НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина: «Программирование»

Контрольное домашнее задание

Вариант 29

Выполнил: Рябичев Алексей,

Студент гр. БПИ176.

Преподаватель: Дударев В.А.

Дата: « »\_\_\_\_\_\_\_\_\_\_ 2017г.

Подпись выполнившего: .

Москва 2017

**Оглавление**

**Условие задачи2**

**Функции разрабатываемого приложения2**

Варианты использования2

Описание интерфейса пользователя2

**Структура приложения3**

Диаграмма классов3

Описание классов, их полей и методов3

**Распределение исходного кода по файлам проекта5**

**Контрольный пример и описание результатов5**

**Текст (код) программы (только части, написанные вручную)7**

**Список литературы20**

**Условие задачи**

Вариант 29.

Разработать оконное приложение (Windows Forms Application), которое будет представлять аналитический инструмент. Входными данными для программы должна служить таблица формата csv или excel (на выбор разработчика) с следующими столбцами: CT, BPAD, BPAN, BPAN2, BWAD, BWAN, BWAN2, BMAD, BMAN, BMAN2.

1. Основное окно программы должно поддерживать функции загрузки таблиц .csv формата и отображение значений этих таблиц (для отображения использовать DataGridVew).
2. Возможность отображения зависимости между парами столбцов данных.
3. Каждый график создавать в отдельном окне, которое не блокирует основное.
4. Возможность открытия нескольких окон с графиками.
5. Возможность изменения границ графика при помощи TrackBar.
6. Обработка пропущенных значений

**Функции разрабатываемого приложения**

Варианты использования

Приложение используется для анализа таблиц .csv формата.

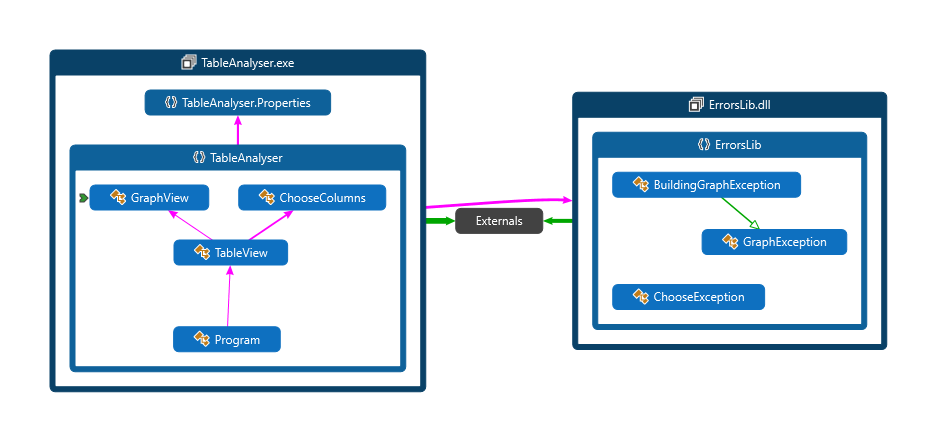
Описание интерфейса пользователя

Приложение позволяет открыть файл формата .csv и просмотреть его содержимое.

Приложение позволяет проанализировать (построить график) зависимость между парами столбцов и экспортировать полученные данные в .png формате.

**Структура приложения**

Диаграмма классов

Описание классов, их полей и методов

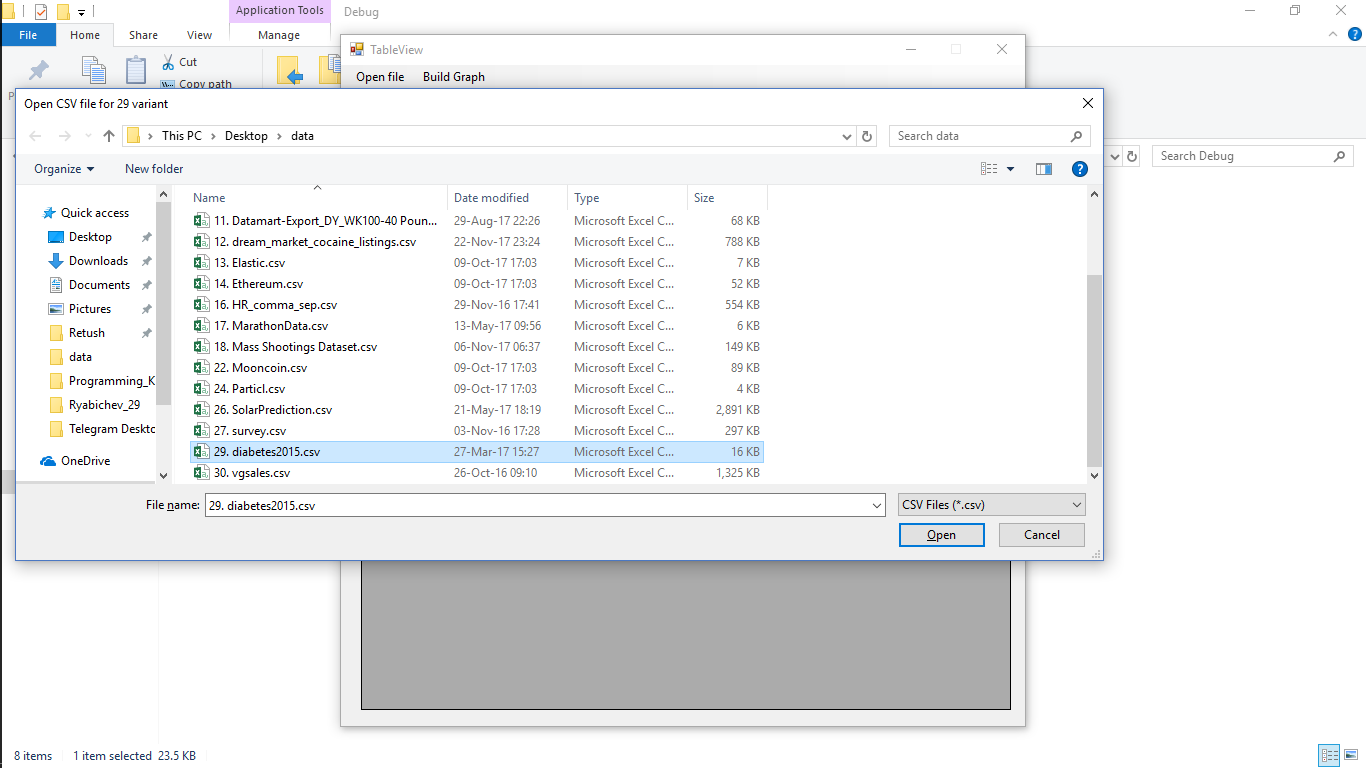
1. Класс Program – главный класс программы. Вызывает TableView
2. Класс TableView – класс, описывающий форму отображения таблицы
   1. \_dataTable – таблица данных
   2. \_dataGridView – элемент отображения таблицы
   3. toolTip – элемент отображения всплывающей подсказки
   4. \_fileReady – переменная, обозначающая открыт ли уже файл
   5. \_openFileDialog – диалог для выбора файла
   6. menuStrip – меню
   7. buildGraphToolStripMenuItem – элемент меню
   8. openFileToolStripMenuItem – элемент меню
   9. ShowTable – отображение таблицы
   10. \_dataGridView\_MouseHover – отображение подсказки для таблицы
   11. SmartSplit – деление строки по запятой и возвращение object[]
   12. BuildGraphToolStripMenuItem\_Click – вызывает BuildGraph
   13. OpenFileToolStripMenuItem\_Click – вызывает OpenFile
   14. OpenFileToolStripMenuItem\_MouseHover – отображение подсказки для меню
   15. OpenFile – дает возможность выбрать файл
   16. FillTable – заолняет таблицу данными из файла
   17. TableView – конструктор данного класса
   18. TableView\_Load – лоадер данного класса
3. Класс ChooseColumns – класс, описывающий форму выбора столбцов для анализа
   1. FirstColumn – название первого столбца
   2. SecondColumn – навзание второго столбца
   3. toolTip - элемент отображения всплывающей подсказки
   4. button – кнопка «ОК»
   5. checkedListBox – элемент отображения списка названий столбцов
   6. \_tableHeaders - список названий столбцов
   7. \_dataTable – таблица данных
   8. Button\_MouseHover – отображение подсказки для кнопки
   9. CheckedLisBox\_MouseHover - отображение подсказки для списка
   10. Button\_Click – обработка нажатия кнопки «ОК»
   11. ChooseColumns – конструктор данного класса
   12. ChooseColumns\_Load – лоадер данного класса
4. Класс GraphView – класс, описывающий форму отображения графика
   1. changeColorToolStripMenuItem – элемент меню для смены цвета графика
   2. labelMaximumValue – текстовый элемент
   3. labelMinimumValue – тексотвый элемент
   4. exportToolStripMenuItem – элемент меню для экспорта графика
   5. menuStrip - меню
   6. \_yXDictionary – словарь координат первого столбца от второго
   7. \_xYDicitionary - словарь координат второго столбца от первого
   8. \_dataTable - таблица
   9. toolTip – элемент отображения всплывающей подсказки
   10. chart - график
   11. \_xY – список значений координат второго столбца от первого
   12. \_yX - список значений координат первого столбца от второго
   13. \_flag – переменная обозначающая какая зависимость сейчас отображается
   14. \_firstColumn – название первого столбца
   15. \_secondColumn – название второго столбца
   16. trackBarMinimum – трэкбар для минимального значения
   17. trackBarMaximum – ирэкбар для максимального значения
   18. exportToolStripMenuItem\_MouseHover – отображение подсказки для меню
   19. trackBarMinimum\_MouseHover – отображение подсказки для трэкбара
   20. trackBarMaximum\_MouseHover – отображение подсказки для трэкбара
   21. Chart\_MouseHover – отображение подсказки для графика
   22. trackBarMaximum\_Scroll – изменение значения максимального значения
   23. trackBarMinimum\_Scroll – изменение значения для минимального значения
   24. exportToolStripMenuItem\_Click – экспорт графика
   25. BuildChart – отображение графика
   26. SortDicitonaries – сортировка координат по возрастанию
   27. Chart\_Click – смена осей x и y
   28. ParseTable – парсинг таблицы
   29. SetTrackBarsMinMax – настройка минимальных и максимальных значений трэкбаров
   30. GraphView – конструктор данного класса
5. ErrorsLib – библиотека классов пользовательских ошибок (Exception)
   1. ChooseException - наследник класса Exception
      1. ChooseException – конструктор от одной переменной
   2. GraphException - наследник класса Exception
      1. GraphException – конструктор от двух переменных
      2. GraphException – конструктор от одной переменной
   3. BuildingGraphException - наследник класса GraphException
      1. BuildingGraphException – конструктор от одной переменной

**Распределение исходного кода программы по файлам проекта**

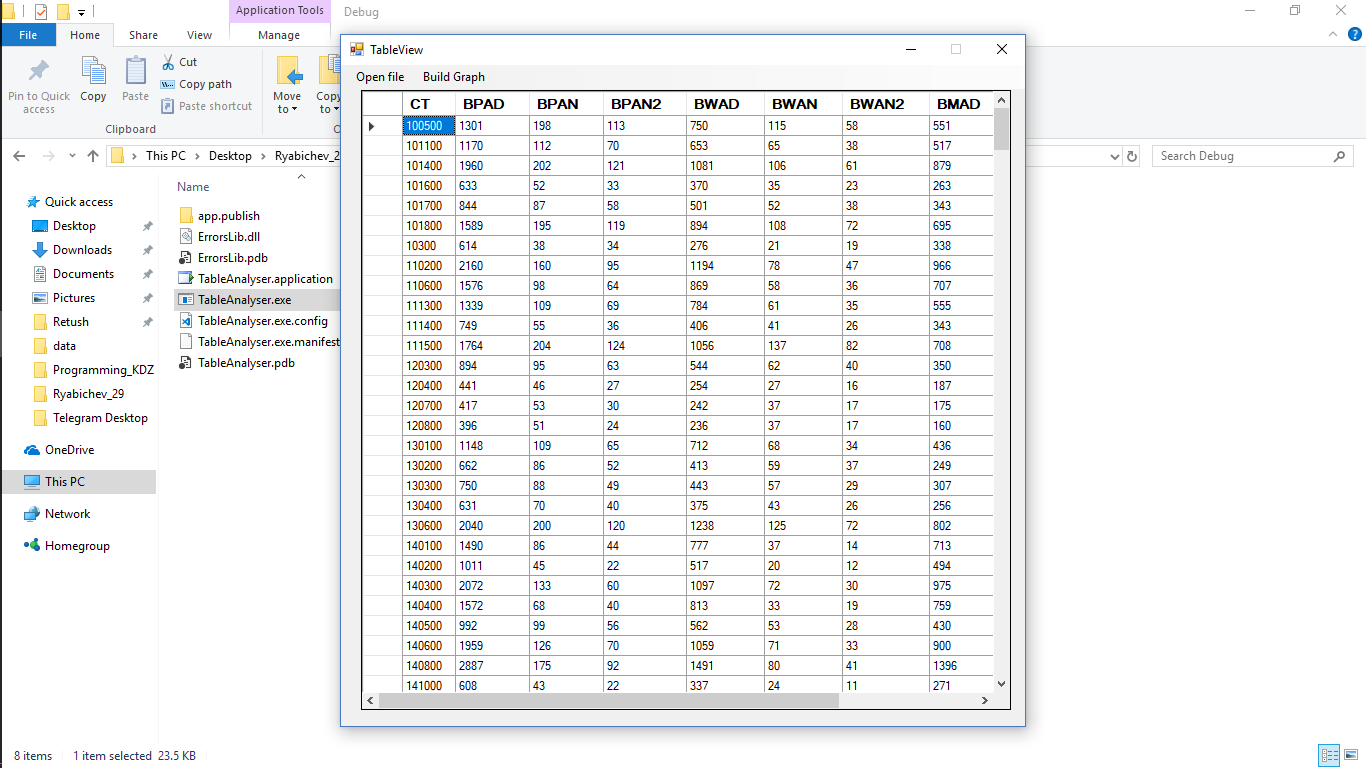
Классы ChooseException, GraphException, BuildingGraphException находятся в файлах ChooseException.cs, GraphException.cs, BuildingGraphException.cs соответственно и в библиотеке файлов ErrorsLib, которая находится в файле ErrorsLib.dll

Классы GraphView, ChooseColumns, TableView, Program находятся в файлах GraphView.cs, ChooseColumns.cs, TableView.cs, Program.cs соответственно и в проекте TableAnalyser, который находится в файле TableAnalyser.exe

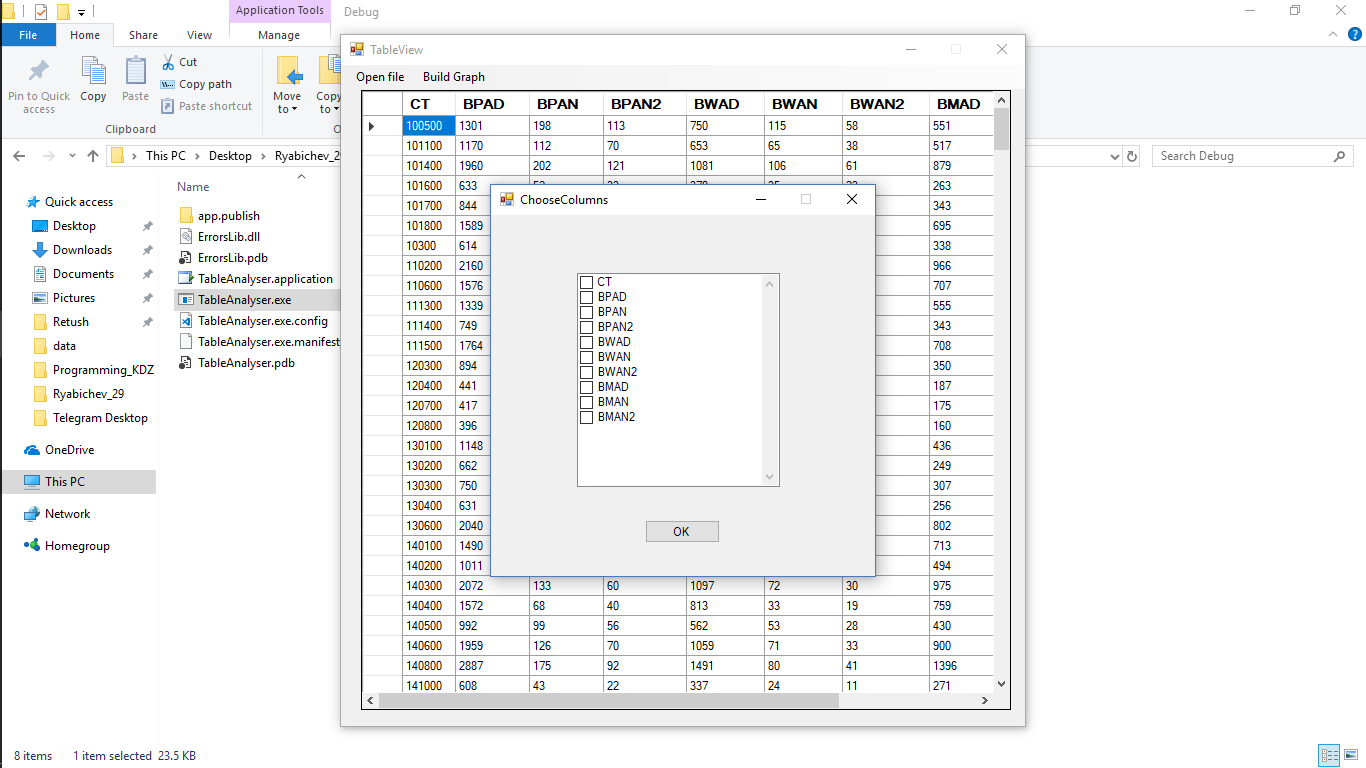
**Контрольный пример и описание результатов**



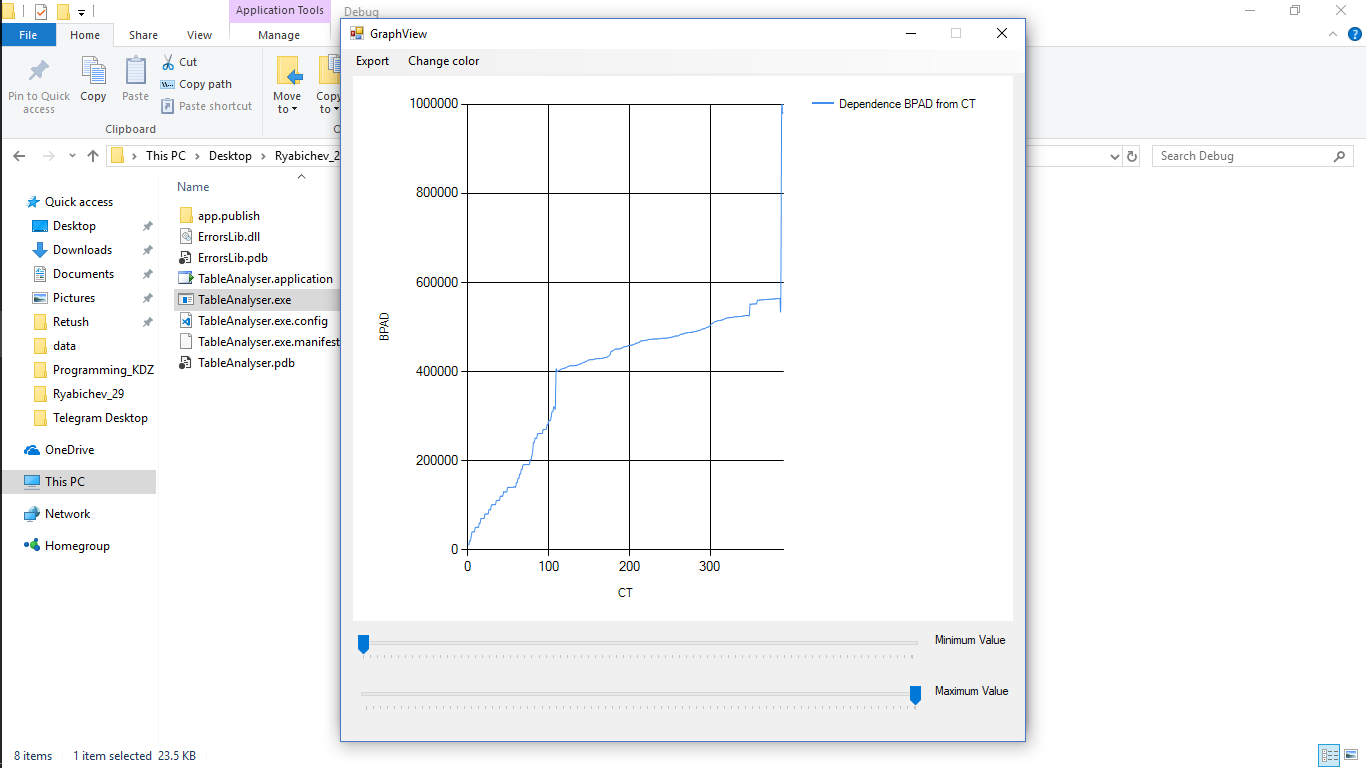
Выбор файла



Отображение таблицы



Выбор столбцов



Отображение графика

**Текст (код) программы (только части, написанные вручную)**

**TableView.cs**

using System;

using System.Data;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

using ErrorsLib;

using TableAnalyser.Properties;

namespace TableAnalyser

{

public sealed partial class TableView : Form

{

private readonly OpenFileDialog \_openFileDialog = new OpenFileDialog

{

Filter = Resources.\_openFileDialog\_Filter,

Title = Resources.\_openFileDialog\_Title

};

private DataTable \_dataTable;

private bool \_fileReady;

/// <summary>

/// Create form of TableView

/// </summary>

public TableView()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

FormBorderStyle = FormBorderStyle.FixedSingle;

Font = new Font("Bandera", 8);

Load += TableView\_Load;

}

/// <summary>

/// Loader for TableView

/// </summary>

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

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

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

{

Show();

OpenFile();

}

/// <summary>

/// Open file

/// </summary>

private void OpenFile()

{

bool flag = false;

while (!flag)

try

{

DialogResult dialogResult = \_openFileDialog.ShowDialog();

if (dialogResult != DialogResult.OK && !\_fileReady)

throw new Exception("Please, choose file firstly");

Cursor.Current = Cursors.WaitCursor;

FillTable();

Cursor.Current = Cursors.Default;

Show();

ShowTable();

flag = true;

\_fileReady = true;

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

}

}

/// <summary>

/// Fill table of elements

/// </summary>

private void FillTable()

{

\_dataTable = new DataTable();

string[] tmp = File.ReadAllLines(\_openFileDialog.FileName);

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

if (i == 0)

foreach (string s in tmp[i].Split(','))

\_dataTable.Columns.Add(s);

else

\_dataTable.Rows.Add(\_dataTable.NewRow().ItemArray = SmartSplit(tmp[i]));

}

/// <summary>

/// Show table of elements

/// </summary>

private void ShowTable()

{

\_dataGridView.DataSource = \_dataTable;

\_dataGridView.ColumnHeadersDefaultCellStyle.Font =

new Font(FontFamily.GenericSansSerif, 10, FontStyle.Bold);

\_dataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

\_dataGridView.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

\_dataGridView.ScrollBars = ScrollBars.Both;

\_dataGridView.Anchor = AnchorStyles.Top;

}

/// <summary>

/// Clicker for open file menu

/// </summary>

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

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

private void OpenFileToolStripMenuItem\_Click(object sender, EventArgs e) => OpenFile();

/// <summary>

/// Clicker for build graph menu

/// </summary>

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

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

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

{

ChooseColumns chooseColumns = new ChooseColumns(\_dataTable);

try

{

DialogResult dialogResult = chooseColumns.ShowDialog();

if (dialogResult == DialogResult.OK)

{

Cursor.Current = Cursors.WaitCursor;

GraphView graphView =

new GraphView(\_dataTable, chooseColumns.FirstColumn, chooseColumns.SecondColumn);

graphView.Show();

}

else if (dialogResult != DialogResult.Abort && dialogResult != DialogResult.Cancel)

{

throw new ChooseException();

}

}

catch (ChooseException exception)

{

MessageBox.Show(Resources.ChooseExceptionMessage + exception.Message);

BuildGraphToolStripMenuItem\_Click(sender, e);

}

catch (GraphException exception)

{

if (exception.InnerException != null) MessageBox.Show(exception.InnerException.Message);

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

}

}

/// <summary>

/// Split string by coma

/// </summary>

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

/// <returns>object[]</returns>

private object[] SmartSplit(string s)

{

if (s == null) throw new ArgumentNullException(nameof(s));

object[] objects = new object[\_dataTable.Columns.Count];

for (int j = 0; j < s.Split(',').Length; j++)

objects[j] = string.IsNullOrEmpty(s.Split(',')[j]) ? "0" : s.Split(',')[j];

return objects;

}

/// <summary>

/// Listener for mouse hover of table

/// </summary>

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

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

private void \_dataGridView\_MouseHover(object sender, EventArgs e) => toolTip.SetToolTip(\_dataGridView, "Table of elements from file");

/// <summary>

/// Listener for mouse hover of menu

/// </summary>

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

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

private void OpenFileToolStripMenuItem\_MouseHover(object sender, EventArgs e) => toolTip.SetToolTip(menuStrip, "Menu");

}

}

**GraphView.cs**

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using ErrorsLib;

using TableAnalyser.Properties;

using Cursor = System.Windows.Forms.Cursor;

namespace TableAnalyser

{

public partial class GraphView : Form

{

private readonly DataTable \_dataTable;

private readonly string \_firstColumn;

private readonly string \_secondColumn;

private readonly Dictionary<int, int> \_xYDictionary = new Dictionary<int, int>();

private readonly Dictionary<int, int> \_yXDictionary = new Dictionary<int, int>();

private bool \_flag;

private List<KeyValuePair<int, int>> \_xY = new List<KeyValuePair<int, int>>();

private List<KeyValuePair<int, int>> \_yX = new List<KeyValuePair<int, int>>();

/// <summary>

/// Creates GraphView Form

/// </summary>

/// <param name="dataTable">Table of elements</param>

/// <param name="firstColumn">Name of first column</param>

/// <param name="secondColumn">Name of second column</param>

public GraphView(DataTable dataTable, string firstColumn, string secondColumn)

{

InitializeComponent();

\_dataTable = dataTable;

\_firstColumn = firstColumn;

\_secondColumn = secondColumn;

FormBorderStyle = FormBorderStyle.FixedSingle;

\_flag = false;

ParseTable();

SortDictionaries();

Cursor.Current = Cursors.Default;

BuildChart(\_xY, \_firstColumn, \_secondColumn);

MessageBox.Show(Resources.GraphViewChartClickHint);

}

/// <summary>

/// Method for revert chart

/// </summary>

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

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

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

{

\_flag = !\_flag;

if (\_flag)

BuildChart(\_xY, \_firstColumn, \_secondColumn);

else

BuildChart(\_yX, \_secondColumn, \_firstColumn);

}

/// <summary>

/// Build chart

/// </summary>

/// <param name="xy">Dicitonary of coordinates</param>

/// <param name="ox">Name for Ox</param>

/// <param name="oy">Name for Oy</param>

private void BuildChart(List<KeyValuePair<int, int>> xy, string ox, string oy)

{

chart.Series[0].Points.Clear();

chart.Series[0].Name = "Graph";

chart.Series[0].LegendText = "Dependence " + oy + " from " + ox;

chart.ChartAreas[0].AxisX.Title = ox;

chart.ChartAreas[0].AxisY.Title = oy;

chart.Series[0].ChartType = SeriesChartType.Spline;

foreach (KeyValuePair<int, int> pair in xy)

chart.Series[0].Points.Add(pair.Key, pair.Value);

trackBarMinimum.Minimum = (int) chart.ChartAreas[0].AxisX.Minimum;

trackBarMinimum.Maximum = (int) chart.ChartAreas[0].AxisX.Maximum;

trackBarMaximum.Minimum = (int) chart.ChartAreas[0].AxisX.Minimum;

trackBarMaximum.Maximum = (int) chart.ChartAreas[0].AxisX.Maximum;

trackBarMaximum.TickFrequency = 5;

trackBarMinimum.TickFrequency = 5;

trackBarMaximum.Value = (int) chart.ChartAreas[0].AxisX.Maximum;

trackBarMinimum.Value = (int) chart.ChartAreas[0].AxisX.Minimum;

}

private void SetTrackbarsMinMax()

{

trackBarMaximum.Minimum = trackBarMinimum.Value;

trackBarMinimum.Maximum = trackBarMaximum.Value;

}

/// <summary>

/// Parsing table of elements

/// </summary>

private void ParseTable()

{

foreach (DataRow row in \_dataTable.Rows)

try

{

if (\_xYDictionary.ContainsKey(

int.Parse(row[\_firstColumn] as string ?? throw new BuildingGraphException())))

\_xYDictionary[int.Parse(row[\_firstColumn] as string)] =

(\_xYDictionary[int.Parse(row[\_firstColumn] as string)] + (row[\_secondColumn] == null

? 0

: int.Parse(row[\_secondColumn] as string ?? throw new BuildingGraphException()))) / 2;

else

\_xYDictionary.Add(

row[\_firstColumn] == null

? 0

: int.Parse(row[\_firstColumn] as string ?? throw new BuildingGraphException()),

row[\_secondColumn] == null

? 0

: int.Parse(row[\_secondColumn] as string ?? throw new BuildingGraphException()));

if (\_yXDictionary.ContainsKey(

int.Parse(row[\_secondColumn] as string ?? throw new BuildingGraphException())))

\_yXDictionary[int.Parse(row[\_secondColumn] as string)] =

(\_yXDictionary[int.Parse(row[\_secondColumn] as string)] + (row[\_firstColumn] == null

? 0

: int.Parse(row[\_firstColumn] as string ?? throw new BuildingGraphException()))) / 2;

else

\_yXDictionary.Add(

row[\_secondColumn] == null

? 0

: int.Parse(row[\_secondColumn] as string ?? throw new BuildingGraphException()),

row[\_firstColumn] == null

? 0

: int.Parse(row[\_firstColumn] as string ?? throw new BuildingGraphException()));

}

catch (Exception exception)

{

throw new GraphException(exception);

}

}

/// <summary>

/// Listener for maximum trackbar scrolling

/// </summary>

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

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

private void TrackBarMaximum\_Scroll(object sender, EventArgs e)

{

SetTrackbarsMinMax();

chart.ChartAreas[0].AxisX.Maximum = trackBarMaximum.Value;

}

/// <summary>

/// Listener for minimum trackbar scrolling

/// </summary>

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

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

private void TrackBarMinimum\_Scroll(object sender, EventArgs e)

{

SetTrackbarsMinMax();

chart.ChartAreas[0].AxisX.Minimum = trackBarMinimum.Value;

}

/// <summary>

/// Cast dicitonaries to lists and sort them

/// </summary>

private void SortDictionaries()

{

\_xY = \_xYDictionary.ToList();

\_xY.Sort((pair1, pair2) => pair1.Key.CompareTo(pair2.Key));

\_yX = \_yXDictionary.ToList();

\_yX.Sort((pair1, pair2) => pair1.Key.CompareTo(pair2.Key));

}

/// <summary>

/// Change color of chart

/// </summary>

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

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

public void ChangeColorToolStripMenuItem\_Click(object sender, EventArgs e)

{

ColorDialog colorDialog = new ColorDialog();

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

chart.Series[0].Color = colorDialog.Color;

}

/// <summary>

/// Export chart as png

/// </summary>

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

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

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

{

FolderBrowserDialog folderBrowserDialog = new FolderBrowserDialog();

if (folderBrowserDialog.ShowDialog() != DialogResult.OK) return;

if (\_flag)

chart.SaveImage(

folderBrowserDialog.SelectedPath + @"\" + "Dependence" + \_secondColumn + "from" + \_firstColumn +

".png", ChartImageFormat.Png);

else

chart.SaveImage(

folderBrowserDialog.SelectedPath + @"\" + "Dependence" + \_firstColumn + "from" + \_secondColumn+

".png", ChartImageFormat.Png);

MessageBox.Show(Resources.FileSave + folderBrowserDialog.SelectedPath);

}

/// <summary>

/// Listener for mouse hover of chart

/// </summary>

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

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

private void Chart\_MouseHover(object sender, EventArgs e) => toolTip.SetToolTip(chart, "Chart");

/// <summary>

/// Listener for mouse hover of menu

/// </summary>

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

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

private void ExportToolStripMenuItem\_MouseHover(object sender, EventArgs e) => toolTip.SetToolTip(menuStrip, "Menu");

/// <summary>

/// Listener for mouse hover of maximum trackbar

/// </summary>

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

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

private void TrackBarMaximum\_MouseHover(object sender, EventArgs e) => toolTip.SetToolTip(trackBarMaximum, "Maximum trackbar");

/// <summary>

/// Listener for mouse hover of minimum tracbar

/// </summary>

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

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

private void TrackBarMinimum\_MouseHover(object sender, EventArgs e) => toolTip.SetToolTip(trackBarMinimum, "Minimum trackbar");

}

}

**ChooseColumns.cs**

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using ErrorsLib;

namespace TableAnalyser

{

public partial class ChooseColumns : Form

{

private readonly DataTable \_dataTable;

private string[] \_tableHeaders;

/// <summary>

/// Create form of ChooseColumn

/// </summary>

/// <param name="dataTable"></param>

public ChooseColumns(DataTable dataTable)

{

InitializeComponent();

Size = new Size(400, 400);

StartPosition = FormStartPosition.CenterScreen;

FormBorderStyle = FormBorderStyle.FixedSingle;

\_dataTable = dataTable;

Load += ChooseColumns\_Load;

}

public string FirstColumn { get; private set; }

public string SecondColumn { get; private set; }

/// <summary>

/// Loader for ChooseColumn

/// </summary>

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

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

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

{

\_tableHeaders = new string[\_dataTable.Columns.Count];

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

{

\_tableHeaders[i] = \_dataTable.Columns[i].ColumnName;

checkedListBox.Items.Add(\_tableHeaders[i]);

}

}

/// <summary>

/// Clicker for button

/// </summary>

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

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

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

{

if (checkedListBox.CheckedIndices.Count != 2)

throw new ChooseException("Choose only two columns");

FirstColumn = \_tableHeaders[checkedListBox.CheckedIndices[0]];

SecondColumn = \_tableHeaders[checkedListBox.CheckedIndices[1]];

DialogResult = DialogResult.OK;

Close();

}

/// <summary>

/// Listener for mouse hover of listbox

/// </summary>

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

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

private void CheckedListBox\_MouseHover(object sender, EventArgs e) => toolTip.SetToolTip(checkedListBox, "List of column names");

/// <summary>

/// Listener for mouse hover of button

/// </summary>

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

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

private void Button\_MouseHover(object sender, EventArgs e) => toolTip.SetToolTip(button, "OK button");

}

}

**BuildingGraphException.cs**

using System;

namespace ErrorsLib

{

public class BuildingGraphException : GraphException

{

public BuildingGraphException(string message = "Some of elements are null") : base(message) { }

}

}

**GraphException.cs**

using System;

namespace ErrorsLib

{

public class GraphException : Exception

{

public GraphException(Exception exception) : base(null, exception) { }

public GraphException(string message) : base(message) { }

}

}

**ChooseException.cs**

using System;

namespace ErrorsLib

{

public class ChooseException : Exception

{

public ChooseException(string message = "Choose columns firstly") : base(message) { }

}

}

**Список литературы**

1. MSDN Library: <https://msdn.microsoft.com/en-us/library/ms310241>