lines[1];  
coefficients = lines[2].Split().Select(int.Parse).ToArray();

Вывод (возврат):  
new Polynomial(coefficients, variable, functionName)

КОНЕЦ

Ввод:  
filename

Чтение из файла:  
lines = File.ReadAllLines(fileName)

Рисунок 12 - Блок-схема метода LoadFromFile файла Polynomial.cs

1. **Подбор тестовых примеров**

Создаём два экземпляра класса со следующими данными:

1) Id = 1

variable = x

functionName = G

coefficients = { 1, 7, -2, 1 }

2) Id = 1

variable = z

functionName = R

coefficients = { 0, -1, 0, 9, 0, -1 }

1. **Описание методов и операторов класса**

**5.1 Метод SaveToFile**

Метод сохранения полинома в файл.

* 1. **Метод LoadFromFile**

Метод загрузки полинома из файла.

* 1. **Оператор +**

Перегрузка оператора сложения двух полиномов.

* 1. **Оператор -**

Перегрузка оператора вычитания двух полиномов.

* 1. **Оператор \***

Перегрузка оператора умножения двух полиномов.

* 1. **Оператор ++**

Увеличение степени полинома.

* 1. **Оператор - -**

Уменьшение степени полинома.

* 1. **ChangeVariableName**

Метод изменения имени переменной полинома.

**5.8 ChangeFunctionName**

Метод изменения имени функции полинома.

* 1. **GetCoefficients**

Получение массива коэффициентов, используется в функциональности изменения коэффициентов.

* 1. **ResetCoefficients**

Вспомогательный метод, используется для переназначения коэффициентов полинома.

* 1. **SetCoefficient**

Метод переназначения коэффициентов полинома.

* 1. **ToString**

Метод для печати полинома на экране.

* 1. **Equals**  
     Метод сравнения двух полиномов.   
     Возвращает true, если степени полиномов совпадают.  
     Возвращает false в других случаях.
  2. **GetHashCode**

Получение hash-кодов полиномов.

1. **Листинг**

**Программа Program.cs**

using System;

class Program

{

static Polynomial G, R;

static string fileG = "poly1.txt";

static string fileR = "poly2.txt";

static void Main()

{

CreatePoly();

int choice = -1; //номер пункта меню заведомо ложный,

//после выбора пользователя здесь будет корректный номер

while (choice != 22)

{

DisplayMenu();

choice = GetUserChoice();

ProcessChoice(choice);

Console.WriteLine();

}

}

static void CreatePoly()

{

G = new Polynomial(new int[] { 1, 7, -2, 1 }, "x", "G");

R = new Polynomial(new int[] { 0, -1, 0, 9, 0, -1 }, "z", "R");

}

static void DisplayMenu()

{

Console.WriteLine("Текущее состояние полиномов:");

Console.WriteLine(G);

Console.WriteLine(R);

Console.WriteLine();

//пункты меню

Console.WriteLine("Выберите операцию:");

Console.WriteLine("1. Сложение полиномов");

Console.WriteLine("2. Вычитание полиномов");

Console.WriteLine("3. Умножение полиномов");

Console.WriteLine("4. Увеличение степени полинома #1");

Console.WriteLine("5. Уменьшение степени полинома #1");

Console.WriteLine("6. Увеличение степени полинома #2");

Console.WriteLine("7. Уменьшение степени полинома #2");

Console.WriteLine("8. Сравнение полиномов");

Console.WriteLine("9. Вывести хэш-коды полиномов");

Console.WriteLine("10. Вывести вектор коэффициентов полинома #1");

Console.WriteLine("11. Вывести вектор коэффициентов полинома #2");

Console.WriteLine("12. Смена имени переменной аргумента полинома #1");

Console.WriteLine("13. Смена имени переменной аргумента полинома #2");

Console.WriteLine("14. Смена имени функции полинома #1");

Console.WriteLine("15. Смена имени функции полинома #2");

Console.WriteLine("16. Переназначение всех коэффициентов полинома #1");

Console.WriteLine("17. Переназначение всех коэффициентов полинома #2");

Console.WriteLine("18. Переназначение коэффициента при указанной степени в полиноме #1");

Console.WriteLine("19. Переназначение коэффициента при указанной степени в полиноме #2");

Console.WriteLine("20. Сохранить полиномы в файлы");

Console.WriteLine("21. Загрузить полиномы из файлов");

Console.WriteLine("22. Выход");

}

static int GetUserChoice()

{

Console.Write("Введите номер операции: ");

return int.Parse(Console.ReadLine());

}

static void ProcessChoice(int choice)

{

switch (choice)

{

case 1:

PerformAddition(); //сложение

break;

case 2:

PerformSubtraction(); //вычитание

break;

case 3:

PerformMultiplication(); //умножение

break;

case 4:

IncreaseDegreeG(); //увеличение степени полином #1

break;

case 5:

DecreaseDegreeG(); //уменьшение степени полином #1

break;

case 6:

IncreaseDegreeR(); //увеличение степени полином #2

break;

case 7:

DecreaseDegreeR(); //уменьшение степени полином #2

break;

case 8:

ComparePolynomials(); //сравнение полиномов

break;

case 9:

PrintHashCodes(); //печать hash-кодов

break;

case 10:

PrintCoefficientsG(); //печать коэффициентов (векторов) полинома #1

break;

case 11:

PrintCoefficientsR(); //печать коэффициентов (векторов) полинома #2

break;

case 12:

ChangeVariableNameG(); //изменение переменной полинома #1

break;

case 13:

ChangeVariableNameR(); //изменение переменной полинома #2

break;

case 14:

ChangeFunctionNameG(); //изменение функции полинома #1

break;

case 15:

ChangeFunctionNameR(); //изменение функции полинома #2

break;

case 16:

ResetCoefficientsG(); //переназначение (сброс) всех коэффициентов полинома #1

break;

case 17:

ResetCoefficientsR(); //переназначение (сброс) всех коэффициентов полинома #2

break;

case 18:

SetCoefficientG(); //установка коэффициента для указанной степени полинома #1

break;

case 19:

SetCoefficientR(); //установка коэффициента для указанной степени полинома #2

break;

case 20:

SavePolynomialsToFile(); //сохранение полиномов в файлы

break;

case 21:

LoadPolynomialsFromFile(); //чтение полиномов из файлов

break;

case 22:

break; //выход из программы

default:

Console.WriteLine("Неверный выбор!"); //если ввели меньше 1 или больше 22,

//выводится сообщение об ошибке

break;

}

}

static void PerformAddition()

//сложение полиномов

{

Polynomial result = G + R;

Console.WriteLine("Результат сложения: " + result);

}

static void PerformSubtraction()

//вычитание полиномов

{

Polynomial result = G - R;

Console.WriteLine("Результат вычитания: " + result);

}

static void PerformMultiplication()

//умножние полиномов

{

Polynomial result = G \* R;

Console.WriteLine("Результат умножения: " + result);

}

static void IncreaseDegreeG()

//увеличение степени полинома

{

G++;

Console.WriteLine("Результат увеличения степени: " + G);

}

static void DecreaseDegreeG()

//уменьшение степени полинома

{

G--;

Console.WriteLine("Результат уменьшения степени: " + G);

}

static void IncreaseDegreeR()

//увеличение степени полинома

{

R++;

Console.WriteLine("Результат увеличения степени: " + R);

}

static void DecreaseDegreeR()

//уменьшение степени полинома

{

R--;

Console.WriteLine("Результат уменьшения степени: " + R);

}

static void ComparePolynomials()

//сравнение полиномов

{

if (G.Equals(R))

{

Console.WriteLine("Полиномы G и R равны по степени.");

}

else

{

Console.WriteLine("Полиномы G и R не равны по степени.");

}

}

static void PrintHashCodes()

//печать hash-кодов

{

Console.WriteLine($"HashCode G(x) = {G.GetHashCode()}");

Console.WriteLine($"HashCode R(z) = {R.GetHashCode()}");

}

static void PrintCoefficientsG()

//печать векторов полинома #1

{

Console.WriteLine("Вектор коэффициентов полинома G(x):");

PrintArray(G.GetCoefficients());

}

static void PrintCoefficientsR()

//печать векторов полинома #2

{

Console.WriteLine("Вектор коэффициентов полинома R(z):");

PrintArray(R.GetCoefficients());

}

static void ChangeVariableNameG()

//изменение названия переменной полинома #1

{

Console.Write("Введите новое имя переменной аргумента для полинома G: ");

string newVarG = Console.ReadLine();

G.ChangeVariableName(newVarG);

}

static void ChangeVariableNameR()

//изменение названия переменной полинома #2

{

Console.Write("Введите новое имя переменной аргумента для полинома R: ");

string newVarR = Console.ReadLine();

R.ChangeVariableName(newVarR);

}

static void ChangeFunctionNameG()

//изменение названия функции полинома #1

{

Console.Write("Введите новое имя функции для полинома G: ");

string newFuncG = Console.ReadLine();

G.ChangeFunctionName(newFuncG);

}

static void ChangeFunctionNameR()

//изменение названия функции полинома #2

{

Console.Write("Введите новое имя функции для полинома R: ");

string newFuncR = Console.ReadLine();

R.ChangeFunctionName(newFuncR);

}

static void ResetCoefficientsG()

//ввод всех новых коэффициентов для полинома #1

{

Console.Write("Введите новые коэффициенты через пробел для полинома G(x) (начиная с самой высокой степени): ");

int[] newCoeffsG = Console.ReadLine().Split().Select(int.Parse).Reverse().ToArray();

G.ResetCoefficients(newCoeffsG);

}

static void ResetCoefficientsR()

//ввод всех новых коэффициентов для полинома #2

{

Console.Write("Введите новые коэффициенты через пробел для полинома R(z) (начиная с самой высокой степени): ");

int[] newCoeffsR = Console.ReadLine().Split().Select(int.Parse).Reverse().ToArray();

R.ResetCoefficients(newCoeffsR);

}

static void SetCoefficientG()

//ввод коэффициента для определенной степени для полинома #1

{

Console.Write("Введите степень для изменения коэффициента в полиноме G(x): ");

int degreeG = int.Parse(Console.ReadLine());

Console.Write($"Введите новый коэффициент для степени {degreeG}: ");

int coeffG = int.Parse(Console.ReadLine());

G.SetCoefficient(degreeG, coeffG);

}

static void SetCoefficientR()

//ввод коэффициента для определенной степени для полинома #2

{

Console.Write("Введите степень для изменения коэффициента в полиноме R(z): ");

int degreeR = int.Parse(Console.ReadLine());

Console.Write($"Введите новый коэффициент для степени {degreeR}: ");

int coeffR = int.Parse(Console.ReadLine());

R.SetCoefficient(degreeR, coeffR);

}

static void SavePolynomialsToFile()

//сохранение полиномов в файлы

{

G.SaveToFile(fileG);

R.SaveToFile(fileR);

Console.WriteLine("Полиномы сохранены в файлы.");

}

static void LoadPolynomialsFromFile()

//загрузка полиномов из файлов

{

G = Polynomial.LoadFromFile(fileG);

R = Polynomial.LoadFromFile(fileR);

Console.WriteLine("Полиномы загружены из файлов.");

}

static void PrintArray(int[] array)

//печать коэффициентов (векторов) полиномов

{

for (int i = array.Length - 1; i >= 0; i--)

{

Console.Write(array[i] + " ");

}

Console.WriteLine();

Console.ReadKey(true);

}

}

**Код класса Polynomial.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

public class Polynomial

{

private int Id; //Id

private List<int> coefficients; //список коэффициентов полинома, начиная с младшего члена

private string variable; //имя переменной аргумента

private string functionName; //имя функции полинома

public Polynomial(IEnumerable<int> coefficients, string variable = "x", string functionName = "f")

{

this.coefficients = coefficients.ToList();

this.variable = variable;

this.functionName = functionName;

}

public void SaveToFile(string fileName)

//метод для сохранения полинома в файл

{

using (StreamWriter writer = new StreamWriter(fileName))

{

writer.WriteLine(this.variable);

writer.WriteLine(this.functionName);

writer.WriteLine(string.Join(" ", this.coefficients));

}

}

public static Polynomial LoadFromFile(string fileName)

//метод для загрузки полинома из файла

{

string[] lines = File.ReadAllLines(fileName);

string variable = lines[0];

string functionName = lines[1];

int[] coefficients = lines[2].Split().Select(int.Parse).ToArray();

return new Polynomial(coefficients, variable, functionName);

}

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

//сложение двух полиномов

{

int maxDegree = Math.Max(p1.coefficients.Count, p2.coefficients.Count);

List<int> resultCoefficients = new List<int>(new int[maxDegree]);

// Если у какого-то полинома нет определенной степени, дополняем нулем

for (int i = 0; i < maxDegree; i++)

{

int thisCoeff = i < p1.coefficients.Count ? p1.coefficients[i] : 0;

int otherCoeff = i < p2.coefficients.Count ? p2.coefficients[i] : 0;

resultCoefficients[i] = thisCoeff + otherCoeff;

}

return new Polynomial(resultCoefficients, p1.variable, p1.functionName);

}

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

//вычитание двух полиномов

{

int maxDegree = Math.Max(p1.coefficients.Count, p2.coefficients.Count);

List<int> resultCoefficients = new List<int>(new int[maxDegree]);

// Если у какого-то полинома нет определенной степени, дополняем нулем

for (int i = 0; i < maxDegree; i++)

{

int thisCoeff = i < p1.coefficients.Count ? p1.coefficients[i] : 0;

int otherCoeff = i < p2.coefficients.Count ? p2.coefficients[i] : 0;

resultCoefficients[i] = thisCoeff - otherCoeff;

}

return new Polynomial(resultCoefficients, p1.variable, p1.functionName);

}

public static Polynomial operator \*(Polynomial p1, Polynomial p2)

//умножение двух полиномов

{

int newDegree = p1.coefficients.Count + p2.coefficients.Count - 1;

List<int> resultCoefficients = new List<int>(new int[newDegree]);

for (int i = 0; i < p1.coefficients.Count; i++)

{

for (int j = 0; j < p2.coefficients.Count; j++)

{

resultCoefficients[i + j] += p1.coefficients[i] \* p2.coefficients[j];

}

}

return new Polynomial(resultCoefficients, p1.variable, p1.functionName);

}

public static Polynomial operator ++(Polynomial p)

//увеличение степени полинома

{

List<int> resultCoefficients = new List<int> { 0 };

resultCoefficients.AddRange(p.coefficients);

return new Polynomial(resultCoefficients, p.variable, p.functionName);

}

public static Polynomial operator --(Polynomial p)

//уменьшение степени полинома

{

if (p.coefficients.Count == 1)

throw new InvalidOperationException("Cannot decrease the degree of a zero polynomial.");

List<int> resultCoefficients = p.coefficients.Skip(1).ToList();

return new Polynomial(resultCoefficients, p.variable, p.functionName);

}

public void ChangeVariableName(string newVariable)

//метод для изменения имени переменной полинома

{

this.variable = newVariable;

}

public void ChangeFunctionName(string newFunctionName)

//метод для изменения имени функции полинома

{

this.functionName = newFunctionName;

}

public int[] GetCoefficients()

//получение массива коэффициентов

{

// Возвращаем массив коэффициентов, начиная с младшего члена (степени 0)

return this.coefficients.ToArray();

}

public void ResetCoefficients(IEnumerable<int> newCoefficients)

{

this.coefficients = newCoefficients.ToList();

}

public void SetCoefficient(int degree, int coefficient)

//метод: установка коэффициентов полинома

{

if (degree < 0 || degree >= this.coefficients.Count)

throw new ArgumentOutOfRangeException(nameof(degree), "Degree out of range.");

this.coefficients[degree] = coefficient;

}

public override string ToString()

//перегрузка метода ToString - вывод полинома на экран

{

var terms = new List<string>();

for (int i = this.coefficients.Count - 1; i >= 0; i--)

{

if (this.coefficients[i] == 0) continue;

string term = this.coefficients[i].ToString();

if (i > 0) term += this.variable;

if (i > 1) term += "^" + i;

if (terms.Count > 0 && this.coefficients[i] > 0)

{

term = "+" + term;

}

terms.Add(term);

}

return $"{this.functionName}({this.variable}) = " + string.Join(" ", terms);

}

public override bool Equals(object obj)

//метод сравнения двух полиномов

//полиномы считаются равными, если их степени равны

{

if (obj == null || GetType() != obj.GetType())

return false;

Polynomial other = (Polynomial)obj;

// Проверяем равенство степеней полиномов

return this.coefficients.Count == other.coefficients.Count;

}

public override int GetHashCode()

//метод получения hash-кода полинома

{

int hash = this.coefficients.Sum(c => Math.Abs(c));

hash += char.ToUpper(this.variable[0]) - 'A' + 1;

hash += char.ToUpper(this.functionName[0]) - 'A' + 1;

return hash;

}

}

1. **Расчет тестовых примеров на ПК**

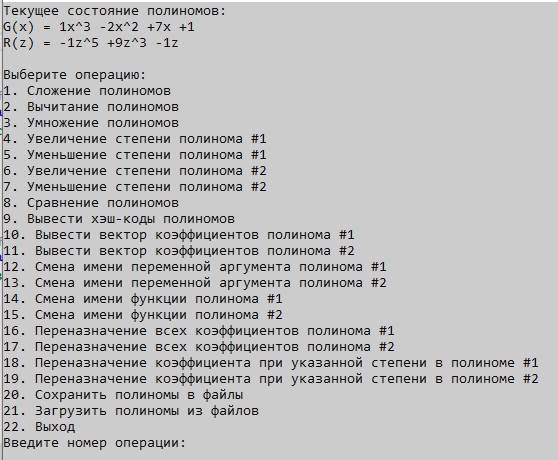


Рисунок 13 - Меню выбора функций программы

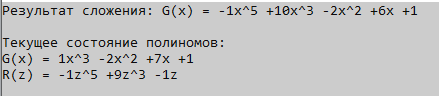


Рисунок 14 - Результат сложения полиномов

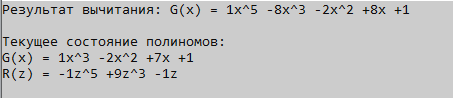
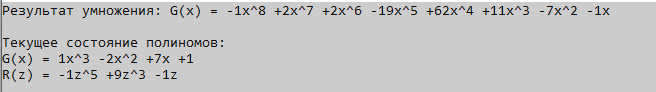


Рисунок 15 - Результат вычитания полиномов

Рисунок 16 - Результат умножения полиномов

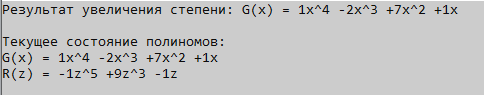


Рисунок 17 - Результат увеличения степени полинома

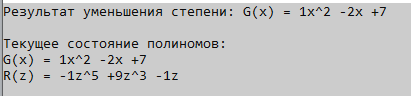


Рисунок 18 - Результат уменьшения степени полинома

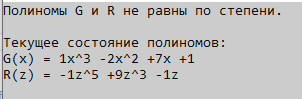


Рисунок 19 - Результат сравнения полиномов – случай, когда не равны

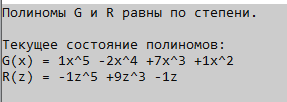


Рисунок 20 - Результат сравнения полиномов – случай, когда равны

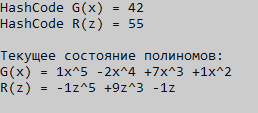


Рисунок 21 - Hash-коды

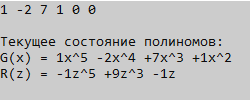


Рисунок 22 - Вектор полинома #1

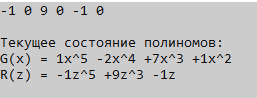


Рисунок - 23 Вектор полинома #2

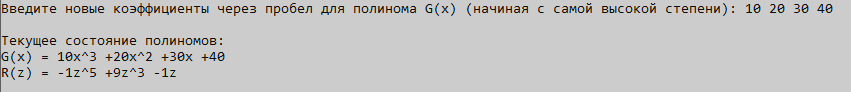


Рисунок 24 - Результат переназначения всех коэффициентов полинома



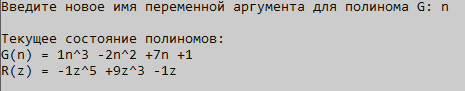
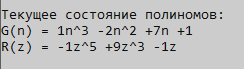


Рисунок - 25 Результат переименования переменной полинома

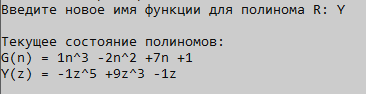


Рисунок 26 - Результат переименования функции полинома

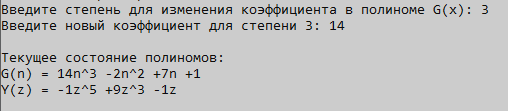


Рисунок 27 - Результат изменения коэффициента по выбранной степени полинома

1. **Вывод по работе**

Научилась создавать и работать с классами и его объектами, которые включают в себя конструкторы, методы, операторы, а также компоненты класса, соответствующие заданным требованиям, включая математические функции полинома, переименование переменной, переназначение всех коэффициентов и т.д. Освоила навык работы с Хешированием (GatHashCode()). Также включила в работу методы для чтения и записи текстовых надписей и их параметров в текстовый файл.