

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Разработка программных систем | | |
| Курс | II | | | Группа | 404 |

Отчёт по контрольной работе № 3

Вариант № 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 403 |  |  |  | Азаров Даниил Константинович |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |

# 1. Постановка задачи

Студент выполняет задание обычной или повышенной сложности. Необходимо написать приложение с использованием технологии WinForms для построения графика функции Декартового листа и вывода таблицы значений функции. Пользователь задает правую и левую границу, шаг, коэффициенты (при их наличии). При невозможности построить график функции в заданном интервале пользователю выдается предупреждение об этом с предложением сменить границы построения. Если график функции из-за коэффициентов вырождается в точку или не может быть построен пользователь также видит предупреждение. Необходимо выполнить задание из вариантов обычной сложности добавив в программу возможность экспорта исходных данных и результатов расчета в MS Excell

# 2. Исходные данные

В качестве исходных данных программа использует три, введенных пользователем, числа – коэффициент “a”, размер отображения, шаг.

# 3. Особые ситуации

Если коэффициент равен нулю, график не существует.

# 4. Математические методы и алгоритмы решения задачи

Для нахождения координат точек графика использовались следующие формулы:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
|  | (3) |

# 5. Форматы представления данных

Таблица 1 — Классы, используемые в программе.

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Имя переменной** | **Тип** | **Описание** |
| FoliumOfDescartes | - | - | Рассчитывает точки графика Декартового листа |
| **Метод** | **Тип** | **Описание** |
| PointsArray | PointF[] | Возвращает массив точек графика Декартового листа |

# 6. Структура программы

Таблица 2 — Модули, на которые разбита программа.

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Main | Запуск программы |
| Graphics | Модуль, содержащий реализацию класса FoliumOfDescartes. |
| UI | Модуль, содержащий реализацию класса UI. |

На рисунке 1 представлена блок-схема алгоритма, представленная в программе:

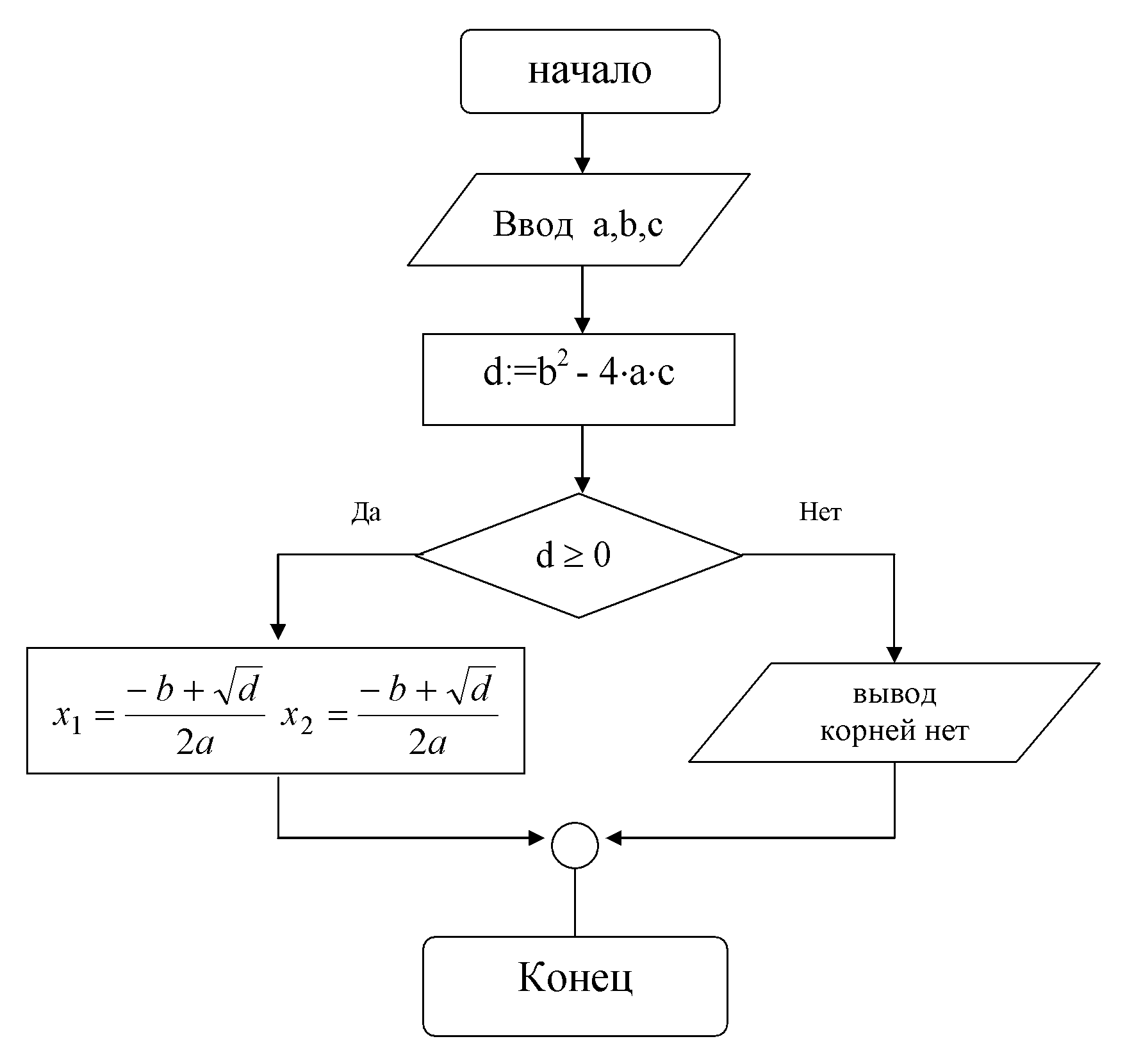


Рисунок 1 — блок-схема алгоритма.

# 7. Описание хода выполнения лабораторной работы

1. Были реализованы пользовательский интерфейс.
2. Были написан класс, возвращающий точки графика.
3. Была исправлена функция, возвращающая точки графика в классе.
4. Было добавлено меню.
5. Была добавлена форма вывода информации о программе.
6. Была добавлена возможность сохранения в файл.
7. Была добавлена возможность ввода из файла.
8. Были написаны модульные тесты.
9. Был написан отчёт.

# 8. Результаты работы программы

На рисунках 2 - 3 представлены скриншоты работы программы.

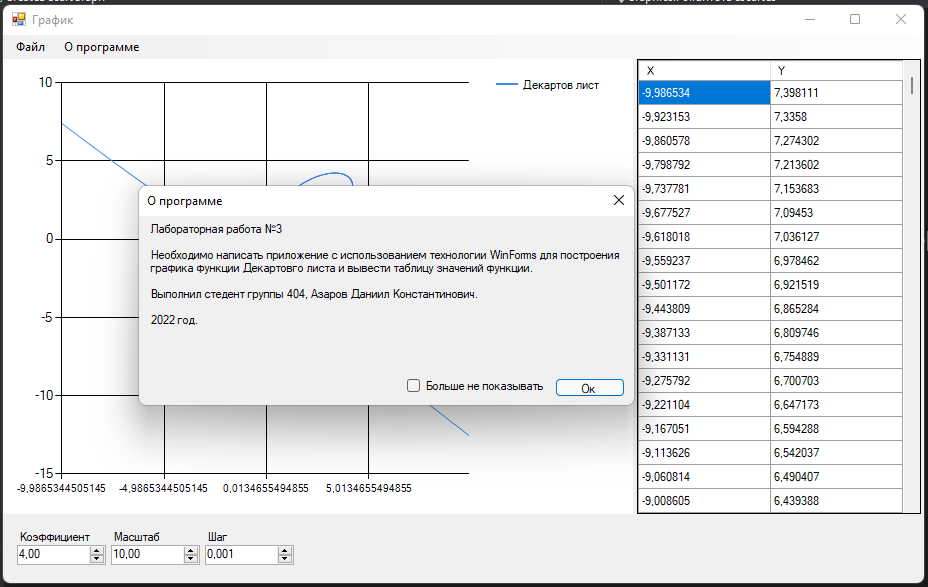


Рисунок 2 — Начало работы программы.

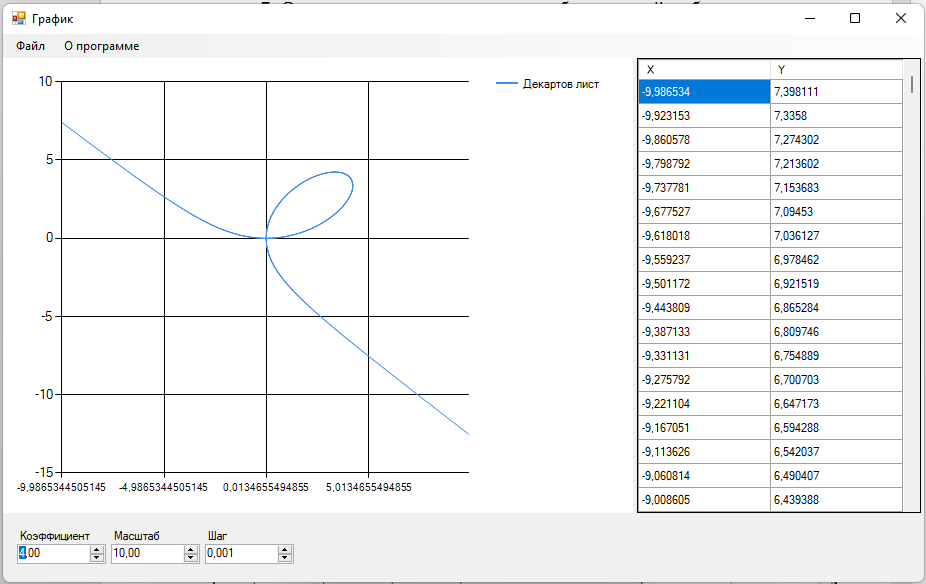


Рисунок 2 — Выбор параметров.

# 9. Исходный текст программы

[Начало FoliumOfDescartes.cs ---]

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Graphics

{

public static class FoliumOfDescartes

{

public static PointF[] PointsArray(double a = 4, double xBorder = 10, double step = 0.001)

{

if (step <= 0)

{

throw new ArgumentException("Step cannot be less than or equal to zero.");

}

List<PointF> firstQuarterChart = new List<PointF>();

List<PointF> secondQuarterChart = new List<PointF>();

List<PointF> thirdQuarterChart = new List<PointF>();

List<PointF> fourthQuarterChart = new List<PointF>();

for (double angle = 0; angle < Math.PI; angle += step)

{

double r = (2 \* a \* Math.Cos(angle) \* Math.Sin(angle)) /

(Math.Pow(Math.Cos(angle), 3) + Math.Pow(Math.Sin(angle), 3));

float x = (float)(r \* Math.Cos(angle));

float y = (float)(r \* Math.Sin(angle));

if (Math.Abs(x) > xBorder)

{

continue;

}

if (x >= 0 && y >= 0)

{

firstQuarterChart.Add(new PointF(x, y));

}

else if(x <= 0 && y >= 0)

{

secondQuarterChart.Add(new PointF(x, y));

}

else if (x >= 0 && y <= 0)

{

fourthQuarterChart.Add(new PointF(x, y));

}

else if (y <= 0 && x <= 0)

{

thirdQuarterChart.Add(new PointF(x, y));

}

}

List<PointF> graphicPoints;

if (a >= 0)

{

graphicPoints = new List<PointF>(secondQuarterChart);

graphicPoints.AddRange(firstQuarterChart);

graphicPoints.AddRange(fourthQuarterChart);

}

else

{

graphicPoints = new List<PointF>(fourthQuarterChart);

graphicPoints.AddRange(thirdQuarterChart);

graphicPoints.AddRange(secondQuarterChart);

}

return graphicPoints.ToArray();

}

}

}

[--- Конец FoliumOfDescartes.cs]

[Начало CreateDecartGraphUnitTest.cs ---]

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using System.Drawing;

using Graphics;

namespace CreateDecartGraphTesting

{

[TestClass]

public class CreateDecartGraphUnitTest

{

[TestMethod]

public void PointsArrayTest()

{

// Arrange

double a = 4;

double xBorder = 10;

double step = 0.001;

PointF[] correctPoints = CorrectPointsArrayForGraphicFoliumOfDescartes(a, xBorder, step);

// Act

PointF[] resultPoints = FoliumOfDescartes.PointsArray(a, xBorder, step);

// Assert

Assert.AreEqual(correctPoints.Length, resultPoints.Length);

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

{

Assert.AreEqual(correctPoints[i], resultPoints[i]);

}

}

private static PointF[] CorrectPointsArrayForGraphicFoliumOfDescartes(double a = 4, double xBorder = 10, double step = 0.001)

{

List<PointF> firstQuarterChart = new List<PointF>();

List<PointF> secondQuarterChart = new List<PointF>();

List<PointF> thirdQuarterChart = new List<PointF>();

List<PointF> fourthQuarterChart = new List<PointF>();

for (double angle = 0; angle < Math.PI; angle += step)

{

double r = (2 \* a \* Math.Cos(angle) \* Math.Sin(angle)) /

(Math.Pow(Math.Cos(angle), 3) + Math.Pow(Math.Sin(angle), 3));

float x = (float)(r \* Math.Cos(angle));

float y = (float)(r \* Math.Sin(angle));

if (Math.Abs(x) > xBorder)

{

continue;

}

if (x >= 0 && y >= 0)

{

firstQuarterChart.Add(new PointF(x, y));

}

else if (x <= 0 && y >= 0)

{

secondQuarterChart.Add(new PointF(x, y));

}

else if (x >= 0 && y <= 0)

{

fourthQuarterChart.Add(new PointF(x, y));

}

else if (y <= 0 && x <= 0)

{

thirdQuarterChart.Add(new PointF(x, y));

}

}

List<PointF> graphicPoints;

if (a >= 0)

{

graphicPoints = new List<PointF>(secondQuarterChart);

graphicPoints.AddRange(firstQuarterChart);

graphicPoints.AddRange(fourthQuarterChart);

}

else

{

graphicPoints = new List<PointF>(fourthQuarterChart);

graphicPoints.AddRange(thirdQuarterChart);

graphicPoints.AddRange(secondQuarterChart);

}

return graphicPoints.ToArray();

}

}

}

[--- Конец CreateDecartGraphUnitTest.cs]