**Национальный исследовательский университет «Высшая школа экономики»**

**Факультет компьютерных наук**

Департамент

**Программной инженерии**

***Контрольное домашнее задание по дисциплине***

***«Программирование»***

Тема работы: Работа с файлами

Выполнил(а): студент группы 183 (2) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Имамов Р.М. тел.89877528126

e-mail адрес: rmimamov@edu.hse.ru

Преподаватель: Чуйкин.Н.К.

Москва, 2019 год. Модуль 3

**Оглавление**

1. **УСЛОВИЕ ЗАДАЧИ 3**
2. **ФУНКЦИИ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ 4**
3. **СТРУКТУРА ПРИЛОЖЕНИЯ 6**
4. **РАСПРЕДЕЛЕНИЕ ИСХОДНОГО КОДА ПО ФАЙЛАМ ПРОЕКТА 9**
5. **КОНТРОЛЬНЫЙ ПРИМЕР И ОПИСАНИЕ РЕЗУЛЬТАТОВ 10**
6. **СООБЩЕНИЯ ПОЛЬЗОВАТЕЛЮ 11**
7. **СПИСОК ЛИТЕРАТУРЫ 14**
8. **ТЕКСТ (КОД) ПРОГРАММЫ 15**

# Условие задачи

Вариант 6

Программа Контрольного домашнего задания представляет собой **WindowsForms** приложение и предназначена для просмотра, обработки и сохранения результатов обработки данных из файла.

**Кинотеатры версия 2.3 от 14.10.2014.csv**

1. Требования к основным классам приложения
   1. Основная информация о Кинотеатрах г. Москвы хранится в объектах класса **Кинотеатр**. Набор полей класса задаётся полями файла **Кинотеатры версия 2.3 от 14.10.2014.csv**, кроме полей, содержащих информацию об административном округе, представленны полем типа **Округ**. Класс **Кинотеатр** находится в *отношении агрегации* с классом **Округ**.
   2. Класс **Округ** представляет округа города Москвы и содержит поля, заданные полями CSV-файла: **AdmArea**, **District** и метод, возвращающий количество районов в данном округе.
   3. Дополнительные классы, необходимые для решения задачи.
2. Приложение должно поддерживать следующие функции
   1. Открыть CSV-файл (**\*.csv**) с исходными данными и проверить корректность данных в нём.
   2. Загрузить данные из CSV-файла в объекты классов **Кинотеатр**, **Округ** (если объект **Округ** с данными об определённом округе существует, то он является общим для всех объектов **Кинотеатр** этого округа) и др.
   3. Отобразить данные из объектов в оконной форме.
   4. Создать новую запись о Кинотеатре.
   5. Удалить уже существующую запись о Кинотеатре
   6. Отредактировать существующую запись о Кинотеатре
   7. Отсортировать данные по полям: **FullName**, **District**
   8. Отсортировать данные по количеству районов в округе.
   9. Отфильтровать данные по полям: **AdmArea**, **District.** Данные для фильтрации вводятся пользователем.
   10. Сохранять результаты редактирований, сортировок и фильтраций в CSVфайл. *Режимы* *сохранения в файл*: создание нового файла, замена содержимого уже существующего файла, добавление сохраняемых данных к содержимому существующего файла.
3. Требования к интерфейсу
   1. При управлении файлом (загрузка, сохранение) использовать **OpenFileDialog** и **SaveFileDialog**.
   2. Для отображения данных использовать сетку **DataGridView**
   3. Количество отображаемых в сетке элементов (***N***) выбирается пользователем, ***N* > 1** и не превышает количества записей в файле **Кинотеатры версия 2.3 от**

**14.10.2014.csv**

1. Требования к устойчивости приложения

4.1. В случае ошибок открывания/сохранения файла или некорректных данных программа должна выводить сообщение.

4.2. Аварийные ситуации должны обрабатываться, пользователю должны выводиться информативные сообщения.

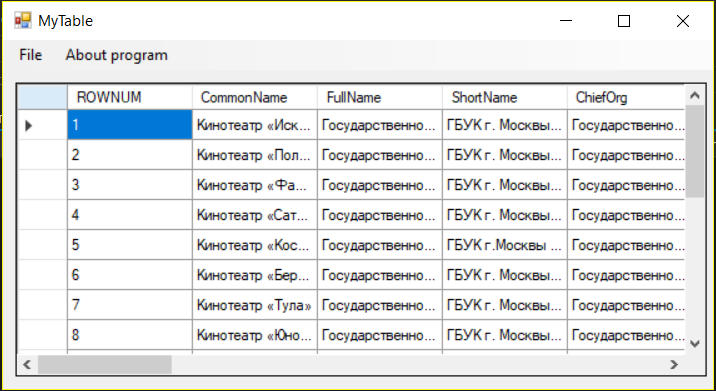
# Функции разрабатываемого приложения

* 1. Варианты использования. Данное приложение может использоваться

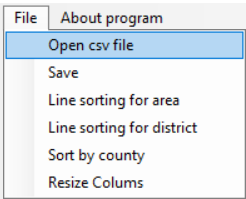
Для офисной работы для структурирования какой либо информации. Также оно способно открывать .csv файлы, так что оно способно интегрироваться уже в рабочую систему.

2.2. Описание интерфейса пользователя:

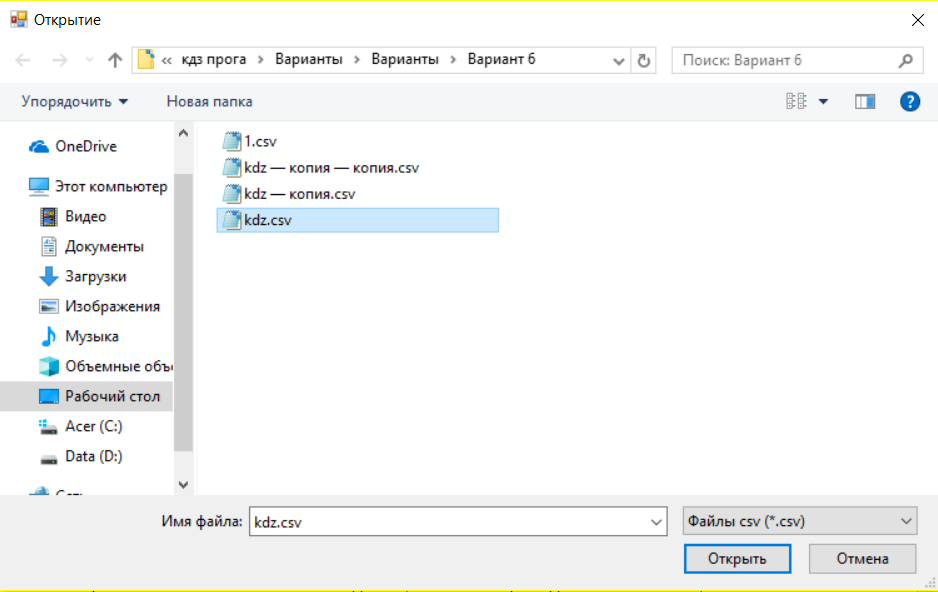
2.3. Основное окно программы



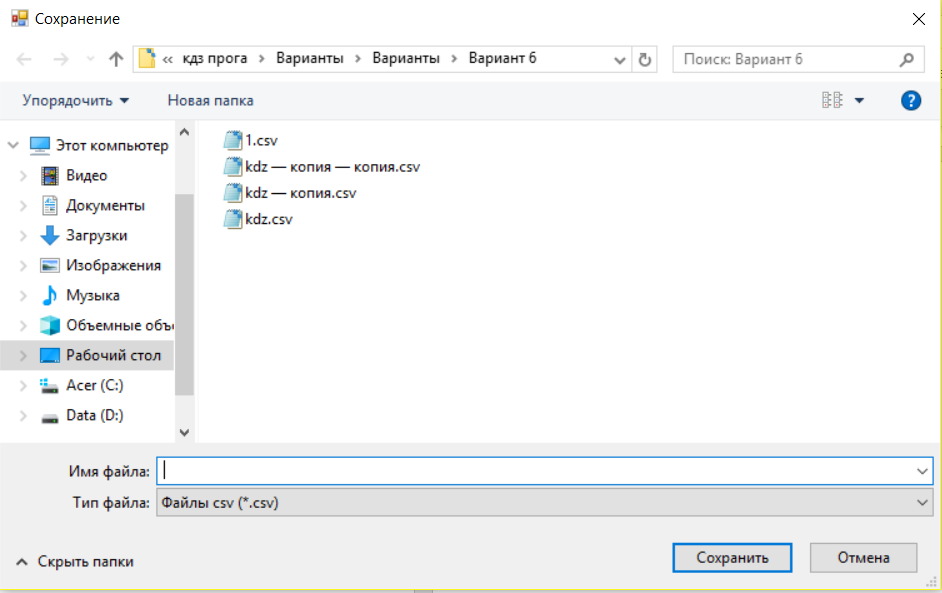
2.4. Выплывающий список для работы с программой:



2.5. Окно для открытия .csv файла

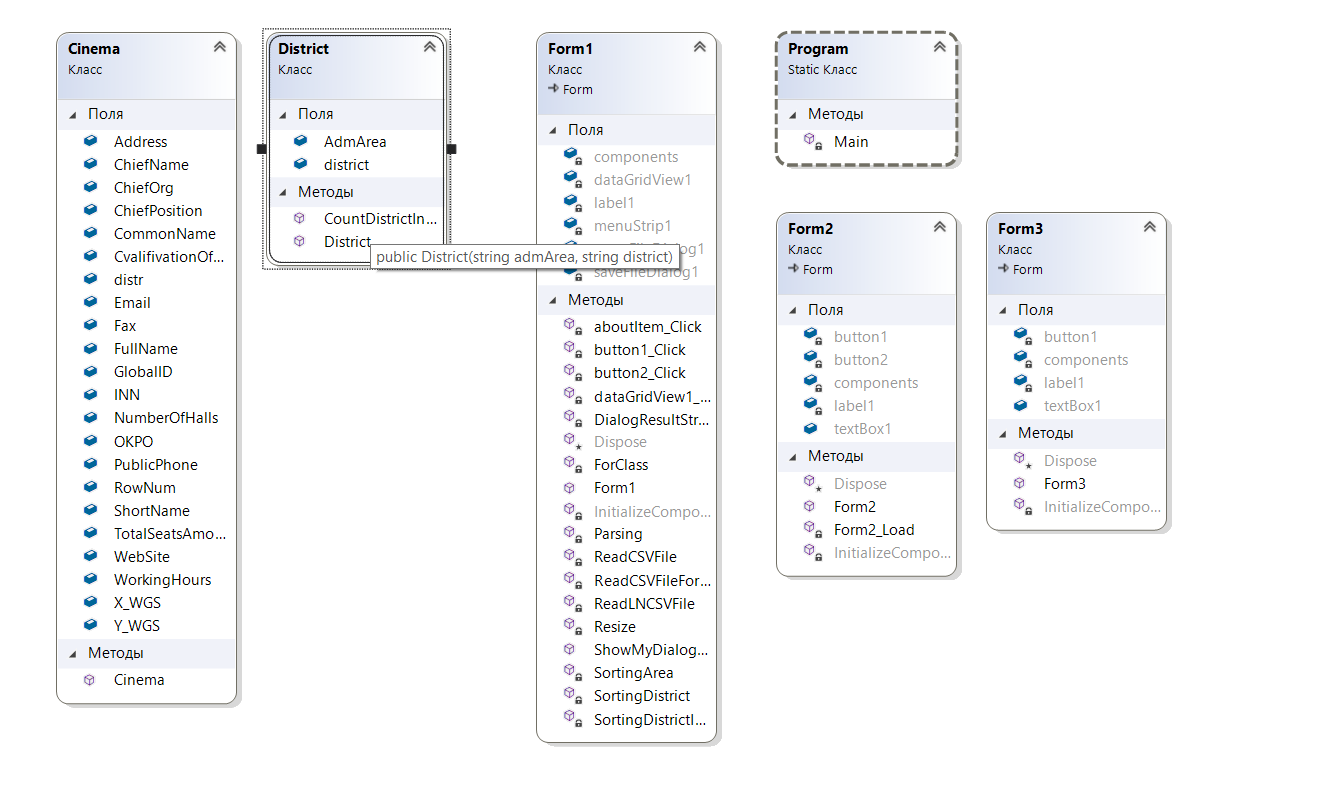


2.6. Окно для сохранения .csv файла



# Структура приложения

* 1. **Диаграмма классов:**



**3.2. Описание классов, их полей и методов Классы:**

* **public partial class Form** класс, содержащий метод Main()
* **public partial class Form1: Form** форма, содержащая основные методы для занесения информации в таблицу.
* **public partial class From2: Form** класс с окном ввода количества выводимых строк.
* **public partial class From3: Form** класс с окном ввода подстроки для фильтрации полей District и AdmArea.
* **Public class Cinema** класс содержащий поля, соответсвующии стобцам таблицы.
* **Public class Distict** класс состоящий в состоянии агрегации с классом Cinema и содержащий в себе 2 поля и метод для вычисления районов в округе.

**Методы:**

* **public partial class Form1: Form** 
  + **private void SortingDistrictInArea** метод сортирующий поля округов по количеству в них районов.
  + **private void ReadCSVFileForDistrict** метод заносящий информация в таблицу для преыдущего метода.
  + **private void SortingDistrict** сортировка районов по подстроке
  + **new void Resize** экранированный метод, подгоняющий ширину столбцов под длину строк находящихся в нем.
  + **private void SortingArea** метод сортировке округов по подстроке
  + **private void ReadLNCSVFile** метод заносящий информацию для методов сортировке по подстроке.
  + **private string DialogResultString** метод вызова диалогового окна ввода подстроки
  + **void aboutItem\_Click** метод выводящий информацию о программе.
  + **private string[][] Parsing** метод осуществляющий парсинг информации из .csv формата.
  + **private DataTable ReadCSVFile** метод заполняющий таблицу по .csv файлу.
  + **private void button1\_Click** метод вызывающий диалоговое окно для открытия .csv файла
  + **public string ShowMyDialogBox** метод вызывающий диалоговое окно в элементов ввода количества выводимых строк
  + **private Cinema[] ForClass** метод заносящий информацию в класс
  + **private void button2\_Click** метод вызывающий открытие окно для сохранения .csv файла
  + **private void dataGridView1\_CellMouseDown** метод реализующий удаление на нажатие правой кнопкой мыши.
* **public class Fractal** o
* **public class Cinema**
* **public class District**

o **public static int CountDistrictInArea** метод считающий количество районов в округе.

**Поля:**

* **public class Cinema**
  + **District distr**
  + **String RowNum**
  + **String CommonName**
  + **String FirstName**
  + **String ShortName**
  + **String ChiefOrg**
  + **String address**
  + **String ChiefPosition**
  + **Sgtring PublicPhone**
  + **String Fax**
  + **String Email**
  + **String WorkingHours**
  + **String CvalificationOfWorkingHours**
  + **String WebSite**
  + **String OKPO**
  + **String INN**
  + **String NumbersOfHall**
  + **String TotalSeatsAmount**
  + **String X\_WGS**
  + **String W\_WGY**
  + **String GlobalID**
* **public class District**
  + **String AdmArea**
  + **String district**

# Распределение исходного кода по файлам проекта

1. Form1.cs – основное окно программы
2. Form2.cs – окно для ввода количества отображаемых строк
3. Form3.cs – окно для ввода подстроки для фильтрации
4. Cinema.cs – класс содержащий поля, соответсвующие столбцам .csv файла
5. District.cs - оставшиеся поля файла
6. Program.cs - начало программы

# Контрольный пример и описание результатов

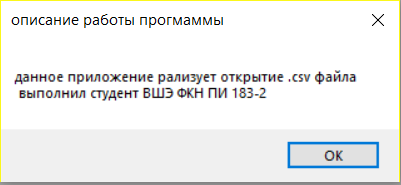
Программа Контрольного домашнего задания представляет собой WindowsForms приложение и предназначена для просмотра, обработки и сохранения результатов обработки данных из файла.

Результаты тестирования:

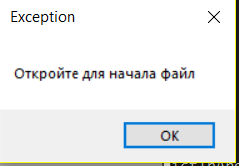
1. При запуске программы появляется пустое окно.
2. Для открытия файла нужно навести на файл и открыть его через диалоговое окно.
3. После можно произвести некоторые сортировки.
4. Также можно узнать информацию о программе.
5. Модно редактировать таблицу.
6. После всего можно сохранить таблицу.

# Сообщения пользователю

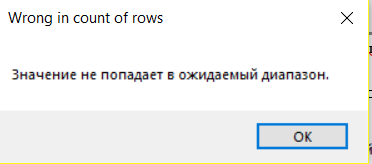
* 1. При нажатии на кнопку о программе:



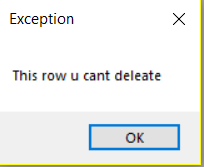
* 1. При поптыке сортировки до открытия файла:



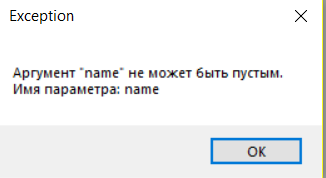
* 1. Если для вывода вводится больше количество строк:



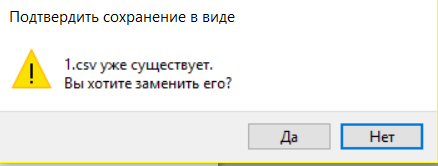
* 1. При попытке удалить строку добавления новых строк:



* 1. При попытке открыть некорректный файл:



* 1. При попытке заменить файл при сохранении:



Остальные сообщения не появляются при корректной работе программы, но при непредвиденных ситуациях появляется окно с текстом «Что-то Пошло не так»!

# Список литературы

1. C# 4.0. Полное руководство

Герберт Шилдт 2011

1. Руководство по C# [Электронный ресурс]

URL: <https://msdn.microsoft.com/ru-ru/>

1. Полное руководство по C# 7.0. [Электронный ресурс]

URL: https://metanit.com/sharp/

# Тест код Программы

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

ToolStripMenuItem fileItem = new ToolStripMenuItem("File");

ToolStripMenuItem newItem = new ToolStripMenuItem("Open csv file");

newItem.Click += button1\_Click;

fileItem.DropDownItems.Add(newItem);

ToolStripMenuItem saveItem = new ToolStripMenuItem("Save");

saveItem.Click += button2\_Click;

fileItem.DropDownItems.Add(saveItem);

ToolStripMenuItem SortArea = new ToolStripMenuItem("Line sorting for area");

SortArea.Click += SortingArea;

fileItem.DropDownItems.Add(SortArea);

ToolStripMenuItem SortDistrict = new ToolStripMenuItem("Line sorting for district");

SortDistrict.Click += SortingDistrict;

fileItem.DropDownItems.Add(SortDistrict);

ToolStripMenuItem SortDistrictInArea = new ToolStripMenuItem("Sort by county");

SortDistrictInArea.Click += SortingDistrictInArea;

fileItem.DropDownItems.Add(SortDistrictInArea);

ToolStripMenuItem Colum = new ToolStripMenuItem("Resize Colums");

Colum.Click += Resize;

fileItem.DropDownItems.Add(Colum);

menuStrip1.Items.Add(fileItem);

ToolStripMenuItem aboutItem = new ToolStripMenuItem("About program");

aboutItem.Click += aboutItem\_Click;

menuStrip1.Items.Add(aboutItem);

DoubleBuffered = true;

}

/// <summary>

/// сортируем округа по количеству в них районов

/// </summary>

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

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

private void SortingDistrictInArea(object sender, EventArgs e)

{

try

{

string[][] file = new string[dataGridView1.RowCount - 1][];

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

{

file[i] = new string[23];

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

file[i][j] = dataGridView1.Rows[i].Cells[j].Value.ToString();

}

}

string[][] newfile = new string[dataGridView1.RowCount - 1][];

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

{

newfile[i] = new string[23];

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

newfile[i][j] = dataGridView1.Rows[i].Cells[j].Value.ToString();

}

}

string[] test = new string[dataGridView1.ColumnCount];

for (int i = 0; i < dataGridView1.RowCount - 2; i++)

{

//newfile[i] = file[i];

for (int j = i + 1; j < dataGridView1.RowCount - 1; j++)

{

if (District.CountDistrictInArea(newfile, file[i][5]) <= District.CountDistrictInArea(newfile, file[j][5]))

{

test = file[i];

file[i] = file[j];

file[j] = test;

}

}

}

ReadCSVFileForDistrict(file);

}

catch(Exception)

{

MessageBox.Show("Откройте для начала файл", "Exception");

}

}

/// <summary>

/// ввод в таблицу для сортивке по ройонвм

/// </summary>

/// <param name="pathfile"></param>

private void ReadCSVFileForDistrict(string[][] pathfile)

{

try

{

for (int i = 0; i < pathfile.GetLength(0); i++)

{

for (int j = 0; j < pathfile[0].Length; j++)

{

dataGridView1.Rows[i].Cells[j].Value = pathfile[i][j];

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Exception 3");

}

}

/// <summary>

/// сортировка по строке для районов

/// </summary>

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

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

private void SortingDistrict(object sender, EventArgs e)

{

try

{

string line = DialogResultString();

if (line == null)

throw new FormatException();

int count = 0;

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

{

if (dataGridView1.Rows[i].Cells[6].Value.ToString().Contains(line))

{

count++;

}

}

string[][] file = new string[count][];

int capacity = 0;

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

{

if (dataGridView1.Rows[i].Cells[6].Value.ToString().Contains(line))

{

file[capacity] = new string[23];

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

file[capacity][j] = dataGridView1.Rows[i].Cells[j].Value.ToString();

}

capacity++;

}

}

ReadLNCSVFile(file);

}

catch (FormatException ex)

{

MessageBox.Show(ex.Message);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Раскрывает все столбцы

/// </summary>

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

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

new void Resize(object sender, EventArgs e)

{

dataGridView1.AutoResizeColumns();

}

/// <summary>

/// все виды сортировок

/// </summary>

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

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

private void SortingArea(object sender, EventArgs e)

{

try

{

string line = DialogResultString();

if (line == null)

throw new FormatException();

int count = 0;

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

{

if(dataGridView1.Rows[i].Cells[5].Value.ToString().Contains(line))

{

count++;

}

}

string[][] file = new string[count][];

int capacity = 0;

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

{

if (dataGridView1.Rows[i].Cells[5].Value.ToString().Contains(line))

{

file[capacity] = new string[23];

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

file[capacity][j] = dataGridView1.Rows[i].Cells[j].Value.ToString();

}

capacity++;

}

}

ReadLNCSVFile(file);

}

catch (FormatException ex)

{

MessageBox.Show(ex.Message);

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Заполнение табицы после ее сортиовки

/// </summary>

/// <param name="pathfile"></param>

private void ReadLNCSVFile(string[][] pathfile)

{

try

{

for (int i = 0; i < pathfile.GetLength(0); i++)

{

for (int j = 0; j < pathfile[0].Length; j++)

{

dataGridView1.Rows[i].Cells[j].Value = pathfile[i][j];

}

}

int count = dataGridView1.RowCount - 1 - pathfile.GetLength(0);

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

{

dataGridView1.Rows.RemoveAt(pathfile.GetLength(0));

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Exception 3");

}

}

/// <summary>

/// ввод строки для фильтрации

/// </summary>

/// <returns></returns>

private string DialogResultString()

{

Form3 testDialog = new Form3();

string n;

// Show testDialog as a modal dialog and determine if DialogResult = OK.

if (testDialog.ShowDialog(this) == DialogResult.OK)

{

// Read the contents of testDialog's TextBox.

n = testDialog.textBox1.Text;

}

else

{

testDialog.Close();

return null;

}

testDialog.Dispose();

return n;

}

/// <summary>

/// о программе

/// </summary>

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

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

void aboutItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("данное приложение рализует открытие .csv файла \n выполнил студент ВШЭ ФКН ПИ 183-2","описание работы прогмаммы");

System.Diagnostics.Process.Start("https://www.youtube.com/watch?v=pv4Raf3Bfsc");

}

/// <summary>

/// парсинг

/// </summary>

/// <param name="file"></param>

/// <returns></returns>

private string[][] Parsing(string[] file, uint n)

{

int[] ar = new int[0];

string[][] newar= new string[n][];

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

{

newar[i] = new string[23];

string[] arr = new string[23];

int count = 0;

arr[count] = "";

for (int j = 0; j < file[i].Length; j++)

{

if(file[i][j] == '"')

{

if(file[i][j+1] == '"' && file[i][j+2]!=';')

{

arr[count] += '"';

j += 2;

continue;

}

else

{

j++;

while(file[i][j]!='"')

{

arr[count] += file[i][j];

j++;

}

continue;

}

}

if (file[i][j] == ';')

{

newar[i][count] = arr[count];

count++;

if(count<23)

arr[count] = "";

continue;

}

arr[count] += file[i][j];

}

}

return newar;

}

/// <summary>

/// записываем информация в таблицу

/// </summary>

/// <param name="pathfile"></param>

/// <returns></returns>

private DataTable ReadCSVFile(string[][] pathfile)

{

//создаём таблицу

DataTable dt = new DataTable("Cinemas");

//создаём колонки

for (int i = 0; i < pathfile[0].Length; i++)

{

DataColumn col = new DataColumn(pathfile[0][i], typeof(String));

dt.Columns.AddRange(new DataColumn[] { col });

}

try

{

DataRow dr = null;

for (int i = 1; i < pathfile.GetLength(0); i++)

{

dr = dt.NewRow();

for (int j = 0; j < pathfile[0].Length; j++)

{

dr[pathfile[0][j]] = pathfile[i][j];

}

dt.Rows.Add(dr);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message,"Exception");

}

return dt;

}

/// <summary>

/// открыть csv файл

/// </summary>

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

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

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

{

try

{

openFileDialog1.Filter = "Файлы csv|\*.csv";

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

string[] file = File.ReadAllLines(openFileDialog1.FileName);

uint n = uint.Parse(ShowMyDialogBox());

if (n < 2 || n > file.Length) throw new ArgumentException();

Cinema[] data = ForClass(Parsing(file , n));

dataGridView1.DataSource = ReadCSVFile(Parsing(file, n));

}

}

catch(ArgumentException ex)

{

MessageBox.Show(ex.Message,"Wrong in count of rows");

}

catch(FormatException ex)

{

MessageBox.Show(ex.Message);

}

catch (IOException ex)

{

MessageBox.Show(ex.Message);

}

catch(Exception)

{

MessageBox.Show("Something went wrong","Exception1");

}

}

/// <summary>

/// Ввод числа строк

/// </summary>

/// <returns></returns>

public string ShowMyDialogBox()

{

Form2 testDialog = new Form2();

string n;

// Show testDialog as a modal dialog and determine if DialogResult = OK.

if (testDialog.ShowDialog(this) == DialogResult.OK)

{

// Read the contents of testDialog's TextBox.

n = testDialog.textBox1.Text;

}

else

{

testDialog.Close();

return null;

}

testDialog.Dispose();

return n;

}

/// <summary>

/// заполняем массив классов

/// </summary>

/// <param name="line"></param>

/// <returns></returns>

private Cinema[] ForClass(string[][] line)

{

Cinema[] data=null;

try

{

data = new Cinema[line.GetLength(0)];

data[1] = new Cinema(new District(line[1][5], line[1][6]), line[1][0], line[1][1], line[1][2], line[1][3], line[1][4],

line[1][7], line[1][8], line[1][9], line[1][10], line[1][11], line[1][12],

line[1][13], line[1][14], line[1][15], line[1][16], line[1][17], line[1][18], line[1][19],

line[1][20], line[1][21], line[1][22]);

for (int i = 2; i < line.GetLength(0); i++)

{

District distric = new District(line[i][5], line[i][6]);

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

{

if (distric.AdmArea == data[j].distr.AdmArea && distric.district == data[j].distr.district)

{

data[i] = new Cinema(data[j].distr, line[i][0], line[i][1], line[i][2], line[i][3], line[i][4],

line[i][7], line[i][8], line[i][9], line[i][10], line[i][11], line[i][12],

line[i][13], line[i][14], line[i][15], line[i][16], line[i][17], line[i][18], line[i][19],

line[i][20], line[i][21], line[i][22]);

}

else

{

data[i] = new Cinema(new District(line[i][5], line[i][6]), line[i][0], line[i][1], line[i][2], line[i][3], line[i][4],

line[i][7], line[i][8], line[i][9], line[i][10], line[i][11], line[i][12],

line[i][13], line[i][14], line[i][15], line[i][16], line[i][17], line[i][18], line[i][19],

line[i][20], line[i][21], line[i][22]);

}

}

}

}

catch (Exception)

{

MessageBox.Show("Проблемма при заполнении класса Cinema", "Exception");

}

return data;

}

/// <summary>

/// сохранение

/// </summary>

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

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

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

{

Stream myStream;

saveFileDialog1.Filter = "Файлы csv|\*.csv";

//if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

// return;

// получаем выбранный файл

string filename = saveFileDialog1.FileName;

// сохраняем текст в файл

saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

if ((myStream = saveFileDialog1.OpenFile()) != null)

{

StreamWriter myWritet = new StreamWriter(myStream);

try

{

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

myWritet.Write(dataGridView1.Columns[i].Name.ToString() + ";");

}

myWritet.WriteLine();

for (int i = 0; i < dataGridView1.RowCount; i++)

{

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

if(dataGridView1.Rows[i].Cells[j].Value!=null)

myWritet.Write('"' + dataGridView1.Rows[i].Cells[j].Value.ToString() + '"' + ";");

else

myWritet.Write(";");

}

myWritet.WriteLine();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

myWritet.Close();

}

}

myStream.Close();

}

}

/// <summary>

/// удаление строки

/// </summary>

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

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

private void dataGridView1\_CellMouseDown(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

{

DialogResult result = MessageBox.Show

(

"Delete the row to which the selected cell belongs?",

"Message",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button1

);

if (result == DialogResult.Yes)

{

try

{

dataGridView1.Rows.RemoveAt(dataGridView1.SelectedCells[0].RowIndex);

}

catch(Exception)

{

MessageBox.Show("This row u cant deleate","Exception");

}

}

}

}

}

public class Cinema

{

public District distr;

public string RowNum;

public string CommonName;

public string FullName;

public string ShortName;

public string ChiefOrg;

public string Address;

public string ChiefName;

public string ChiefPosition;

public string PublicPhone;

public string Fax;

public string Email;

public string WorkingHours;

public string CvalifivationOfWOrkingHours;

public string WebSite;

public string OKPO;

public string INN;

public string NumberOfHalls;

public string TotalSeatsAmount;

public string X\_WGS;

public string Y\_WGS;

public string GlobalID;

public Cinema(District district, string rowNum, string commonName, string fullName, string shortName, string chiefOrg, string address, string chiefName, string chiefPosition, string publicPhone, string fax, string email, string workingHours, string cvalifivationOfWOrkingHours, string webSite, string oKPO, string iNN, string numberOfHalls, string totalSeatsAmount, string x\_WGS, string y\_WGS, string globalID)

{

distr = district;

RowNum = rowNum;

CommonName = commonName;

FullName = fullName;

ShortName = shortName;

ChiefOrg = chiefOrg;

Address = address;

ChiefName = chiefName;

ChiefPosition = chiefPosition;

PublicPhone = publicPhone;

Fax = fax;

Email = email;

WorkingHours = workingHours;

CvalifivationOfWOrkingHours = cvalifivationOfWOrkingHours;

WebSite = webSite;

OKPO = oKPO;

INN = iNN;

NumberOfHalls = numberOfHalls;

TotalSeatsAmount = totalSeatsAmount;

X\_WGS = x\_WGS;

Y\_WGS = y\_WGS;

GlobalID = globalID;

}

}

public class District

{

public string AdmArea;

public string district;

public District(string admArea, string district)

{

AdmArea = admArea;

this.district = district;

}

public static int CountDistrictInArea(string[][] arr, string area)

{

int count = 0;

List<string> dis = new List<string>();

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

{

if(arr[i][5]==area)

{

bool flag = false;

for (int j = 0; j < dis.Count; j++)

{

if (string.Compare(arr[i][6], dis[j])==0) flag = true;

}

dis.Add(arr[i][6]);

if (flag) continue;

else count++;

}

}

return count;

}

}