Московский авиационный институт

(национальный исследовательский университет)

Институт №8 "Информационные технологии и прикладная математика"

**Курсовой проект**

**по курсу**

**"Инструментальные средства**

**разработки прикладных программных систем"**

**по теме "Случайный поиск".**

Выполнил студент группы: М8О-205Б-21

Бондарева Елена Евгеньевна

Преподаватель: Кузнецова Светлана Валентиновна

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 24.12.2022

Москва, 2022

Номер по списку: **2**

**9 тема**: *Random Search* (или случайный поиск).

**Постановка задачи и описание алгоритма:**

Используя алгоритм *Случайный поиск*, найти лучшую функцию из заданного вначале множества функций.

Предположим, есть массив точек (x, y), составляющих график, и есть определенное количество разных функций, имеющих разные графики. Нахождение *лучшей* будет осуществляться так: как близко точки выбранной функции находятся к точкам исходного массива точек (вводимого пользователем).

В некоторых случаях так можно подобрать функцию и на ее основе предсказывать следующие точки после исходных.

**Диаграмма классов:**

**Form1**

**PointAddFrom**

**ChartFrom**

**Point**

**Candidate**

**Реализация алгоритма на C# и пользовательского интерфейса:**

**I**

Код основной формы(Form1), при помощи которой появится другая форма, позволяющая задать пользователю любое количество точек, а также задать любые координаты этих самых точек, на основе которых будет строиться график. Эта основная форма запрашивает у пользователя время (в секундах), после которого будет выводится график лучшей функции(из списка введенных пользователем функций), а также той функции, которая будет получена в результате введенных точек в начале работы приложения. Она же непосредственно будет запрашивать у пользователя разрешение для вывода и построения графиков функций.

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 Algorythm9\_Libarary;

namespace Algorythm9\_Gui

{

public partial class Form1 : Form

{

private List<Algorythm9\_Libarary.Point> Points = new List<Algorythm9\_Libarary.Point>();

public Form1()

{

InitializeComponent();

}

// после нажатия кнопки выводится форма с графиками

private void button1\_Click(object sender, EventArgs e)

{

var randomCandidates = Candidate.GenerateCandidates();

var bestCandidate = Candidate.RandomSearch(Points, randomCandidates, new TimeSpan(0,0, int.Parse(DeltaTimeTextBox.Text)));

var chartForm = new ChartFrom(Points, bestCandidate);

chartForm.Show();

}

bool pointsAddFormCreated = false;

// после нажатия кнопки появляется новая форма для ввода пользователем координат точек

private void button2\_Click(object sender, EventArgs e)

{

if (pointsAddFormCreated)

return;

var pointsAddForm = new PointsAddFrom(Points);

pointsAddFormCreated = true;

pointsAddForm.FormClosed += PointsAddFrom\_FormClosed;

pointsAddForm.Show();

}

private void PointsAddFrom\_FormClosed(object sender, FormClosedEventArgs e)

{

pointsAddFormCreated = false;

}

}

}

**II**

Код формы(PointsAddFrom), благодаря которой пользователь может задать любое количество точек, а также координаты этих точек по своему усмотрению.

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 Algorythm9\_Libarary;

// будет происходить ввод координат точек для построения графика

namespace Algorythm9\_Gui

{

public partial class PointsAddFrom : Form

{

private List<Algorythm9\_Libarary.Point> \_points;

public PointsAddFrom(List<Algorythm9\_Libarary.Point> points)

{

InitializeComponent();

\_points = points;

}

// запоминание координат точки, введенных пользователем

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

{

var point = new Algorythm9\_Libarary.Point();

point.X = double.Parse(XTextBox.Text);

point.Y = double.Parse(YTextBox.Text);

\_points.Add(point);

}

}

}

**III**

Код формы, которая выводит в Oxy график, построенный на основе введенных пользователем точек, а также лучшей функции, то есть той, которая наиболее схожа к выведенной из всех функций, которые рассматривались.

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.Windows.Forms.DataVisualization.Charting;

using Algorythm9\_Libarary;

// работа формы, демонстрирующей графики:

namespace Algorythm9\_Gui

{

public partial class ChartFrom : Form

{

public ChartFrom(List<Algorythm9\_Libarary.Point> points, Candidate candidate)

{

InitializeComponent();

ChartPoints.ChartAreas.Add("Chart");

Series seriesSource = new Series("f(x) = ?");

seriesSource.ChartType = SeriesChartType.Line;

foreach (var point in points)

{

seriesSource.Points.AddXY(point.X, point.Y);

}

Series seriesCandidate = new Series(candidate.ToString());

seriesCandidate.ChartType = SeriesChartType.Line;

foreach (var point in points)

{

seriesCandidate.Points.AddXY(point.X, candidate.YfromX(point.X));

}

ChartPoints.Series.Add(seriesSource);

ChartPoints.Series.Add(seriesCandidate);

ChartPoints.Legends.Add("f(x) = ?");

ChartPoints.Legends.Add(candidate.ToString());

}

}

}

Была создана библиотека Algorythm9\_Libarary:

**IV:**

Код *кандидатов, претендентов на звание лучшей функции.* В данном коде идет выбор лучшей функции из заданных с самого начала функций. Выбирается та функция, которая наиболее близка и схожа со значениями к той, что будет впоследствии построена на основе точек, введенных пользователем.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Algorythm9\_Libarary

{

public class Candidate

{

public Candidate(Func<double, double> yFromX, string asText)

{

YfromX = yFromX;

\_asText= asText;

}

private string \_asText;

public override string ToString()

{

return \_asText;

}

public readonly Func<double, double> YfromX;

public double Quality(List<Point> source)

{

double sumOfOtklon = 0;

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

{

var y = YfromX(source[i].X);

sumOfOtklon += Math.Abs(y - source[i].Y);

}

if (sumOfOtklon == 0)

return 1;

double sredOtklon = sumOfOtklon / source.Count;

double quality = 1 / sredOtklon;

return quality;

}

// алгоритм нахождения лучшего из множества

static public Candidate RandomSearch(List<Point> source, List<Candidate> candidates, TimeSpan deltaTime)

{

Random random = new Random(DateTime.Now.Millisecond);

var Best = candidates[random.Next(0, candidates.Count)];

DateTime endTime = DateTime.Now.Add(deltaTime);

while (Best.Quality(source) < 1 && DateTime.Now < endTime)

{

Candidate S = candidates[random.Next(0, candidates.Count)];

Console.WriteLine("now quality: " + S.Quality(source));

if (S.Quality(source) > Best.Quality(source))

{

Best = S;

}

}

return Best;

}

// задаем претендентов

static public List<Candidate> GenerateCandidates()

{

List<Candidate> candidates = new List<Candidate>();

candidates.Add(new Candidate((double x) => x \* x, "f(x) = x\*x"));

candidates.Add(new Candidate((double x) => x + 2, "f(x) = x+2"));

candidates.Add(new Candidate((double x) => x \* x - 50, "f(x) = x\*x - 50"));

candidates.Add(new Candidate((double x) => x + 2 + x \* x, "f(x) = x+2 + x\*x"));

candidates.Add(new Candidate((double x) => x \* 2 \* x, "f(x) = x\*x \* 2"));

candidates.Add(new Candidate((double x) => x + 2 - 999, "f(x) = x + 2 - 999"));

return candidates;

}

}

}

**V:**

Points.cs – для точек в Oxy(координаты)

namespace Algorythm9\_Libarary

{

public struct Point

{

public Point(double x, double y)

{

X = x;

Y = y;

}

public double X;

public double Y;

}

}

**VI**

**Код Programm.cs(фундамент):**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Algorythm9\_Gui

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

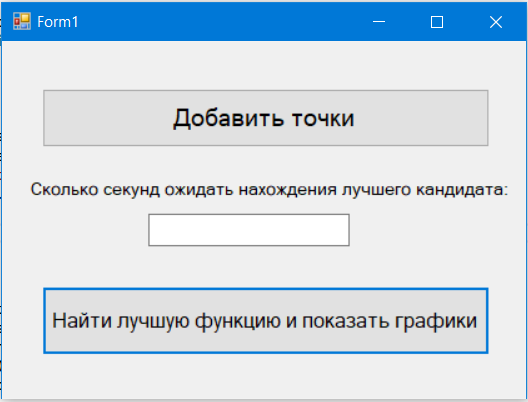
Application.Run(new Form1());

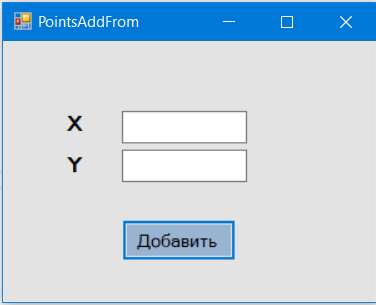
}

}

}

**Демонстрация функциональных возможностей приложения(скриншоты):**





/\* Первая форма вызывает вторую, когда пользователь нажимает на кнопку "Добавить точки". \*/

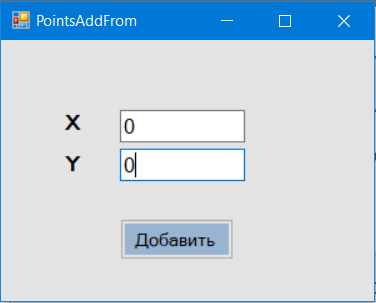
**Тесты. Протокол работы(скриншоты):**

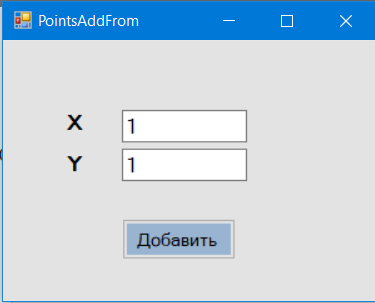
Пусть изначально рассматриваются такие функции для выбора наилучшей, как:

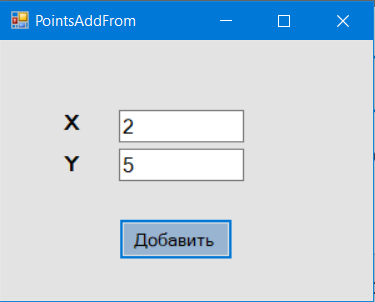
F(x)=x\*x; F(x)=x+2; F(x)=x\*x-50; F(x)=x+2+x\*x; F(x)=x\*x\*2; F(x)=x+2-999.

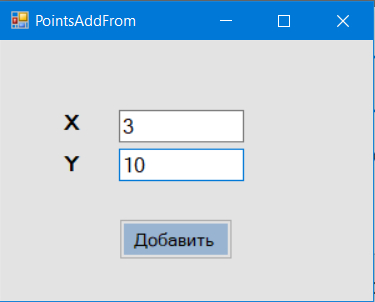
**1 тест:**

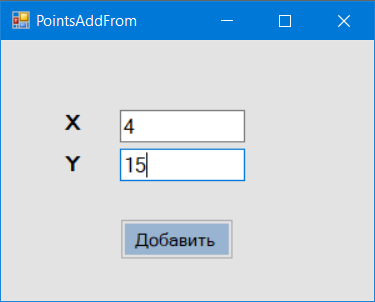
1) Происходит добавление точек пользователем. В данном случае пользователь задает 6 точек со следующими координатами:

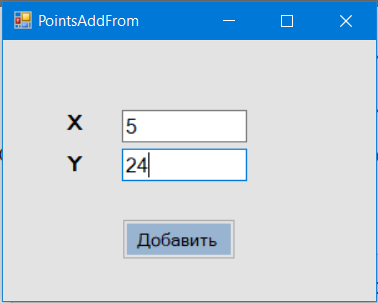
****

****

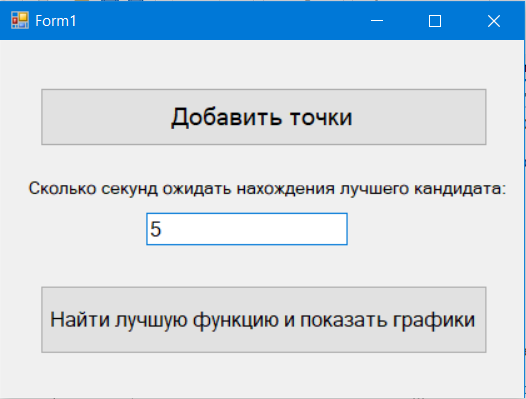
****

****

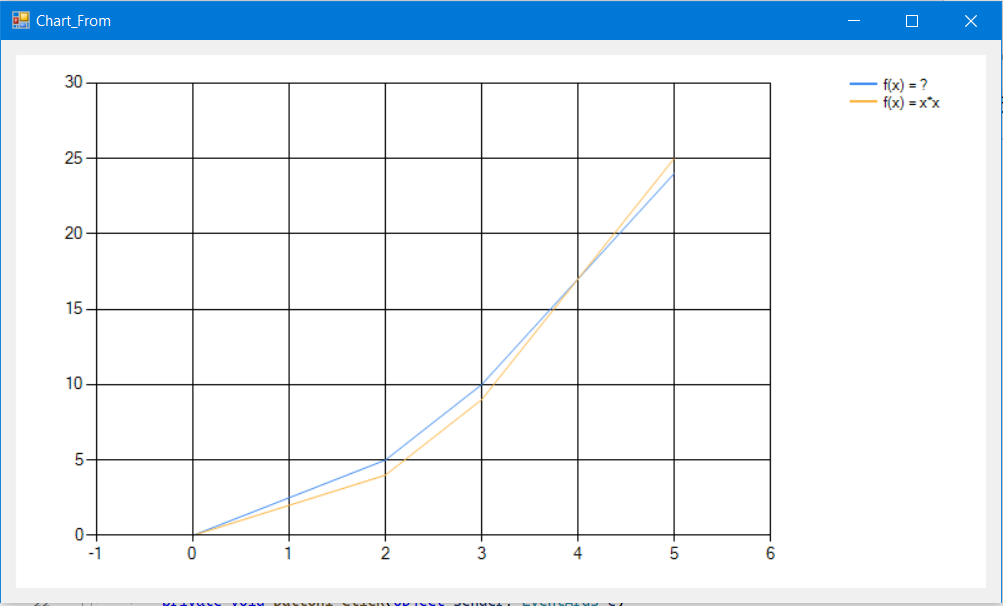
****

****

2) Далее пользователь выбирает время(в сек.), по истечении которого получит график, полученный из введенных им точек, и *лучшую* функцию среди рассматриваемых кандидатов. В данном случае, время - 5сек.

****

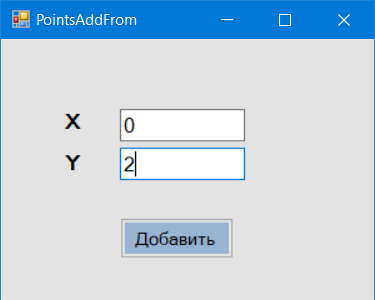
*В результате пользователь получит:*

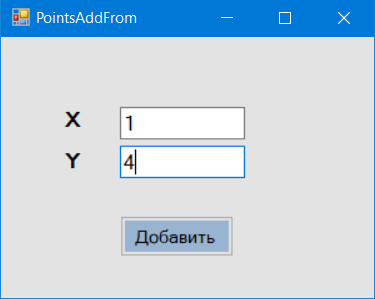
****

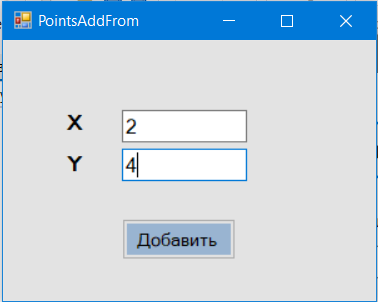
Таким образом, пользователь получает лучшую функцию, и она: F(x)=x\*x(оранжевый цвет), синий цвет- пользователя.

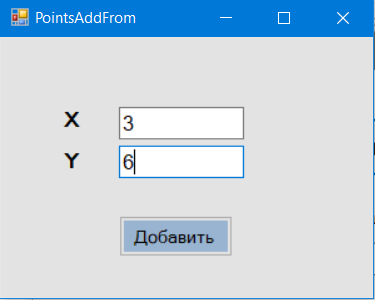
**2 тест:**

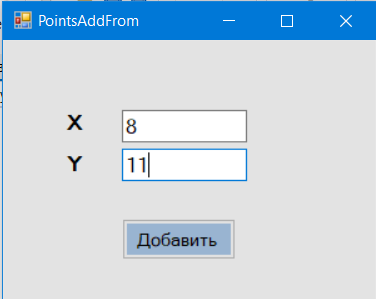
1) Происходит добавление точек пользователем. В данном случае пользователь задает 5 точек со следующими координатами:



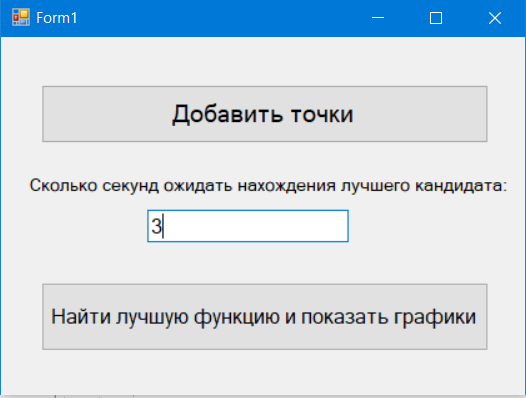




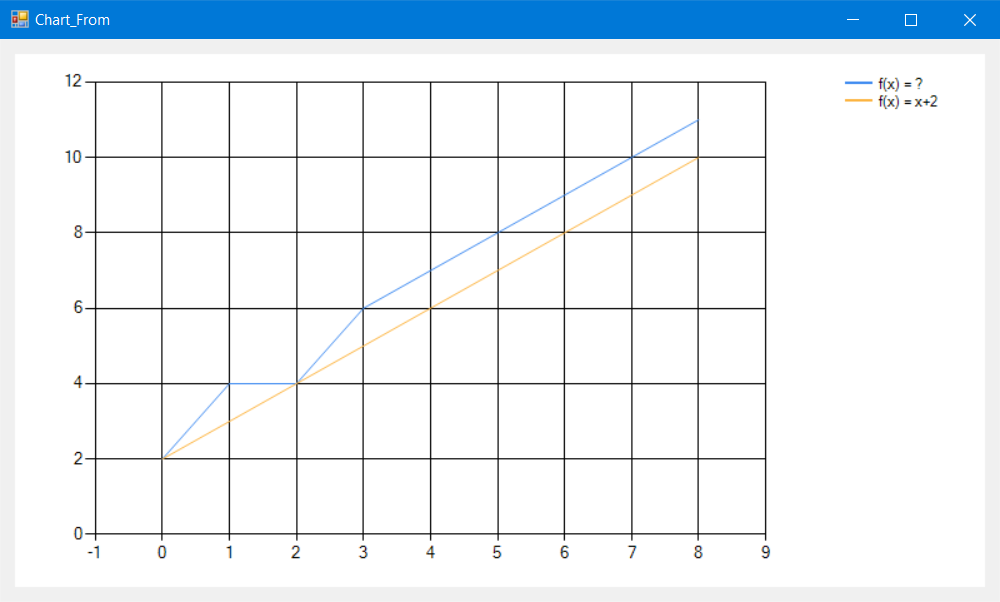




2) Далее пользователь выбирает время(в сек.), по истечении которого получит график, полученный из введенных им точек, и *лучшую* функцию среди рассматриваемых кандидатов. В данном случае: время – 3 сек.

****

*В результате пользователь получит:*



Таким образом, пользователь получает лучшую функцию среди заданных, и она: F(x)=x+2(оранжевый цвет), а синий цвет – пользователя.

Использовалась такая среда разработки, как Visual Studio (2022). Для работы с приложением необходима операционная система Windows.

**Выводы:** в результате работы приложения был продемонстрирован *Случайный поиск*, то есть *рассматривался поиск лучшего из некого множества.* В моем случае, - это нахождение(поиск) лучшей функции из множество заданных функций. Таким образом, приложение позволяет находить из рассматриваемых функций, ту, лучшую, которая ближе всего к тому графику, составленному из точек, введенных пользователем; а также больше всего схожа с ним (имеют точки с одинаковыми координатами, а также расстояние между точками двух графиков - не такое значительное, по сравнению с другими.