|  |
| --- |
| ***Инв. № Подп. и дата Взам. инв. № Инв. № дубл. Подп. и дата*** |

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО | УТВЕРЖДАЮ |
| Доцент факультета | Академический руководитель |
| компьютерных наук департамента | образовательной программы |
| Программной инженерии | «Программная инженерия» |
|  | профессор департамента программной |
|  | инженерии, канд. техн. наук |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_E.Ю. Песоцкая |  |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов |
|  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

**ПРОГРАММА ОЦЕНКИ И ИДЕНТИФИКАЦИИ РИСКОВ**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.09-01 01-1-ЛУ**

Исполнитель

студент группы БПИ183

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Д.С. Капур/

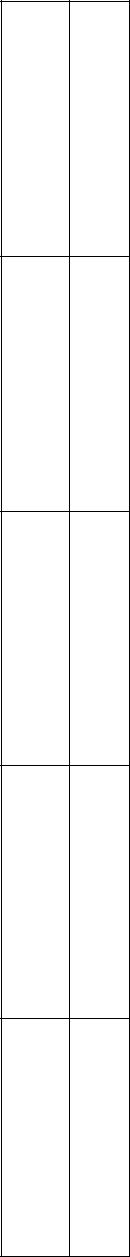
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Москва 2019

УТВЕРЖДЕН

RU.17701729. 04.09-01 01-1

|  |
| --- |
| ***Инв. № подл Подп. и дата Взам. инв. № Инв. № дубл. Подп. и дата*** |



**ПРОГРАММА ОЦЕНКИ И ИДЕНТИФИКАЦИИ РИСКОВ**

**Текст программы**

**RU.17701729.04.09-01 01-1-ЛУ**

**Листов 48**

Москва 2019

|  |  |
| --- | --- |
|  | 2 |
|  | RU.17701729. 04.09-01 01-1 |
|  | СОДЕРЖАНИЕ |
| 1. ТЕКСТ ПРОГРАММЫ | ........................................................................................................................ 3 |

|  |  |
| --- | --- |
| Класс Form1............................................................................................................................................ | 3 |
| Класс Form1............................................................................................................................................ | 3 |
|  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| RU.17701729.507900-01 12 |  |  |  |  |
| Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата |

3

RU.17701729. 04.09-01 01-1

1. **ТЕКСТ ПРОГРАММЫ**

**1.1.** **Класс MainWindow**

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 WindowsFormsApp1;

using System.Data.OleDb;

namespace CourseWork

{

public partial class Form1 : Form

{

private OleDbConnection myConnection;

public static string connectString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Users\\User\\Desktop\\Пересдача курсовой работы\\LP1.accdb;";

int[] TimeArray = new int[0];

int[] TimeArrayp = new int[0];

int[] FinanceArray = new int[0];

int[] FinanceArrayp = new int[0];

int[] StaffArray = new int[0];

int[] StaffArrayp = new int[0];

int[] QualityArray = new int[0];

int[] QualityArrayp = new int[0];

RiskEstimate form2 = new RiskEstimate();

int[] getarrtype()

{

switch (comboBox1.SelectedIndex)

{

case 0:

{

return TimeArray;

break;

}

case 1:

{

return FinanceArray;

break;

}

case 2:

{

return StaffArray;

break;

}

case 3:

{

return QualityArray;

break;

}

}

return null;

}

int[] getarrptype()

{

switch (comboBox1.SelectedIndex)

{

case 0:

{

return TimeArrayp;

break;

}

case 1:

{

return FinanceArrayp;

break;

}

case 2:

{

return StaffArrayp;

break;

}

case 3:

{

return QualityArrayp;

break;

}

}

return null;

}

void resizearrs()

{

if (TimeArray.Length < time\_risksBindingSource.Count) Array.Resize(ref TimeArray, time\_risksBindingSource.Count);

if (TimeArrayp.Length < time\_risksBindingSource.Count) Array.Resize(ref TimeArrayp, time\_risksBindingSource.Count);

if (FinanceArray.Length < financial\_risksBindingSource.Count) Array.Resize(ref FinanceArray, financial\_risksBindingSource.Count);

if (FinanceArrayp.Length < financial\_risksBindingSource.Count) Array.Resize(ref FinanceArrayp, financial\_risksBindingSource.Count);

if (StaffArray.Length < staff\_risksBindingSource.Count) Array.Resize(ref StaffArray, staff\_risksBindingSource.Count);

if (StaffArrayp.Length < staff\_risksBindingSource.Count) Array.Resize(ref StaffArrayp, staff\_risksBindingSource.Count);

if (QualityArray.Length < quality\_risksBindingSource.Count) Array.Resize(ref QualityArray, quality\_risksBindingSource.Count);

if (QualityArrayp.Length < quality\_risksBindingSource.Count) Array.Resize(ref QualityArrayp, quality\_risksBindingSource.Count);

}

int getcr(DataGridViewRow CurrRow1 = null)

{

if (CurrRow1 == null) CurrRow1 = dataGridView2.CurrentRow;

return (int)CurrRow1.Cells[0].Value - 1;

}

public Form1()

{

InitializeComponent();

comboBox2.SelectedIndex = 0;

Text = "Identification";

comboBox3.Visible = false;

label7.Visible = false;

// создаем экземпляр класса OleDbConnection

myConnection = new OleDbConnection(connectString);

// открываем соединение с БД

myConnection.Open();

}

public void Form1\_Update()

{

}

/// <summary>

/// Remove all

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

DialogResult dialogResult = MessageBox.Show("Вы уверены, что хотите сбросить все риски?", "Внимание!", MessageBoxButtons.OKCancel);

if (dialogResult == DialogResult.OK)

{

int[] arr = null;

arr = TimeArray;

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

{

arr[i] = 0;

}

arr = FinanceArray;

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

{

arr[i] = 0;

}

arr = StaffArray;

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

{

arr[i] = 0;

}

arr = QualityArray;

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

{

arr[i] = 0;

}

var table = dataGridView2;

while (table.Rows.Count > 0)

table.Rows.Remove(table.Rows[0]);

table = dataGridView3;

employer1TableAdapter.Fill(risk\_databaseDataSet.Employer1);

while (table.Rows.Count > 0)

table.Rows.Remove(table.Rows[0]);

employer1BindingSource.EndEdit();

employer1TableAdapter.Update(risk\_databaseDataSet.Employer1);

table = dataGridView5;

employer2TableAdapter.Fill(risk\_databaseDataSet.Employer2);

while (table.Rows.Count > 0)

table.Rows.Remove(table.Rows[0]);

employer2BindingSource.EndEdit();

employer2TableAdapter.Update(risk\_databaseDataSet.Employer2);

table = dataGridView4;

employer3TableAdapter.Fill(risk\_databaseDataSet.Employer3);

while (table.Rows.Count > 0)

table.Rows.Remove(table.Rows[0]);

employer3BindingSource.EndEdit();

employer3TableAdapter.Update(risk\_databaseDataSet.Employer3);

comboBox1\_SelectedIndexChanged(sender, e);

}

if (dialogResult == DialogResult.Cancel)

{

return;

}

}

public void CheckEntry()

{

if (WhoEntered == 1)

{

dataGridView1.Visible = false;

comboBox1.Visible = false;

dataGridView2.Visible = false;

label1.Visible = false;

label2.Visible = false;

button1.Visible = false;

textBox1.Visible = false;

textBox2.Visible = false;

button2.Visible = false;

dataGridView3.Visible = true;

dataGridView4.Visible = false;

dataGridView5.Visible = false;

button3.Visible = false;

comboBox2.Visible = true;

label3.Visible = true;

label4.Visible = true;

textBox3.Visible = true;

textBox4.Visible = true;

button4.Visible = true;

label6.Visible = true;

}

if (WhoEntered == 3)

{

dataGridView1.Visible = false;

comboBox1.Visible = false;

dataGridView2.Visible = false;

label1.Visible = false;

label2.Visible = false;

button1.Visible = false;

textBox1.Visible = false;

textBox2.Visible = false;

button2.Visible = false;

dataGridView3.Visible = false;

dataGridView4.Visible = false;

dataGridView5.Visible = true;

button3.Visible = false;

comboBox2.Visible = true;

label3.Visible = true;

label4.Visible = true;

textBox3.Visible = true;

textBox4.Visible = true;

button4.Visible = true;

label6.Visible = true;

}

if (WhoEntered == 2)

{

dataGridView1.Visible = false;

comboBox1.Visible = false;

dataGridView2.Visible = false;

label1.Visible = false;

label2.Visible = false;

button1.Visible = false;

textBox1.Visible = false;

textBox2.Visible = false;

button2.Visible = false;

dataGridView3.Visible = false;

dataGridView4.Visible = true;

dataGridView5.Visible = false;

button3.Visible = false;

comboBox2.Visible = true;

label3.Visible = true;

label4.Visible = true;

textBox3.Visible = true;

textBox4.Visible = true;

button4.Visible = true;

label6.Visible = true;

}

if (WhoEntered == 4)

{

dataGridView1.Visible = true;

comboBox1.Visible = true;

dataGridView2.Visible = true;

label1.Visible = false;

label2.Visible = false;

button1.Visible = false;

textBox1.Visible = false;

textBox2.Visible = false;

button2.Visible = true;

dataGridView3.Visible = true;

dataGridView4.Visible = false;

dataGridView5.Visible = false;

button3.Visible = true;

label5.Visible = true;

comboBox3.Visible = true;

label7.Visible = true;

}

}

public int WhoEntered;

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

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

{

// текст запроса

string query = "SELECT \* FROM LP1";

// создаем объект OleDbCommand для выполнения запроса к БД MS Access

OleDbCommand command = new OleDbCommand(query, myConnection);

OleDbDataReader ObjReader = command.ExecuteReader();

while (ObjReader.Read())

{

}

//if (textBox1.Text == "E1" && textBox2.Text == "EP1")

//{

// MessageBox.Show("Entry Succeded, Employer1!");

// WhoEntered = 1;

// CheckEntry();

// return;

//}

//if (textBox1.Text == "E2" && textBox2.Text == "EP2")

//{

// MessageBox.Show("Entry Succeded, Employer2!");

// WhoEntered = 2;

// CheckEntry();

// return;

//}

//if (textBox1.Text == "E3" && textBox2.Text == "EP3")

//{

// MessageBox.Show("Entry Succeded, Employer3!");

// WhoEntered = 3;

// CheckEntry();

// return;

//}

if (textBox1.Text == "M1" && textBox2.Text == "MP1")

{

MessageBox.Show("Entry Succeded, Manager1!");

WhoEntered = 4;

CheckEntry();

return;

}

else

{

MessageBox.Show("Incorrect login or password!");

return;

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0)

{

dataGridView1.DataSource = time\_risksBindingSource;

resizearrs();

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

dataGridView1.Rows[i].Visible = TimeArray[i] == 0 ? true : false;

}

}

if (comboBox1.SelectedIndex == 1)

{

dataGridView1.DataSource = financial\_risksBindingSource;

resizearrs();

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

dataGridView1.Rows[i].Visible = FinanceArray[i] == 0 ? true : false;

}

}

if (comboBox1.SelectedIndex == 2)

{

dataGridView1.DataSource = staff\_risksBindingSource;

resizearrs();

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

dataGridView1.Rows[i].Visible = StaffArray[i] == 0 ? true : false;

}

}

if (comboBox1.SelectedIndex == 3)

{

dataGridView1.DataSource = quality\_risksBindingSource;

resizearrs();

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

dataGridView1.Rows[i].Visible = QualityArray[i] == 0 ? true : false;

}

}

}

private void Form1\_Load(object sender, EventArgs e)

{

try

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.Employer1". При необходимости она может быть перемещена или удалена.

this.employer1TableAdapter.Fill(this.risk\_databaseDataSet.Employer1);

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.Employer2". При необходимости она может быть перемещена или удалена.

this.employer2TableAdapter.Fill(this.risk\_databaseDataSet.Employer2);

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.Employer3". При необходимости она может быть перемещена или удалена.

this.employer3TableAdapter.Fill(this.risk\_databaseDataSet.Employer3);

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.ManagerTable". При необходимости она может быть перемещена или удалена.

this.managerTableTableAdapter.Fill(this.risk\_databaseDataSet.ManagerTable);

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.Time\_risks". При необходимости она может быть перемещена или удалена.

this.time\_risksTableAdapter.Fill(this.risk\_databaseDataSet.Time\_risks);

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.Staff\_risks". При необходимости она может быть перемещена или удалена.

this.staff\_risksTableAdapter.Fill(this.risk\_databaseDataSet.Staff\_risks);

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.Quality\_risks". При необходимости она может быть перемещена или удалена.

this.quality\_risksTableAdapter.Fill(this.risk\_databaseDataSet.Quality\_risks);

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.Quality\_risks". При необходимости она может быть перемещена или удалена.

this.financial\_risksTableAdapter.Fill(this.risk\_databaseDataSet.Financial\_risks);

// TODO: данная строка кода позволяет загрузить данные в таблицу "risk\_databaseDataSet.Quality\_risks". При необходимости она может быть перемещена или удалена.

this.financial\_risksTableAdapter.Fill(this.risk\_databaseDataSet.Financial\_risks);

}

catch (Exception)

{

MessageBox.Show("Проверьте подключение баз данных с рисками");

this.Close();

}

if (dataGridView2.RowCount <= 0)

{

}

}

private void financial\_risksBindingSource\_CurrentChanged(object sender, EventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

int CurrRow1 = dataGridView1.CurrentRow.Index;

getarrtype()[CurrRow1] = -1; //Пометка, что он лежит в очереди на распределение

if (dataGridView1.Rows.Count < 1)

{

throw new IndexOutOfRangeException();

}

managerTableBindingSource.Add(null);

dataGridView2.Rows[dataGridView2.Rows.Count - 1].Cells[5].Value = "";

for (int i = 0; i < 5; i++)

{

dataGridView2.Rows[dataGridView2.Rows.Count - 1].Cells[i].Value = dataGridView1.CurrentRow.Cells[i].Value;

}

financial\_risksBindingSource.MoveNext();

time\_risksBindingSource.MoveNext();

staff\_risksBindingSource.MoveNext();

quality\_risksBindingSource.MoveNext();

try

{

dataGridView1.Rows[CurrRow1].Visible = false;

}

catch (Exception ex)

{

//if (dataGridView1.Rows[CurrRow1].Cells[0].Value == null) throw new IndexOutOfRangeException();

//if(comboBox1.SelectedIndex==1)financial\_risksBindingSource.Add(null);

//if (comboBox1.SelectedIndex == 0) time\_risksBindingSource.Add(null);

//if (comboBox1.SelectedIndex == 2) staff\_risksBindingSource.Add(null);

//if (comboBox1.SelectedIndex == 3) quality\_risksBindingSource.Add(null);

if (dataGridView1.CurrentCell == null) throw new IndexOutOfRangeException();

dataGridView1.CurrentCell = null;

dataGridView1.Rows[CurrRow1].Visible = false;

}

managerTableBindingSource.MoveNext();

}

catch (IndexOutOfRangeException)

{

MessageBox.Show("Вы все выбрали");

}

catch (NullReferenceException)

{

MessageBox.Show("Выберите новую базу данных");

}

}

int findin(DataGridView d, DataGridViewRow r)

{

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

{

bool f = true;

for (int j = 1; j < Math.Min(d.Rows[i].Cells.Count, r.Cells.Count); j++)

{

if ((string)d.Rows[i].Cells[j].Value != (string)r.Cells[j].Value)

{

f = false;

break;

}

}

if (f)

{

return i;

}

}

return -1;

}

private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

resizearrs();

if (e.ColumnIndex != 5)

{

return;

}

int CurrRow1 = getcr();//dataGridView2.CurrentRow.Index;

#region Извлечение

int retfi;

switch (comboBox1.SelectedIndex)

{

//Time

//Finance

//Staff

//Quality

case 0:

{

if (TimeArray[CurrRow1] > 0)

{

DataGridView table = null;

switch (TimeArray[CurrRow1])

{

case 1:

{

table = dataGridView3;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer1TableAdapter.Fill(risk\_databaseDataSet.Employer1);

employer1TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value, (string)table.Rows[retfi].Cells[5].Value);

table.Rows.Remove(table.Rows[retfi]);

employer1BindingSource.EndEdit();

//employer1TableAdapter.Update(risk\_databaseDataSet.Employer1);

}

break;

}

case 2:

{

table = dataGridView5;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer2TableAdapter.Fill(risk\_databaseDataSet.Employer2);

employer2TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value);

table.Rows.Remove(table.Rows[retfi]);

employer2BindingSource.EndEdit();

//employer2TableAdapter.Update(risk\_databaseDataSet.Employer2);

}

break;

}

case 3:

{

table = dataGridView4;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer3TableAdapter.Fill(risk\_databaseDataSet.Employer3);

employer3TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value);

table.Rows.Remove(table.Rows[retfi]);

employer3BindingSource.EndEdit();

//employer3TableAdapter.Update(risk\_databaseDataSet.Employer3);

}

break;

}

}

table.CurrentCell = null;

TimeArray[CurrRow1] = 0;

}

break;

}

case 1:

{

if (FinanceArray[CurrRow1] > 0)

{

DataGridView table = null;

switch (FinanceArray[CurrRow1])

{

case 1:

{

table = dataGridView3;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer1TableAdapter.Fill(risk\_databaseDataSet.Employer1);

employer1TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value, (string)table.Rows[retfi].Cells[5].Value);

table.Rows.Remove(table.Rows[retfi]);

employer1BindingSource.EndEdit();

//employer1TableAdapter.Update(risk\_databaseDataSet.Employer1);

}

break;

}

case 2:

{

table = dataGridView5;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer2TableAdapter.Fill(risk\_databaseDataSet.Employer2);

employer2TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value);

table.Rows.Remove(table.Rows[retfi]);

employer2BindingSource.EndEdit();

//employer2TableAdapter.Update(risk\_databaseDataSet.Employer2);

}

break;

}

case 3:

{

table = dataGridView4;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer3TableAdapter.Fill(risk\_databaseDataSet.Employer3);

employer3TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value);

table.Rows.Remove(table.Rows[retfi]);

employer3BindingSource.EndEdit();

//employer3TableAdapter.Update(risk\_databaseDataSet.Employer3);

}

break;

}

}

table.CurrentCell = null;

FinanceArray[CurrRow1] = 0;

}

break;

}

case 2:

{

if (StaffArray[CurrRow1] > 0)

{

DataGridView table = null;

switch (StaffArray[CurrRow1])

{

case 1:

{

table = dataGridView3;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer1TableAdapter.Fill(risk\_databaseDataSet.Employer1);

employer1TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value, (string)table.Rows[retfi].Cells[5].Value);

table.Rows.Remove(table.Rows[retfi]);

employer1BindingSource.EndEdit();

//employer1TableAdapter.Update(risk\_databaseDataSet.Employer1);

}

break;

}

case 2:

{

table = dataGridView5;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer2TableAdapter.Fill(risk\_databaseDataSet.Employer2);

employer2TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value);

table.Rows.Remove(table.Rows[retfi]);

employer2BindingSource.EndEdit();

//employer2TableAdapter.Update(risk\_databaseDataSet.Employer2);

}

break;

}

case 3:

{

table = dataGridView4;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer3TableAdapter.Fill(risk\_databaseDataSet.Employer3);

employer3TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value);

table.Rows.Remove(table.Rows[retfi]);

employer3BindingSource.EndEdit();

//employer3TableAdapter.Update(risk\_databaseDataSet.Employer3);

}

break;

}

}

table.CurrentCell = null;

StaffArray[CurrRow1] = 0;

}

break;

}

case 3:

{

if (QualityArray[CurrRow1] > 0)

{

DataGridView table = null;

switch (QualityArray[CurrRow1])

{

case 1:

{

table = dataGridView3;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer1TableAdapter.Fill(risk\_databaseDataSet.Employer1);

employer1TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value, (string)table.Rows[retfi].Cells[5].Value);

table.Rows.Remove(table.Rows[retfi]);

employer1BindingSource.EndEdit();

//employer1TableAdapter.Update(risk\_databaseDataSet.Employer1);

}

break;

}

case 2:

{

table = dataGridView5;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer2TableAdapter.Fill(risk\_databaseDataSet.Employer2);

employer2TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value);

table.Rows.Remove(table.Rows[retfi]);

employer2BindingSource.EndEdit();

//employer2TableAdapter.Update(risk\_databaseDataSet.Employer2);

}

break;

}

case 3:

{

table = dataGridView4;

retfi = findin(table, dataGridView1.Rows[CurrRow1]);

if (retfi > -1)

{

//employer3TableAdapter.Fill(risk\_databaseDataSet.Employer3);

employer3TableAdapter.Delete((int)table.Rows[retfi].Cells[0].Value, (string)table.Rows[retfi].Cells[1].Value, (string)table.Rows[retfi].Cells[2].Value, (string)table.Rows[retfi].Cells[3].Value, (string)table.Rows[retfi].Cells[4].Value);

table.Rows.Remove(table.Rows[retfi]);

employer3BindingSource.EndEdit();

//employer3TableAdapter.Update(risk\_databaseDataSet.Employer3);

}

break;

}

}

QualityArray[CurrRow1] = 0;

}

break;

}

}

#endregion

switch ((string)dataGridView2.CurrentRow.Cells[5].Value)

{

case "Employer1":

string type = "";

//employer1TableAdapter.Fill(risk\_databaseDataSet.Employer1);

//employer1BindingSource.Add(null);

switch (comboBox1.SelectedIndex)

{

//Time

//Finance

//Staff

//Quality

case 0:

{

type = "Time";

TimeArray[CurrRow1] = 1;

TimeArrayp[CurrRow1] = dataGridView3.Rows.Count - 1;

break;

}

case 1:

{

type = "Finance";

FinanceArray[CurrRow1] = 1;

FinanceArrayp[CurrRow1] = dataGridView3.Rows.Count - 1;

break;

}

case 2:

{

type = "Staff";

StaffArray[CurrRow1] = 1;

StaffArrayp[CurrRow1] = dataGridView3.Rows.Count - 1;

break;

}

case 3:

{

type = "Quality";

QualityArray[CurrRow1] = 1;

QualityArrayp[CurrRow1] = dataGridView3.Rows.Count - 1;

break;

}

}

//dataGridView2.CurrentRow.Cells[5].Value = "Employer1";

//int maxn = -1;

//for (int i = 1; i < dataGridView3.Rows.Count - 1; i++)

//{

// maxn = Math.Max(maxn, (int)dataGridView3.Rows[i].Cells[0].Value);

//}

//for (int i = 1; i < 5; i++)

//{

// dataGridView3.Rows[dataGridView3.Rows.Count - 1].Cells[i].Value = dataGridView2.CurrentRow.Cells[i].Value;

//}

//dataGridView3.Rows[dataGridView3.Rows.Count - 1].Cells[0].Value = Math.Max(maxn + 1, (int)dataGridView3.Rows[dataGridView3.Rows.Count - 1].Cells[0].Value);

Validate();

employer1BindingSource.EndEdit();

employer1TableAdapter.Insert(type, (string)dataGridView2.CurrentRow.Cells[1].Value, (string)dataGridView2.CurrentRow.Cells[2].Value, (string)dataGridView2.CurrentRow.Cells[3].Value, (string)dataGridView2.CurrentRow.Cells[4].Value);

//employer1TableAdapter.Update(risk\_databaseDataSet.Employer1);

employer1TableAdapter.Fill(risk\_databaseDataSet.Employer1);

Validate();

break;

case "Employer2":

//employer2TableAdapter.Fill(risk\_databaseDataSet.Employer2);

//employer2BindingSource.Add(null);

switch (comboBox1.SelectedIndex)

{

//Time

//Finance

//Staff

//Quality

case 0:

{

TimeArray[CurrRow1] = 2;

TimeArrayp[CurrRow1] = dataGridView5.Rows.Count - 1;

break;

}

case 1:

{

FinanceArray[CurrRow1] = 2;

FinanceArrayp[CurrRow1] = dataGridView5.Rows.Count - 1;

break;

}

case 2:

{

StaffArray[CurrRow1] = 2;

StaffArrayp[CurrRow1] = dataGridView5.Rows.Count - 1;

break;

}

case 3:

{

QualityArray[CurrRow1] = 2;

QualityArrayp[CurrRow1] = dataGridView5.Rows.Count - 1;

break;

}

}

//dataGridView2.CurrentRow.Cells[5].Value = "Employer2";

//maxn = -1;

//for (int i = 1; i < dataGridView5.Rows.Count - 1; i++)

//{

// maxn = Math.Max(maxn, (int)dataGridView5.Rows[i].Cells[0].Value);

//}

//for (int i = 1; i < 5; i++)

//{

// dataGridView5.Rows[dataGridView5.Rows.Count - 1].Cells[i].Value = dataGridView2.CurrentRow.Cells[i].Value;

//}

//dataGridView5.Rows[dataGridView5.Rows.Count - 1].Cells[0].Value = Math.Max(maxn + 1, (int)dataGridView5.Rows[dataGridView5.Rows.Count - 1].Cells[0].Value);

employer2BindingSource.EndEdit();

//employer2TableAdapter.Update(risk\_databaseDataSet.Employer2);

employer2TableAdapter.Insert((string)dataGridView2.CurrentRow.Cells[1].Value, (string)dataGridView2.CurrentRow.Cells[2].Value, (string)dataGridView2.CurrentRow.Cells[3].Value, (string)dataGridView2.CurrentRow.Cells[4].Value);

employer2TableAdapter.Fill(risk\_databaseDataSet.Employer2);

Validate();

break;

case "Employer3":

//employer3TableAdapter.Fill(risk\_databaseDataSet.Employer3);

//employer3BindingSource.Add(null);

switch (comboBox1.SelectedIndex)

{

//Time

//Finance

//Staff

//Quality

case 0:

{

TimeArray[CurrRow1] = 3;

TimeArrayp[CurrRow1] = dataGridView4.Rows.Count - 1;

break;

}

case 1:

{

FinanceArray[CurrRow1] = 3;

FinanceArrayp[CurrRow1] = dataGridView4.Rows.Count - 1;

break;

}

case 2:

{

StaffArray[CurrRow1] = 3;

StaffArrayp[CurrRow1] = dataGridView4.Rows.Count - 1;

break;

}

case 3:

{

QualityArray[CurrRow1] = 3;

QualityArrayp[CurrRow1] = dataGridView4.Rows.Count - 1;

break;

}

}

//dataGridView2.CurrentRow.Cells[5].Value = "Employer3";

//maxn = -1;

//for (int i = 1; i < dataGridView4.Rows.Count - 1; i++)

//{

// maxn = Math.Max(maxn, (int)dataGridView4.Rows[i].Cells[0].Value);

//}

//for (int i = 1; i < 5; i++)

//{

// dataGridView4.Rows[dataGridView4.Rows.Count - 1].Cells[i].Value = dataGridView2.CurrentRow.Cells[i].Value;

//}

//dataGridView4.Rows[dataGridView4.Rows.Count - 1].Cells[0].Value = Math.Max(maxn + 1, (int)dataGridView4.Rows[dataGridView4.Rows.Count - 1].Cells[0].Value);

employer3BindingSource.EndEdit();

//employer3TableAdapter.Update(risk\_databaseDataSet.Employer3);

employer3TableAdapter.Insert((string)dataGridView2.CurrentRow.Cells[1].Value, (string)dataGridView2.CurrentRow.Cells[2].Value, (string)dataGridView2.CurrentRow.Cells[3].Value, (string)dataGridView2.CurrentRow.Cells[4].Value);

employer3TableAdapter.Fill(risk\_databaseDataSet.Employer3);

Validate();

break;

}

}

private void textBox6\_TextChanged(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

bool f = true;

double v = 1;

DataGridView d = dataGridView3;

if (WhoEntered == 2) d = dataGridView5;

if (WhoEntered == 3) d = dataGridView4;

try

{

if (d.Rows.Count < 1)

{

f = false;

throw new Exception();

}

if (Convert.ToDouble(textBox3.Text) > 1 || Convert.ToDouble(textBox3.Text) < 0)

{

f = false;

throw new OverflowException();

}

else

{

v \*= Convert.ToDouble(textBox3.Text);

}

}

catch (OverflowException)

{

MessageBox.Show("Введите значение вероятности между 0 и 1");

return;

}

catch (FormatException)

{

MessageBox.Show("Введите корректное значение вероятности");

return;

}

catch (Exception)

{

MessageBox.Show("У вас нет рисков для оценки!");

}

try

{

if (Convert.ToDouble(textBox4.Text) > 1 || Convert.ToDouble(textBox4.Text) < 0)

{

f = false;

throw new OverflowException();

}

else

{

v \*= Convert.ToDouble(textBox4.Text);

}

}

catch (OverflowException)

{

MessageBox.Show("Введите значение воздействия между 0 и 1");

return;

}

catch (FormatException)

{

MessageBox.Show("Введите корректное значение воздействия");

return;

}

if (f)

{

form2.addselectedrisk(d, v, Convert.ToDouble(textBox3.Text), Convert.ToDouble(textBox4.Text), (string)comboBox2.SelectedItem);

if (!form2.IsAccessible)

form2.ShowDialog(); // OLD TABLE

//form2.draw\_risk\_Click(sender, e);

}

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

private void dataGridView2\_CellContentClick\_1(object sender, DataGridViewCellEventArgs e)

{

}

private void textBox1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

button1\_Click(sender, e);

}

}

/// <summary>

///Рисуем карту рисков

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void draw\_risk\_Click(object sender, EventArgs e)

{

try

{

DataSender ds = new DataSender();

ds.n = 1;

ds.points = new List<ColorPoint>();

double scalecoeff = 50;

// проходимся по дереву выбранных атрибутов, берем значение вероятности и влияния для выбранного элемента

for (int ns = 10; ns < dataGridView1.Rows.Count; ns++)// перебираем столбцы (перебирая системы)

{

double x = (double)dataGridView1.Rows[ns].Cells[5].Value; // вероятность

double y = (double)dataGridView1.Rows[ns].Cells[6].Value; // влияние

string name = $"{dataGridView1.Rows[ns].Cells[1]}\r\n{dataGridView1.Rows[ns].Cells[2]}\r\n{dataGridView1.Rows[ns].Cells[3]}\r\n{dataGridView1.Rows[ns].Cells[4]}\r\n{dataGridView1.Rows[ns].Cells[5]}\r\n{comboBox2.SelectedItem}";// $"Система - {rsk[ns - 1]}, \r\nЭлемент - {(string)dataGridView3.Rows[oldrowindex].Cells[0].Value}, \r\nАтрибут - {(string)dataGridView3.Rows[i + oldrowindex].Cells[0].Value}";

ColorPoint cp = new ColorPoint(x, y, 10/\*Радиус точки для отрисовки\*/, name, Color.Black, scalecoeff);

ds.points.Add(cp);

}

FormGraphics fr = new FormGraphics(ds, $"Карта рисков для элемента \"{"Change some string"}\"", own: this);

fr.Show();

}

catch (Exception ex)

{

}

}

private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

{

WhoEntered = comboBox3.SelectedIndex + 1;

if (WhoEntered == 2) WhoEntered = 4;

CheckEntry();

}

private void label7\_Click(object sender, EventArgs e)

{

}

public void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

// обработчик события закрытия формы

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

// заркываем соединение с БД

myConnection.Close();

}

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| RU.17701729.507900-01 12 |  |  |  |  |
| Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата |

**1.2. КлассRiskEstimate**

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;

namespace CourseWork

{

publicpartialclassRiskEstimate : Form

{

public RiskEstimate()

{

InitializeComponent();

#regionДобавлениетаблицысозначениямирисков

dataGridView1.Rows.Add();

dataGridView1.Rows[0].Cells[0].Value = "Вероятность";

this.dataGridView1.Rows.Add(5);

dataGridView1.Rows[1].Cells[0].Value = 0.9;

dataGridView1.Rows[2].Cells[0].Value = 0.7;

dataGridView1.Rows[3].Cells[0].Value = 0.5;

dataGridView1.Rows[4].Cells[0].Value = 0.3;

dataGridView1.Rows[5].Cells[0].Value = 0.1;

for(int i = 1; i < 6; i++)

{

for(int j = 2; j < 7; j++)

{

double n = Convert.ToDouble(dataGridView1.Columns[j].HeaderText);

dataGridView1.Rows[i].Cells[j].Value = (double)dataGridView1.Rows[i].Cells[0].Value \* n;

if ((double)dataGridView1.Rows[i].Cells[j].Value >= 0.04) dataGridView1.Rows[i].Cells[j].Style.BackColor = Color.Orange;

if ((double)dataGridView1.Rows[i].Cells[j].Value >= 0.18) dataGridView1.Rows[i].Cells[j].Style.BackColor = Color.Red;

}

}

dataGridView1.Rows.Add();

dataGridView1.Rows[dataGridView1.Rows.Count - 1].DefaultCellStyle.BackColor = Color.OldLace;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[0].Value = "";

dataGridView1.Rows.Add();

dataGridView1.Rows[dataGridView1.Rows.Count - 1].DefaultCellStyle.BackColor = Color.OldLace;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[0].Value = "";

dataGridView1.Rows.Add();

dataGridView1.Rows[dataGridView1.Rows.Count - 1].DefaultCellStyle.BackColor = Color.OldLace;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[0].Value = "";

#endregion

this.dataGridView1.Rows.Add();

dataGridView1.Rows[dataGridView1.Rows.Count - 1].DefaultCellStyle.BackColor = Color.PowderBlue;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[0].Value = "Названияриска";

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[1].Value = "Источникриска";

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[2].Value = "Последствияриска";

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[3].Value = "Описаниериска";

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[4].Value = "Оценкариска";

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[5].Value = "Вероятностьриска";

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[6].Value = "Воздействиериска";

}

publicvoid addselectedrisk(DataGridView d, double v, double ver, double vozd, string str)

{

this.dataGridView1.Rows.Add();

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[0].Value = d.CurrentRow.Cells[1].Value;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[1].Value = d.CurrentRow.Cells[2].Value;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[2].Value = d.CurrentRow.Cells[3].Value;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[3].Value = d.CurrentRow.Cells[4].Value;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[4].Value = v;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[5].Value = ver;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[6].Value = vozd;

dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[7].Value = str;

if ((double)dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[4].Value >= 0.04) dataGridView1.Rows[dataGridView1.Rows.Count - 1].DefaultCellStyle.BackColor = Color.Orange;

if ((double)dataGridView1.Rows[dataGridView1.Rows.Count - 1].Cells[4].Value >= 0.18) dataGridView1.Rows[dataGridView1.Rows.Count - 1].DefaultCellStyle.BackColor = Color.Red;

}

privatevoid RiskEstimate\_Load(object sender, EventArgs e)

{

}

}

}

**1.3. Класс DrawingWindow**

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;

namespace WindowsFormsApp1

{

/// <summary>

/// Класс для формы, на которой будут рисоваться графики

/// </summary>

public partial class FormGraphics : Form

{

private float posx, posy, pox, poy; // положение левого верхнего угла битмапа на форме (нужно для перемещения)

private ForGraphics gr; // класс для графика

public DataSender ds; // данные для построения графика

private int nowp = -1; // номер точки, с которой начнётся проверка выбора при следующем двойном нажатии на левую кнопку мышки (для случая совпадания точек на графике)

private bool f = false; // Происходит ли перемещение графика относительно формы

Form owner = null;

private Bitmap bmph = null;

/// <summary>

/// конструктор

/// </summary>

/// <param name="ds"></param>

public FormGraphics(DataSender ds, string name = "Graphics", Form own = null)

{

try

{

this.ds = ds;

this.owner = own;

this.Owner = null;

DoubleBuffered = true;

gr = new ForGraphics(this);

InitializeComponent();

Invalidate();

Init(); // задание изначальной позиции верхнего левого угла битмапа

this.Focus();

this.Text = name;

//this.TopMost = true; // будет всегда поверх остальных окон

//this.TopMost = false; // будет поверх остальных окон, но уже не всегда

this.MinimumSize = new Size(Screen.PrimaryScreen.WorkingArea.Width / 2, Screen.PrimaryScreen.WorkingArea.Height / 2);

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Построение графика

/// </summary>

void DrawChart(bool firstdraw = false)

{

if (f) Init();

float scl = gr.scale;

gr.scale = Math.Max(20, gr.scale);

string text\_help = "СПРАВКА:" +

"\r\nПеремещать график можно с помощью мышки. " +

"\r\nУвеличивать/уменьшать изображение на графике можно с помощью колесика мышки или клавиш \"W\" / \"S\"." +

"\r\nДля возврата к исходному положению и размеру графика нажмите \"R\".";

switch (ds.n) // по значению n - определяем, какой из графиков рисовать

{

case 0:

{

string histog = "На гистограмме расположены группы столбцов. " +

"Каждая группа задает влияние/вероятность риска соответствующего атрибута (в элементе, выбранном на 2 вкладке) для всех рассматриваемых ИТ систем. " +

"Номер системы задается цифрой над соответствующим столбцом. Тип атрибута задается названием над соответствующей группой столбцов. " +

"Столбцы гистограмм окрашены в красный, желтый, зеленый цвета в зависимости от относительной значения высоты. " +

"Кол-во групп равно кол-ву атрибутов, выбранных на 2-ой вкладке. ";

//textBox2.Text/\*справка\*/ = text\_help + "\r\n"+ histog;

richTextBox1.Text = "СПРАВКА:";

text\_help = "\r\nПеремещать график можно с помощью мышки. " +

"\r\nУвеличивать/уменьшать изображение на графике можно с помощью колесика мышки или клавиш \"W\" / \"S\"." +

"\r\nДля возврата к исходному положению и размеру графика нажмите \"R\".";

richTextBox1.SelectionStart = richTextBox1.TextLength;

richTextBox1.SelectionLength = 0;

richTextBox1.SelectionColor = Color.Blue;

richTextBox1.AppendText(text\_help + "\r\n");

richTextBox1.SelectionColor = richTextBox1.ForeColor;

richTextBox1.SelectionColor = Color.Black;

richTextBox1.AppendText(histog);

richTextBox1.SelectionColor = richTextBox1.ForeColor;

textBox1.Text = "Для вывода информации о столбцах на графике два раза нажмите на столбец."; ;

if (firstdraw)

{

gr.DrawHists(ds);

this.bmph = new Bitmap((Image)gr.bmp.Clone());

}

gr.scale = scl;

gr.DrawHists(ds);

break;

}

case 1:

{

textBox1.Text = "Для вывода информации о точках на графике два раза нажмите на точку.";

//textBox2.Text/\*справка\*/ = text\_help + "\r\nТочки выводятся зеленым, желтым и красным цветом для обозначения низкого, среднего и высокого уровней риска соответственно.";

richTextBox1.Text = "СПРАВКА:";

text\_help = "\r\nПеремещать график можно с помощью мышки. " +

"\r\nУвеличивать/уменьшать изображение на графике можно с помощью колесика мышки или клавиш \"W\" / \"S\"." +

"\r\nДля возврата к исходному положению и размеру графика нажмите \"R\".";

richTextBox1.SelectionStart = richTextBox1.TextLength;

richTextBox1.SelectionLength = 0;

richTextBox1.SelectionColor = Color.Blue;

richTextBox1.AppendText(text\_help + "\r\n");

richTextBox1.SelectionColor = richTextBox1.ForeColor;

richTextBox1.SelectionColor = Color.Black;

richTextBox1.AppendText("Точки выводятся зеленым, желтым и красным цветом для обозначения низкого, среднего и высокого уровней риска соответственно.");

richTextBox1.SelectionColor = richTextBox1.ForeColor;

if (firstdraw)

{

gr.DrawPoints(ds);

this.bmph = new Bitmap((Image)gr.bmp.Clone());

}

gr.scale = scl;

gr.DrawPoints(ds);

break;

}

case 2:

{

textBox1.Text = "Для вывода информации о точках на графике два раза нажмите на точку.";

//textBox2.Text/\*справка\*/ = text\_help + "\r\nДиаметр круга, соответствующего системе, пропорционален стоимости данной системы.";

richTextBox1.Text = "СПРАВКА:";

text\_help = "\r\nПеремещать график можно с помощью мышки. " +

"\r\nУвеличивать/уменьшать изображение на графике можно с помощью колесика мышки или клавиш \"W\" / \"S\"." +

"\r\nДля возврата к исходному положению и размеру графика нажмите \"R\".";

richTextBox1.SelectionStart = richTextBox1.TextLength;

richTextBox1.SelectionLength = 0;

richTextBox1.SelectionColor = Color.Blue;

richTextBox1.AppendText(text\_help + "\r\n");

richTextBox1.SelectionColor = richTextBox1.ForeColor;

richTextBox1.SelectionColor = Color.Black;

richTextBox1.AppendText("Диаметр круга, соответствующего системе, пропорционален стоимости данной системы.");

richTextBox1.SelectionColor = richTextBox1.ForeColor;

if (firstdraw)

{

gr.DrawBigPoints(ds);

this.bmph = new Bitmap((Image)gr.bmp.Clone());

}

gr.scale = scl;

gr.DrawBigPoints(ds);

break;

}

}

firstdraw = false;

//gr.bmp = new Bitmap(((Image)gr.bmp.Clone()), (int)(gr.bmp.Width \* scl / gr.scale), (int)(gr.bmp.Height \* scl / gr.scale));

Invalidate(); // будет вызван OnPaint

}

/// <summary>

/// Переопределение перерисовки окна - вызывается при вызове Invalidate().

/// (Переопределение стандартного метода OnPaint)

/// </summary>

/// <param name="e"></param>

protected override void OnPaint(PaintEventArgs e)

{

try

{

if (gr.bmp != null)

{

e.Graphics.DrawImage(gr.bmp, posx, posy);// отрисовываем битмап на нашей форме

}

base.OnPaint(e);

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Для выбора точек (к которым выведется информация) с помощью двойного щелчка левой кнопкой мыши

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox\_MouseDoubleClick(object sender, MouseEventArgs e)

{

try

{

Point nowpoint = new Point((int)(e.X - posx)/\*переход в относительные координаты битмапа из координат на форме\*/, (int)(e.Y - posy));

if (ds.n == 0)

{

if (nowp < 0) { nowp = gr.mas\_rect.Count - 1; }

for (int i = 0; i < gr.mas\_rect.Count; i++, nowp = (nowp - 1 + gr.mas\_rect.Count) % gr.mas\_rect.Count) // проходим по массиву отрисованных столбцов

{

var debe = gr.mas\_rect[nowp];

if (nowpoint < gr.mas\_rect[nowp])

{

this.textBox1.Text = "" + gr.mas\_rect[nowp].name;

nowp = (nowp - 1 + gr.mas\_rect.Count) % gr.mas\_rect.Count; // чтобы при нажатии на то же самое место на карте - выводились системы поочередно, заключенные в данной точке

//(остаток так как мы можем проходить массив не с 0-го элемента и переходить через последний элемент к 0-му элементу)

break;

}

}

}

else

{

if (nowp < 0) { nowp = gr.mas\_point.Count - 1; }

for (int i = 0; i < gr.mas\_point.Count; i++, nowp = (nowp - 1 + gr.mas\_point.Count) % gr.mas\_point.Count) // проходим по массиву отрисованных точек

{

if (nowpoint + gr.mas\_point[nowp] < gr.mas\_point[nowp].rv)

{

this.textBox1.Text = "" + gr.mas\_point[nowp].name;

nowp = (nowp - 1 + gr.mas\_point.Count) % gr.mas\_point.Count; // чтобы при нажатии на то же самое место на карте - выводились системы поочередно, заключенные в данной точке

//(остаток так как мы можем проходить массив не с 0-го элемента и переходить через последний элемент к 0-му элементу)

break;

}

}

}

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

#region Перемещение графика

/// <summary>

/// Левая кнопка мыши нажата

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox\_MouseDown(object sender, MouseEventArgs e)

{

try

{

if (gr.bmp == null) return;

if (!f)

{

pox = e.X;

poy = e.Y;

f = true;

return;

}

else

{

}

}

catch (Exception ex)

{

////ApplicationClosingByException(ex);

}

}

/// <summary>

/// Левая кнопка мыши отпущена

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox\_MouseUp(object sender, MouseEventArgs e)

{

try

{

if (gr.bmp == null) return;

if (f)

{

f = false;

}

}

catch (Exception ex)

{

////ApplicationClosingByException(ex);

}

}

/// <summary>

/// Перемещение курсора

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

try

{

if (gr.bmp == null) return;

if (f)

{

float dx = e.X - pox, dy = e.Y - poy;

pox += dx;

poy += dy;

posx += dx;

posy += dy;

posx = Math.Min(this.Width - 18, Math.Max(-gr.bmp.Width + 1, posx)); //чтобы не отдалялось за рамки - часть битмапа всегда внутри формы

posy = Math.Min(this.Height - 38, Math.Max(-gr.bmp.Height + 1, posy));

Invalidate();

}

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

#endregion

#region Масштабирование графика

/// <summary>

/// Приближение в конкретной точке.

/// Реагирует на курсор.

/// </summary>

/// <param name="e"></param>

void ZoomUp(MouseEventArgs e)

{

try

{

//gr.scale = Math.Min(50,Math.Max(4, gr.scale \*(float)1.5));

gr.scale \*= (float)1.5;

if (gr.scale < 4)

{

gr.scale /= (float)1.5;

return;

}

if (gr.scale > 50)

{

gr.scale /= (float)1.5;

return;

}

pox -= (posx - e.X) - (posx - e.X) \* (float)(1.5); // Чтобы увеличивать на том месте , к которому направлена мышка

poy -= (posy - e.Y) - (posy - e.Y) \* (float)(1.5);

posx -= (posx - e.X) - (posx - e.X) \* (float)(1.5);

posy -= (posy - e.Y) - (posy - e.Y) \* (float)(1.5);

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Приближение в левом верхнем углу.

/// Реагирует на кнопку для быстрого действия и приближает "в левом верхнем углу", а не к "курсору".

/// </summary>

void ZoomUp()

{

try

{

//gr.scale = Math.Min(50, Math.Max(4, gr.scale \*(float)1.5));

gr.scale \*= (float)1.5;

if (gr.scale < 4)

{

gr.scale /= (float)1.5;

return;

}

if (gr.scale > 50)

{

gr.scale /= (float)1.5;

return;

}

Rewrite();

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Отдаление в конкретной точке.

/// Реагирует на курсор.

/// </summary>

/// <param name="e"></param>

void ZoomDown(MouseEventArgs e)

{

try

{

//gr.scale= Math.Min(50, Math.Max(4,gr.scale / (float)1.5));

gr.scale /= (float)1.5;

if (gr.scale < 4)

{

gr.scale \*= (float)1.5;

return;

}

if (gr.scale > 50)

{

gr.scale \*= (float)1.5;

return;

}

pox -= (posx - e.X) - (posx - e.X) / (float)(1.5);

poy -= (posy - e.Y) - (posy - e.Y) / (float)(1.5);

posx -= (posx - e.X) - (posx - e.X) / (float)(1.5);

posy -= (posy - e.Y) - (posy - e.Y) / (float)(1.5);

//Rewrite();

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Отдаление в левом вехнем углу экрана.

/// Реагирует на кнопку для быстрого действия и приближает "в левом верхнем углу", а не к "курсору".

/// </summary>

void ZoomDown()

{

try

{

//gr.scale = Math.Min(50, Math.Max(4, gr.scale / (float)1.5));

gr.scale /= (float)1.5;

if (gr.scale < 4)

{

gr.scale \*= (float)1.5;

return;

}

if (gr.scale > 50)

{

gr.scale \*= (float)1.5;

return;

}

Rewrite();

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Масштабирование изображения.

/// Реагирует на прокрутку колесика мышки.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void pictureBox\_MouseWheel(object sender, MouseEventArgs e)

{

try

{

if (gr.bmp == null) return;

if (e.Delta > 0)

{

ZoomUp(e);

}

else

{

ZoomDown(e);

}

Rewrite();

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

#endregion

/// <summary>

/// Отлов нажатий клавиш.

/// Клавиши для быстрых операций.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

try

{

if (gr.bmp == null) return;

if (e.KeyCode == Keys.R)

{

Init();

}

if (e.KeyCode == Keys.W)

{

ZoomUp();

}

if (e.KeyCode == Keys.S)

{

ZoomDown();

}

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Перерисовываем фрактал

/// </summary>

void Rewrite()

{

try

{

DrawChart();

}

catch (System.ArgumentException ex)

{

Problem("Слишком большое приближение/удаление \n");

Init();

}

catch (Exception ex)

{

Problem("\n" + ex.Message);

}

Invalidate();

}

/// <summary>

/// Сохранение изображения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

try

{

try

{

if (gr.bmp == null) throw (new NullReferenceException());

SaveFileDialog FBD = new SaveFileDialog();

FBD.Filter = "Image Files(\*.BMP)|\*.BMP|Image Files(\*.JPG)|\*.JPG|Image Files(\*.PNG)|\*.PNG|All Files(\*.\*)|\*.\*";

FBD.FilterIndex = 2; // задаем по умолчанию JPG

FBD.FileName = "Сhart"; // имя по умолчанию

if (FBD.ShowDialog() == DialogResult.OK)

{

DrawChart(true);

this.bmph.Save(FBD.FileName);

}

}

catch (ArgumentNullException ex)

{

Problem("Неверный путь");

}

catch (System.Runtime.InteropServices.ExternalException ex)

{

Problem("Невозможно сохранить из-за ошибки файловой системы");

}

catch (NullReferenceException ex)

{

Problem("Невозможно сохранить несуществующий объект\n" + ex.Message);

}

catch (Exception ex)

{

Problem("" + ex.Message);

}

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Вывод сообщения об ошибке

/// </summary>

/// <param name="s"></param>

void Problem(string s)

{

MessageBox.Show(s);

}

/// <summary>

/// Стартовые значения позиции и размера

/// </summary>

/// <param name="draw">Перерисовать фрактал?Да:Нет</param>

private void Init(bool draw = true)

{

try

{

if (gr.bmp == null) return;

posx = gr.space;

posy = gr.space + textBox1.Height;

gr.scale = 10;

if (draw)

{

DrawChart();

Invalidate();

}

}

catch (Exception ex)

{

//ApplicationClosingByException(ex);

}

}

/// <summary>

/// Закрытие приложения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

void Form1Closed(object sender, EventArgs e)

{

Dispose();

}

///// <summary>

///// Метод для закрытия приложения при возникновении исключения

///// </summary>

//private void ApplicationClosingByException(Exception ex = null, string s = "")

//{

// Program.formclosedbyexception = true;

// if (ex != null)

// {

// Program.exceptionmessage = $"Возникло исключение. Форма будет перещапущена. \r\nДополнительная информация: \r\n{ex.Message}";

// }

// else

// {

// Program.exceptionmessage = s;

// }

// foreach (var i in this.OwnedForms)

// {

// i.Close();

// }

// bool f = MessageBox.Show(Program.exceptionmessage, "Перезапустить приложение?", MessageBoxButtons.YesNo) == DialogResult.Yes;

// this.owner?.Close();

// this.Close();

// Program.formclosedbyexception = f;

//}

}

}

**1.4. Класс ForGraphics**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

/// <summary>

/// Класс для отрисовки графиков

/// </summary>

public class ForGraphics

{

public Bitmap bmp; // битмап

Graphics graph;

Brush brush; // специальная кисть для рисования

public float scale = 10; //Масштаб

public int dspace = 3; // Ширина белой рамки (зазора) вокруг графика (т.е. дополнительная зона около осей - для надписи)

public int space { get { return Math.Max((int)(this.dspace \* scale),15); } } // Ширина белой рамки (зазора) вокруг графика (т.е. дополнительная зона около осей - для надписи)

public double vscale = 6; // Отношение значения риска атрибута к высоте столбца гистограммы (столбец диаграммы будет в vscale раз выше, чем вычисленный риск атрибута )

public float TextSize = 5f; // размер шрифта

public List<ColorPoint> mas\_point = new List<ColorPoint>(); // массив точек

public List<ColorRect> mas\_rect = new List<ColorRect>(); // массив точек

/// <summary>

/// конструктор о умолчаниию

/// </summary>

public ForGraphics() { }

public ForGraphics(Form f, float scale = 1)

{

this.scale = scale;

this.bmp = new Bitmap((int)(f.Width \* this.scale), (int)(f.Width \* this.scale));// создаем битмап размеров с форму

this.graph = Graphics.FromImage(bmp); // график, который лежит в битмапе

}

/// <summary>

/// построение стрелок и подписи к ним

/// </summary>

/// <param name="xe">максимальное значение на оси x</param>

/// <param name="ye">максимальное значение на оси y</param>

/// <param name="zv">нулевое значение (начало координат)</param>

/// <param name="xs">подпись под осью x</param>

/// <param name="ys">подпись под осью y</param>

void DrawAxis(int xe = -1, int ye = -1, int zv = 0, string xs = null, string ys = null)

{

graph.FillRectangle(System.Drawing.Brushes.White, 0, 0, bmp.Width, bmp.Height);// белая область размером с наш график

float textsize = (float)(TextSize \* 2.5) \* scale / 8;// размер шрифта

float arroww = 0.1f \* this.scale;

// надо ли рисовать нулевое значение

if (xe != -1 || ye != -1) graph.DrawString(zv.ToString(), new Font("Tempus Sans ITC", textsize, FontStyle.Bold), new SolidBrush(System.Drawing.Color.Black), (3), bmp.Height - space + 2);

// прямая ox и две "палочки для стрелки"

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

{

graph.DrawLine(new Pen(Color.Black), new Point(space, bmp.Height - space + i), new Point(bmp.Width - space, bmp.Height - space + i));

graph.DrawLine(new Pen(Color.Black), new Point(bmp.Width - space - (int)(5 \* scale/10), bmp.Height - space - (int)(3 \* scale/10) + i), new Point(bmp.Width - space, bmp.Height - space + i));

graph.DrawLine(new Pen(Color.Black), new Point(bmp.Width - space - (int)(5 \* scale/10), bmp.Height - space + (int)(3 \* scale/10) + (int)(arroww) - i), new Point(bmp.Width - space, bmp.Height - space + (int)(arroww) - i));

}

if (xs != null) graph.DrawString(xs, new Font("Tempus Sans ITC", textsize, FontStyle.Bold), new SolidBrush(System.Drawing.Color.Black), (bmp.Width - 2 \* space) / 2 - (int)(xs.Length \* textsize / 5.5)/\*половина длины поля для рисования\*/, bmp.Height - space + 2); // Текст будет по центру

// если задано максимальное, рисуем его

if (xe != -1) graph.DrawString(xe.ToString(), new Font("Tempus Sans ITC", textsize, FontStyle.Bold), new SolidBrush(System.Drawing.Color.Black), (bmp.Width - space), bmp.Height - space + 2);

// прямая oy и две "палочки для стрелки"

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

{

graph.DrawLine(new Pen(Color.Black), new Point(space - i, bmp.Height - space), new Point(space - i, space));

graph.DrawLine(new Pen(Color.Black), new Point(space - (int)(3 \* scale/10) - (int)(arroww) + i, space + (int)(5 \* scale/10)), new Point(space - (int)(arroww) + i, space));

graph.DrawLine(new Pen(Color.Black), new Point(space + (int)(3 \* scale/10) - i, space + (int)(5 \* scale/10)), new Point(space - i, space));

}

if (ys != null) graph.DrawString(ys, new Font("Tempus Sans ITC", textsize, FontStyle.Bold), new SolidBrush(System.Drawing.Color.Black), space + 5, space - 7); // Правее чем стрелочка

// если задано максимальное, рисуем его

if (ye != -1) graph.DrawString(ye.ToString(), new Font("Tempus Sans ITC", textsize, FontStyle.Bold), new SolidBrush(System.Drawing.Color.Black), 3, space - 7);

}

/// <summary>

/// Рисуем прямоугольник (Только для гистограммы)

/// </summary>

/// <param name="x">координата левого верхнего угла</param>

/// <param name="y">координата левого верхнего угла</param>

/// <param name="w">ширина</param>

/// <param name="h">высота</param>

/// <param name="c">цвет</param>

/// <param name="s">название</param>

/// <param name="f">рисовать столбец или текст</param>

ColorRect DrawRect(int x, int y, int w, int h, Color c, string s = null, bool f=true)

{

float textsize = (float)(TextSize \* 2.5) \* scale / 16;// размер шрифта

if (f)

{

brush = new System.Drawing.SolidBrush(c); // кисточка нужного цвета

graph.FillRectangle(this.brush, x, y, w, h);

}

else

{

// Добавление имени над прямоугольником

if (s != null) graph.DrawString(s, new Font("Tempus Sans ITC", textsize, FontStyle.Bold), new SolidBrush(System.Drawing.Color.Black), (float)(x + w / 2.0 - s.Length \* textsize / 5.5), (float)(y - 28 \* scale / 10.0));

}

return new ColorRect(x, y, w, h, s, c);

}

/// <summary>

/// Рисование точки (карта рисков и карта соотношений)

/// </summary>

/// <param name="x">координата верхнего левого угла прямоугольника, в который вписана окружность</param>

/// <param name="y"></param>

/// <param name="r">радиус</param>

/// <param name="c"></param>

/// <param name="s"></param>

/// <returns></returns>

ColorPoint DrawPoint(int x, int y, int r, Color c, string s)

{

float textsize = (float)(TextSize \* 2.5); // размер шрифта

brush = new System.Drawing.SolidBrush(c);

graph.FillEllipse(this.brush, x - r, y - r, r \* 2, r \* 2);

return new ColorPoint(x, y, r, s, c);

}

/// <summary>

/// Функция для рисования гистограммы

/// </summary>

/// <param name="ds"></param>

/// <param name="histogramwigth"></param>

/// <param name="spaceingroup"></param>

/// <param name="spacebetweengroups"></param>

public void DrawHists(DataSender ds, int histogramwigth = 5, int spaceingroup = 2, int spacebetweengroups = 5)

{

mas\_rect = new List<ColorRect>();

List<List<int>> histogram = new List<List<int>>(); // массив, высот столбцов гистограммы приведённый к int

int counter = 0;

for (int i = 0; i < ds.histogram\_h.Count; i++)

{

histogram.Add(new List<int>());

for (int j = 0; j < ds.histogram\_h[i].Count; j++)

{

histogram[i].Add((int)(Math.Max(1,ds.histogram\_h[i][j] \* vscale)));

counter++;

}

}

int wigth = (int)((((histogramwigth) + (spaceingroup)) \* (counter) + (spacebetweengroups - spaceingroup) \* (histogram.Count) + spacebetweengroups) \* scale + space \* 2);

List<List<string>> names = ds.histogram\_name; // массив, имён столбцов гистограммы

// ширина белой области, на которой рисуется график

// ищем максимальное значение риска для нахождения высоты белой области, на которой рисуется график

int maxv = histogram[0][0];

foreach (var i in histogram)

{

foreach (var j in i)

{

maxv = Math.Max(maxv, j);

}

}

int heigth = (int)((((maxv) + 28 / 10.0) \* scale + space \* 2 + space)); // spaсe - ширина рамки вокруг графика (для надписей)

bmp = new Bitmap(wigth, heigth);

graph = Graphics.FromImage(bmp);

DrawAxis(xs: "Атрибуты", ys: " ", ye: 1); //рисуем оси и создаем белое поле нужного размера

// поэтапно рисуем прямоугольники для гистограммы

int nowx = (int)(space + (spacebetweengroups) \* scale); // нынешняя позиция по оси x

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

{

for (int j = 0; j < histogram[i].Count; j++)

{

Color c = Color.Green; // далеее, смотря по высоте прямоугольника, при необходимости поменяем на желтый или красный

if (histogram[i][j] >= maxv / 3 && histogram[i][j] < maxv \* 2 / 3) c = Color.Yellow;

if (histogram[i][j] >= maxv \* 2 / 3) c = Color.Red;

mas\_rect.Add(DrawRect((int)(nowx), (int)(bmp.Height - space - (int)(histogram[i][j] \* scale)), (int)(histogramwigth \* scale), (int)(histogram[i][j] \* scale), c, names[i][j]));

nowx += (int)((histogramwigth + spaceingroup) \* scale);

}

nowx += (int)((spacebetweengroups - spaceingroup) \* scale);

}

nowx = (int)(space + (spacebetweengroups) \* scale); // нынешняя позиция по оси x

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

{

maxv = 0;

for (int j = 0; j < histogram[i].Count; j++)

{

maxv = Math.Max(maxv, histogram[i][j]);

}

for (int j = 0; j < histogram[i].Count; j++)

{

Color c = Color.Green; // далеее, смотря по высоте прямоугольника, при необходимости поменяем на желтый или красный

if (histogram[i][j] >= maxv / 3 && histogram[i][j] < maxv \* 2 / 3) c = Color.Yellow;

if (histogram[i][j] >= maxv \* 2 / 3) c = Color.Red;

if (j == histogram[i].Count / 2)

{

DrawRect((int)(nowx), (int)(bmp.Height - space - (int)(maxv \* scale + 28)), (int)(histogramwigth \* scale), (int)(maxv \* scale + 28), c, ds.histogram\_attr\_name[i], false);

}

DrawRect((int)(nowx), (int)(bmp.Height - space - (int)(histogram[i][j] \* scale)), (int)(histogramwigth \* scale), (int)(histogram[i][j] \* scale), c, (j + 1).ToString(), false);

nowx += (int)((histogramwigth + spaceingroup) \* scale);

}

nowx += (int)((spacebetweengroups - spaceingroup) \* scale);

}

}

/// <summary>

/// Функция для построение 2-го графика (карта рисков)

/// </summary>

/// <param name="ds"></param>

/// <param name="f1"></param>

/// <param name="f2"></param>

public void DrawPoints(DataSender ds, DataSender.Func f1 = null, DataSender.Func f2 = null)

{

mas\_point = new List<ColorPoint>();

if (f1 == null)

{

if (ds.f1 != null)

{

f1 = ds.f1;

}

else

{

f1 = delegate (double x) { return ((1.0 / (Math.Max(0,x \* 100.0 - 32))) + 32) / 100.0; };

}

}

if (f2 == null)

{

if (ds.f2 != null)

{

f2 = ds.f2;

}

else

{

f2 = delegate (double x) { return ((1.0 / (Math.Max(0,x \* 100.0 - 70))) + 70) / 100.0; };

}

}

int maxvx = ds.points[0].X; // ищем максимальное значение по x и y, чтобы понять, какую область надо выделять под график

int maxvy = ds.points[0].Y;

int maxvr = (int)ds.points[0].r; // ищем максимальный радиус, чтобы относительно него вычислить размер области, на которой рисуется график

foreach (var i in ds.points)

{

maxvx = Math.Max(maxvx, i.X);

maxvy = Math.Max(maxvy, i.Y);

maxvr = Math.Max(maxvr, (int)i.r);

}

int wigth = (int)(Math.Max((maxvx + 3 \* maxvr), 50) \* scale + space \* 3); //определяем ширину бимпама

int heigth = (int)(Math.Max((maxvy + 3 \* maxvr), 50) \* scale + space \* 3 + space); // определяем высоту бимпама

bmp = new Bitmap(wigth, heigth);

graph = Graphics.FromImage(bmp);

DrawAxis(xs: "Вероятность", ys: "Влияние", xe: 1, ye: 1);// xe: 1, ye: 1 - для отображения 0 и 1 на осях графиков

// отрисовка точек

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

{

Color c = Color.Green; // аналогично опредеелнию цвета для графика №1

if (ds.points[i].y >= f1(ds.points[i].x) && ds.points[i].y < f2(ds.points[i].x)) c = Color.Yellow;

if (ds.points[i].y >= f2(ds.points[i].x)) c = Color.Red;

string name = ds.points[i].ToString(3,2); // 2 - номер графика (нумерация здесь с 1)

mas\_point.Add(DrawPoint((int)(ds.points[i].X \* scale + space \* 2), (int)(bmp.Height - space \* 2 - (int)(ds.points[i].Y \* scale)), (int)(ds.points[i].r), c, name));

}

}

/// <summary>

/// Функция для отрисовки 3-го графика (карта соотношений)

/// </summary>

/// <param name="ds"></param>

public void DrawBigPoints(DataSender ds)

{

mas\_point = new List<ColorPoint>();

int maxvx = ds.points[0].X; // ищем максимальное значение по x и y, чтобы понять, какую область надо выделять под график

int maxvy = ds.points[0].Y;

int maxvr = (int)ds.points[0].r; // ищем максимальный радиус, чтобы относительно него вычислить размер области, на которой рисуется график

foreach (var i in ds.points)

{

maxvx = Math.Max(maxvx, i.X);

maxvy = Math.Max(maxvy, i.Y);

maxvr = Math.Max(maxvr, (int)i.r);

}

int wigth = (int)(Math.Max((maxvx + 3 \* maxvr), 50) \* scale + space \* 2); //определяем ширину бимпама

int heigth = (int)(Math.Max((maxvy + 3 \* maxvr), 50) \* scale + space \* 2 + space); // определяем высоту бимпама

bmp = new Bitmap(wigth, heigth);

graph = Graphics.FromImage(bmp);

DrawAxis(xs: "Качество системы", ys: "Средневзвешенный риск системы", xe: 1, ye: 1);

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

{

string name = ds.points[i].ToString(3,3); //3 - номер графика (нумерация здесь с 1)

mas\_point.Add(DrawPoint((int)(ds.points[i].X \* scale + space), (int)(bmp.Height - space - (int)(ds.points[i].Y \* scale)), (int)(Math.Max(1,ds.points[i].r \* scale)), ds.points[i].c, name));

}

}

}

}

**1.5. Класс DataSender**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

/// <summary>

/// Класс для передачи данных для рисования графика

/// </summary>

public class DataSender

{

public int n = 0; // Номер функции для рисования графика

// для n==0

public List<List<double>> histogram\_h = new List<List<double>>(); // высоты столбцов для гистограммы

public List<List<string>> histogram\_name = new List<List<string>>(); // имена для гистограммы

public List<string> histogram\_attr\_name = new List<string>(); // имена для груп гистограмм для одного атрибута гистограммы

// для n==1 || n==2

public List<ColorPoint> points = new List<ColorPoint>(); // второй и третий график

public delegate double Func(double x); // делегат для подачи функций разграничения цветов точек на графике

public Func f1 = null; // функция для карты рисков - граница зеленого и желтого

public Func f2 = null; // функция для карты рисков - граница желтого и красного

}

}

**1.6. Класс ColorPoint**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

/// <summary>

/// класс для цветной точки на графиках: "карта рисков" и "карта соотношений"

/// </summary>

public class ColorPoint

{

public double x = 0, y = 0; // Координаты точки

public int X { get { return (int)(x \* scalecoeff); } } // Данные координаты использует функция рисования при отображении точки

public int Y { get { return (int)(y \* scalecoeff); } }

public double r { get { return Math.Max(0, rv \* coeff); } } // Радиус видимой точки - радиус нарисованной точки, а не исходный радиус

public double rv = 3; // исходный подаваемый радиус точки

public string name = "";

public Color c = Color.Black;

public double scalecoeff = 1;

public double coeff = 1;

public ColorPoint() { }

public ColorPoint(double x, double y, double r, string name, double scalecoeff = 1, double coeff = 1)

{

this.x = x; // координаты точки

this.y = y;// координаты точки

this.rv = r; // радиус при выделении

this.name = name; // имя при наведении

this.scalecoeff = scalecoeff;

this.coeff = coeff;

}

public ColorPoint(double x, double y, double r, string name, Color c, double scalecoeff = 1, double coeff = 1)

{

this.x = x; // координаты точки

this.y = y;// координаты точки

this.rv = r; // радиус при выделении

this.name = name; // имя при наведении

this.c = c; // цвет

this.scalecoeff = scalecoeff;

this.coeff = coeff;

}

// приведение типов

public static implicit operator Point(ColorPoint p)

{

return new Point((int)(p.x), (int)(p.y));

}

public static implicit operator ColorPoint(Point p)

{

return new ColorPoint(p.X, p.Y, 3, "", Color.Black);

}

//Оператор получения длины вектора по двум точкам с помощью знака +

public static double operator +(ColorPoint a, ColorPoint b)

{

if (a == null || b == null) throw new NullReferenceException();

return Math.Sqrt(((a.X - b.X) \* (a.X - b.X)) + ((a.Y - b.Y) \* (a.Y - b.Y)));

}

//переопределенный метод ToString()

public override string ToString()

{

return ToString(3,3);

}

//Информация об объекте

public string ToString(int prec, int num = 2)

{

if (num == 2)

{

int p = 1;

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

{

p \*= 10;

}

return $"{name}\r\nВероятность атрибута {getstr(x, p)}\r\nВлияние атрибута {getstr(y, p)}\r\n";

}

if (num == 3)

{

int p = 1;

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

{

p \*= 10;

}

return $"{name}\r\nКачество {getstr(x, p)}\r\nСредневзвешенный риск системы {getstr(y, p)}\r\nCтоимость {getstr(rv, p)} тыс $";

}

return "";

}

/// <summary>

/// Вывод не больше 3 знаков после запятой

/// </summary>

/// <param name="x"></param>

/// <param name="p"></param>

/// <returns></returns>

public string getstr(double x, int p)

{

//string rs = "" + (int)(x);

//if (((int)(x \* p) % p) > 0)

//{

// rs += ",";

// string rs2 = "";

// int p2 = p;

// bool flaga = false;

// while (p2 / 10 > 0)

// {

// int n = (((int)(x \* p2) % p2)) % 10;

// if (flaga || n != 0)

// {

// flaga = true;

// rs2 += $"{ n}";

// }

// p2 /= 10;

// }

// rs += new string((rs2.Reverse()).ToArray<char>());

//}

//return rs;

string x\_s = String.Format("{0:0.###}", x);

return x\_s;

}

}

}

RU.17701729.507900-01 81 01-1

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов | № документа | Входящий № | Подпись | Дата |
|  | измененных | замененных | новых | аннулирован | (страниц) в |  | сопроводительного |  |  |
|  |  |  |  | ных | документе |  | документа и дата |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |