ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО	*УТВЕРЖДАЮ*
Преподаватель департамента программной инженерии факультета компьютерных наук	Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук
________________ Н. К. Чуйкин	________________ В. В. Шилов
«____» ____________ 2019 г.	«____» ____________ 2019 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА ДЛЯ ВИЗУАЛИЗАЦИИ ГРАФА ДРУЗЕЙ СОЦИАЛЬНОЙ СЕТИ "ВКОНТАКТЕ"

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.13-01 51 01-1-ЛУ

Исполнитель: студент группы БПИ185
________________ А. А. Мануйлов
«____» ________________ 2019 г.

— Москва 2019 —

УТВЕРЖДЕН
RU.17701729.04.13-01 34 01-1-ЛУ

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	

РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА ДЛЯ ВИЗУАЛИЗАЦИИ ГРАФА ДРУЗЕЙ СОЦИАЛЬНОЙ СЕТИ "ВКОНТАКТЕ"

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.13-01 51 01-1-ЛУ

Листов 25

— Москва 2019 —

Содержание

1 Текст программы

1.1 Form1.cs

```
using System;
using System.Net;
using System.IO;
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.Threading;
using Newtonsoft.Json.Linq;
using GraphX.PCL.Common.Enums;
using GraphX.PCL.Logic.Algorithms.OverlapRemoval;
using GraphX.PCL.Logic.Models;
using GraphX.Controls;
using GraphX.Controls.Models;
using QuickGraph;



namespace Курсач
{
public partial class Form1 : Form
{
Bitmap map;
Graphics gr;
SolidBrush br;
Random rnd;
PointF mousePos, center, p;
Point pictureBoxLocation;
Vertex[] vertexes;
Edge[] edges;
PointF[] startConfiguration;
Pen pen;
Dictionary<string, string[]> graph;
string[] friends;
float vertexRadius = 8;
bool flag = true;
bool isAbleToForce = true;
string token;
string id;
```

```
public Form1()
{
InitializeComponent();
this.WindowState = FormWindowState.Maximized;
pictureBoxLocation = pictureBox1.Location;
MouseWheel += new MouseEventHandler(This_MouseWheel);
rnd = new Random();
map = new Bitmap(pictureBox1.Width, pictureBox1.Height);
gr = Graphics.FromImage(map);
br = new SolidBrush(Color.FromArgb(255, 0, 0, 255));
graph = new Dictionary<string, string[]>();
pen = new Pen(Color.FromArgb(100, 43, 255, 242));
MinimumSize = new Size(Screen.PrimaryScreen.WorkingArea.Width / 2,
 Screen.PrimaryScreen.WorkingArea.Height / 2);
menuStrip1.BringToFront();
label1.BringToFront();
pictureBox2.BringToFront();
pictureBox3.BringToFront();
label2.BringToFront();
pictureBox4.Visible = true;
pictureBox4.Refresh();
pictureBox4.Visible = false;
}

public static string ObjToStr(object obj) { return obj.ToString();

private void Draw()
{
try
{
pictureBox1.Image = map;
gr.Clear(Color.White);
for (int i = 0; i < edges.Length; i++) рисуем// всеребрамеждудрузьями
{
if (edges[i].IsVertexPicked())
{
gr.DrawLine(new Pen(Color.Red), edges[i].V1.Coord.X + vertexRadius
 / 2, edges[i].V1.Coord.Y + vertexRadius / 2,
edges[i].V2.Coord.X + vertexRadius / 2, edges[i].V2.Coord.Y
+ vertexRadius / 2);
}
else
{
gr.DrawLine(pen, edges[i].V1.Coord.X + vertexRadius / 2,
 edges[i].V1.Coord.Y + vertexRadius / 2,
edges[i].V2.Coord.X + vertexRadius / 2, edges[i].V2.Coord.Y
+ vertexRadius / 2);
}
}

for (int i = 0; i < vertexes.Length; i++) рисуем// ребрасцентральнымпе
{
if (vertexes[i].IsPicked)
{
gr.DrawLine(new Pen(Color.Red), center.X + vertexRadius / 2,
 center.Y + vertexRadius / 2,
vertexes[i].Coord.X + vertexRadius / 2, vertexes[i].Coord.Y
+ vertexRadius / 2);
}
else
{
gr.DrawLine(new Pen(Color.Silver), center.X + vertexRadius / 2,
 center.Y + vertexRadius / 2,
vertexes[i].Coord.X + vertexRadius / 2, vertexes[i].Coord.Y
+ vertexRadius / 2);
}
}

gr.FillEllipse(new SolidBrush(Color.DarkRed), center.X, center.Y,
vertexRadius, vertexRadius); центральный// персонаж
gr.DrawEllipse(new Pen(Color.Black), center.X, center.Y,
 vertexRadius, vertexRadius);

for (int i = 0; i < vertexes.Length; i++) рисуем// всевершины
{
if (vertexes[i].IsPicked)
{
gr.FillEllipse(new SolidBrush(Color.Green), vertexes[i].Coord.X,
vertexes[i].Coord.Y, vertexRadius, vertexRadius);
}
else
{
gr.FillEllipse(br, vertexes[i].Coord.X, vertexes[i].Coord.Y,
 vertexRadius, vertexRadius);
}
gr.DrawEllipse(new Pen(Color.Black), vertexes[i].Coord.X,
 vertexes[i].Coord.Y, vertexRadius, vertexRadius);
}
pictureBox1.Refresh();
}
catch (OverflowException e)
{
gr.Clear(Color.White);
throw new OverflowException();
}
catch (Exception)
{
```

```
throw;
}
}

/// <summary>
///
/// </summary>
/// <param name="id"></param>
/// <param name="token"></param>
/// <param name="pack_f"Часть><\ друзейюзеранебольше      100>
/// <returns></returns>
Dictionary<string, string[]> FriendsGetMutual(string sourse_id,
string token, string[] pack_f)
{
try
{
Dictionary<string, string[]> result =
new Dictionary<string, string[]>();
List<string> requests = StringSeparator(pack_f);
foreach (string pack_friends in requests)
{
string address = $"https://api.vk.com/method/friends.getMutual?
␣␣␣␣␣␣␣␣v=5.52&source_uids={sourse_id}&target_uids={pack_friends}&
␣␣␣␣␣␣␣␣access_token={token}";
string response = string.Empty;
using (WebClient webClient = new WebClient())
{
response = webClient.DownloadString(address);
}
JObject obj = JObject.Parse(response);
IList<JToken> res = obj["response"].ToList();
foreach (JToken dict in res)
{
string[] r = Array.ConvertAll(dict["common_friends"].ToArray(),
 new Converter<object, string>(ObjToStr));
result.Add(dict["id"].ToString(), r);
}
}
return result;
}
catch (Exception)
{
throw new Exception();
}
}

/// <summary>
/// получаемсписокдрузейюзерапоего          id
/// </summary>
/// <param name="id"></param>
/// <param name="tok"></param>
/// <returns></returns>
string[] GetListFriendsByID(string id, string tok)
{
try
{
string address = $"https://api.vk.com/method/friends.get?v=5.52&
␣␣␣␣␣␣␣␣user_id={id}&fields=nickname&access_token={tok}";
string response = string.Empty;
using (WebClient webClient = new WebClient())
{
response = webClient.DownloadString(address);
}
JObject obj = JObject.Parse(response);
List<string> res = new List<string>();
for (int i = 0; i < obj["response"]["items"].Count(); i++)
{
if (!obj["response"]["items"][i].ToString().Contains("deactivated")
{
res.Add(obj["response"]["items"][i]["id"].ToString());
}
}
//string[] array = Array.ConvertAll(obj["response"]["items"].ToArra
 new Converter<object, string>(ObjToStr));
return res.ToArray();
}
catch (NullReferenceException e)
{
MessageBox.Show("Проблемы␣с␣авторизацией.\n␣запустите␣приложение␣заново.
return null;
}
catch (Exception)
{
MessageBox.Show("Какието-␣проблемы.");
return null;
}
}

string[] UserInfo(string id)
{
try
{
string address = $"https://api.vk.com/method/users.get?v=5.52&
␣␣␣␣␣␣␣␣user_id={id}&fields=photo_100&access_token={token}";
string response = string.Empty;
using (WebClient webClient = new WebClient())
{
response = webClient.DownloadString(address);
```

```
}
JObject obj = JObject.Parse(response);
return new string[] { obj["response"][0]["first_name"].ToString(),
 obj["response"][0]["last_name"].ToString(),
  obj["response"][0]["photo_100"].ToString() };
}
catch (NullReferenceException e)
{
MessageBox.Show(e.Message);
return null;
}
catch (Exception)
{
MessageBox.Show("Какието–␣проблемы.");
return null;
}
}

List<string> StringSeparator(string[] arr)
{
List<string> result = new List<string>();
string s = string.Empty;
for (int i = 0; i < arr.Length; i++)
{
s += arr[i] + ",";
if (i % 99 == 0 && i != 0)
{
result.Add(s);
s = string.Empty;
}
}
result.Add(s);
return result;
}

private void Form1_SizeChanged(object sender, EventArgs e)
{
if (flag) // длятогочтобыпризагрузкенепадало
{
flag = false;
return;
}
try
{
if (pictureBox1.Width * pictureBox1.Height != 0)
{
int x = map.Width;
int y = map.Height;
map = new Bitmap(pictureBox1.Width, pictureBox1.Height);
```

```
gr = Graphics.FromImage(map);
double coefx = map.Width * 1.0 / x;
double coefy = map.Height * 1.0 / y;
if (vertexes != null)
{
for (int i = 0; i < vertexes.Length; i++)
{
vertexes[i].Coord.X *= (float)coefx;
vertexes[i].Coord.Y *= (float)coefy;
}
center.X *= (float)coefx;
center.Y *= (float)coefy;
Draw();
}
if (startConfiguration != null)
{
for (int i = 0; i < startConfiguration.Length; i++)
{
startConfiguration[i].X *= (float)coefx;
startConfiguration[i].Y *= (float)coefy;
}
}
}
}
catch (ArgumentException ext)
{
MessageBox.Show(ext.Message, "Exception");
return;
}
catch (OverflowException ext)
{
MessageBox.Show(ext.Message, "Exception");
return;
}
catch (Exception ext)
{
MessageBox.Show(ext.Message, "Exception");
return;
}
}

/// <summary>
/// силовойалгоритмразмещенияграфанаплоскости
/// </summary>
private void ForceDirectedAlgorithm()
{
try
{
double C1 = 400, // притяжение
```

```
C2 = Math.Sqrt(pictureBox1.Width * pictureBox1.Width +
 pictureBox1.Height * pictureBox1.Height) / 20,
  // идеальноерасстояниемеждувершинами
C3 = 100000, // отталкивание
C4 = Math.Sqrt(pictureBox1.Width * pictureBox1.Width +
 pictureBox1.Height * pictureBox1.Height) / 7;притяжение
 // кцентру
float delta = 0.0001F;
PointF center = new PointF(pictureBox1.Width / 2,
 pictureBox1.Height / 2);
for (int i = 0; i < 200; i++)
{
PointF[] vectors = new PointF[vertexes.Length]; // векторсилынакажду
for (int j = 0; j < vertexes.Length; j++) смотрим// какиесилыдействую
{
PointF vect = new PointF(0, 0);
for (int h = 0; h < vertexes.Length; h++)
{
if (j != h)
{
double f;
if (vertexes[j].IsVertexCommon(vertexes[h]))
{
f = Math.Log10(Distanse(vertexes[j].Coord,
 vertexes[h].Coord) / C2) * C1;
}
else
{
double d = Distanse(vertexes[j].Coord, vertexes[h].Coord);
f = -C3 / (d * d);
}
vect.X += (float)((vertexes[h].Coord.X - vertexes[j].Coord.X)
 *(f * (Math.Abs(vertexes[h].Coord.X - vertexes[j].Coord.X) /
  Distanse(vertexes[j].Coord, vertexes[h].Coord))));
vect.Y += (float)((vertexes[h].Coord.Y - vertexes[j].Coord.Y)
 *(f * (Math.Abs(vertexes[j].Coord.Y - vertexes[h].Coord.Y) /
  Distanse(vertexes[j].Coord, vertexes[h].Coord))));
}
}
vect.X += (float)((Distanse(vertexes[j].Coord, center)) *
 (center.X - vertexes[j].Coord.X) * 0.5);
vect.Y += (float)((Distanse(vertexes[j].Coord, center)) *
 (center.Y - vertexes[j].Coord.Y) * 0.5);
vectors[j] = vect;
}

for (int j = 0; j < vertexes.Length; j++)
{
vertexes[j].Coord.X += (float)Math.Round(vectors[j].X * delta, 2);
vertexes[j].Coord.Y += (float)Math.Round(vectors[j].Y * delta, 2);
}
Draw();
}
}
catch (OverflowException ex)
{
throw new OverflowException();
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
throw new Exception();
}
finally
{
pictureBox4.Visible = false;
}
}

private void pictureBox1_MouseMove(object sender, MouseEventArgs e
{
if (e.Button == MouseButtons.Left &&кластеризоватьГраф
 ToolStripMenuItem.Enabled)
{
try
{
pictureBox1.Location = new Point((int)(p.X +
 MousePosition.X - mousePos.X),
(int)(p.Y + MousePosition.Y - mousePos.Y));
isAbleToForce = false;
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
}
}
}

private void pictureBox1_MouseDown(object sender, MouseEventArgs e
{
mousePos = MousePosition;
p = pictureBox1.Location;
}

private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
if (vertexes != null)
```

```
{
try
{
int deltax = pictureBox1.Location.X − pictureBoxLocation.X;
int deltay = pictureBox1.Location.Y − pictureBoxLocation.Y;
for (int i = 0; i < vertexes.Length; i++)
{
vertexes[i].Coord.X += deltax;
vertexes[i].Coord.Y += deltay;
}
center.X += deltax;
center.Y += deltay;
Draw();
pictureBox1.Location = pictureBoxLocation;
}
catch (Exception)
{
MessageBox.Show("Чтото−␣пошло␣не␣так");
return;
}
}
}

void This_MouseWheel(object sender, MouseEventArgs e)
{
if (e.Delta != 0)
{
try
{
float coef = 1.1F;
if (e.Delta <= 0)
{
coef = 0.9F;
}
if (vertexes != null)
{
float oldx = center.X, oldy = center.Y;
center.X *= coef;
center.Y *= coef;

float dx = center.X − oldx;
float dy = center.Y − oldy;

center.X −= dx;
center.Y −= dy;
for (int i = 0; i < vertexes.Length; i++)
{
vertexes[i].Coord.X *= coef;
vertexes[i].Coord.Y *= coef;
vertexes[i].Coord.X -= dx;
vertexes[i].Coord.Y -= dy;
}
vertexRadius *= coef;
Draw();
isAbleToForce = false;
}
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
}
}
}

private void pictureBox1_MouseDoubleClick(object sender,
 MouseEventArgs e)
{
if (vertexes != null)
{
try
{
for (int i = 0; i < vertexes.Length; i++)
{
if (Distanse(e.Location, new PointF(vertexes[i].Coord.X +
 vertexRadius / 2, vertexes[i].Coord.Y + vertexRadius / 2))
  <= vertexRadius)
{
for (int j = 0; j < vertexes.Length; j++)
vertexes[j].IsPicked = false;

vertexes[i].IsPicked = true;
pictureBox1.Refresh();
string[] mas = UserInfo(vertexes[i].ID);
if (mas == null) return;
pictureBox3.Load(mas[2]);
pictureBox3.Visible = true;
label2.Text = Encoding.UTF8.GetString(Encoding.GetEncoding
("windows-1251").GetBytes(mas[0] + "␣" + mas[1]));
label2.Visible = true;
break;
}
}
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
```

```
}
}
}

/// <summary>
/// Алгаритмкластеризации   LPA
/// </summary>
private void Clustering()
{
try
{
vertexRadius = 8;
bool flag = true;
int k = 0;
int[] indexes = new int[vertexes.Length];
for (int i = 0; i < vertexes.Length; i++)
indexes[i] = i;

while (flag)
{
flag = false;
int[] shakedNumbers = indexes.OrderBy(x => rnd.Next()).ToArray();
// перемешиваемвершиныдлякорректнойработыалго
for (int i = 0; i < shakedNumbers.Length; i++)
{
Dictionary<string, int> dict = new Dictionary<string, int>();
foreach (Vertex v in vertexes) // считаме средисмежныхвершинчастотумет
{
if (vertexes[shakedNumbers[i]].IsVertexCommon(v) && v.ID !=
 vertexes[shakedNumbers[i]].ID)
{
if (dict.Keys.Contains(v.Group)) dict[v.Group] += 1;
else dict[v.Group] = 1;
}
}

int m = -1;
foreach (string s in dict.Keys) // ищеммаксимальнуючастотувхождений
{
if (dict[s] > m)
m = dict[s];
}

List<string> mas = new List<string>(); // создаемлистнаиболеечастыхвхож
foreach (string s in dict.Keys)
{
if (dict[s] == m)
mas.Add(s);
}
int ind = rnd.Next(0, mas.Count);
if (mas.Count != 0 && mas[ind] !=
 vertexes[shakedNumbers[i]].Group)
{
flag = true;
vertexes[shakedNumbers[i]].Group = mas[ind];
}
}
}

List<string> clusters = new List<string>(); // id кластеров
for (int i = 0; i < vertexes.Length; i++)
{
if (!clusters.Contains(vertexes[i].Group))
 clusters.Add(vertexes[i].Group);
}
PointF[] clustCoords = VerticesOfRegularPolygon(clusters.Count);
// координатыцентровкластеров
int a = TheoremCos(pictureBox1.Height / 3,
 Math.PI * 2 / clusters.Count);
if (clustCoords.Length == 1) a = Math.Min(pictureBox1.Width,
 pictureBox1.Height) / 5;
for (int i = 0; i < vertexes.Length; i++) распределяем// вершиныпокла
{
for (int j = 0; j < clusters.Count; j++)
{
if (vertexes[i].Group == clusters[j])
{
vertexes[i].Coord = new PointF(rnd.Next((int)clustCoords[j].X - a,
 (int)clustCoords[j].X + a) + (float)rnd.NextDouble(),
rnd.Next((int)clustCoords[j].Y - a,
 (int)clustCoords[j].Y + a) + (float)rnd.NextDouble());
break;
}
}
}
Draw();
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
}
}

private int TheoremCos(double R, double a)
{
try
```

```
{
return (int)(Math.Sqrt(2 * R * R * (1 - Math.Cos(a))) / 2);
 // возвращаетполовинуотсторонымногоугольника

}
catch (Exception)
{
throw new Exception();
}
}

private void Method(object sender, CancelEventArgs e)
{
try
{
id = ((browser)sender).userId;
token = ((browser)sender).accessToken;
if (id != null && token != null)
{
string[] m = UserInfo(id);
label1.Text = "Вы␣вошли␣как␣" + Encoding.UTF8.GetString(Encoding.G
("windows-1251").GetBytes(m[0] + "␣" + m[1]));
label1.Visible = true;
pictureBox2.Load(m[2]);
pictureBox2.Visible = true;отобразитьГраф
ToolStripMenuItem.Enabled = true;
}
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
}
}

private void авторизироватьсяToolStripMenuItem_Click(object sender,
 EventArgs e)
{
try
{
browser browser = new browser();
browser.Owner = this;
browser.Show();
browser.Closing += Method;
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
```

```
}
}

private void отобразитьГрафToolStripMenuItem_Click(object sender,
 EventArgs e)
{
try
{
center = new PointF(pictureBox1.Width / 2, pictureBox1.Height / 2);
vertexRadius = 8;
friends = GetListFriendsByID(id, token);
if (friends == null) return;
graph = FriendsGetMutual(id, token, friends);
vertexes = new Vertex[friends.Length];
for (int i = 0; i < friends.Length; i++)
{
PointF P = new PointF(rnd.Next(10, pictureBox1.Width) - 10,
 rnd.Next(10, pictureBox1.Height) - 10);
vertexes[i] = new Vertex(friends[i], P, graph[friends[i]]);
}
edges = CreateEdgesArray();
startConfiguration = new PointF[vertexes.Length];
for (int i = 0; i < vertexes.Length; i++)
{
startConfiguration[i] = new PointF(vertexes[i].Coord.X,
 vertexes[i].Coord.Y);
}
Draw();
label2.Visible = false;
pictureBox3.Visible = false;кластеризоватьГраф
ToolStripMenuItem.Enabled = true;сброситьМасштаб
ToolStripMenuItem.Enabled = true;запуститьСиловойАлгоритм
ToolStripMenuItem.Enabled = false;
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
}
}

private Edge[] CreateEdgesArray() // избежатьповторенияребер
{
try
{
List<Edge> res = new List<Edge>();
for (int i = 0; i < vertexes.Length; i++)
{
for (int j = 0; j < vertexes.Length; j++)
{
if (vertexes[i].IsVertexCommon(vertexes[j]))
{
res.Add(new Edge(vertexes[i], vertexes[j]));
}
}
}
return res.ToArray();
}
catch (Exception)
{
throw new Exception();
}
}

private void кластеризоватьГрафToolStripMenuItem_Click(object sender,
 EventArgs e)
{
try
{
Clustering();запуститьСиловойАлгоритм
ToolStripMenuItem.Enabled = true;
for (int i = 0; i < vertexes.Length; i++)
{
startConfiguration[i] = new PointF(vertexes[i].Coord.X,
 vertexes[i].Coord.Y);
}
}
catch (OverflowException ex)
{
MessageBox.Show("Попробуйте␣заново");
return;
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
}

Draw();
}

private void запуститьСиловойАлгоритмToolStripMenuItem_Click
(object sender, EventArgs e)
{
ToStartPos();
if (isAbleToForce)
{
try
```

```
{
pictureBox4.Visible = true;
pictureBox4.Refresh();
pictureBox4.BringToFront();
ForceDirectedAlgorithm();
for (int i = 0; i < vertexes.Length; i++)
{
startConfiguration[i] = new PointF(vertexes[i].Coord.X,
 vertexes[i].Coord.Y);
}
}
catch (OverflowException ex)
{
MessageBox.Show("Попробуйте␣заново");
return;
}
catch (Exception)
{
MessageBox.Show("Чтото−␣пошло␣не␣так");
return;
}

}
else
MessageBox.Show("Для␣запуска␣силового␣алгоритма␣сначала␣необходимо
␣␣␣␣␣␣␣␣сбросить␣масштаб");
}

private void сброситьМасштабToolStripMenuItem_Click(object sender,
 EventArgs e)
{
ToStartPos();
}

private void ToStartPos()
{
try
{
center = new PointF(pictureBox1.Width / 2, pictureBox1.Height / 2);
for (int i = 0; i < vertexes.Length; i++)
{
vertexes[i].Coord = new PointF(startConfiguration[i].X,
 startConfiguration[i].Y);
}
vertexRadius = 8;
isAbleToForce = true;
Draw();
}
catch (Exception)
```

```
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
}
}

private void сохранитьИзображениеToolStripMenuItem_Click
(object sender, EventArgs e)
{
try
{
SaveFileDialog save = new SaveFileDialog();
save.ShowDialog();
string filename = save.FileName + ".jpg";
map.Save(filename, System.Drawing.Imaging.ImageFormat.Jpeg);
}
catch (Exception)
{
MessageBox.Show("Чтото-␣пошло␣не␣так");
return;
}
}

private void справкаToolStripMenuItem_Click(object sender,
 EventArgs e)
{
MessageBox.Show("Выполнил␣студент␣первого␣курса,␣185␣группы␣\n␣" +
"отделения␣Программной␣Инженерии␣\n" +
"Мануйлов␣Александр␣\n␣\n␣" +
"Научный␣руководитель:␣Чуйкин␣Н.␣К.␣\n␣\n" +
"Связь␣с␣автором:␣aamanuylov@edu.hse.ru");
}

/// <summary>
/// Ищеткоординатыцентровкластеровпоокружностивокругцентраполя
/// </summary>
/// <param name="n"></param>
/// <returns></returns>
PointF[] VerticesOfRegularPolygon(int n)
{
try
{
PointF[] res = new PointF[n];
res[0] = new PointF(pictureBox1.Width / 2 +
(center.X - pictureBox1.Width / 2), pictureBox1.Height / 6 +
 (center.Y - pictureBox1.Height / 2));
double rad = pictureBox1.Height / 3;
double A = Math.PI * 2 / n;
for (int i = 1; i < n; i++)
{
res[i] = new PointF((float)(rad * Math.Sin(i * A) +
 pictureBox1.Width / 2) + (center.X - pictureBox1.Width / 2),
pictureBox1.Height / 2 - (float)(rad * Math.Cos(i * A)) +
 (center.Y - pictureBox1.Height / 2));
}
return res;
}
catch (Exception)
{
throw new Exception();
}
}

double Distanse(PointF a, PointF b)
{
try
{
return Math.Sqrt((a.X - b.X) * (a.X - b.X) + (a.Y - b.Y) *
 (a.Y - b.Y));
}
catch (Exception)
{
throw new Exception();
}
}
}
}
```

1.2 browser.cs

```
using System;
using System.Text.RegularExpressions;
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;

namespace Курсач
{
public partial class browser : Form
{
string client_id = "5906178";
public string accessToken;
```

```
        public string userId;

        public browser()
        {
        InitializeComponent();
        string address = $"https://oauth.vk.com/authorize?client_id=
␣␣␣␣␣␣␣␣{client_id}&display=popup&redirect_uri=https://oauth.vk.com/blank.
␣␣␣␣␣␣␣␣html&scope=friends&response_type=token&v=5.52&revoke=1";
        webBrowser1.Navigate(address);
        Size = webBrowser1.ClientSize;
        }

        public void webBrowser1_DocumentCompleted(object sender,
         WebBrowserDocumentCompletedEventArgs e)
        {

        if (e.Url.ToString().IndexOf("access_token") != -1)
        {
        Regex myReg = new Regex(@"(?<name>[\w\d\x5f]+)=(?<value>
␣␣␣␣␣␣␣␣[^\x26\s]+)", RegexOptions.IgnoreCase | RegexOptions.Singleline);
        foreach (Match m in myReg.Matches(e.Url.ToString()))
        {
        if (m.Groups["name"].Value == "access_token")
        {
        accessToken = m.Groups["value"].Value;
        }
        else if (m.Groups["name"].Value == "user_id")
        {
        userId = m.Groups["value"].Value;
        }
        }
        if (String.IsNullOrEmpty(accessToken))
        {
        MessageBox.Show("Ошибка.␣Ключ␣доступа␣не␣найден.", "Ошибка",
         MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
        }
        }
        if (userId != null && accessToken != null)
        this.Close();

        }
        }
        }
```

1.3 Vertex.cs

```
        class Vertex
```

```
{
        public PointF Coord;
        public Vertex(string iD, PointF coord, string[] com_vert)
        {
                ID = iD;
                Coord = coord;
                Common_Vertexes = com_vert;
                Group = iD;
        }
        public string[] Common_Vertexes { get; }
        public string ID { get; }
        public string Group { get; set; }
        public bool IsPicked { get; set; }

        public bool IsVertexCommon(Vertex v)
        {
                if (this.Common_Vertexes.Contains(v.ID))
                {
                        return true;
                }
                return false;
        }
}
```

1.4 Edge.cs

```
class Edge
{
        public Vertex V1, V2;
        public Edge(Vertex v1, Vertex v2)
        {
                V1 = v1;
                V2 = v2;
        }

        public bool IsVertexPicked()
        {
                if (V1.IsPicked || V2.IsPicked) return true;
                return false;
        }
}
```

Лист регистрации изменений									
Изм.	Номера листов				Всего листов в документе	№ документа	Входящий № сопроводит. докум. и дата	Подпись	Дата
	изменен-ных	заменен-ных	новых	аннулиро-ванных					