МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе №5 по дисциплине

“Коды, исправляющие ошибки”

Выполнил студент группы ПМИ-3301-02-00     / А.Л. Вохмянин /

Проверил преподаватель кафедры ПМИ     / И.А. Пушкарёв /

Киров 2019

**Цель:** реализовать программу, реализующую деление многочленов с остатком и представление в линейном виде.

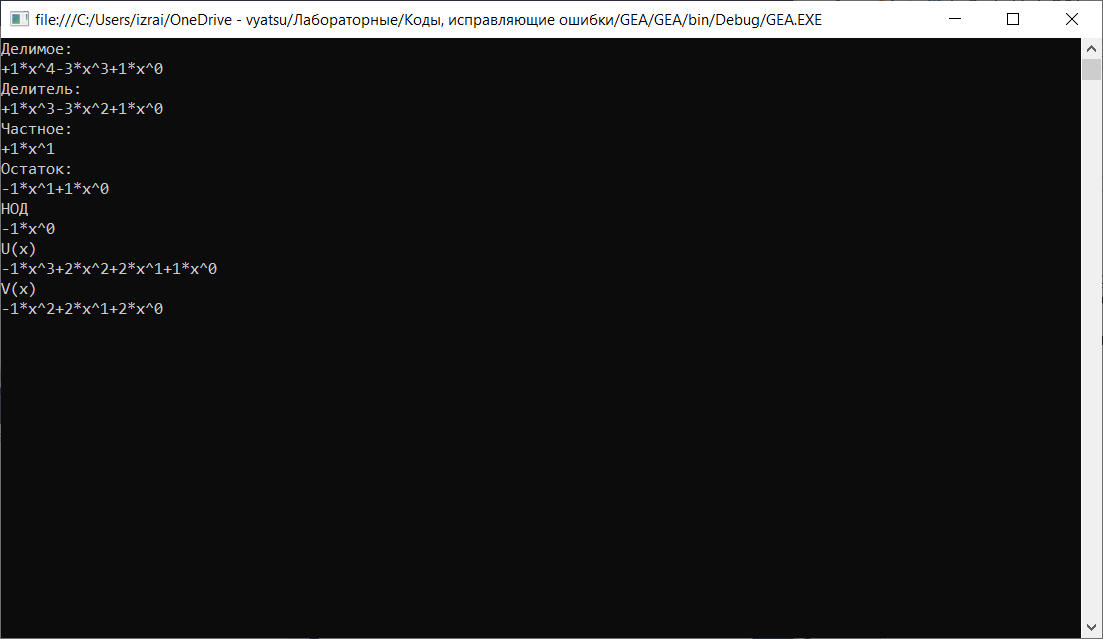
Пусть a и b — целые числа, не равные одновременно нулю, и последовательность чисел

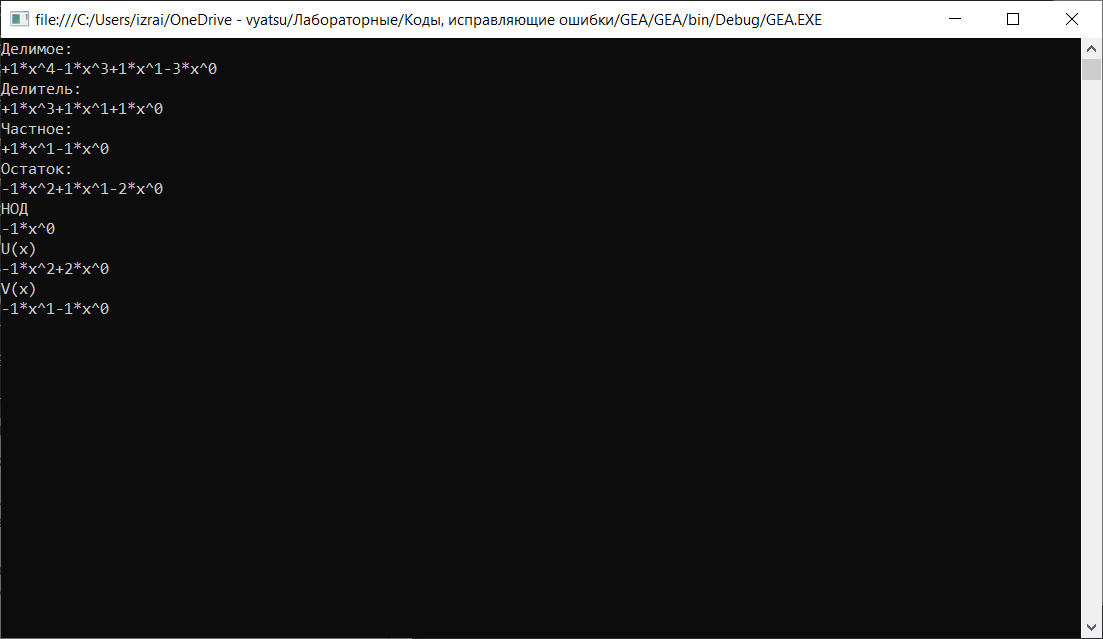
определена тем, что каждое — это остаток от деления предпредыдущего числа на предыдущее, а предпоследнее делится на последнее нацело.

Тогда НОД(a, b), наибольший общий делитель a и b, равен , последнему ненулевому члену этой последовательности

Алгоритм Евклида и расширенный алгоритм Евклида естественным образом обобщается на [кольцо многочленов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D1%8C%D1%86%D0%BE_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD%D0%BE%D0%B2) k[x] от одной переменной над произвольным [полем](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_(%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0)) k, поскольку для таких многочленов определена операция деления с остатком. При выполнении алгоритма Евклида для многочленов аналогично алгоритму Евклида для целых чисел получается последовательность полиномиальных остатков

**Результаты выполнения программы:**



****

**Вывод:** былиизучены и получены практические навыки использования алгоритма Евклида для многочленов.

**Листинг:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace GEA

{

class Program

{

const double eps = 0.0001;

public static class MyMath

{

public static double[] Multiplication(List<double> polynom1, List<double> polynom2)

{

double[] result = new double[polynom1.Count + polynom2.Count - 1];

for (int i = 0; i < polynom1.Count; i++)

for (int j = 0; j < polynom2.Count; j++)

{

if (result[i + j] != 0)

{

result[i + j] += polynom1[i] \* polynom2[j];

}

else

result[i + j] = polynom1[i] \* polynom2[j];

}

return result;

}

public static void Deconv(List<double> dividend, List<double> divisor, out double [] quotient, out List<double> remainder)

{

while (dividend.Last() == 0)

{

dividend.RemoveAt(divisor.Count - 1);

}

while (divisor.Last() == 0)

{

divisor.RemoveAt(divisor.Count - 1);

}

remainder = new List<double>(dividend);

quotient = new double[remainder.Count - divisor.Count + 1];

for (int i = 0; i < quotient.Length; i++)

{

double coeff = remainder[remainder.Count - i - 1] / divisor.Last();

quotient[quotient.Length - i - 1] = coeff;

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

{

remainder[remainder.Count - i - j - 1] -= coeff \* divisor[divisor.Count - j - 1];

}

}

}

}

// деление многочлена с остатком

// предаставить в виде умножения + остаток

// Необходимо найти их НОД и представить его в линейном виде

// https://lektsii.org/14-31260.html

static void Main(string[] args)

{

List<double> dividend = new List<double> {1, 0, 0, -3, 1};

List<double> divisor = new List<double>{1, 0, -3, 1};

double[] quotient;

List<double> remainder;

#region Вывод dividend = divisor \* quotient + remainder

Console.WriteLine("Делимое:");

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

{

if (dividend[dividend.Count - i - 1] != 0)

{

Console.Write("{0}{1}\*x^{2}", dividend[dividend.Count - i - 1] >= 0 ? "+" : "", dividend[dividend.Count - i - 1], dividend.Count - i - 1);

}

}

Console.WriteLine("\nДелитель:");

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

{

if (divisor[divisor.Count - i - 1] != 0)

{

Console.Write("{0}{1}\*x^{2}", divisor[divisor.Count - i - 1] >= 0 ? "+" : "", divisor[divisor.Count - i - 1], divisor.Count - i - 1);

}

}

MyMath.Deconv(dividend, divisor, out quotient, out remainder);

Console.WriteLine("\nЧастное:");

for (int i = 0; i < quotient.Length; i++)

{

if (quotient[quotient.Length - i - 1] != 0)

{

Console.Write("{0}{1}\*x^{2}", quotient[quotient.Length - i - 1] >= 0 ? "+" : "", quotient[quotient.Length - i - 1], quotient.Length - i - 1);

}

}

Console.WriteLine("\nОстаток:");

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

{

if (remainder[remainder.Count - i - 1] != 0)

{

Console.Write("{0}{1}\*x^{2}", remainder[remainder.Count - i - 1] >= 0 ? "+" : "", remainder[remainder.Count - i - 1], remainder.Count - i - 1);

}

}

#endregion

List<double> s1 = new List<double>(quotient);

List<double> temp = new List<double>(remainder);

bool flag = false;

dividend = new List<double>(divisor);

divisor = new List<double>(remainder);

MyMath.Deconv(dividend, divisor, out quotient, out remainder);

List<double> s2 = new List<double>(quotient);

while (true)

{

while (remainder.Last() == 0 || Math.Abs(remainder.Last()) < eps)

{

remainder.RemoveAt(remainder.Count - 1);

if (remainder.Count == 0)

{

remainder = new List<double>(temp);

flag = true;

break;

}

}

if (flag == true)

break;

dividend = new List<double>(divisor);

divisor = new List<double>(remainder);

temp = new List<double>(remainder);

MyMath.Deconv(dividend, divisor, out quotient, out remainder);

}

Console.WriteLine("\nНОД");

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

{

if (remainder[remainder.Count - i - 1] != 0)

{

Console.Write("{0}{1}\*x^{2}", remainder[remainder.Count - i - 1] >= 0 ? "+" : "", remainder[remainder.Count - i - 1], remainder.Count - i - 1);

}

}

// F(x) = x^4 - x^3 + x - 3

// G(x) = x^3 + x + 1

// D(x) = -1

// F(x) \* U(x) + G(x) \* V(x) = D(x)

// D(x) = F(x) \* (1 + S1\*S2) - G(x) \* (S2)

double[] U = MyMath.Multiplication(s1, s2);

U[0] += 1;

List<double> V = new List<double>(s2);

Console.WriteLine("\nU(x)");

for (int i = 0; i < U.Length; i++)

{

if (U[U.Length - i - 1] != 0)

{

Console.Write("{0}{1}\*x^{2}", U[U.Length - i - 1] >= 0 ? "+" : "", U[U.Length - i - 1], U.Length - i - 1);

}

}

Console.WriteLine("\nV(x)");

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

{

if (V[V.Count - i - 1] != 0)

{

Console.Write("{0}{1}\*x^{2}", V[V.Count - i - 1] >= 0 ? "+" : "", V[V.Count - i - 1], V.Count - i - 1);

}

}

Console.ReadKey();

}

}

}