Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных

систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине «**Разработка приложений в визуальных средах**»

на тему: «**Программная модель гармонических колебаний маятника**» 1

Выполнил

студент гр.10701219 Харлап С.А

Принял

доц. Гурский Н.Н.

Минск 2021

Белорусский национальный технический университет

Кафедра программного обеспечения информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту (работе)**

**по дисциплине «**Разработка приложений в визуальных средах»

Тема: «Программная модель гармонических колебаний маятника»

**Исполнитель**: Харлап С. А.

**Студент 2 курса 10701219 группы**

**Руководитель**: доц. Гурский Н.Н.

Минск 2021

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc70963261)

[1 МАТЕМАТИЧЕСКАЯ ФОРМУЛИРОВКА ЗАДАЧИ 4](#_Toc70963262)

[2 ОПИСАНИЕ ПРОГРАММЫ 5](#_Toc70963263)

[2.1 Структурная схема программы 5](#_Toc70963264)

[2.2 Описание разработанного класса 6](#_Toc70963265)

[2.3 Основные возможности программы 7](#_Toc70963266)

[2.4 Средства использования сервисов, предоставляемых MicrosoftOffice 7](#_Toc70963267)

[4 МЕТОДИКА ИСПЫТАНИЙ 11](#_Toc70963268)

[ЗАКЛЮЧЕНИЕ 13](#_Toc70963269)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 14](#_Toc70963270)

ПРИЛОЖЕНИЕ А………………………………………………………………………15

# ВВЕДЕНИЕ

Компьютерная модель - компьютерная программа, работающая на отдельном компьютере, суперкомпьютере или множестве взаимодействующих компьютеров (вычислительных узлов), реализующая абстрактную, то есть информационную модель некоторой системы. Компьютерные модели стали обычным инструментом численно-математического моделирования и применяются в физике, астрофизике, механике, химии, биологии, экономике, социологии, метеорологии, других науках и прикладных задачах в различных областях радиоэлектроники, машиностроения, автомобилестроения и проч. Компьютерные модели используются для получения новых знаний о моделируемом объекте или для приближенной оценки поведения систем, слишком сложных для логико-аналитического исследования.

Компьютерное моделирование является одним из эффективных методов изучения сложных систем. Компьютерные модели проще и удобнее исследовать в силу их возможности проводить так называемые "вычислительные эксперименты", которые на самом деле экспериментами не являются, так как информация берется не из физического эксперимента, физической реальности, а из модельного представления о ней, проводят в тех случаях, когда реальные эксперименты затруднены из-за финансовых или физических препятствий, или могут дать непредсказуемо опасный результат. В случае корректной логики и корректной формализации на этапе создания компьютерных моделей имеется возможность выявить основные факторы, определяющие количественные свойства изучаемого объекта-оригинала (или целого класса объектов), в частности, исследовать отклик моделируемой физической системы на изменения ее параметров и начальных условий.

Построение компьютерной модели базируется на абстрагировании от конкретной природы явлений или изучаемого объекта-оригинала и состоит из двух этапов — сначала создание качественно-логической, а затем и количественно-математической модели. Компьютерное же моделирование заключается в проведении серии "вычислительных экспериментов" на компьютере, целью которых является анализ на внутреннюю непротиворечивость модели и получение количественных данных о процессе функционирования модели.

Затем исследователь производит интерпретацию, то есть объяснение этих количественных результатов и их содержательное, то есть неформальное сопоставление с реальным поведением изучаемого объекта, а также частое и многократное последующее уточнение модели и т. д.

Целью курсовой работы является разработка приложения отображения гармонических колебаний маятника. Для работы был выбран язык программирования C# (VunForms), который даёт возможность быстрого создания приложений с графическим интерфейсом, а также позволяет реализовать концепции ООП.

# 1 МАТЕМАТИЧЕСКАЯ ФОРМУЛИРОВКА ЗАДАЧИ

Треугольник – геометрическая фигура, образованная тремя отрезками, соединяющими три точки, которые не лежат на одной прямой. Прямоугольный треугольник имеет прямой угол, равный 90°. Если вычертить треугольник на плоскости, то, взяв середины сторон за вершины нового треугольника, то можно рекурсивно получить изображение, составленное из треугольников.

Подобные треугольники — это треугольники, у которых углы равны, а стороны пропорциональны. В программе все треугольники будут подобными, за счёт этого и будет получаться изображение.

Функция рекурсивна, если в ней содержится одно или несколько обращений к самой себе или к другим функциям, в которых есть обращение к такой функции. При входе в обычную функцию выход из неё всегда происходит раньше, чем повторный вход, но для рекурсивной функции это необязательно.

# 2 ОПИСАНИЕ ПРОГРАММЫ

# 2.1 Структурная схема программы

Структурно программа состоит из главной формы и второстепенных форм, одна из которых служит для показа изображения, а вторая является оригинальной заставкой (см. рисунок 2.1). Программа содержит некоторое количество компонентов, предназначенных для взаимодействия с Word, вызова информации о программе, формирования изображения и создания самого треугольника.

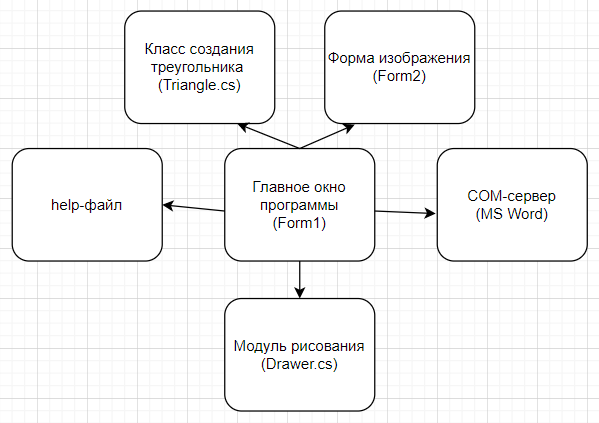
****

Рисунок 2.1 – Структурная схема связей модулей программы.

# 2.2 Описание разработанного класса

В результате анализа предметной области для её описания и моделирования был разработан класс Triangle, инкапсулирующий поля, методы и свойства, применимые к рассматриваемому объекту.

public double A {get;set;} //свойство, обеспечивающее доступ к полю, содержащему величину первого катета треугольника;

public double B {get;set;} //свойство, обеспечивающее доступ к полю, содержащему величину второго катета треугольника;

public double C {get;set;} //свойство, обеспечивающее доступ к полю, содержащему величину гипотенузы треугольника;

public List<Point> TrianglePoints {get;} //свойство, дающее доступ ко списку точек треугольника;

public Triangle (double firstCathet, double secondCathet, Point start){} //конструктор для создания объекта класса Triangle;

public void Draw(Pen pen, List<Point> points, Graphics gr){} //метод для рисования незакрашенных треугольников;

public void FillTriangles(Brush br, Point[] points, Graphics gr){} //метод для рисования закрашенных треугольников;

public List<Point> GetPoints(double firstCathet, double secondCathet, Point startPoint){} //метод получения списка точек треугольника;

# 2.3 Основные возможности программы

Для взаимодействия с программой пользователю предоставлена главное окно (Form1). Функционал, который предоставляет форма:

* Поля для ввода величин сторон треугольника.
* Поле для ввода глубины рекурсии
* Вывод данных в:
  + Блокнот
  + MS Word
* Вызов помощи окна помощи.
* Выбор типа треугольников (закрашенный или не закрашенный)

Пользователь может использовать кнопку «Рисовать», которая вызывает метод. Данный метод рисует треугольники в новой форме.

При выборе кнопки «Файл» и «Сохранить данные» у пользователя есть выбор между MSWord и блокнот.

При использовании кнопки «Помощь» вызывается help-файл с руководством пользователя и описанием математической модели задачи.

# 2.4 Средства использования сервисов, предоставляемых MicrosoftOffice

Программа обеспечивает возможность вывода данных не только на экран внутри самого приложения, но и с помощью средств MSOffice. Программа обеспечивает взаимодействие с серверами автоматизации приложений MSWord.

* Метод OfficeWordOutput() служит для вывода информации в MS Word вместе с полученным изображением.

**3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Для запуска приложения необходимо вызвать файл SimilarTrianglesMaker.exe. После вызова приложения появится оригинальная заставка, изображённая на рисунке 3.1.

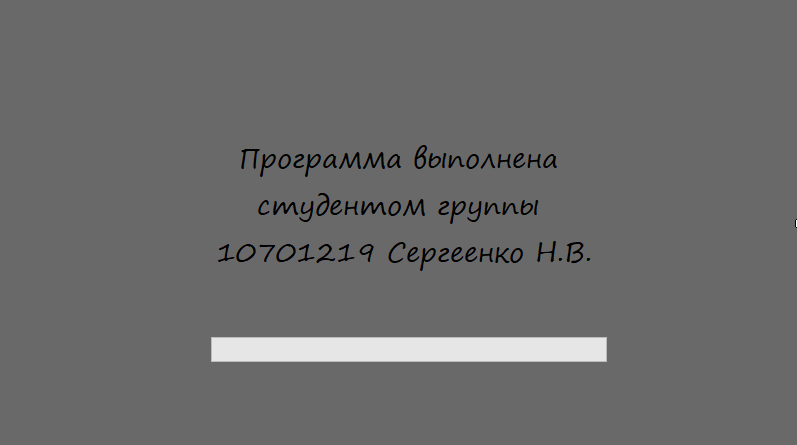


Рисунок 3.1 – Приветственное окно

После заставки появляется главное окно программы (см. рисунок 3.2).

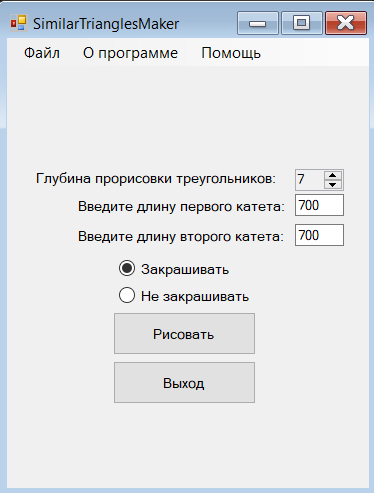


Рисунок 3.2 - Главное окно программы

Главная форма имеет:

* Главное меню (см. рисунок 3.3)
* Области ввода данных
* Управляющие кнопки (см. рисунок 3.8)



Рисунок 3.3 – Пункты главного меню

Пункт меню «Файл» имеет свои подпункты (см. рисунок 3.4).

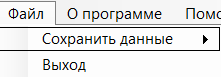


Рисунок 3.4 – Подпункты элемента меню «Файл»

Подпункт пункта «Сохранить данные» имеет свои подпункты (см. рисунок 3.5).



Рисунок 3.5 – Подпункты пункта «Сохранить данные»

Подпункты меню «Файл» экспортируют данные в Word и блокнот.

Пункт меню «О программе» выводит информацию о создателе программы.

Пункт меню «Помощь» вызывает help-файл с небольшим руководством пользователя и описанием математической модели (см. рисунок 3.6)

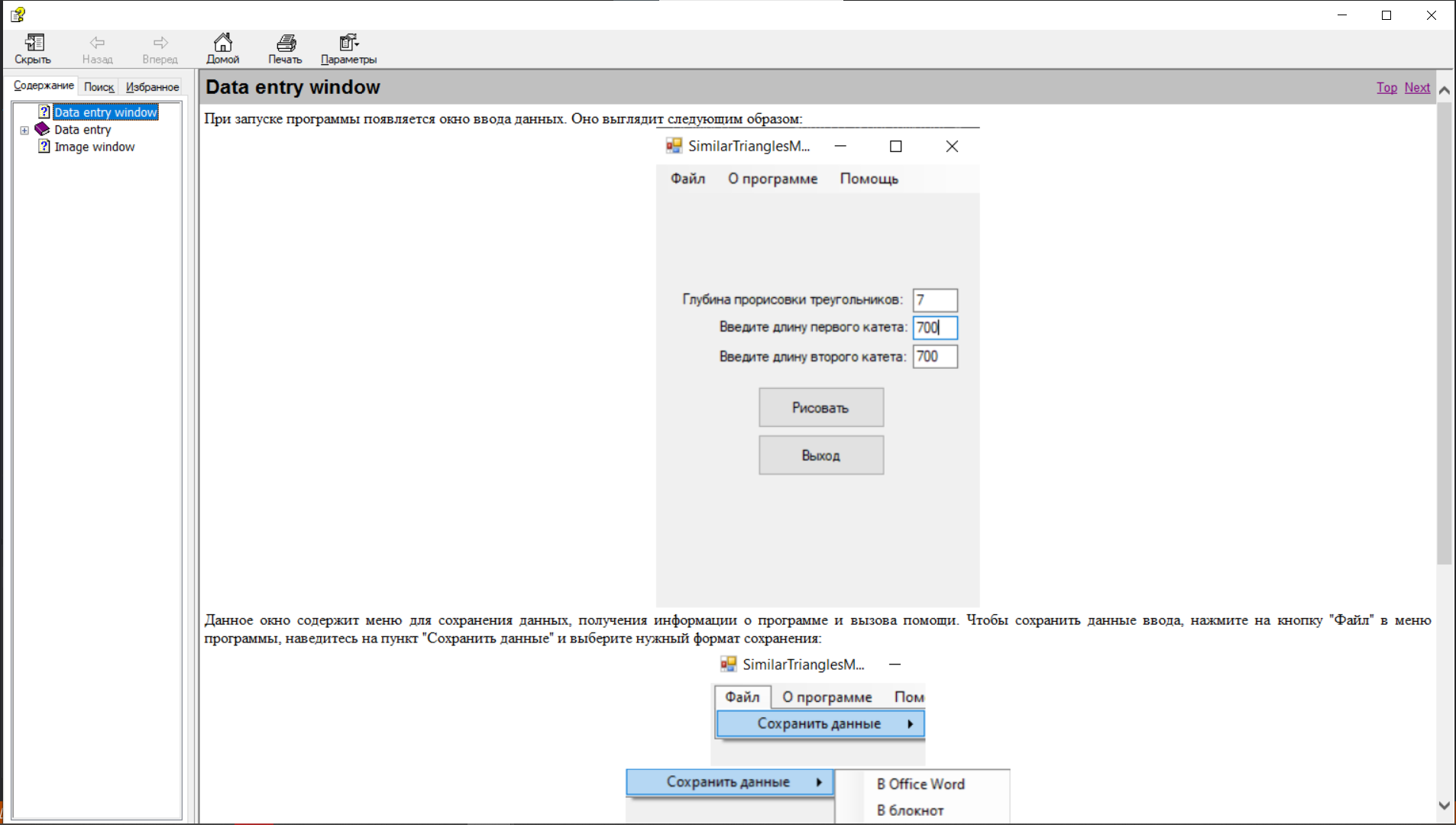


Рисунок 3.6 – Руководство пользователя, вызываемое клавишей «Помощь»

При выборе кнопки «Рисовать» открывается другая форма, на которой рисуется изображение. Форма изображена на рисунке 3.7.

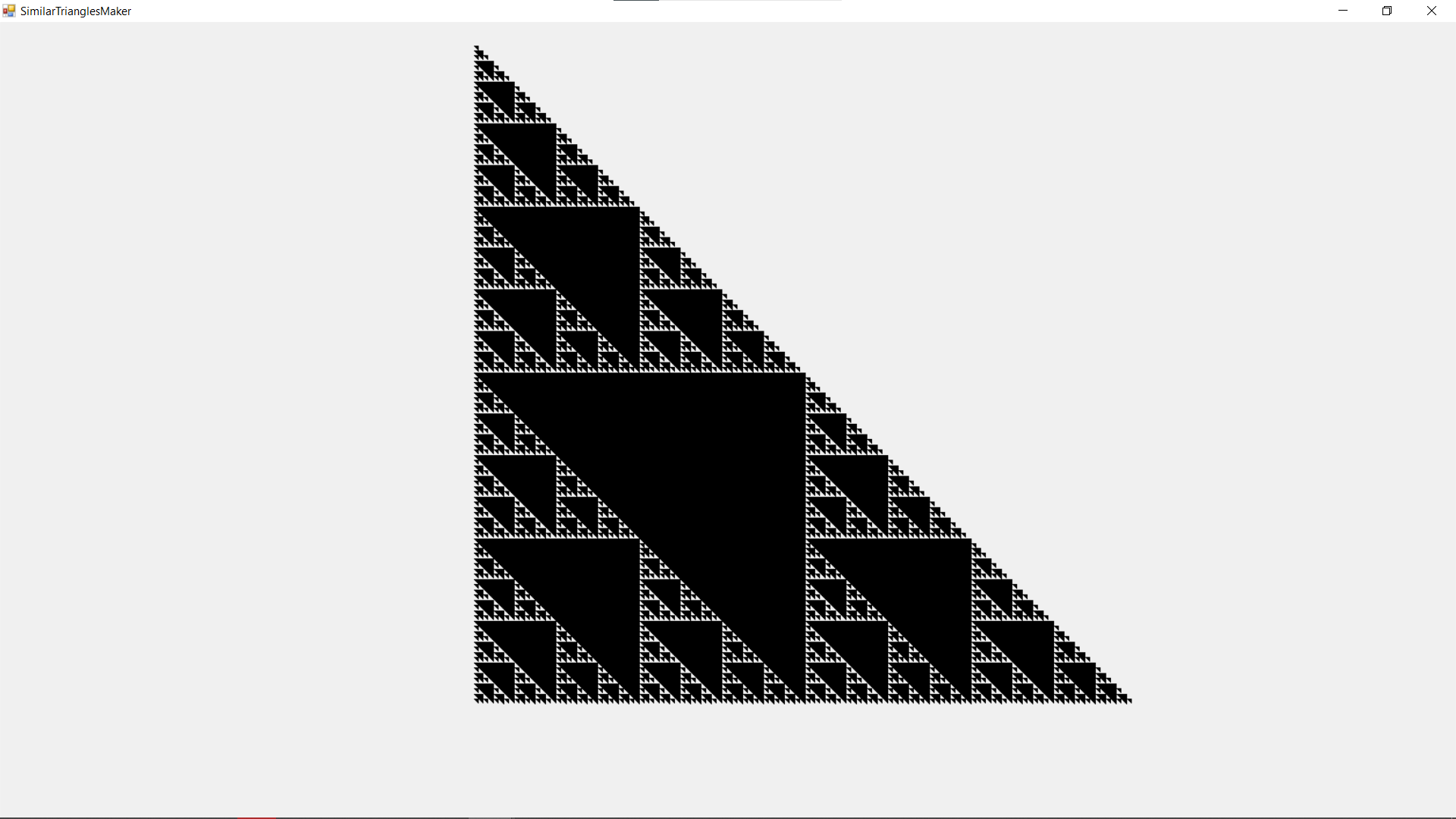


Рисунок 3.7 – Форма, показываемая после нажатия кнопки «Рисовать»

Кнопка «Выход» завершает работу приложения.

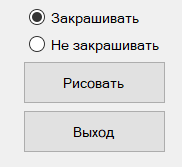


Рисунок 3.8 – управляющие кнопки

# 4 МЕТОДИКА ИСПЫТАНИЙ

Целью проведения испытаний является проверка работоспособности (надежности) программы при различных условиях её функционирования.

Программа должна обеспечивать корректность ввода исходных данных (путем осуществления соответствующих проверок и информирования пользователя о возникших неточностях в работе), а также получение непротиворечивого результата.

Для обеспечения нормальной работы программы требуется наличие необходимых динамических библиотек, а также приложений Word и Excel.

Первое испытание будет проведено при корректных данных (см. рисунок 4.1, 4.2).

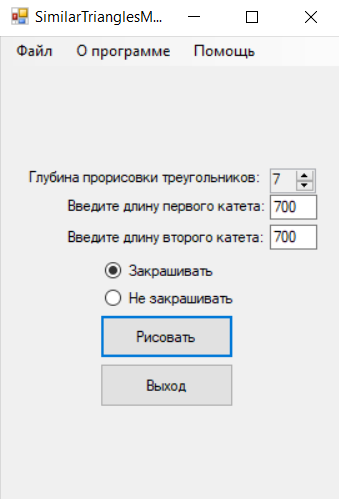


Рисунок 4.1 – Ввод корректных данных

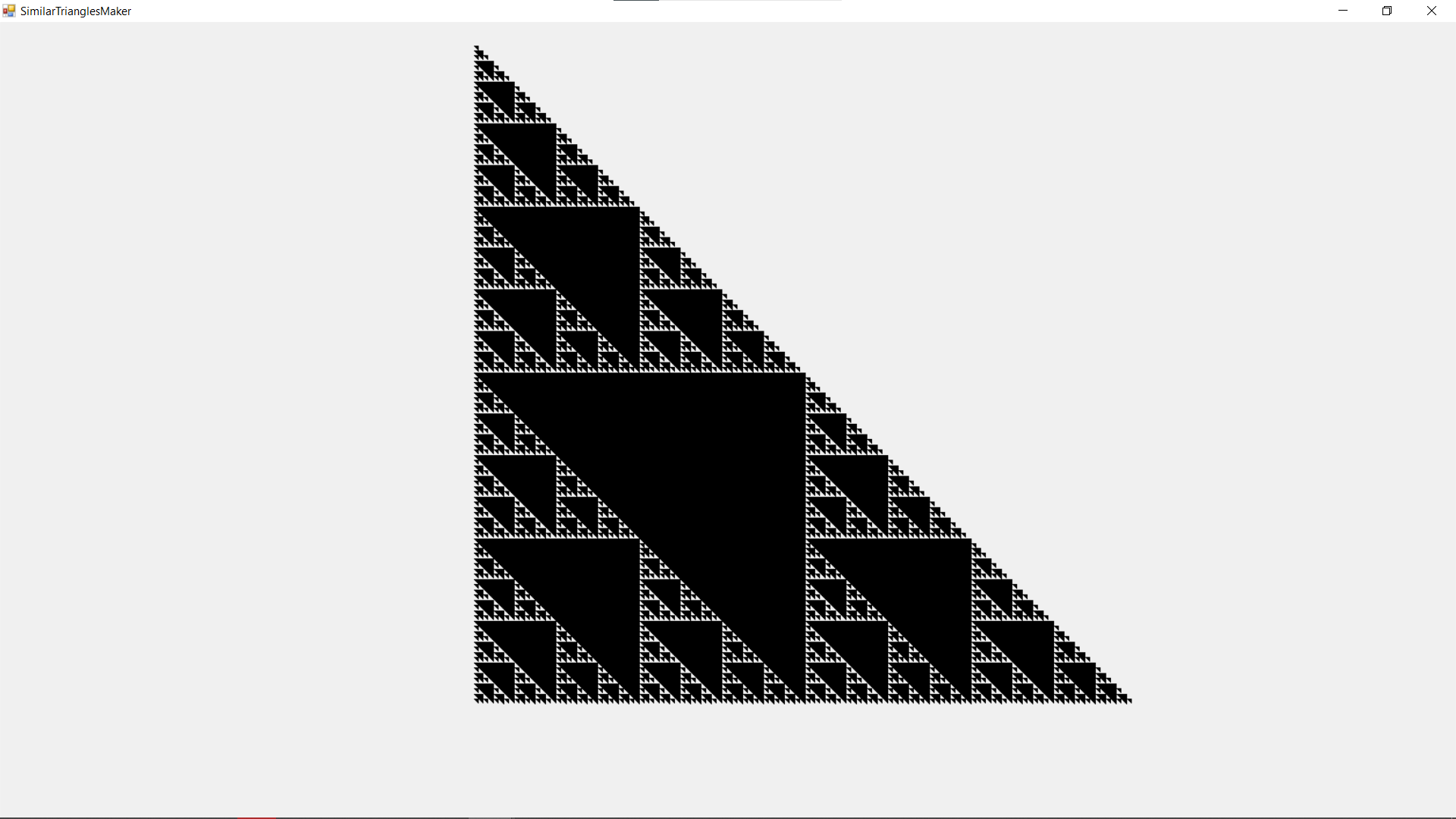


Рисунок 4.2 – Вывод второй формы в случае корректных данных.

Далее проведем проверку с некорректными параметрами, например, неверными сторонами треугольника, которые должен быть больше нуля (см. рисунок 4.3).

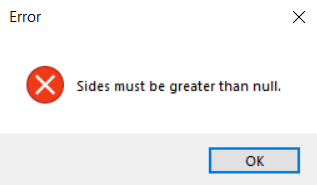


Рисунок 4.3 – Окно предупреждения о том, что стороны должны быть больше нуля.

В результате тестирования приложения ошибок обнаружено не было. Следует считать, что в целом программа протестирована, отвечает поставленным требованием и вполне работоспособна.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был приобретен опыт при разработке объектно-ориентированных программ, закреплен опыт, полученный при прохождении курса «Разработка приложений в визуальных средах», было изучено взаимодействие с серверами автоматизации приложений MicrosoftOffice.

Программа имеет удобный для пользователя интерфейс, различные формы вывода информации. Программа может использоваться для получения изображений из подобных треугольников. В дальнейшем можно доработать программу, чтобы она рисовала не только изображения из треугольников, а, например, изображения из квадратов, окружностей и других геометрических фигур.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Гринчишин Я.Т. и др. Алгоритмы и программы на Бейсике: Учеб. Пособие для студентов пед. Ин-тов по физ.-мат. Спец./Я. Т. Гринчишин, В. И. Ефимов, А. Н. Ломакович. –М.: Просвещение. 1988.-160с.
2. CLRviaC#. Программирование на платформе Microsoft .NETFramework 4.5 на языке C#. 4-е изд. – СПб.; Питер, 2013 – 896с.
3. С# для чайников.: Пер. с англ. – СПб. : ООО «Диалектика», 2019. – 608 с.

**ПРИЛОЖЕНИЕ A**

**Файл Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using Word = Microsoft.Office.Interop.Word;

using Excel = Microsoft.Office.Interop.Excel;

namespace SimilarTrianglesMaker

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Form2 form2;

private void Button1\_Click(object sender, EventArgs e) //метод для получения данных и рисования изображения

{

double a = 0, b = 0;

int c = Convert.ToInt32(domainUpDown1.SelectedItem);

if (!double.TryParse(textBox1.Text, out a) || !double.TryParse(textBox2.Text, out b))

{

MessageBox.Show("Invalid data", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else if (Convert.ToDouble(textBox1.Text) < 0 || Convert.ToDouble(textBox2.Text) < 0)

{

MessageBox.Show("Sides must be greater than null.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

if (form2 == null || form2.IsDisposed)

{

form2 = new Form2();

form2.Show();

}

Graphics gr = form2.CreateGraphics();

gr.Clear(form2.BackColor);

Triangle t = new Triangle(Convert.ToDouble(a), Convert.ToDouble(b), new Point(500, 20));

Pen pen = new Pen(Brushes.Black);

List<Point> points = t.TrianglePoints;

Drawer.Draw(pen, t, gr, points[0], points[1], points[2], c, !radioButton1.Checked);

// create the bitmap to copy the screen shot to

Bitmap bitmap = new Bitmap(1920, 1080);

// now copy the screen image to the graphics device from the bitmap

using (Graphics g = Graphics.FromImage(bitmap))

{

g.CopyFromScreen(form2.Location, Point.Empty, bitmap.Size);

}

bitmap.Save(@"D:\Разработка приложений\РПВС\Курсовой проект\SimilarTrianglesMaker\img.png", System.Drawing.Imaging.ImageFormat.Png);

}

}

private void ВБлокнотToolStripMenuItem\_Click(object sender, EventArgs e)

{

string msg = $"Triangle cathets are {textBox1.Text}, {textBox2.Text}. Recursion depth is {domainUpDown1.SelectedItem}\n";

string path = @"D:\Разработка приложений\РПВС\Курсовой проект\SimilarTrianglesMaker\text.txt";

File.AppendAllText(path, msg);

}

private void ВOfficeWordToolStripMenuItem\_Click(object sender, EventArgs e)

{

Word.Application wordapp = new Word.Application();

wordapp.Visible = true;

//Открываем существующий документ

Object filename = @"D:\Разработка приложений\РПВС\Курсовой проект\SimilarTrianglesMaker\a1.docx";

Object confirmConversions = true;

Object readOnly = false;

Object addToRecentFiles = true;

Object passwordDocument = Type.Missing;

Object passwordTemplate = Type.Missing;

Object revert = false;

Object writePasswordDocument = Type.Missing;

Object writePasswordTemplate = Type.Missing;

Object format = Type.Missing;

Object encoding = Type.Missing; ;

Object oVisible = Type.Missing;

Object openConflictDocument = Type.Missing;

Object openAndRepair = Type.Missing;

Object documentDirection = Type.Missing;

Object noEncodingDialog = false;

Object xmlTransform = Type.Missing;

Word.Document worddocument = wordapp.Documents.Open(ref filename,

ref confirmConversions, ref readOnly, ref addToRecentFiles,

ref passwordDocument, ref passwordTemplate, ref revert,

ref writePasswordDocument, ref writePasswordTemplate,

ref format, ref encoding, ref oVisible,

ref openAndRepair, ref documentDirection, ref noEncodingDialog,

ref xmlTransform);

//Получаем ссылки на параграфы документа

Word.Paragraphs wordparagraphs = worddocument.Paragraphs;

//Будем работать с первым параграфом

Word.Paragraph wordparagraph = (Word.Paragraph)wordparagraphs[1];

//Выводим текст в первый параграф

object missing = System.Reflection.Missing.Value;

wordparagraph.Range.Text += $"Triangle cathets are {textBox1.Text} and {textBox2.Text}. Recursion depth is {domainUpDown1.SelectedItem}\n";

wordparagraph.Range.InlineShapes.AddPicture(@"D:\Разработка приложений\РПВС\Курсовой проект\SimilarTrianglesMaker\img.png", ref missing, ref missing, ref missing);

}

private void ОПрограммеToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Программа разработана студентом группы 10701219 Сергеенко Н.В.", "О программе", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void ВыходToolStripMenuItem\_Click(object sender, EventArgs e)

{

Close();

}

private void Button2\_Click(object sender, EventArgs e)

{

Close();

}

private void ПомощьToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Help.ShowHelp(sender as Button, "D:\\Application\_creating\\NewProject.chm");

}

}

}

**Файл Triangle.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace SimilarTrianglesMaker

{

public class Triangle

{

private double a;

private double b;

private double c;

public double A {

get { return this.a; }

set {

if (value > 0)

{

this.a = value;

}

else { this.a = 0; }

}

} //first cathet of triangle

public double B {

get { return this.b; }

set

{

if (value > 0)

{

this.b = value;

}

else { this.b = 0; }

}

} //second cathet of triangle

public double C {

get { return this.c; }

set

{

if (value > 0)

{

this.c = value;

}

else { this.c = 0; }

}

} //hypotenuse of triangle

private Point start;

public Point Start {

get { return this.start; }

set

{

if (value.X >= 0 && value.Y >= 0)

{

this.start = value;

}

else { this.start = new Point(0,0); }

}

} //start point of triangle

private List <Point> trianglePoints = new List<Point>(); //list of triangles points

private Point[] arrayPoints = new Point[3];

public List<Point> TrianglePoints { get { return this.trianglePoints; } }

public Point[] ArrayPoints { get { return this.arrayPoints; } }

public Triangle() //default constructor

{

GetPoints(A, B, Start);

ChangeArrayPoints(TrianglePoints);

}

public Triangle(double firstCathet, double secondCathet) //constructor with two cathets

{

this.A = firstCathet;

this.B = secondCathet;

this.C = Math.Sqrt(Math.Pow(firstCathet, 2) + Math.Pow(secondCathet, 2));

GetPoints(A, B, Start);

ChangeArrayPoints(TrianglePoints);

}

public Triangle(double firstCathet, double secondCathet, double hypotenuse) //constructor with two cathets and hypotenuse

{

this.A = firstCathet;

this.B = secondCathet;

this.C = hypotenuse;

GetPoints(A, B, Start);

ChangeArrayPoints(TrianglePoints);

}

public Triangle(double firstCathet, double secondCathet, Point start) //constructor with two cathets and start point

{

this.A = firstCathet;

this.B = secondCathet;

this.C = Math.Sqrt(Math.Pow(firstCathet, 2) + Math.Pow(secondCathet, 2));

this.Start = start;

GetPoints(A, B, Start);

ChangeArrayPoints(TrianglePoints);

}

public Triangle(double firstCathet, double secondCathet, double hypotenuse, Point start) //constructor with two cathets, hypotenuse and start point

{

this.A = firstCathet;

this.B = secondCathet;

this.C = hypotenuse;

this.Start = start;

GetPoints(A, B, Start);

ChangeArrayPoints(TrianglePoints);

}

public void Draw(Pen pen, List<Point> points, Graphics gr) //draws triangle

{

gr.DrawLine(pen, points[0], points[1]);

gr.DrawLine(pen, points[1], points[2]);

gr.DrawLine(pen, points[2], points[0]);

}

public void FillTriangles(Brush br, Point[] p, Graphics gr)

{

gr.FillPolygon(br, p);

}

public List<Point> GetPoints(double firstCathet, double secondCathet, Point startPoint) //returns list of points

{

trianglePoints.Clear();

trianglePoints.Add(startPoint);

Point p2 = new Point(startPoint.X, startPoint.Y + Convert.ToInt32(firstCathet));

trianglePoints.Add(p2);

Point p3 = new Point(p2.X + Convert.ToInt32(secondCathet), p2.Y);

trianglePoints.Add(p3);

ChangeArrayPoints(trianglePoints);

return trianglePoints;

}

public void ChangeArrayPoints(List<Point> points)

{

for(int i = 0; i < 3; i++)

{

ArrayPoints[i] = points[i];

}

}

public string GetInfo()

{

return $"Triangle sides: {A}, {B}, {C}\n";

}

public void ChangeValues(double firstCathet, double secondCathet)

{

this.A = firstCathet;

this.B = secondCathet;

this.C = Math.Sqrt(Math.Pow(firstCathet, 2) + Math.Pow(secondCathet, 2));

GetPoints(A, B, Start);

ChangeArrayPoints(TrianglePoints);

}

public void ChangePoints(Point point1, Point point2, Point point3)

{

this.TrianglePoints.Clear();

this.TrianglePoints.Add(point1);

this.TrianglePoints.Add(point2);

this.TrianglePoints.Add(point3);

ChangeArrayPoints(TrianglePoints);

}

}

}

**Файл Drawer.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Drawing;

using System.Text;

using System.Threading.Tasks;

namespace SimilarTrianglesMaker

{

public static class Drawer

{

public static void Draw(Pen pen, Triangle tr, Graphics gr, Point p1, Point p2, Point p3, int recursionDepth, bool flag)

{

if (recursionDepth > 0)

{

Point p4 = new Point((p1.X + p2.X) / 2, (p1.Y + p2.Y) / 2);

Point p5 = new Point((p2.X + p3.X) / 2, (p2.Y + p3.Y) / 2);

Point p6 = new Point((p1.X + p3.X) / 2, (p1.Y + p3.Y) / 2);

tr.ChangePoints(p4, p5, p6);

if (flag)

{

tr.Draw(pen, tr.TrianglePoints, gr);

}

else

{

tr.FillTriangles(Brushes.Black, tr.ArrayPoints, gr);

}

Draw(pen, tr, gr, p1, p4, p6, recursionDepth - 1, flag);

Draw(pen, tr, gr, p2, p4, p5, recursionDepth - 1, flag);

Draw(pen, tr, gr, p3, p5, p6, recursionDepth - 1, flag);

}

}

}

}