

Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Департамент

Программной инженерии

*Контрольное домашнее задание
по дисциплине
«Программирование»*

Тема работы: Вариант 14. Выходы из пешеходных тоннелей

Выполнил: студент группы БПИ181(2)

_____ Матевосян А.А.

тел. +7 (999) 905 6164

e-mail адрес: aamatevosyan@edu.hse.ru

Преподаватель: Береснева Екатерина Николаевна

Москва, 2019 год. Модуль 3

Оглавление

1. Условие задачи	3
2. Функции разрабатываемого приложения	4
2.1. Варианты использования	4
2.2. Описание интерфейса пользователя	4
3. Структура приложения	8
3.1. Диаграмма классов	8
3.2. Описание классов, их полей и методов	9
4. Распределение исходного кода по файлам проекта	15
5. Контрольный пример и описание результатов	16
6. Текст (код) программы	17
AddRowForm.cs	17
Area.cs	18
AreaList.cs	19
EditRowForm.cs	19
IDGenerator.cs	21
InnerTunnel.cs	21
InnerTunnelList.cs	23
MainForm.cs	23
OpenForm.cs	23
Program.cs	29
Tunnel.cs	30
TunnelList.cs	31
7. Список литературы	36

1. Условие задачи

Программа Контрольного домашнего задания представляет собой **WindowsForms** приложение и предназначена для просмотра, обработки и сохранения результатов обработки данных из файла. (Вариант 14)

1. Требования к основным классам приложения
 - 1.1. Основная информация о Пешеходных тоннелях хранится в объектах класса **Тоннель**. Набор полей класса задаётся полями CSV-файла **Выходы из пешеходных тоннелей.csv**, кроме полей, содержащих информацию об административном округе и районе, представленных полем типа **Район**. Класс **Тоннель** находится в отношении агрегации с классом **Район**. Один из методов класса **Тоннель** возвращает значение **global_id**, сохранённое в поле **Tunnel**.
 - 1.2. Класс **Район** представляет округа города Москвы и содержит поля, заданные полями CSV-файла: **AdmArea, District**.
 - 1.3. Дополнительные классы, необходимые для решения задачи (объявляет автор программы).
2. Приложение должно поддерживать следующие функции:
 - 2.1. Открыть CSV-файл (*.csv) с исходными данными и проверить корректность данных в нём.
 - 2.2. Загрузить данные из CSV-файла в объекты классов **Район**, «**Район**» (если объект **Район** с данными об определённом районе существует, то он является общим для всех объектов **Тоннель** этого района) и др.
 - 2.3. Отобразить данные из объектов в оконной форме.
 - 2.4. Создать новую запись о Тоннеле.
 - 2.5. Удалить уже существующую запись о Тоннеле
 - 2.6. Отредактировать существующую запись о Тоннеле
 - 2.7. Отсортировать данные по алфавиту по полям: **Name, AdmArea**
 - 2.8. Отсортировать данные по количеству районов в округах.
 - 2.9. Отфильтровать данные полям: **Tunnel{global_id}, AdmArea**. Данные для фильтрации вводятся пользователем.
 - 2.10. Сохранять результаты редактирований, сортировок и фильтров в CSV-файл. *Режимы сохранения в файл*: создание нового файла, замена содержимого уже существующего файла, добавление сохраняемых данных к содержимому существующего файла.
3. Требования к интерфейсу
 - 3.1. При управлении файлом (загрузка, сохранение) использовать **OpenFileDialog** и **SaveFileDialog**.
 - 3.2. Для отображения данных использовать сетку **DataGridView**
 - 3.3. Количество отображаемых в сетке элементов (*N*) выбирается пользователем, *N* > 1 и не превышает количества записей в файле **Выходы из пешеходных тоннелей.csv**.
4. 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.

2. Функции разрабатываемого приложения

2.1. Варианты использования

Программа предназначена для просмотра, обработки и сохранения результатов обработки данных из csv файла. С его помощью можно открыть данные о пешеходных тоннелях и представить их в виде удобной для восприятия таблицы.

Программу можно использовать в качестве инструмента для анализа больших данных. Изучать и изменять известные нам данные.

Программа может открывать не только весь файл, а также заданный участок данных. Можно добавлять, удалять и изменять данные, также как отсортировать и отфильтровать их.

2.2. Описание интерфейса пользователя

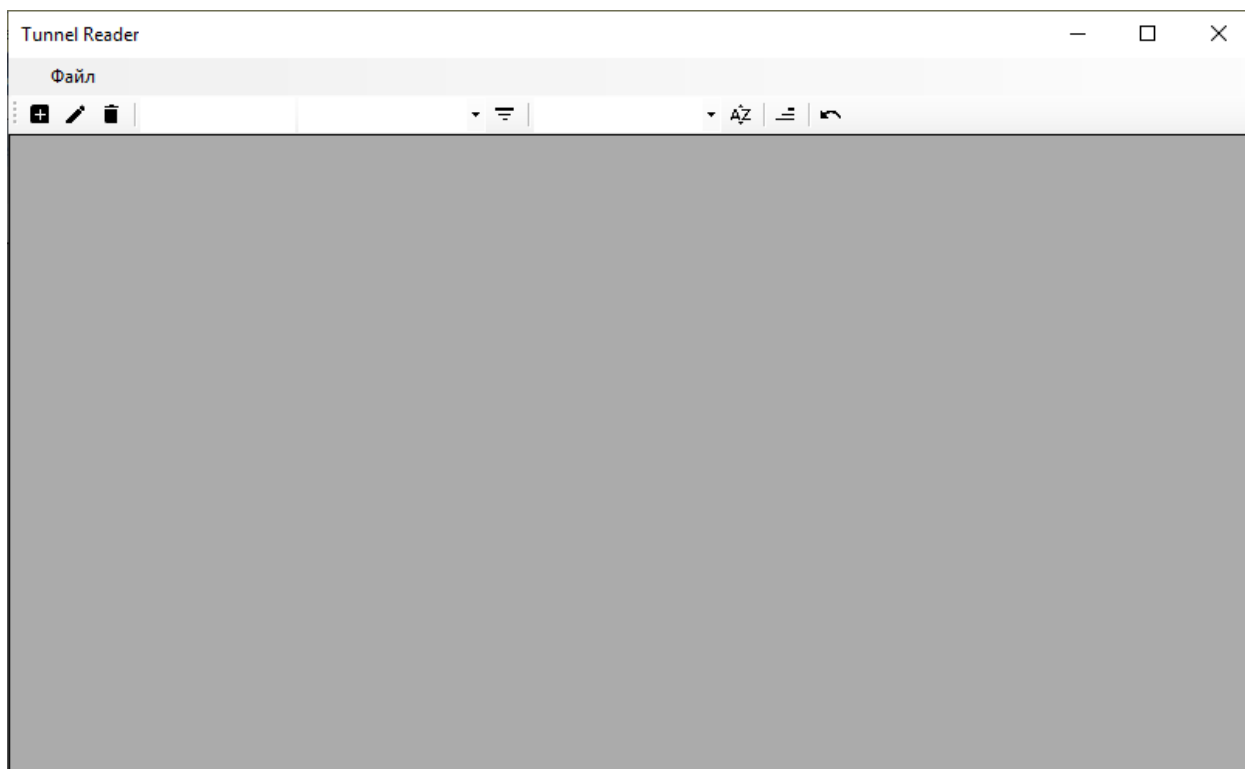


Рисунок (1)

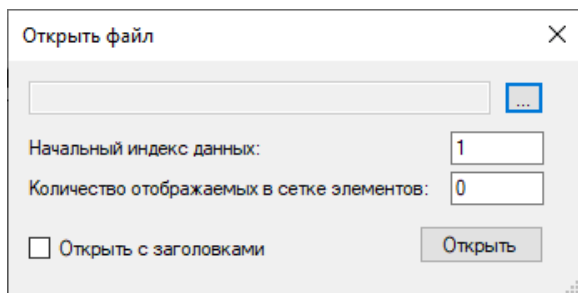


Рисунок (2)

На рис. 1 изображен интерфейс программы. Для того чтобы открыть файл нужно нажать на Файл -> Открыть. Открывается следующее окно (Рис. 2).

При нажатии на кнопку ... можно выбрать нужный файл с помощью появившегося диалогового окна. (Рис. 3)

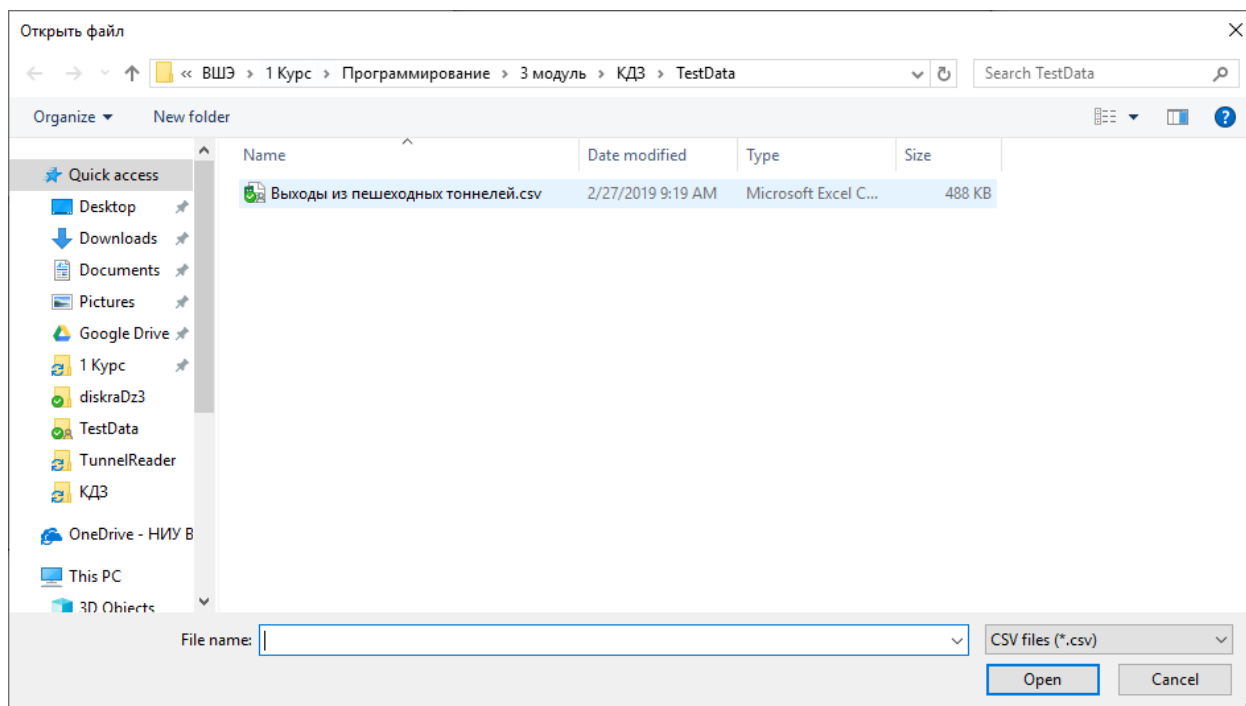
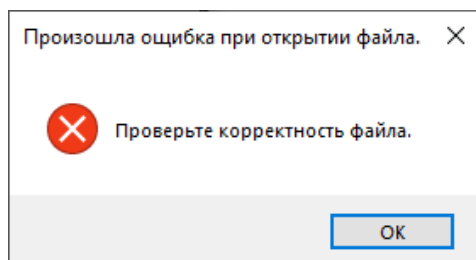


Рисунок (3)



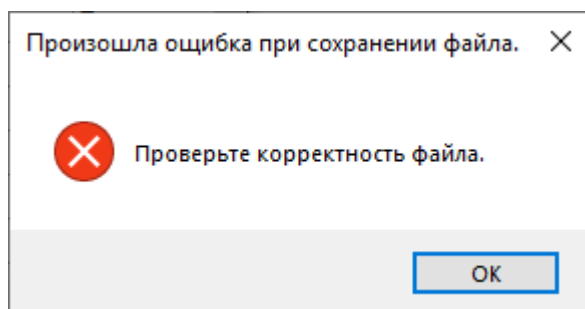
Потом нужно указать с какого индекса и сколько строк из исходного файла нужно открыть и нажать кнопку открыть. При выборе неправильного или поврежденного файла отображается окно. (Рис. 4)

Если выбранный файл корректен, то программа загружает данные из файла. (Рис. 5)

Рисунок (4)

Tunnel Reader				
Файл				
	ROWNUM	Name	Tunnel global_id	Tunnel value
1	1	город Зеленоград, корпус 933, к корпусу 933, северный выход	1697031	Тоннель пеш
2	2	город Зеленоград, корпус 1001, сооружение 1, к озеру Школьное, северный выход	1697032	Тоннель пеш
3	3	Волоколамское шоссе, дом 92, сооружение 1, нечетная сторона, западный выход	1696973	Тоннель пеш
4	4	Дмитровское шоссе, дом 11, сооружение 1, четная сторона, северный выход	1696999	Тоннель пеш
5	5	улица Габричевского, дом 10, корпус 1, нечетная сторона, западный выход	1696958	Тоннель пеш
6	6	город Зеленоград, корпус 1001, сооружение 1, к Панфиловскому проспекту, дом 40, западный выход	1697032	Тоннель пеш
7	7	Волоколамское шоссе, дом 92, сооружение 1, четная сторона, восточный выход	1696973	Тоннель пеш
8	8	Дмитровское шоссе, дом 11, сооружение 1, нечетная сторона, южный выход	1696999	Тоннель пеш
9	9	Дмитровское шоссе, владение 14, сооружение 2, четная сторона, южный выход	1696963	Тоннель пеш
10	10	Гончарный проезд, дом 6, сооружение 1, четная сторона, северный выход	1696939	Тоннель пеш
11	11	Каширское шоссе, дом 45, строение 4, западный выход	1697066	Тоннель пеш
12	12	Дмитровское шоссе, дом 110Д, сооружение 1, четная сторона, северный выход	1696988	Тоннель пеш
13	13	Каширское шоссе, дом 24, сооружение 1, четная сторона, западный выход	1697115	Тоннель пеш
14	14	Каширское шоссе, дом 24, сооружение 1, нечетная сторона, западный выход	1697115	Тоннель пеш
15	15	Дмитровское шоссе, дом 66, сооружение 1, четная сторона, северный выход	1697105	Тоннель пеш
16	16	Каширское шоссе, дом 42, корпус 1, сооружение 1, нечетная сторона, восточный выход	1696952	Тоннель пеш

Рисунок (5)



Для пере сохранения обработанных данных нужно нажать в Файл → Сохранить.

Для сохранения данных в другой файл нужно нажать в Файл → Сохранить как. Появляется окно (Рис. 7), где нужно выбрать где и как нужно сохранить файл. При вводе неправильных данных или при отказе доступа на запись в файл появляется окно вида (Рис.6)

Рисунок (6)

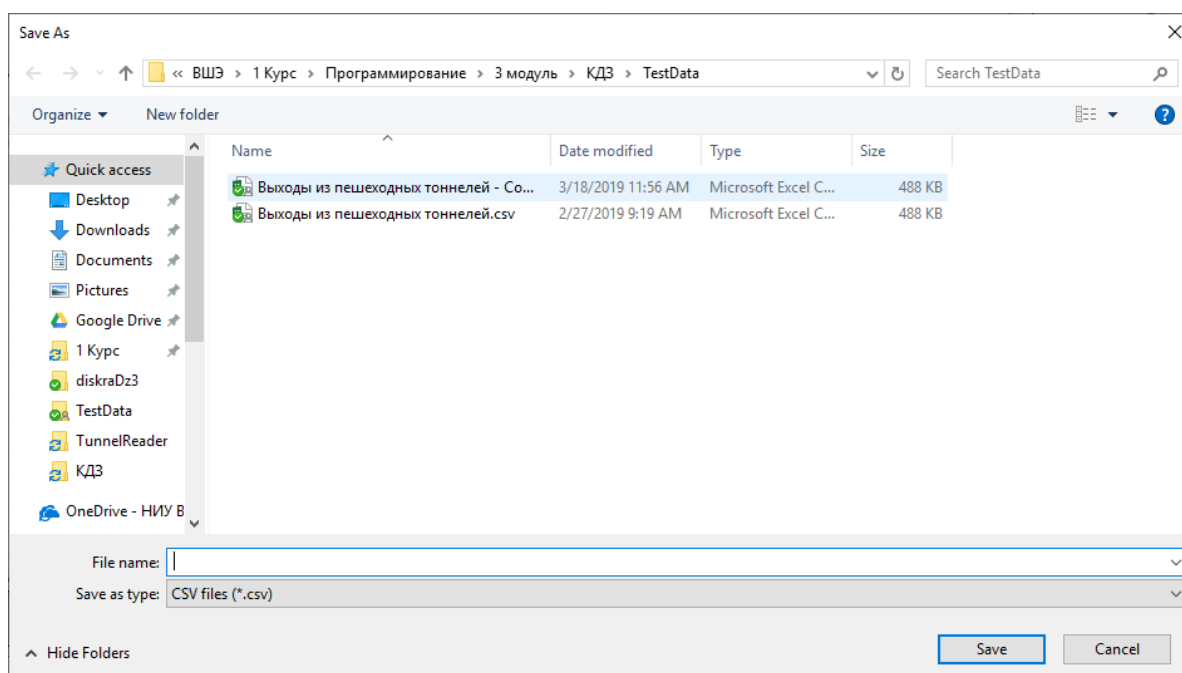


Рисунок (7)

Инструментальная панель

1.		Создать новую запись о Тоннеле.
2.		Отредактировать уже существующую запись о Тоннеле.
3.		Удалить уже существующую запись о Тоннеле.
4.		Отфильтровать данные.
5.		Отсортировать по алфавиту.
6.		Отсортировать по количеству районов в округах.


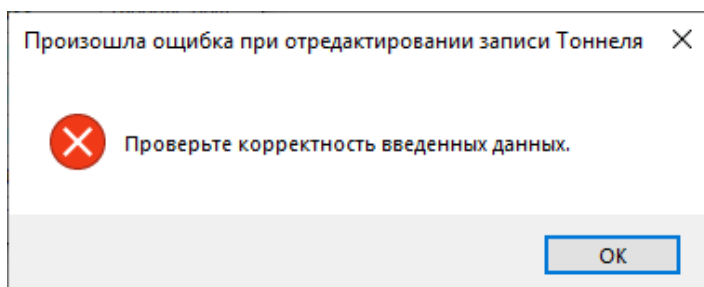
7.		Шаг назад
----	---	-----------

Таблица (1)

Для добавления новой записи в таблицу нужно нажать на кнопку 1 (Таблица 1). Появится окно (Рис. 9) где нужно вводить данные в соответствующих полях. Значение `global_id` генерируется автоматически автоинкрементном.

Так же для изменения данных, нужно выбрать строку, которую нужно изменить и нажать на кнопку 2 (Таблица 1). Появляется окно (Рис. 10) где нужно вводить значения данные в соответствующих полях.



При создании и при редактировании некорректных данных появляется окно вида (Рис. 8).

Для удаления нужно выбрать строку и нажать на кнопку 3 (Таблица 3).

Рисунок 8

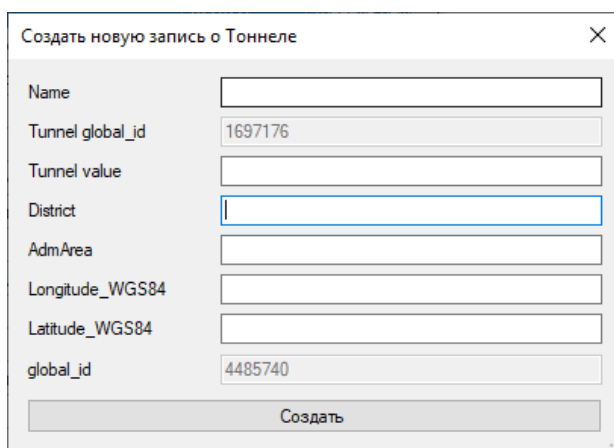


Рисунок 9

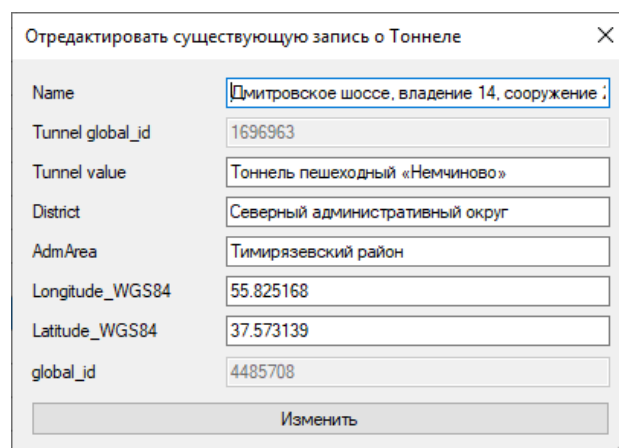


Рисунок 10

Для фильтрации данных нужно выбрать в списке фильтрации название поля по которому будет произведена фильтрация, а затем ввести значение для фильтрации в соседнем поле, и нажать на кнопку 4 (Таблица 1).

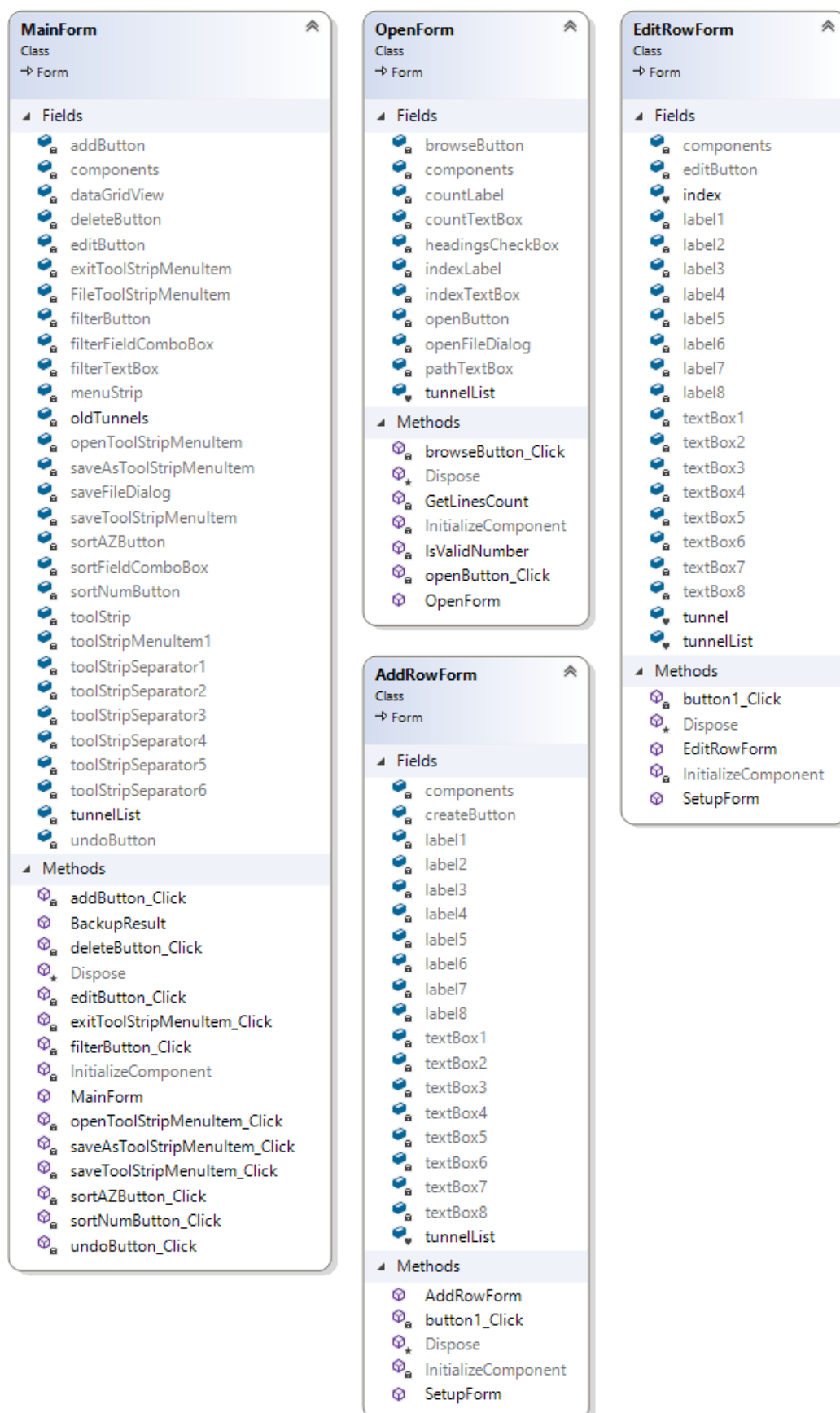
Для сортировки данных в алфавитном порядке нужно выбрать в списке сортировки название поля по которому будет произведена сортировка и нажать на кнопку 5 (Таблица 1).

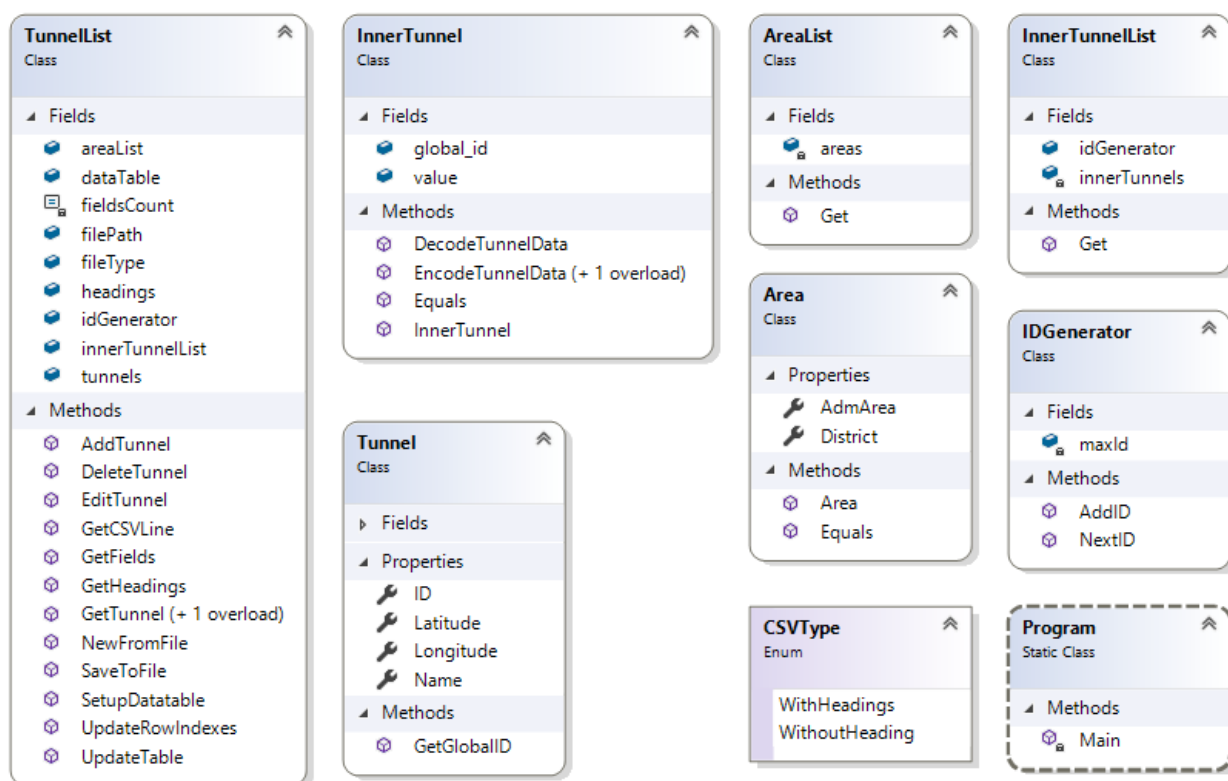
Для сортировки по количеству округов нужно нажать на кнопку 6 (Таблица 1).

В качестве дополнительной возможности, есть кнопка 7 (Таблица 1), при нажатии на которой отменяется предыдущее действие (фильтрация, сортировка ...). При наведении указателя мыши на элементах инструментальной панели появляются подсказки.

3. Структура приложения

3.1. Диаграмма классов





3.2. Описание классов, их полей и методов

Описание и функциональное назначение классов

Класс	Назначение
AddRowForm	Класс оконной формы для создания новой записи Тоннеля
Area	Класс районов
AreaList	Класс для работы со списком районов
EditRowForm	Класс оконной формы для редактирования заданной записи Тоннеля
IDGenerator	Класс генерации ID
InnerTunnel	Класс внутренних тоннелей
InnerTunnelList	Класс для работы со списком внутренних
MainForm	Класс оконной формы главного меню
OpenForm	Класс оконной формы для открытия файла
Program	Класс начальной точки программы
Tunnel	Класс тоннель
TunnelList	Класс для работы с данными

Описание полей, методов и свойств класса MainForm

Поля				
Имя	Модификатор доступа	Тип	Назначение	
oldTunnels	private	List<Tunnel>	Значение тоннелей до совершенного действия	
tunnelList	private	TunnelList	Список тоннелей	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
addButton_Click	private	void	object, EventArgs e	Добавляет новое значение в таблицу
BackupResult	public	void	-	Сохраняет список тоннелей для последующего восстановления
deleteButton_Click	private	void	object, EventArgs e	Удаляет значения из таблицы
editButton_Click	private	void	object, EventArgs e	Редактирует значение из таблицы
filterButton_Click	private	void	object, EventArgs e	Фильтрует данные
MainForm	public	-	-	Конструктор по умолчанию
openToolStripMenuItem_Click	private	void	object, EventArgs e	Показывает окно для последующего открытия файла
saveAsToolStripMenuItem_Click	private	void	object, EventArgs e	Сохраняет данные в заданный файл
saveToolStripMenuItem_Click	private	void	object, EventArgs e	Пересохраняет данные
sortAZButton_Click	private	void	object, EventArgs e	Сортирует данные в алфавитном порядке
sortNumButton_Click	private	void	object, EventArgs e	Сортирует данные по количеству округов
undoButton_Click	private	void	object, EventArgs e	Восстанавливает данные до совершенного действия
exitToolStripMenuItem_Click	private	void	object, EventArgs e	Завершает программу

Описание полей, методов и свойств класса OpenForm

Поля				
Имя	Модификатор доступа	Тип	Назначение	
tunnelList	internal	TunnelList	Список тоннелей	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
browseButton_Click	private	void	object, EventArgs	Возвращает путь к выбранному файлу
GetLinesCount	private	int	string	Возвращает количество строк файла
IsValidNumber	private	bool	string	Проверяет является ли строка числом
openButton_Click	private	void	object, EventArgs	Открывает выбранный файл
OpenForm	public	-	-	Конструктор по умолчанию

Описание полей, методов и свойств класса AddRowForm

Поля				
Имя	Модификатор доступа	Тип	Назначение	
tunnellist	internal	Tunnellist	Список тоннелей	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
AddRowForm	public	-	-	Конструктор по умолчанию
button1_Click	private	void	object, EventArgs	Добавляет данные в таблицу
SetupForm	public	void	-	Инициализирует форму

Описание полей, методов и свойств класса EditRowForm

Поля			
Имя	Модификатор доступа	Тип	Назначение
tunnellist	internal	Tunnellist	Список тоннелей

index	internal	int	Выбранный индекс списка	
tunnel	internal	Tunnel	Выбранный тоннель	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
EditRowForm	public	-	-	Конструктор по умолчанию
button1_Click	private	void	object, EventArgs	Редактирует данные из таблицы
SetupForm	public	void	-	Инициализирует форму

Описание полей, методов и свойств класса Program

Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
Main	private	void	-	Начальная точка приложения

Описание полей, методов и свойств класса Area

Свойства				
Имя	Модификатор доступа	Тип	Назначение	
AdmArea	public	string	Административный округ	
District	public	string	Район округа	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
Area	public	-	string, string	Конструктор
Equals	public	bool	object	Переопределенный метод для сравнения по значению

Описание полей, методов и свойств класса InnerTunnel

Поля				
Имя	Модификатор доступа	Тип	Назначение	
global_id	public	int	ID Внутреннего тоннеля	
value	public	string	Значение внутреннего тоннеля	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
DecodeTunnelData	public	void	string, out string, out string	Декодирует данные из строки CSV

EncodeTunnelData	public	string	InnerTunnel	Кодирует данные
EncodeTunnelData	public	string	string, string	Кодирует данные
InnerTunnel	public	-	int, string	Конструктор
Equals	public	bool	object	Переопределенный метод для сравнения по значению

Описание полей, методов и свойств класса IDGenerator

Поля				
Имя	Модификатор доступа	Тип	Назначение	
maxId	private	int	Максимальный ID	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
AddID	public	void	int	Добавляет ID
NextID	public	string	-	Возвращает следующий ID

Описание полей, методов и свойств класса AreaList

Поля				
Имя	Модификатор доступа	Тип	Назначение	
areas	private	List<Area>	Список районов	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
Get	public	Area	string, string	Возвращает район с заданными значениями

Описание полей, методов и свойств класса InnerTunnelList

Поля				
Имя	Модификатор доступа	Тип	Назначение	
innerTunnels	private	List<InnerTunnel>	Список внутренних тоннелей	
idGenerator	public	IDGenerator	Класс для генерирования ID	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение

Get	public	InnerTunnel	int, string	Возвращает внутренний тоннель с заданными значениями
-----	--------	-------------	----------------	---

Описание полей, методов и свойств класса Tunnel

Поля				
Имя	Модификатор доступа	Тип	Назначение	
area	public	Area	Район тоннеля	
innerTunnel	public	InnerTunnel	Внутренний тоннель	
Свойства				
ID	public	int	ID тоннеля	
Latitude	public	double	Широта	
Longitude	public	double	Долгота	
Name	public	string	Имя тоннеля	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
GetGlobalID	public	int	-	Возвращает ID внутреннего тоннеля

Описание полей, методов и свойств класса TunnelList

Поля				
Имя	Модификатор доступа	Тип	Назначение	
areaList	public	AreaList	Список Районов	
dataTable	public	DataTable	Таблица данных	
fieldsCount	public	int	Число столбцов в таблице	
filePath	public	string	Путь к файлу	
fileType	public	CSVType	Тип Файла	
headings	public	string[]	Заголовки	
idGenerator	public	IDGenerator	Класс для генерации ID	
innerTunnelList	public	InnerTunnelList	Список внутренних тоннелей	
tunnels	public	List<Tunnel>	Список тоннелей	
Методы				
Имя	Модификатор доступа	Тип	Аргументы	Назначение
AddTunnel	public	void	string[]	Добавляет тоннель к списку
DeleteTunnel	public	int	int	Удаляет заданный тоннель
EditTunnel	public	void	string[], int	Редактирует заданный тоннель

GetCSVLine	public	string	Tunnel, int	Возвращает строку данных CSV
GetFields	public	string[]	Tunnel	Возвращает значения для столбцов
GetHeadings	public	string	-	Возвращает заголовки
GetTunnel	public	Tunnel	string	Возвращает тоннель с заданными данными
GetTunnel	public	Tunnel	string[]	Возвращает тоннель с заданными данными
NewFromFile	public	TunnellList	string, CSVType, int, int	Создает новый объект из файла
SaveToFile	public	void	string	Сохраняет объект в файл
SetupDatatable	public	void	-	Инициализирует таблицу
UpdateRowIndexes	public	void	-	Обновляет индексы строк
UpdateTable	public	void	-	Обновляет таблицу

4. Распределение исходного кода по файлам проекта

Класс	Назначение
<i>AddRowForm.cs</i>	Реализация класса AddRowForm
<i>Area.cs</i>	Реализация класса Area
<i>AreaList.cs</i>	Реализация класса AreaList
<i>EditRowForm.cs</i>	Реализация класса EditRowForm
<i>IDGenerator.cs</i>	Реализация класса IDGenerator
<i>InnerTunnel.cs</i>	Реализация класса InnerTunnel
<i>InnerTunnellList.cs</i>	Реализация класса InnerTunnellList
<i>MainForm.cs</i>	Реализация класса MainForm
<i>OpenForm.cs</i>	Реализация класса OpenForm
<i>Program.cs</i>	Реализация класса Program
<i>Tunnel.cs</i>	Реализация класса Tunnel
<i>TunnellList.cs</i>	Реализация класса TunnellList и enum CSVType

5. Контрольный пример и описание результатов

Программа была протестирована по заданному тестовому файлу. В ходе тестирования были проверены следующие возможности программы

1. Открытие файлов, проверка открытия некорректного файла и возможность открытия заданной части данных
2. Переименование файла, сохранения данных в другой файл, проверка доступа к файлу
3. Сортировка данных
4. Фильтрация данных
5. Возможность добавления, удаления и редактирования данных, и работа с некорректно введенными данными
6. Была проверена работоспособность дополнительной возможности Шаг назад

6. Текст (код) программы

AddRowForm.cs

```

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 TunnelReader
{
    /// <summary>
    /// Форма для создания новой записи тоннеля
    /// </summary>
    public partial class AddRowForm : Form
    {
        /// <summary>
        /// Список тоннелей
        /// </summary>
        internal TunnelList tunnelList;
        /// <summary>
        /// Конструктор по умолчанию
        /// </summary>
        public AddRowForm()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Инициализирует форму
        /// </summary>
        public void SetupForm()
        {
            string[] headings = tunnelList.headings;
            label1.Text = headings[1];
            label2.Text = headings[2] + " global_id";
            label3.Text = headings[2] + " value";
            label4.Text = headings[3];
            label5.Text = headings[4];
            label6.Text = headings[5];
            label7.Text = headings[6];
            label8.Text = headings[7];

            textBox2.Text = tunnelList.innerTunnelList.idGenerator.NextID().ToString();
            textBox8.Text = tunnelList.idGenerator.NextID().ToString();
            textBox2.Enabled = false;
            textBox8.Enabled = false;
        }

        /// <summary>
        /// Добавляет данные в таблицу
        /// </summary>
        private void button1_Click(object sender, EventArgs e)
        {
            try

```

```

        {
            string[] fields = new string[]
            {
                "", textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text,
                textBox5.Text, textBox6.Text,
                textBox7.Text, textBox8.Text
            };
            tunnelList.AddTunnel(fields);
            this.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Проверьте корректность введенных данных.", "Произошла
ошибка при создании новой записи Тоннеля", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

Area.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TunnelReader
{
    /// <summary>
    /// Класс Район
    /// </summary>
    class Area
    {
        /// <summary>
        /// Административный округ
        /// </summary>
        public string AdmArea { get; set; }
        /// <summary>
        /// Район округа
        /// </summary>
        public string District { get; set; }

        /// <summary>
        /// Конструктор
        /// </summary>
        public Area(string admArea, string district)
        {
            AdmArea = admArea;
            District = district;
        }

        /// <summary>
        /// Переопределенный метод для сравнения по значению
        /// </summary>
        public override bool Equals(object obj)
        {
            Area area = obj as Area;
            return (area.AdmArea == this.AdmArea) && (area.District == this.District);
        }
    }
}

```

AreaList.cs

```

using System.Collections.Generic;

namespace TunnelReader
{
    /// <summary>
    /// Класс для работы со списком Районов
    /// </summary>
    internal class AreaList
    {
        /// <summary>
        /// Список районов
        /// </summary>
        private readonly List<Area> areas = new List<Area>();

        /// <summary>
        /// Возвращает район с заданными значениями
        /// </summary>
        public Area Get(string admArea, string district)
        {
            var area = new Area(admArea, district);
            var result = areas.Find(a => a.Equals(area));
            if (result != null) return result;

            areas.Add(area);
            return area;
        }
    }
}

```

EditRowForm.cs

```

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 TunnelReader
{
    /// <summary>
    /// Форма для редактирования записи тоннеля
    /// </summary>
    public partial class EditRowForm : Form
    {
        /// <summary>
        /// Список тоннелей
        /// </summary>
        internal Tunnellist tunnelliList;
        /// <summary>
        /// Выбранный индекс списка
        /// </summary>
        internal int index;
        /// <summary>
        /// Выбранный тоннель
        /// </summary>
        internal Tunnel tunnel;
    }
}

```

```

/// <summary>
/// Конструктор по умолчанию
/// </summary>
public EditRowForm()
{
    InitializeComponent();
}

/// <summary>
/// Инициализирует форму
/// </summary>
public void SetupForm()
{
    string[] headings = tunnellist.headings;
    tunnel = tunnellist.tunnels[index];
    label11.Text = headings[1];
    label12.Text = headings[2] + " global_id";
    label13.Text = headings[2] + " value";
    label14.Text = headings[3];
    label15.Text = headings[4];
    label16.Text = headings[5];
    label17.Text = headings[6];
    label18.Text = headings[7];

    string[] fields = tunnellist.GetFields(tunnel);
    textBox1.Text = fields[1];
    textBox2.Text = fields[2];
    textBox3.Text = fields[3];
    textBox4.Text = fields[4];
    textBox5.Text = fields[5];
    textBox6.Text = fields[6];
    textBox7.Text = fields[7];
    textBox8.Text = fields[8];

    //textBox2.Text = tunnellist.innerTunnellist.idGenerator.NextID().ToString();
    //textBox8.Text = tunnellist.idGenerator.NextID().ToString();
    textBox2.Enabled = false;
    textBox8.Enabled = false;
}

/// <summary>
/// Редактирует данные из таблицы
/// </summary>
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        string[] fields = new string[]
        {
            "", textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text,
textBox5.Text, textBox6.Text,
            textBox7.Text, textBox8.Text
        };
        tunnellist.EditTunnel(fields, index);
        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Проверьте корректность введенных данных.", "Произошла
ошибка при отредактировании записи Тоннеля", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```
    }
}
```

IDGenerator.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TunnelReader
{
    /// <summary>
    /// Класс для генерации ID
    /// </summary>
    class IDGenerator
    {
        /// <summary>
        /// Максимальный ID
        /// </summary>
        private int maxId = 1;

        /// <summary>
        /// Добавляет ID
        /// </summary>
        public void AddID(int id)
        {
            if (id > maxId)
                maxId = id;
        }

        /// <summary>
        /// Возвращает следующий ID
        /// </summary>
        public int NextID()
        {
            return maxId++;
        }
    }
}
```

InnerTunnel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TunnelReader
{
    /// <summary>
    /// Внутренний тоннель
    /// </summary>
    class InnerTunnel
    {
        /// <summary>
        /// ID Внутреннего тоннеля
        /// </summary>
        public int global_id;
```

```

    /// <summary>
    /// Значение внутреннего тоннеля
    /// </summary>
    public string value;

    /// <summary>
    /// Конструктор
    /// </summary>
    public InnerTunnel(int global_id, string value)
    {
        this.global_id = global_id;
        this.value = value;
    }

    /// <summary>
    /// Декодирует данные из строки CSV
    /// </summary>
    public static void DecodeTunnelData(string encoded, out string strID, out string
strValue)
    {
        const string _id = "\"\"global_id\"\": ";
        const string _idDel = ", ";

        const string _value = "\"\"value\"\": \""";
        const string _valueDel = "\"\" }";

        int startForID = encoded.IndexOf(_id) + _id.Length;
        int endForID = encoded.IndexOf(_idDel);
        strID = encoded.Substring(startForID, endForID - startForID);

        int startForValue = encoded.IndexOf(_value) + _value.Length;
        int endForValue = encoded.IndexOf(_valueDel);
        strValue = encoded.Substring(startForValue, endForValue - startForValue);
    }

    /// <summary>
    /// Кодировать данные
    /// </summary>
    public static string EncodeTunnelData(string strID, string strValue)
    {
        return $"{{ \"\"global_id\"\": {strID}, \"\"value\"\": \"{strValue}\"\"
}}";
    }

    /// <summary>
    /// Кодировать данные
    /// </summary>
    public static string EncodeTunnelData(InnerTunnel innerTunnel)
    {
        return EncodeTunnelData(innerTunnel.global_id.ToString(), innerTunnel.value);
    }

    /// <summary>
    /// Переопределенный метод для сравнения по значению
    /// </summary>
    public override bool Equals(object obj)
    {
        InnerTunnel innerTunnel = obj as InnerTunnel;
        return (this.global_id == innerTunnel.global_id) && (this.value ==
innerTunnel.value);
    }
}

```

InnerTunnelList.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TunnelReader
{
    /// <summary>
    /// Класс для работы со списком Внутренних тоннелей
    /// </summary>
    class InnerTunnelList
    {
        /// <summary>
        /// Список внутренних тоннелей
        /// </summary>
        private readonly List<InnerTunnel> innerTunnels = new List<InnerTunnel>();
        /// <summary>
        /// Класс для генерирования ID
        /// </summary>
        public IDGenerator idGenerator = new IDGenerator();

        /// <summary>
        /// Возвращает внутренний тоннель с заданными значениями
        /// </summary>
        public InnerTunnel Get(int global_id, string value)
        {
            idGenerator.AddID(global_id);
            InnerTunnel innerTunnel = new InnerTunnel(global_id, value);
            InnerTunnel result = innerTunnels.Find((a) => a.Equals(innerTunnel));
            if (result != null)
                return result;
            else
            {
                innerTunnels.Add(innerTunnel);
                return innerTunnel;
            }
        }
    }
}

```

MainForm.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows.Forms;

namespace TunnelReader
{
    /// <summary>
    /// Главная форма программы
    /// </summary>
    public partial class MainForm : Form
    {
        /// <summary>
        /// Значение тоннелей до совершенного действия
        /// </summary>
    }
}

```

```

private List<Tunnel> oldTunnels = new List<Tunnel>();

/// <summary>
///     Список тоннелей
/// </summary>
private Tunnellist tunnellist;

/// <summary>
///     Конструктор по умолчанию
/// </summary>
public MainForm()
{
    InitializeComponent();
}

/// <summary>
///     Добавляет новое значение в таблицу
/// </summary>
private void addButton_Click(object sender, EventArgs e)
{
    if (tunnellist != null)
    {
        BackupResult();
        var addRowForm = new AddRowForm();
        addRowForm.tunnellist = tunnellist;
        addRowForm.SetupForm();
        addRowForm.ShowDialog();
        tunnellist.UpdateRowIndexes();
    }
}

/// <summary>
///     Показывает окно для последующего открытия файла
/// </summary>
private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
    var openForm = new OpenForm();
    openForm.ShowDialog();
    if (openForm.tunnellist != null)
    {
        tunnellist = openForm.tunnellist;
        dataGridView.DataSource = tunnellist.dataTable;

        sortFieldComboBox.ComboBox.Items.Clear();
        sortFieldComboBox.ComboBox.Items.Add(tunnellist.headings[1]);
        sortFieldComboBox.ComboBox.Items.Add(tunnellist.headings[3]);
        sortFieldComboBox.ComboBox.SelectedIndex = 0;

        filterFieldComboBox.ComboBox.Items.Clear();
        filterFieldComboBox.ComboBox.Items.Add(tunnellist.headings[2] + "
global_id");
        filterFieldComboBox.ComboBox.Items.Add(tunnellist.headings[3]);
        filterFieldComboBox.ComboBox.SelectedIndex = 0;

        foreach (DataGridViewColumn column in dataGridView.Columns)
            column.SortMode = DataGridViewColumnSortMode.NotSortable;
    }
}

/// <summary>
///     Пересохраняет данные
/// </summary>
private void saveToolStripMenuItem_Click(object sender, EventArgs e)
{
    try

```



```

        {
            tunnelList.SaveToFile(tunnelList.filePath);
        }
        catch (UnauthorizedAccessException ex)
        {
            MessageBox.Show("Невозможно сохранить файл так как он не доступен для
пользователя",
                "Произошла ошибка при сохранении файла.", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
        catch (IOException ex)
        {
            MessageBox.Show("Проверьте существует ли выбранный файл, используется ли
в других приложениях.",
                "Произошла ошибка при сохранении файла.", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Проверьте корректность файла.", "Произошла ошибка при
сохранении файла.",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    /// <summary>
    ///     Сохраняет данные в заданный файл
    /// </summary>
    private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        try
        {
            saveFileDialog.ShowDialog();
            if (saveFileDialog.FileName != "")
            {
                tunnelList.SaveToFile(saveFileDialog.FileName);
            }
        }
        catch (UnauthorizedAccessException ex)
        {
            MessageBox.Show("Невозможно сохранить файл так как он не доступен для
пользователя",
                "Произошла ошибка при сохранении файла.", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
        catch (IOException ex)
        {
            MessageBox.Show("Проверьте существует ли выбранный файл, используется ли
в других приложениях.",
                "Произошла ошибка при сохранении файла.", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Проверьте корректность файла.", "Произошла ошибка при
сохранении файла.",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    /// <summary>
    ///     Сортирует данные в алфавитном порядке
    /// </summary>
    private void sortAZButton_Click(object sender, EventArgs e)
    {
        if (tunnelList != null)
        {

```

```

        BackupResult();
        if (sortFieldComboBox.SelectedIndex == 0)
            tunnelList.tunnels.Sort((t1, t2) => t1.Name.CompareTo(t2.Name));
        else
            tunnelList.tunnels.Sort((t1, t2) =>
            {
                var res = t1.area.AdmArea.CompareTo(t2.area.AdmArea);
                if (res == 0 && t1.Name != t2.Name)
                    return t1.Name.CompareTo(t2.Name);
                return res;
            });

        tunnelList.UpdateTable();
    }
}

/// <summary>
///     Сортирует данные по количеству округов
/// </summary>
private void sortNumButton_Click(object sender, EventArgs e)
{
    if (tunnelList != null && tunnelList.tunnels.Count > 0)
    {
        BackupResult();
        var districtsDictionary = new Dictionary<string, int>();
        for (var i = 0; i < tunnelList.tunnels.Count; i++)
        {
            var data =
                districtsDictionary.ContainsKey(tunnelList.tunnels[i].area.AdmArea)
                    ? districtsDictionary[tunnelList.tunnels[i].area.AdmArea] + 1
                    : 1;
            districtsDictionary[tunnelList.tunnels[i].area.AdmArea] = data;
        }

        tunnelList.tunnels.Sort((t1, t2) =>
        {
            var res =
                districtsDictionary[t1.area.AdmArea].CompareTo(districtsDictionary[t2.area.AdmArea]);
            if (res == 0 && t1.area.AdmArea != t2.area.AdmArea)
                return t1.area.AdmArea.CompareTo(t2.area.AdmArea);
            return res;
        });
        tunnelList.UpdateTable();
    }
}

/// <summary>
///     Фильтрует данные
/// </summary>
private void filterButton_Click(object sender, EventArgs e)
{
    if (tunnelList != null && tunnelList.tunnels.Count > 0)
    {
        BackupResult();
        if (filterFieldComboBox.SelectedIndex == 0)
            tunnelList.tunnels = tunnelList.tunnels
                .Where(t =>
                    t.GetGlobalID().ToString().IndexOf(filterTextBox.Text) == 0)
                .ToList();
        else
            tunnelList.tunnels = tunnelList.tunnels
                .Where(t => t.area.AdmArea.IndexOf(filterTextBox.Text) == 0)
                .ToList();

        tunnelList.UpdateTable();
    }
}

```

```

    }
}

/// <summary>
///     Восстанавливает данные до совершенного действия
/// </summary>
private void undoButton_Click(object sender, EventArgs e)
{
    if (oldTunnels != null && oldTunnels.Count != 0)
    {
        tunnellist.tunnels = oldTunnels;
        tunnellist.UpdateTable();
        oldTunnels = null;
    }
}

/// <summary>
///     Удаляет значения из таблицы
/// </summary>
private void deleteButton_Click(object sender, EventArgs e)
{
    if (tunnellist != null && tunnellist.tunnels.Count > 0)
    {
        BackupResult();
        var index = dataGridView.SelectedRows[0].Index;
        tunnellist.DeleteTunnel(index);
    }
}

/// <summary>
///     Редактирует значение из таблицы
/// </summary>
private void editButton_Click(object sender, EventArgs e)
{
    if (tunnellist != null && tunnellist.tunnels.Count > 0)
    {
        BackupResult();
        var index = dataGridView.SelectedRows[0].Index;
        var editRowForm = new EditRowForm();
        editRowForm.tunnellist = tunnellist;
        editRowForm.index = index;
        editRowForm.SetupForm();
        editRowForm.ShowDialog();
        tunnellist.UpdateRowIndexes();
    }
}

/// <summary>
///     Сохраняет список тоннелей для последующего восстановления
/// </summary>
public void BackupResult()
{
    oldTunnels = new List<Tunnel>(tunnellist.tunnels);
}

/// <summary>
///     Завершает программу
/// </summary>
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}
}

```

```
}
```

OpenForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TunnelReader
{
    /// <summary>
    /// Форма для открытия файла
    /// </summary>
    public partial class OpenForm : Form
    {
        /// <summary>
        /// Список тоннелей
        /// </summary>
        internal TunnelList tunnelList;
        /// <summary>
        /// Конструктор по умолчанию
        /// </summary>
        public OpenForm()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Проверяет является ли строка числом
        /// </summary>
        private bool IsValidNumber(string str)
        {
            int id;
            return int.TryParse(str, out id) && id > 0;
        }

        /// <summary>
        /// Возвращает путь к выбранному файлу
        /// </summary>
        private void browseButton_Click(object sender, EventArgs e)
        {
            openFileDialog.ShowDialog();
            pathTextBox.Text = openFileDialog.FileName;
            openButton.Enabled = true;
        }

        /// <summary>
        /// Возвращает количество строк файла
        /// </summary>
        private int GetLinesCount(string pathToFile)
        {
            int count = 0;
            FileStream fileStream = new FileStream(pathToFile, FileMode.Open);
            using (StreamReader streamReader = new StreamReader(fileStream))
            {
```

```

        while (streamReader.ReadLine() != null)
            count++;
        streamReader.Close();
    }
    return count;
}

/// <summary>
/// Открывает выбранный файл
/// </summary>
private void openButton_Click(object sender, EventArgs e)
{
    try
    {
        if (!IsValidNumber(indexTextBox.Text))
            throw new ArgumentException("Неправильно введен начальный индекс.");
        if (!IsValidNumber(countTextBox.Text) ||
            int.Parse(countTextBox.Text) > GetLinesCount(pathTextBox.Text) -
            int.Parse(indexTextBox.Text) + 1 + ((headingsCheckBox.Checked) ? -1 : 0))
            throw new ArgumentException("Неправильно введено количество
            отображаемых элементов.");

        tunnellist = TunnelList.NewFromFile(pathTextBox.Text,
            headingsCheckBox.Checked ? CSVType.WithHeadings :
            CSVType.WithoutHeading,
            int.Parse(indexTextBox.Text), int.Parse(countTextBox.Text));
        this.Close();
    }
    catch (UnauthorizedAccessException ex)
    {
        MessageBox.Show("Невозможно открыть файл так как он не допущен для
        пользователя", "Произошла ошибка при открытии файла.", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
    catch (IOException ex)
    {
        MessageBox.Show("Проверьте существует ли выбранный файл, используется ли
        в других приложениях.", "Произошла ошибка при открытии файла.", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Проверьте корректность файла.", "Произошла ошибка при
        открытии файла.", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}
}

```

Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TunnelReader
{
    /// <summary>

```

```

/// Начальная точка программы
/// </summary>
static class Program
{
    /// <summary>
    /// Начальная точка приложения
    /// </summary>
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new MainForm());
    }
}

```

Tunnel.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms.VisualStyles;

namespace TunnelReader
{
    /// <summary>
    /// Тоннель
    /// </summary>
    class Tunnel
    {
        /// <summary>
        /// Долгота
        /// </summary>
        /// <value>Долгота</value>
        public double Longitude { get; set; }
        /// <summary>
        /// Широта
        /// </summary>
        /// <value>Широта</value>
        public double Latitude { get; set; }
        /// <summary>
        /// ID тоннеля
        /// </summary>
        public int ID { get; set; }
        /// <summary>
        /// Внутренний тоннель
        /// </summary>
        public InnerTunnel innerTunnel;
        /// <summary>
        /// Имя тоннеля
        /// </summary>
        /// <value>Имя тоннеля</value>
        public string Name { get; set; }
        /// <summary>
        /// Район тоннеля
        /// </summary>
        public Area area;
    }
}

```

```

        /// <summary>
        /// Возвращает ID внутреннего тоннеля
        /// </summary>
        public int GetGlobalID()
        {
            return innerTunnel.global_id;
        }
    }
}

```

TunnelList.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Windows.Forms;

namespace TunnelReader
{
    /// <summary>
    /// Класс для работы с данными
    /// </summary>
    internal class TunnelList
    {
        /// <summary>
        /// Список тоннелей
        /// </summary>
        public List<Tunnel> tunnels;
        /// <summary>
        /// Тип Файла
        /// </summary>
        public CSVType fileType;
        /// <summary>
        /// Список Районов
        /// </summary>
        public AreaList areaList;

        /// <summary>
        /// Список внутренних тоннелей
        /// </summary>
        public InnerTunnelList innerTunnelList;
        /// <summary>
        /// Таблица данных
        /// </summary>
        public DataTable dataTable;
        /// <summary>
        /// Число столбцов в таблице
        /// </summary>
        private const int fieldsCount = 9;
        /// <summary>
        /// Заголовки
        /// </summary>
        public string[] headings;
        /// <summary>
        /// Путь к файлу
        /// </summary>
        public string filePath;
    }
}

```

```

    /// <summary>
    /// Класс для генерации ID
    /// </summary>
    public IDGenerator idGenerator;

    /// <summary>
    /// Создает новый объект из файла
    /// </summary>
    public static TunnellList NewFromFile(string pathToFile, CSVType fileType, int
from, int count)
    {
        TunnellList tunnellList = new TunnellList();
        tunnellList.areaList = new AreaList();
        tunnellList.innerTunnellList = new InnerTunnellList();
        tunnellList.tunnels = new List<Tunnel>();
        tunnellList.filePath = pathToFile;
        tunnellList.idGenerator = new IDGenerator();
        tunnellList.fileType = fileType;

        var fileStream = new FileStream(pathToFile, FileMode.Open);
        using (var streamReader = new StreamReader(fileStream, Encoding.UTF8))
        {
            if (CSVType.WithHeadings == fileType)
                tunnellList.headings = streamReader.ReadLine()
                    .Split(new char[] { ';' },
StringSplitOptions.RemoveEmptyEntries);
            else
                tunnellList.headings = new string[]
                { "Index", "Name", "Tunnel", "Admin. Area", "District",
"Longitude", "Latitude", "ID" };
            tunnellList.SetupDatatable();

            string line;
            Tunnel tunnel;
            for (var index = 1; index < from; index++)
                streamReader.ReadLine();
            for (var index = 0; index != count && (line = streamReader.ReadLine()) !=
null && line != ""; index++)
            {
                tunnel = tunnellList.GetTunnel(line);
                tunnellList.tunnels.Add(tunnel);
                tunnellList.idGenerator.AddID(tunnel.ID);
                tunnellList.dataTable.Rows.Add(tunnellList.GetFields(tunnel));
            }

            tunnellList.UpdateRowIndex();
        }

        return tunnellList;
    }

    /// <summary>
    /// Сохраняет объект в файл
    /// </summary>
    public void SaveToFile(string filePath)
    {
        var fileStream = new FileStream(filePath, FileMode.Create);
        using (var streamWriter = new StreamWriter(fileStream))
        {
            streamWriter.WriteLine(GetHeadings());
            for (var i = 0; i < tunnels.Count; i++)
                streamWriter.WriteLine(GetCSVLine(tunnels[i], i + 1));
            streamWriter.Flush();
        }
    }

```



```

    }

    /// <summary>
    /// Добавляет тоннель к списку
    /// </summary>
    public void AddTunnel(string[] fields)
    {
        var tunnel = GetTunnel(fields);
        dataTable.Rows.Add(fields);
        tunnels.Add(tunnel);
    }

    /// <summary>
    /// Редактирует заданный тоннель
    /// </summary>
    public void EditTunnel(string[] fields, int index)
    {
        var tunnel = GetTunnel(fields);
        dataTable.Rows[index].ItemArray = fields;
        tunnels[index] = tunnel;
    }

    /// <summary>
    /// Удаляет заданный тоннель
    /// </summary>
    public void DeleteTunnel(int index)
    {
        tunnels.RemoveAt(index);
        dataTable.Rows.RemoveAt(index);
        UpdateRowIndexes();
    }

    /// <summary>
    /// Возвращает тоннель с заданными данными
    /// </summary>
    public Tunnel GetTunnel(string csvLine)
    {
        var fields = csvLine.Split(new char[] { ';' },
StringSplitOptions.RemoveEmptyEntries).ToList();
        for (var i = 0; i < fields.Count; i++) fields[i] = fields[i].Trim(' ');

        string strID, strValue;

        InnerTunnel.DecodeTunnelData(fields[2], out strID, out strValue);
        fields.Insert(2, strID);
        fields[3] = strValue;

        return GetTunnel(fields.ToArray());
    }

    /// <summary>
    /// Возвращает тоннель с заданными данными
    /// </summary>
    public Tunnel GetTunnel(string[] fields)
    {
        for (var i = 1; i < fields.Length; i++)
            if (fields[i] == "")
                throw new ArgumentException("Некоторые из данных пусты!!!");

        var tunnel = new Tunnel();
        tunnel.Name = fields[1];

        tunnel.Latitude = double.Parse(fields[6]);
        tunnel.Longitude = double.Parse(fields[7]);
        tunnel.ID = int.Parse(fields[8]);
    }

```

```

        tunnel.area = arealist.Get(fields[4], fields[5]);
        tunnel.innerTunnel = innerTunnelList.Get(int.Parse(fields[2]), fields[3]);
        return tunnel;
    }

    /// <summary>
    /// Возвращает значения для столбцов
    /// </summary>
    public string[] GetFields(Tunnel tunnel)
    {
        return new[]
        {
            "", tunnel.Name, tunnel.innerTunnel.global_id.ToString(),
            tunnel.innerTunnel.value, tunnel.area.AdmArea,
            tunnel.area.District, tunnel.Longitude.ToString(),
            tunnel.Latitude.ToString(), tunnel.ID.ToString()
        };
    }

    /// <summary>
    /// Обновляет индексы строк
    /// </summary>
    public void UpdateRowIndexes()
    {
        for (var i = 0; i < dataTable.Rows.Count; i++)
            dataTable.Rows[i][0] = (i + 1).ToString();
    }

    /// <summary>
    /// Инициализирует таблицу
    /// </summary>
    public void SetupDatatable()
    {
        dataTable = new DataTable();
        dataTable.Columns.Add(headings[0]);
        dataTable.Columns.Add(headings[1]);
        dataTable.Columns.Add(headings[2] + " global_id");
        dataTable.Columns.Add(headings[2] + " value");
        dataTable.Columns.Add(headings[3]);
        dataTable.Columns.Add(headings[4]);
        dataTable.Columns.Add(headings[5]);
        dataTable.Columns.Add(headings[6]);
        dataTable.Columns.Add(headings[7]);
    }

    /// <summary>
    /// Возвращает заголовки
    /// </summary>
    public string GetHeadings()
    {
        return string.Join(";", headings);
    }

    /// <summary>
    /// Возвращает строку данных CSV
    /// </summary>
    public string GetCSVLine(Tunnel tunnel, int index)
    {
        var fields = new List<string>();
        fields.Add(index.ToString());
        fields.Add(tunnel.Name);
        fields.Add(InnerTunnel.EncodeTunnelData(tunnel.innerTunnel));
        fields.Add(tunnel.area.AdmArea);

```

```

        fields.Add(tunnel.area.District);
        fields.Add(tunnel.Longitude.ToString());
        fields.Add(tunnel.Latitude.ToString());
        fields.Add(tunnel.ID.ToString());

        for (var i = 0; i < fields.Count; i++) fields[i] = "\"" + fields[i] + "\"";
        return string.Join(";", fields);
    }

    /// <summary>
    /// Обновляет таблицу
    /// </summary>
    public void UpdateTable()
    {
        dataTable.Rows.Clear();
        idGenerator = new IDGenerator();
        for (var i = 0; i < tunnels.Count; i++)
        {
            dataTable.Rows.Add(GetFields(tunnels[i]));
            idGenerator.AddID(tunnels[i].ID);
        }

        UpdateRowIndexes();
    }
}

/// <summary>
/// Показывает нужно ли окрыть файл CSV с заголовкой или без
/// </summary>
internal enum CSVType
{
    WithHeadings,
    WithoutHeading
}
}

```

7. Список литературы

В ходе написания программы не было использовано литературы.