МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 3

на тему: «Функциональное тестирование программного обеспечения»

по дисциплине: «Качество и тестирование программного обеспечения»

Вариант 8

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71ПГ

Проверили: Ужаринский А.Ю., Конюхова О.В.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

Орел, 2020 г.

**Задание на лабораторную работу:**

Решить алгебраическое уравнение 3-й степени (кубическое уравнение) ax3 + bx2 + cx + d. Корни приведенного уравнения рассчитать по формулам Кардано.

Причины:

1. a = 0
2. b = 0
3. c = 0
4. d = 0
5. a != 0
6. b != 0
7. c != 0
8. d != 0

Следствия:

101 – нет корней

102 – 1 корень

103 – 2 корня

104 – 3 корня

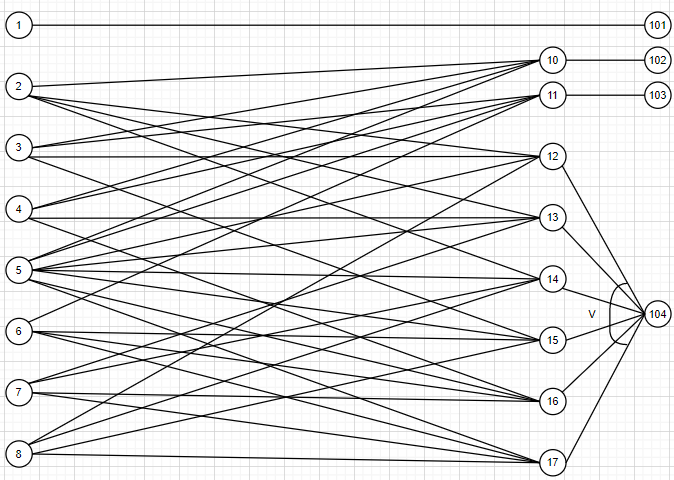


Рисунок 1 – Граф причинно-следственных связей

Список тестов:

1. ИД: a = 0

ОР: нет корней

1. ИД: a != 0, b = 0, c = 0, d = 0

ОР: 1 корень

1. ИД: a != 0, b != 0, c = 0, d = 0

ОР: 2 корня

1. ИД: a != 0, b = 0, c = 0, d != 0

ОР: 3 корня

1. ИД: a != 0, b = 0, c != 0, d = 0

ОР: 3 корня

1. ИД: a != 0, b = 0, c != 0, d != 0

ОР: 3 корня

1. ИД: a != 0, b != 0, c = 0, d != 0

ОР: 3 корня

1. ИД: a != 0, b != 0, c != 0, d = 0

ОР: 3 корня

1. ИД: a != 0, b != 0, c != 0, d != 0

ОР: 3 корня

**Код**

**«Program.cs»**

using System;using System.Collections;using System.Collections.Generic;

using System.Linq;using System.Numerics;

namespace Lab\_3{

class Program {

static void Main(string[] args) {

Console.WriteLine("Enter values: ");

Console.Write("a:");

double a = Convert.ToDouble(Console.ReadLine());

if (a == 0) {

Console.WriteLine("Error: a == 0");

return;

}

Console.Write("b:");

double b = Convert.ToDouble(Console.ReadLine());

Console.Write("c:");

double c = Convert.ToDouble(Console.ReadLine());

Console.Write("d:");

double d = Convert.ToDouble(Console.ReadLine());

Console.WriteLine($"Your expression: {a}x^3 + {b}x^2 + {c}x + {d}");

double p = (3 \* a \* c - Math.Pow(b, 2)) / (3 \* Math.Pow(a, 2));

double q = (2 \* Math.Pow(b, 3) - 9 \* a \* b \* c + 27 \* Math.Pow(a, 2) \* d) / (27 \* Math.Pow(a, 3));

Console.WriteLine($"New expression: y^3 + {p}y + {q}");

double Q = Math.Pow((p / 3), 3) + Math.Pow((q / 2), 2);

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

Complex alphaComplex, betaComplex;

List<Complex> pairAB = AlphaBetaReturn(p, q, Q).ToList();

alphaComplex = pairAB[0];

betaComplex = pairAB[1];

double alpha = Math.Sqrt(Math.Pow(alphaComplex.Real, 2) + Math.Pow(alphaComplex.Imaginary, 2));

double beta = Math.Sqrt(Math.Pow(betaComplex.Real, 2) + Math.Pow(betaComplex.Imaginary, 2));

Console.WriteLine($"Alpha: {alpha}");

Console.WriteLine($"Beta: {beta}");

Complex y1 = alpha + beta + -b / (3 \* a);

Complex y2 = new Complex(-((alpha + beta) / 2) + -b / (3 \* a), ((alpha - beta) / 2) \* Math.Sqrt(3));

Complex y3 = new Complex(-((alpha + beta) / 2) + -b / (3 \* a), -((alpha - beta) / 2) \* Math.Sqrt(3));

Console.WriteLine("Roots:");

Console.WriteLine($"y1: {y1} \n" +

$"y2: {y2} \n" +

$"y3: {y3}");

//Console.ReadKey();

}

public static List<Complex> AlphaBetaReturn(double p, double q, double Q)

{

List<Complex> result = new List<Complex>();

Complex arithmeticAlpha, arithmeticBeta;

if (Q >= 0)

{

arithmeticAlpha = new Complex(Math.Cbrt(-(q / 2) + Math.Sqrt(Q)), 0);

arithmeticBeta = new Complex(Math.Cbrt(-(q / 2) - Math.Sqrt(Q)), 0);

}

else

{

Complex Qpositive = new Complex(0, Math.Sqrt(Math.Abs(Q)));

Complex Qnegative = new Complex(0, -Math.Sqrt(Math.Abs(Q)));

Complex Q1 = new Complex(Math.Cbrt(-(q / 2) + Qpositive.Imaginary), 0 );

Complex Q2 = new Complex((Math.Cbrt(-(q / 2) + Qpositive.Imaginary)) \* (-0.5), Math.Cbrt(-(q / 2) + Qpositive.Imaginary)) \* (Math.Sqrt(3) / 2 );

Complex Q3 = new Complex((Math.Cbrt(-(q / 2) + Qpositive.Imaginary)) \* (-0.5), -(Math.Cbrt(-(q / 2) + Qpositive.Imaginary)) \* (Math.Sqrt(3) / 2 ));

arithmeticAlpha = new Complex(Math.Cbrt(-(q / 2) + Qpositive.Imaginary), 0);

arithmeticBeta = new Complex(Math.Cbrt(-(q / 2) - Qpositive.Imaginary), 0);

}

Complex alpha2 = new Complex(arithmeticAlpha.Real \* (-0.5), arithmeticAlpha.Real \* (Math.Sqrt(3) / 2));

Complex alpha3 = new Complex(arithmeticAlpha.Real \* (-0.5), -arithmeticAlpha.Real \* (Math.Sqrt(3) / 2));

Complex beta2 = new Complex(arithmeticBeta.Real \* (-0.5), arithmeticBeta.Real \* (Math.Sqrt(3) / 2));

Complex beta3 = new Complex(arithmeticBeta.Real \* (-0.5), -arithmeticBeta.Real \* (Math.Sqrt(3) / 2));

List<Complex> alphas = new List<Complex>() {arithmeticAlpha, alpha2, alpha3};

List<Complex> betas = new List<Complex>() {arithmeticBeta, beta2, beta3};

int a = 1;

Console.WriteLine();

foreach (var alpha in alphas)

{

int b = 1;

foreach (var beta in betas)

{

Console.WriteLine($"A: {a} B: {b}");

Console.WriteLine($"Alpha: {alpha} \n" +

$"Beta: {beta}");

Complex ab = alpha \* beta;

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

double multy = Math.Sqrt(Math.Pow(ab.Real, 2) + Math.Pow(ab.Imaginary, 2));

Console.WriteLine($"Alpha \* Beta: {multy} \n" +

$"-p / 3: {-p / 3}");

if (Complex.Abs(multy - (-p / 3.0)) < 0.01)

{

Console.WriteLine("Equals");

result.Add(alpha);

result.Add(beta);

Console.WriteLine();

return result;

}

Console.WriteLine();

b++;

}

a++;

}

result.Add(1);

result.Add(2);

return result;

}

}

}