

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

**Факультет компьютерных наук**

Департамент

**Программной инженерии**

*Контрольное домашнее задание  
по дисциплине  
«Программирование»*

Тема работы:
--------------

Выполнил: студент группы БПИ183(2)  
\_\_\_\_\_ Герасименко Е.Р.

тел. \_\_\_\_\_  
e-mail адрес: ergerasimenko@edu.hse.ru

Преподаватель: \_\_\_\_\_

Москва, 2019 год. Модуль 3

## Содержание

1. Условие задачи .....	3
2. Функции разрабатываемого приложения.....	6
2.1. Варианты использования .....	6
2.2. Описание интерфейса пользователя .....	6
3. Структура приложения .....	8
4. Распределение исходного кода по файлам проекта .....	14
5. Описание интерфейса .....	15
6. Контрольный пример и описание результата .....	17
7. Сообщения пользователю.....	29
8. Текст (код) программы .....	31
9. Список литературы.....	138

## 1. Условие задачи

Общие для всех вариантов КДЗ функциональные требования:

1. Приложение в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие стандартные операции со списком объектов:
  - 1.1. Создание списка путем ввода данных пользователем
  - 1.2. добавление в список нового объекта
  - 1.3. удаление объекта из списка
  - 1.4. вывод сведений об объектах заданного пользователем типа из списка на форму
  - 1.5. сохранение списка объектов в указанном пользователем текстовом файле в режиме:
  - 1.6. сохранения в файле текущего списка объектов
  - 1.7. добавления в файл текущего списка объектов
  - 1.8. построение списка объектов по данным, прочитанным из файла
2. Список объектов может быть реализован в виде массива или какой-либо коллекции, например, списка. Количество создаваемых объектов заранее не известно. Порядок следования объектов разных типов (если такие есть) при создании списка произвольный и определяется пользователем.
3. Приложение обязательно должно корректно открывать и позволять модифицировать, созданные с его помощью файлы с данными.
4. Исходные файлы с данными могут располагаться в любой папке компьютера.
5. Приложение должно корректно работать с файлами, в пути к которым содержатся символы национальных алфавитов, знаки препинания, # и проч., допустимые для путей к файлам в ОС Windows символы.

Общие требования к интерфейсу программы

1. Приложение должно представлять собой оконное *Windows* приложение.

2. При добавлении данных в ИСС исходные данные пользователь вводит с помощью экранных форм, содержащих поля для текстового ввода или списки значений.
3. При отсутствии явных ограничений в варианте задания, результаты отображаются с помощью экранных форм для вывода текста или элементов типа «сетка данных».
4. Сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных и других нештатных ситуациях отображать в всплывающих окнах типа окон сообщений.
5. В приложении в обязательно *должны быть реализованы следующие элементы управления*: меню, инструментальная панель, всплывающие подсказки.
6. Остальные элементы управления используются по усмотрению разработчика.

#### Ограничения

1. При выполнении КДЗ требуется соблюдать корпоративные стандарты НИУ ВШЭ. В дисциплине «Программирование» стандартом в текущем учебном году является применение *Microsoft VS 2017* (не выше).
2. В программной реализации не использовать вспомогательные компоненты и сторонние библиотеки, не входящие в стандартную библиотеку.
3. Не использовать массивы типа **object[]**.
4. Не применять СУБД и базы данных.

#### 1. Требования к основным классам приложения

1.1. Основная информация о Пунктах охраны правопорядка (ОПОП) хранится в объектах класса **ОПОП**. Набор полей класса задаётся полями файла **Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv**, кроме полей, содержащих информацию об административном округе и районе, представленных полем типа **Расположение**. Один из методов класса возвращает набор **ОПОП**, имеющих тот же **OPOPNumber**, что и у данного объекта. Класс **ОПОП** находится в *отношении агрегации* с классом **Расположение**.

1.2. Класс **Расположение** представляет информацию об административном расположении ОПОП, заданном полями CSV-файла: **AdmArea, District**.

1.3. Дополнительные классы, необходимые для решения задачи (объявляет автор программы).

Исполнитель – Герасименко Е. Р. Вариант 2

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 **ОПОП**, **Расположение** (если объект **Расположение** с данными об определённом округе и районе существует, то он является общим для всех объектов **ОПОП** этого округа и района) и др.

2.3. Отобразить данные из объектов в оконной форме.

2.4. Создать новую запись об ОПОП.

2.5. Удалить уже существующую запись об ОПОП.

2.6. Отредактировать существующую запись об ОПОП.

2.7. Отсортировать данные по полям: **OPOPnum**, **GLOBALID**

2.8. Отфильтровать данные по полям: **District**, **AdmArea**. Данные для фильтрации вводятся пользователем.

2.9. Выводить количество и список **ОПОП**, имеющих тот же номер, что и выбранный в списке **ОПОП**.

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**. Количество отображаемых в сетке элементов ( $N$ ) выбирается пользователем,  $N > 1$  и не превышает количества записей в файле **Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv**

4. Требования к устойчивости приложения

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

Исполнитель – Герасименко Е. Р. Вариант 2

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

## **2. Функции разрабатываемого приложения**

### **2.1. Варианты использования**

Программа предназначена для использования на компьютере, содержащем пакеты, необходимые для использования C# и .NET, в образовательных, развлекательных и научных целях требующих открытия редактирования и сохранения файлов формата “.csv”; “.tsv”; “.scsv”.

### **2.2. Описание интерфейса пользователя**

Интерфейс представляет из себя оконный интерфейс Windows Forms и содержит:

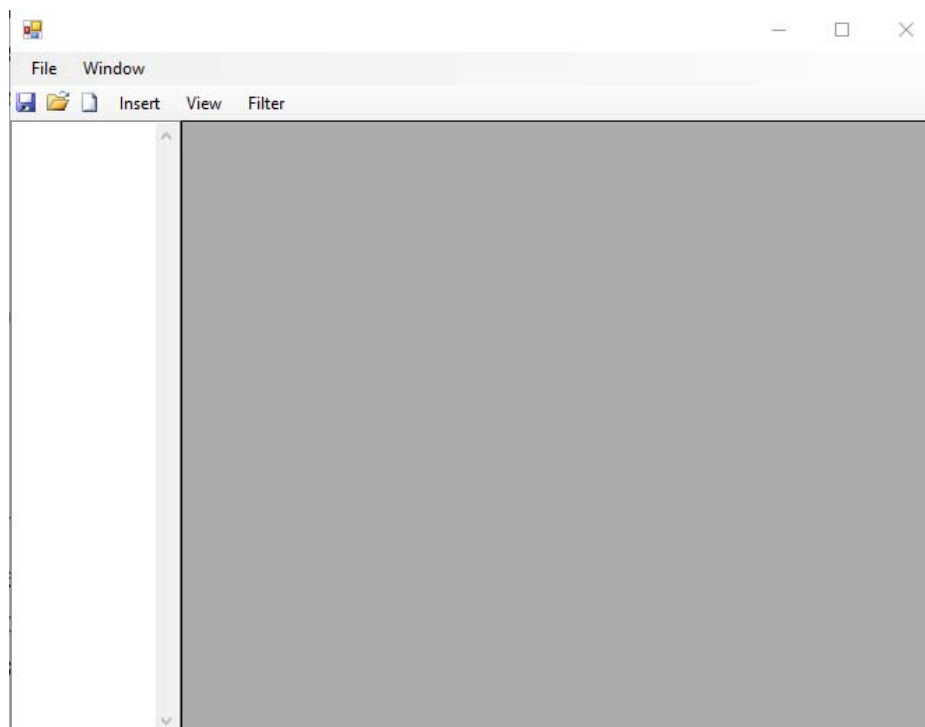


Рисунок 1 - Интерфейс программы

- Меню
  - Элемент меню file для открытия и сохранения файлов
    - New - открывается новое окно фрактала, данное окно становится неактивным
    - Override? – перезаписать или дополнить (галочка – перезаписать)
    - Save – сохранить - программа сохранит под указанным ранее именем, если имя не было у казано ранее – вызовется Save As (\*.csv)
    - Save As – сохранить как - программа попросит выбрать путь и ввести имя
      - Save As (\*.csv) – с выбором параметров файла
      - Save As for Excel (\*.csv) – с параметрами для открытия в Excel
      - Save Dialog – дополнительно выбрать кодировку файла
    - Load – Загрузить данные из файла (выводит диалог выбора файла)
  - Элемент меню Window для закрепления окна поверх остальных окон
    - Over all windows – показывать окно поверх остальных (если галочка – да)
- Панель инструментов
  - Кнопку для быстрого сохранения
  - Кнопку для быстрой загрузки
  - Кнопку для создания новой таблицы
  - Элемент вставка
    - Вставить строку
    - Вставить столбец
  - Элемент вид
    - Выбрать начальный элемент
    - Выбрать конечный элемент
  - Элемент фильтр
    - Фильтровать по столбцу
    - Фильтровать по номеру выбранного ОПОП
- Контекстное меню
  - Открыть с другими параметрами файла
  - Найти во всех ячейках
- Поле для вывода результатов поиска

При возможной потере несохранённых изменений на экран выводится окно WARNING. Данное меню содержит:

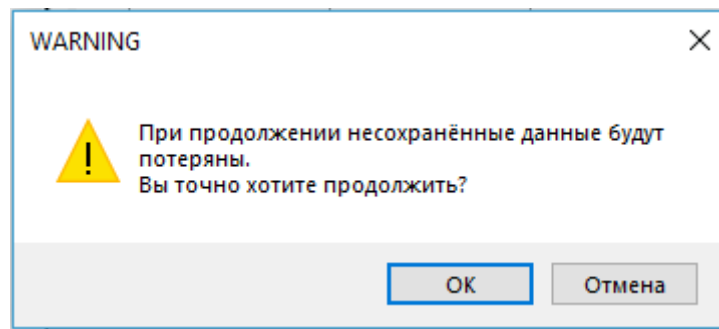


Рисунок 2 - Интерфейс окна WARNING

- Текст «При продолжении несохранённые данные будут потеряны. Вы точно хотите продолжить?»
- Кнопки:
  - Ок – продолжить
  - Отмена – отменить операцию

### 3. Структура приложения

Класс:

- AhoCorasik

Подклассы и подструктуры:

- vertex

Поля:

- next – массив следующих
- leaf - лист
- leafs – строка листа
- p - предок
- pch – символ предка
- link – ссылка на максимального следующего в другой подстроке
- go – массив следующих в других подстроках

Поля:

- t - используется для хранения информации об узлах дерева
- sz - используется для хранения количества узлов

Свойства:

Методы:



- Конструкторы – задаёт дерево поиска:
  - which = подстроки;
  - nmax = 1000; - максимальное количество узлов
- findInString(s, w)- функция поиска строки s в массиве строк w
- find - функция поиска подстрок в строке на основе построенного дерева
- CSVconv
  - Методы:
    - fscanf – считывает данные из файла
    - fprintf – пишет данные в файл
    - LoadCSVtoStr - функция загрузки данных из .csv файла
    - SaveStrtoCSV – функция записи данных в .csv файл
    - ConvertCSVlinetoListstr - функция преобразования строки формата .csv в массив строк
    - ConvertListstrtoCSVline - функция преобразования массива строк в строку формата .csv
  - ОПОП
    - Поля:
      - \_ROWNUM – номер строки
      - \_OPOPNumber – номер ОПОП
      - \_adress – адрес
      - \_PublicPhone – телефон
      - \_GLOBALID – глобальный номер ОПОП
      - Name – имя
      - Address – адрес
      - ExtraInfo – дополнительная информация
      - X\_WGS –
      - Y\_WGS -
    - Свойства:
      - Adress – получить адрес area и район
      - ROWNUM
      - OPOPNumber
      - AdmArea
      - District
      - PublicPhone
      - GLOBALID

- adress

Итераторы:

- Итератор – доступ как к элементам таблицы в соответствии с нумерацией столбцов:

Методы:

- Конструкторы – задаёт ОПОП:

- Расположение

Поля:

- AdmArea – адрес area
- District – адрес район

- Phone

Поля:

- number - номер

Методы:

- get – возвращает номер
- set – строит номер по входной строке
- parse – удаляет все спецсимволы из номера

- Data

Поля:

- flagmb – системный флаг
- rewrite – перезаписать
- issaved – всё сохранено
- isaded – происходит добавление
- sne – старт номер фильтр
- sn – первая строка
- ene – энд номер фильтр
- en – последняя строка
- name – имя окна
- data – что лежит в таблице
- datas – что лежало в файле
- separ – разделитель в .csv файле
- encode - кодировка

- Find – debug interface

- Form1

Поля:

- IS\_AUTO\_UPDATE\_ES – автообновление данных при выходе из контекстного меню
- d – методанные
- opor - список ОПОП
- adr – список адресов

Свойства:

- flagmb – системный флаг
- rewrite – перезаписать
- issaved – всё сохранено
- isaded – происходит добавление
- sne – старт номер фильтр
- sn – первая строка
- ene – энд номер фильтр
- en – последняя строка
- name – имя окна
- data – что лежит в таблице
- datas – что лежало в файле
- separ – разделитель в .csv файле
- encode - кодировка

Методы:

- Конструкторы – создаёт окно
- Form1\_KeyDown – отлов нажатий клавиш
- DropExWindow – вывод сообщений об ошибке
- toolStripMenuItem1\_Click – перезаписать или дозаписать
- saveToolStripMenuItem1\_Click – сохранить таблицу
- saveDialogToolStripMenuItem\_Click - Сохранение таблицы как (старый интерфейс)
- saveAstxtToolStripMenuItem\_Click - Сохранение таблицы как текстовый файл (новый интерфейс)
- saveAsToolStripMenuItem\_Click - Сохранение таблицы как (новый интерфейс)
- saveToolStripButton\_Click – сохранение таблицы
- loadToolStripMenuItem\_Click – загрузка таблицы
- loadToolStripButton\_Click – загрузка таблицы
- ReopenToolStripMenuItem\_Click – загрузить в другой кодировке
- openToolStripButton\_Click – загрузка таблицы

- newToolStripMenuItem\_Click – создать новую таблицу
- newToolStripButton\_Click – создать новую таблицу
- rowToolStripMenuItem\_Click – добавить строку
- dataGridView1\_RowsAdded – добавление строки
- removeRowInData – удаление строки
- dataGridView1\_DeleteRow – удаление строки
- columnToolStripMenuItem\_Click – добавление столбца
- editCellData – изменение значения ячейки
- FindSubstring – создание дерева поиска
- findInString – поиск полных совпадений в строке
- deleteCopy – удаление парных вхождения
- findToolStripMenuItem1\_Click – поиск подстрок в строке
- UpdateData – обновление информации об ОПОП-ах
- dataGridView1\_SortCompare – сортировка между ячейками
- filterToolStripMenuItem\_Click – начало фильтрации
- toolStripMenuItem5\_Click – смена значения фильтрации
- toolStripButton1\_Click – фильтрация по номеру выбранного ОПОП
- Form1\_Load – загрузка формы
- overAllWindowsToolStripMenuItem\_Click – поверх остальных окон
- UpdateGrid – обновление таблицы
- Form1Closed – закрытие формы
- SaveorLose – сохранить с потерей данных?
- OnFormClosing – закрытие формы
- OnPaint – перерисовка окна
- ContextMenuItem\_Closing – закрытие контекстного меню
- fromToolStripMenuItem\_Click – обновление номера начальной колонки
- toToolStripMenuItem\_Click – обновление номера конечной колонки

- SaveFileDialogWithEncoding

Подклассы и подструктуры:

- OPENFILENAME

Поля:

- lStructSize
- hwndOwner
- hInstance
- lpstrFilter

- lpstrCustomFilter
- nMaxCustFilter
- nFilterIndex
- lpstrFile
- nMaxFile
- lpstrFileTitle
- nMaxFileTitle
- lpstrInitialDir
- lpstrTitle
- Flags
- nFileOffset
- nFileExtension
- lpstrDefExt
- lCustData
- lpfnHook
- lpTemplateName
- pvReserved
- dwReserved
- FlagsEx
- RECT
  - Поля:
  - Left
  - Top
  - Right
  - Bottom
- POINT
  - Поля:
  - X
  - Y
- NMHDR
  - Поля:
  - HwndFrom
  - IdFrom
  - Code

Поля:

- Для передачи значений из окна в переопределённое окно

Методы:

- HookProc – переопределение вывода диалогового окна и добавление к нему нового списка для выбора кодировки

- pair

Поля:

- first – значение 1
- second – значение 2

Методы:

- Конструкторы – задаёт значения 1 и 2:
- makepair - функция создания пары из 2-х элементов
- setfirst - функция установления значения 1-му элементу
- setsecond – функция установления значения 2-му элементу

- vector

Поля:

- e – элементы
- length – количество элементов

Методы:

- Конструкторы – задаёт значения 1 и 2:
- fill - функция заполнить элементами списка
- size – получить длину вектора
- iterator – функция доступа к i-му элементу
- append – добавить элемент
- remove – удалить последний
- reverse – развернуть
- toString – строка из элементов
- toarray – получить массив

#### 4. Распределение исходного кода по файлам проекта

- AhoKorasik.cs – Класс поиска подстроки в строке
- CSVconv.cs – Класс считывания и записи файлов \*.csv
- Data.cs – Классы ОПОП и Расположение и Phone и Data
- SaveFileDialog.cs – Класс переопределённого окна сохранения файла

- Form1.cs – Класс Form1 (Интерфейс основного окна приложения)
- Program.cs – Класс Program (Точка запуска приложения)
- Find.cs – debug Класс
- STL.cs – Классы vector и pair

## 5. Описание интерфейса

Состоит из 2-х частей:

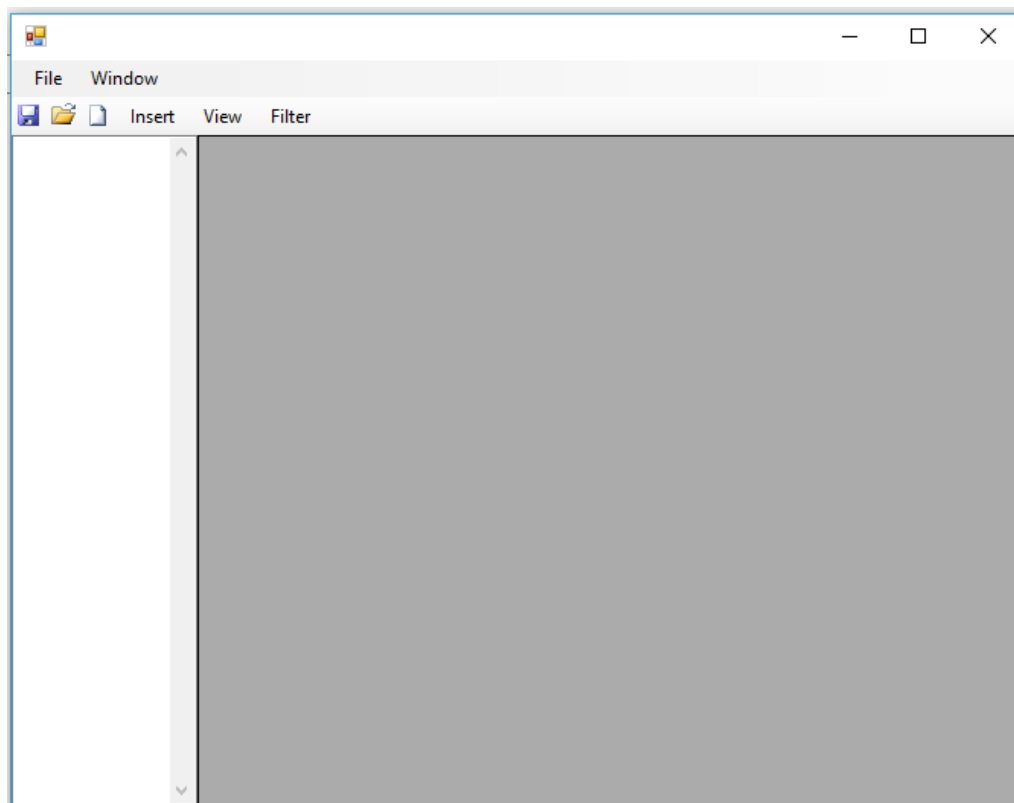
- Окно предупреждения (предупреждает о потере данных)
  - Да для продолжения
  - Отмена для прерывания операции
- Форма:
  - Меню:
    - File:
      - Save As – вывод диалога “сохранить как” (Save As (\*.csv)) (аналог – Ctrl+Shift+S)
        - Save As (\*.csv) – выбрать директорию и имя и сохранить таблицу
        - Save As for Excel (\*.csv) – выбрать директорию и имя и сохранить таблицу
        - Save Dialog – выбрать директорию и имя и сохранить таблицу
        - Save – сохранить таблицу по последнему пути, если пути нет, то вызвать Save As (аналог – Ctrl+S)
      - New – создать новую таблицу (аналог – Ctrl+N)
      - Load – загрузить таблицу (аналог – Ctrl+O)
      - Override? – перезаписывать при сохранении
    - Window:
      - Over all windows – Поверх всех окон (есть галочка – да, нет – нет)
  - Инструменты
    - Сохранить – Save (аналог – Ctrl+S)
    - Загрузить – Load (аналог – Ctrl+L)
    - Новый – New (аналог – Ctrl+N)

- Insert
  - Row – вставить строку перед выбранной (иначе в конце)
  - Column - вставить столбец с названием (следующее поле “Column name”)
- View
  - From – начинать выводить со строки (следующее поле для ввода строки)
  - To – выводить до строки (следующее поле для ввода строки)
- Filter
  - Filtering – производить фильтрацию (галочка)
  - Where – следующая строка для ввода названия колонки
  - What – следующая строка для ввода слов для фильтрации (различные слова для фильтрации вводить через ; )
  - Filter - фильтровать
  - Filter Selected OPOPNumber – фильтровать по OPOPNumber выбранной ОПОП
- Контекстное меню
  - Reopen with encoding – открыть снова в новой кодировке и с новым типом разделителя
    - Encoding Type – тип кодировки
    - Separator Type – тип разделителя
    - Reopen – открыть с новыми параметрами
  - Find – искать
    - Text here (separator = ‘;’) – в следующей строке (по умолчанию “s;se;sep”) вписать слова для поиска через ;
    - Find – искать
- Текстовое поле слева – поле результатов поиска
- Поле таблицы – место для отображения данных из файла \*.scv (место вызова Контекстного меню)

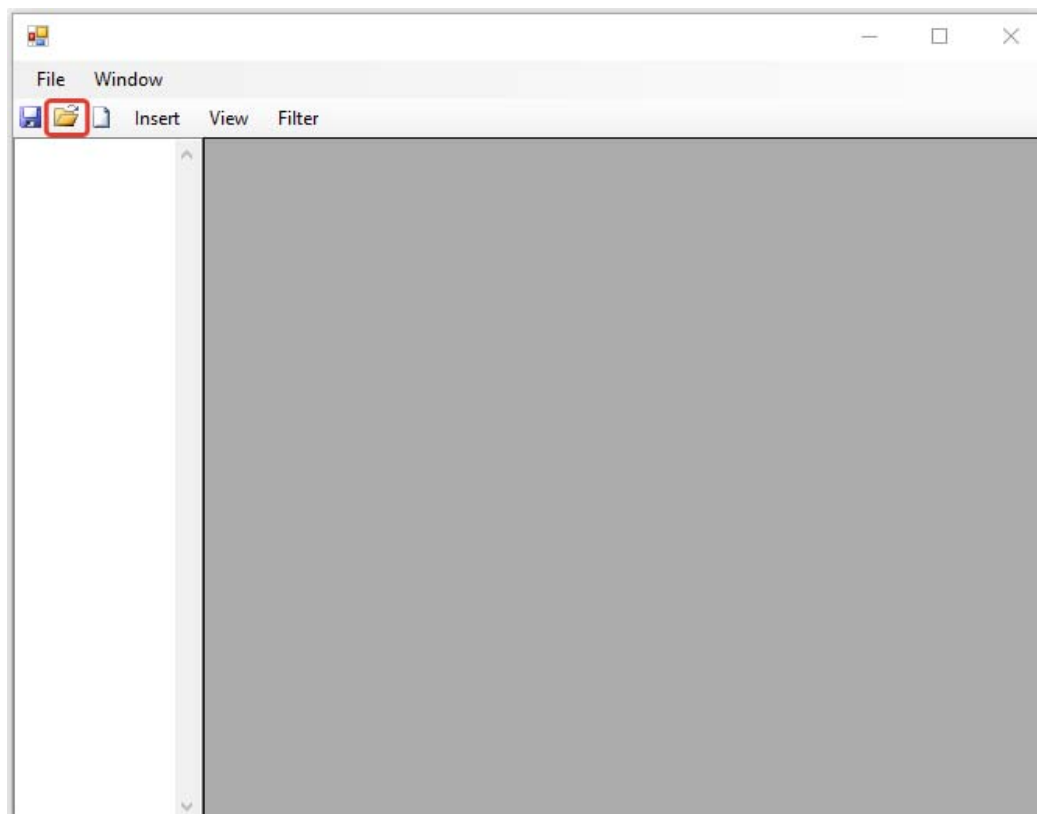


## 6. Контрольный пример и описание результата

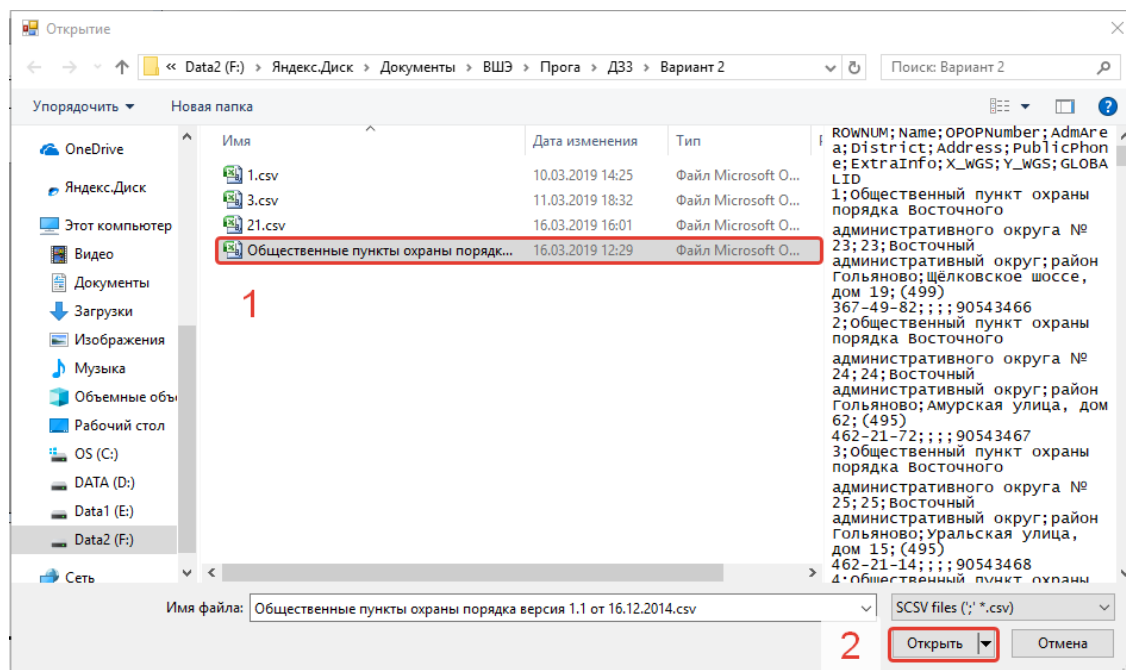
1) Запустить программу



2) Нажать на значок папки



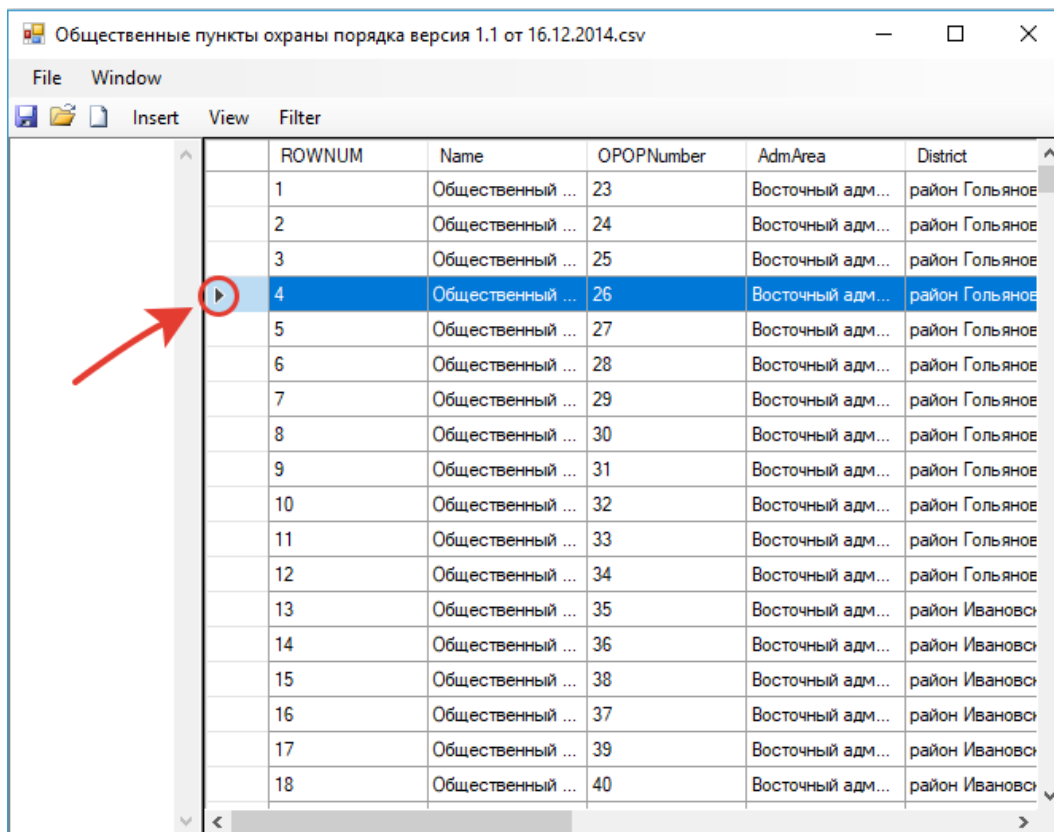
3) Выбрать файл и нажать открыть



#### 4) Увидеть таблицу

Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv					
File Window					
Insert View Filter					
	ROWNUM	Name	OPOPNumber	AdmArea	District
1	1	Общественный ...	23	Восточный адм...	район Гольяное
2	2	Общественный ...	24	Восточный адм...	район Гольяное
3	3	Общественный ...	25	Восточный адм...	район Гольяное
4	4	Общественный ...	26	Восточный адм...	район Гольяное
5	5	Общественный ...	27	Восточный адм...	район Гольяное
6	6	Общественный ...	28	Восточный адм...	район Гольяное
7	7	Общественный ...	29	Восточный адм...	район Гольяное
8	8	Общественный ...	30	Восточный адм...	район Гольяное
9	9	Общественный ...	31	Восточный адм...	район Гольяное
10	10	Общественный ...	32	Восточный адм...	район Гольяное
11	11	Общественный ...	33	Восточный адм...	район Гольяное
12	12	Общественный ...	34	Восточный адм...	район Гольяное
13	13	Общественный ...	35	Восточный адм...	район Ивановс
14	14	Общественный ...	36	Восточный адм...	район Ивановс
15	15	Общественный ...	38	Восточный адм...	район Ивановс
16	16	Общественный ...	37	Восточный адм...	район Ивановс
17	17	Общественный ...	39	Восточный адм...	район Ивановс
18	18	Общественный ...	40	Восточный адм...	район Ивановс

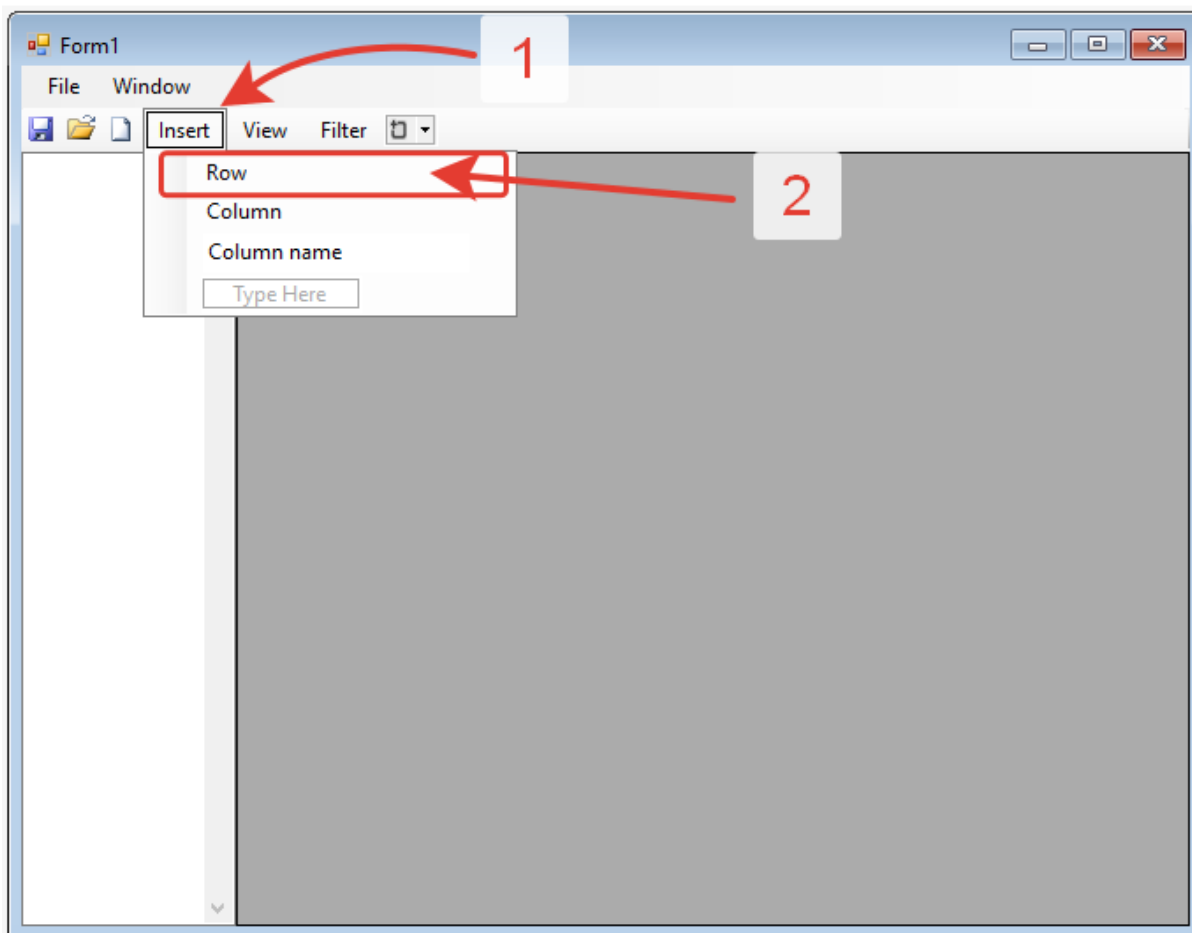
#### 5) Выбрать строку



Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

	ROWNUM	Name	OPOPNumber	AdmArea	District
	1	Общественный ...	23	Восточный адм...	район Гольяное
	2	Общественный ...	24	Восточный адм...	район Гольяное
	3	Общественный ...	25	Восточный адм...	район Гольяное
	4	Общественный ...	26	Восточный адм...	район Гольяное
	5	Общественный ...	27	Восточный адм...	район Гольяное
	6	Общественный ...	28	Восточный адм...	район Гольяное
	7	Общественный ...	29	Восточный адм...	район Гольяное
	8	Общественный ...	30	Восточный адм...	район Гольяное
	9	Общественный ...	31	Восточный адм...	район Гольяное
	10	Общественный ...	32	Восточный адм...	район Гольяное
	11	Общественный ...	33	Восточный адм...	район Гольяное
	12	Общественный ...	34	Восточный адм...	район Гольяное
	13	Общественный ...	35	Восточный адм...	район Ивановс
	14	Общественный ...	36	Восточный адм...	район Ивановс
	15	Общественный ...	38	Восточный адм...	район Ивановс
	16	Общественный ...	37	Восточный адм...	район Ивановс
	17	Общественный ...	39	Восточный адм...	район Ивановс
	18	Общественный ...	40	Восточный адм...	район Ивановс

6) Нажать на Insert и в выпавшем списке выбрать Row и нажать



7) Увидеть добавленную строку ОПОП

Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

File Window

Insert View Filter

	ROWNUM	Name	OPOPNumber	AdmArea	District
	1	Общественный ...	23	Восточный адм...	район Гольяное
	2	Общественный ...	24	Восточный адм...	район Гольяное
	3	Общественный ...	25	Восточный адм...	район Гольяное
	4	Общественный ...	26	Восточный адм...	район Гольяное
	5	Общественный ...	27	Восточный адм...	район Гольяное
	6	Общественный ...	28	Восточный адм...	район Гольяное
	7	Общественный ...	29	Восточный адм...	район Гольяное
	8	Общественный ...	30	Восточный адм...	район Гольяное
	9	Общественный ...	31	Восточный адм...	район Гольяное
	10	Общественный ...	32	Восточный адм...	район Гольяное
	11	Общественный ...	33	Восточный адм...	район Гольяное
	12	Общественный ...	34	Восточный адм...	район Гольяное
	13	Общественный ...	35	Восточный адм...	район Ивановс
	14	Общественный ...	36	Восточный адм...	район Ивановс
	15	Общественный ...	38	Восточный адм...	район Ивановс
	16	Общественный ...	37	Восточный адм...	район Ивановс
	17	Общественный ...	39	Восточный адм...	район Ивановс

8) Выбрать ячейку добавленной ОПОП и нажать на неё

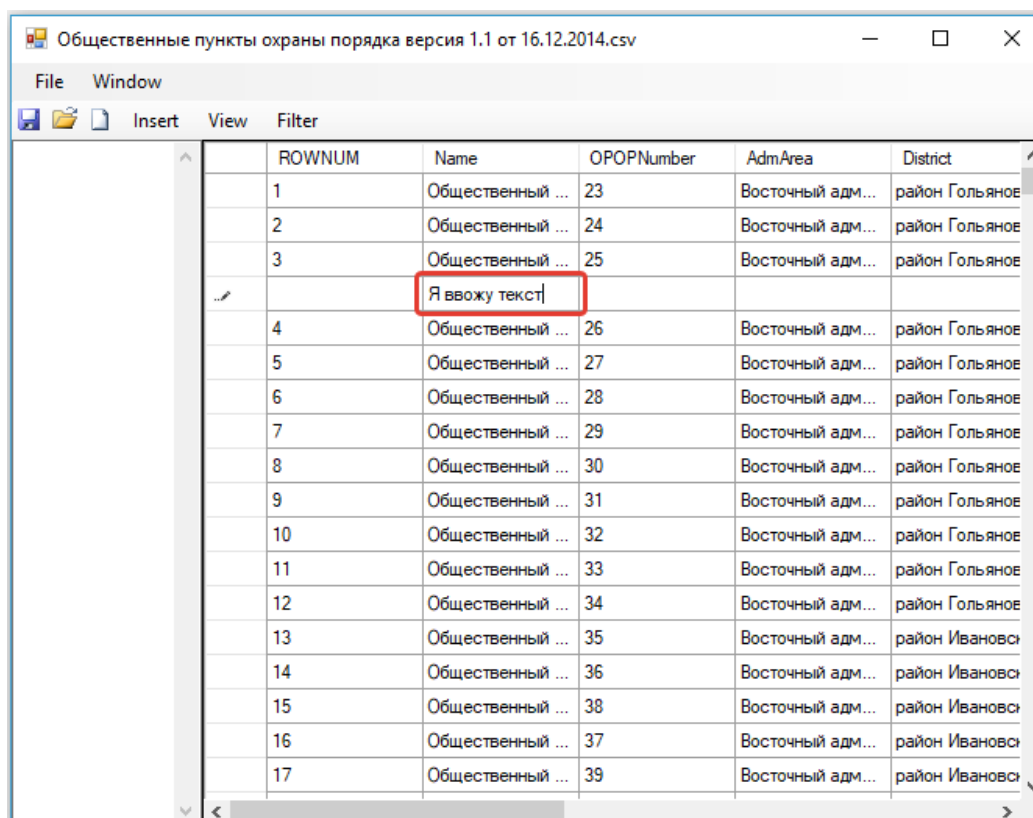
Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

File Window

Insert View Filter

	ROWNUM	Name	OPOPNumber	AdmArea	District
	1	Общественный ...	23	Восточный адм...	район Гольяное
	2	Общественный ...	24	Восточный адм...	район Гольяное
	3	Общественный ...	25	Восточный адм...	район Гольяное
	4	Общественный ...	26	Восточный адм...	район Гольяное
	5	Общественный ...	27	Восточный адм...	район Гольяное
	6	Общественный ...	28	Восточный адм...	район Гольяное
	7	Общественный ...	29	Восточный адм...	район Гольяное
	8	Общественный ...	30	Восточный адм...	район Гольяное
	9	Общественный ...	31	Восточный адм...	район Гольяное
	10	Общественный ...	32	Восточный адм...	район Гольяное
	11	Общественный ...	33	Восточный адм...	район Гольяное
	12	Общественный ...	34	Восточный адм...	район Гольяное
	13	Общественный ...	35	Восточный адм...	район Ивановс
	14	Общественный ...	36	Восточный адм...	район Ивановс
	15	Общественный ...	38	Восточный адм...	район Ивановс
	16	Общественный ...	37	Восточный адм...	район Ивановс
	17	Общественный ...	39	Восточный адм...	район Ивановс

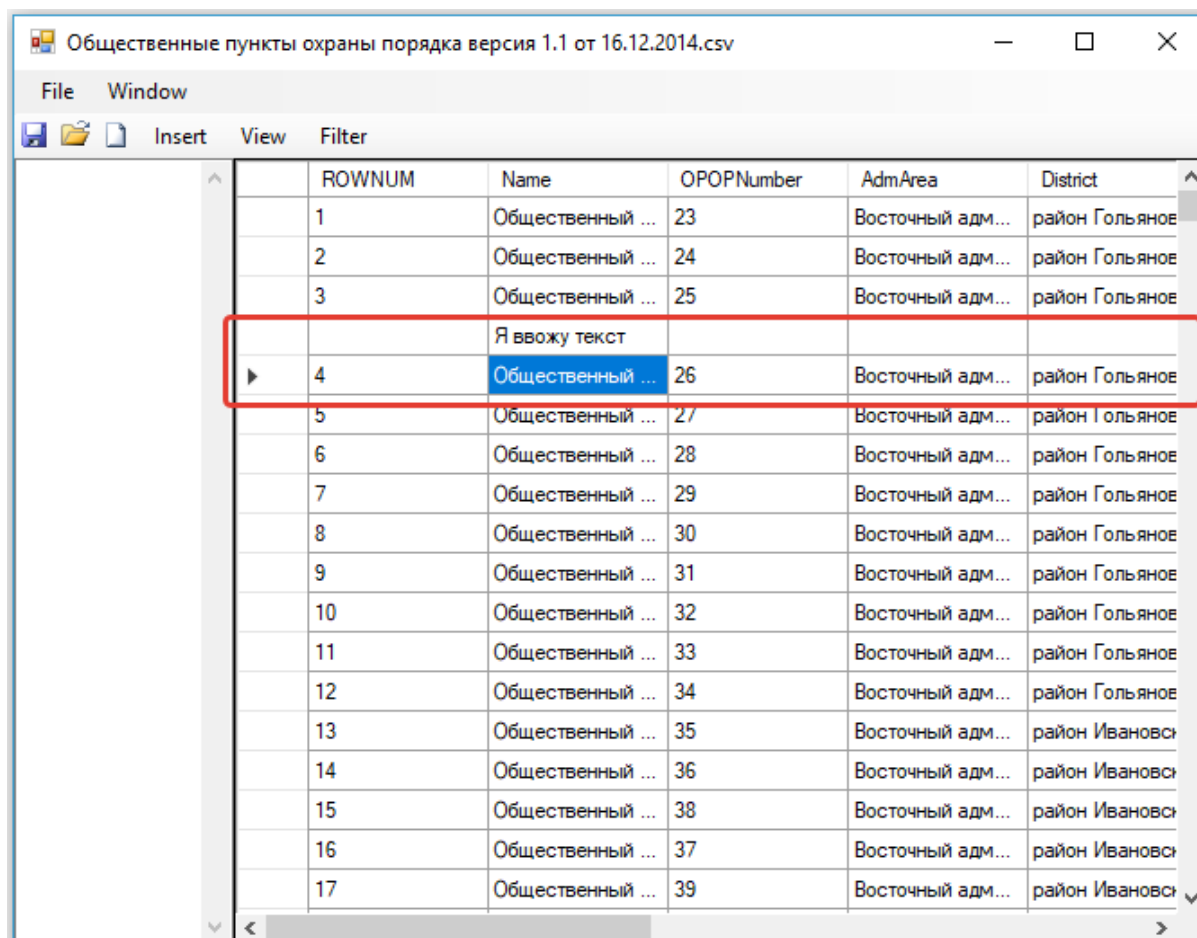
9) Ввести текст «Я ввожу текст»



Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

ROWNUM	Name	OPOPNumber	AdmArea	District
1	Общественный ...	23	Восточный адм...	район Гольяное
2	Общественный ...	24	Восточный адм...	район Гольяное
3	Общественный ...	25	Восточный адм...	район Гольяное
	Я ввожу текст			
4	Общественный ...	26	Восточный адм...	район Гольяное
5	Общественный ...	27	Восточный адм...	район Гольяное
6	Общественный ...	28	Восточный адм...	район Гольяное
7	Общественный ...	29	Восточный адм...	район Гольяное
8	Общественный ...	30	Восточный адм...	район Гольяное
9	Общественный ...	31	Восточный адм...	район Гольяное
10	Общественный ...	32	Восточный адм...	район Гольяное
11	Общественный ...	33	Восточный адм...	район Гольяное
12	Общественный ...	34	Восточный адм...	район Гольяное
13	Общественный ...	35	Восточный адм...	район Ивановс
14	Общественный ...	36	Восточный адм...	район Ивановс
15	Общественный ...	38	Восточный адм...	район Ивановс
16	Общественный ...	37	Восточный адм...	район Ивановс
17	Общественный ...	39	Восточный адм...	район Ивановс

10) Нажать Enter или перейти на любую другую ячейку.



Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

ROWNUM	Name	OPOPNumber	AdmArea	District
1	Общественный ...	23	Восточный адм...	район Гольяное
2	Общественный ...	24	Восточный адм...	район Гольяное
3	Общественный ...	25	Восточный адм...	район Гольяное
	Я ввожу текст			
4	Общественный ...	26	Восточный адм...	район Гольяное
5	Общественный ...	27	Восточный адм...	район Гольяное
6	Общественный ...	28	Восточный адм...	район Гольяное
7	Общественный ...	29	Восточный адм...	район Гольяное
8	Общественный ...	30	Восточный адм...	район Гольяное
9	Общественный ...	31	Восточный адм...	район Гольяное
10	Общественный ...	32	Восточный адм...	район Гольяное
11	Общественный ...	33	Восточный адм...	район Гольяное
12	Общественный ...	34	Восточный адм...	район Гольяное
13	Общественный ...	35	Восточный адм...	район Ивановс
14	Общественный ...	36	Восточный адм...	район Ивановс
15	Общественный ...	38	Восточный адм...	район Ивановс
16	Общественный ...	37	Восточный адм...	район Ивановс
17	Общественный ...	39	Восточный адм...	район Ивановс

11) Выделить строку с ОПОП.

Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

File Window

Insert View Filter

	ROWNUM	Name	OPOPNumber	AdmArea	District
	1	Общественный ...	23	Восточный адм...	район Гольяное
	2	Общественный ...	24	Восточный адм...	район Гольяное
	3	Общественный ...	25	Восточный адм...	район Гольяное
	4	Общественный ...	26	Восточный адм...	район Гольяное
	5	Общественный ...	27	Восточный адм...	район Гольяное
	6	Общественный ...	28	Восточный адм...	район Гольяное
	7	Общественный ...	29	Восточный адм...	район Гольяное
	8	Общественный ...	30	Восточный адм...	район Гольяное
	9	Общественный ...	31	Восточный адм...	район Гольяное
	10	Общественный ...	32	Восточный адм...	район Гольяное
	11	Общественный ...	33	Восточный адм...	район Гольяное
	12	Общественный ...	34	Восточный адм...	район Гольяное
	13	Общественный ...	35	Восточный адм...	район Ивановс
	14	Общественный ...	36	Восточный адм...	район Ивановс
	15	Общественный ...	38	Восточный адм...	район Ивановс
	16	Общественный ...	37	Восточный адм...	район Ивановс
	17	Общественный ...	39	Восточный адм...	район Ивановс

12) Нажать Del.

Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

File Window

Insert View Filter

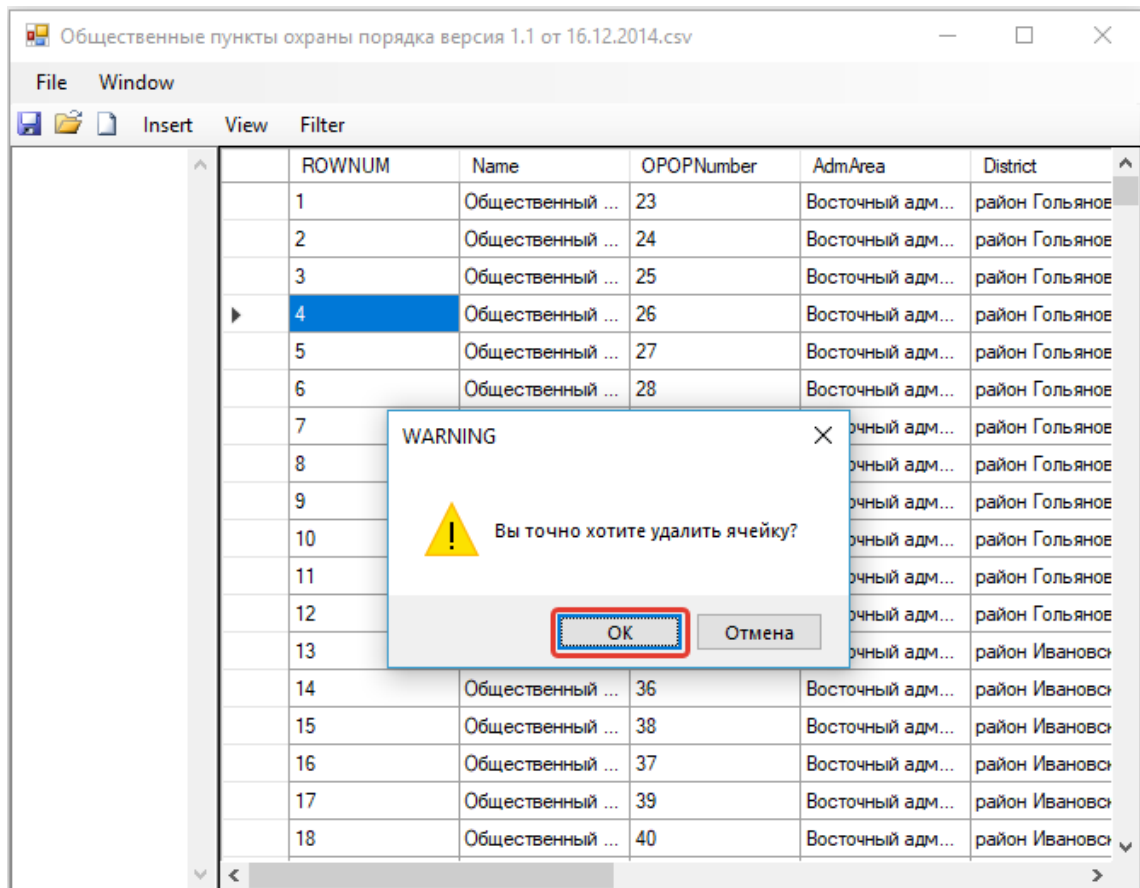
	ROWNUM	Name	OPOPNumber	AdmArea	District
	1	Общественный ...	23	Восточный адм...	район Гольяное
	2	Общественный ...	24	Восточный адм...	район Гольяное
	3	Общественный ...	25	Восточный адм...	район Гольяное
	4	Общественный ...	26	Восточный адм...	район Гольяное
	5	Общественный ...	27	Восточный адм...	район Гольяное
	6	Общественный ...	28	Восточный адм...	район Гольяное
	7	Общественный ...	29	Восточный адм...	район Гольяное
	8	Общественный ...	30	Восточный адм...	район Гольяное
	9	Общественный ...	31	Восточный адм...	район Гольяное
	10	Общественный ...	32	Восточный адм...	район Гольяное
	11	Общественный ...	33	Восточный адм...	район Гольяное
	12	Общественный ...	34	Восточный адм...	район Гольяное
	13	Общественный ...	35	Восточный адм...	район Ивановс
	14	Общественный ...	36	Восточный адм...	район Ивановс
	15	Общественный ...	38	Восточный адм...	район Ивановс
	16	Общественный ...	37	Восточный адм...	район Ивановс
	17	Общественный ...	39	Восточный адм...	район Ивановс
	18	Общественный ...	40	Восточный адм...	район Ивановс

WARNING

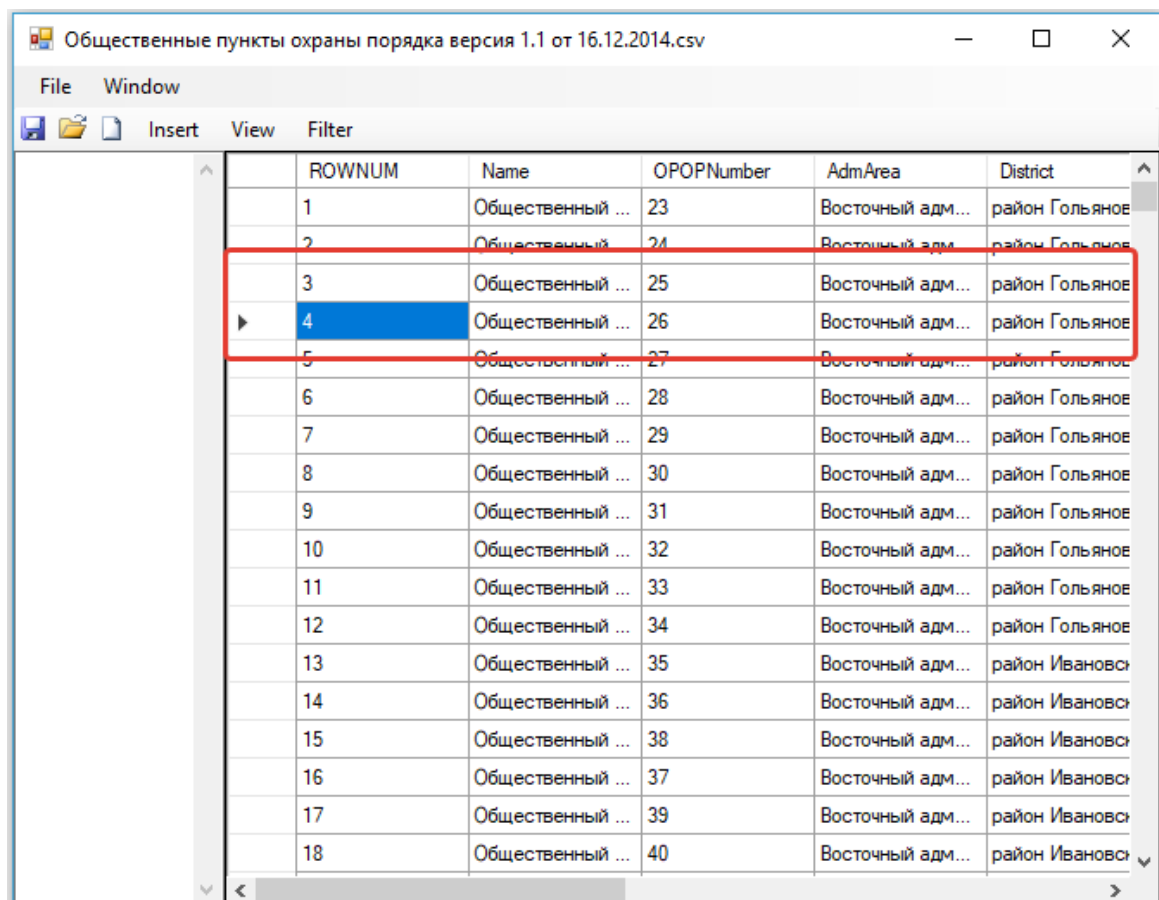
Вы точно хотите удалить ячейку?

OK Отмена

13) Подтвердить удаление (нажать Ок).



14) Насладиться отсутствием записи об ОПОП.



15) Начать писать в выбранной ячейке и нажать Enter.



Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

File Window

Insert View Filter

	ROWNUM	Name	OPOPNumber	AdmArea	District
	1	Общественный ...	23	Восточный адм...	район Гольяное
	2	Общественный ...	24	Восточный адм...	район Гольяное
	3	Общественный ...	25	Восточный адм...	район Гольяное
	45	Общественный ...	26	Восточный адм...	район Гольяное
	5	Общественный ...	27	Восточный адм...	район Гольяное
	6	Общественный ...	28	Восточный адм...	район Гольяное
	7	Общественный ...	29	Восточный адм...	район Гольяное
	8	Общественный ...	30	Восточный адм...	район Гольяное
	9	Общественный ...	31	Восточный адм...	район Гольяное
	10	Общественный ...	32	Восточный адм...	район Гольяное
	11	Общественный ...	33	Восточный адм...	район Гольяное
	12	Общественный ...	34	Восточный адм...	район Гольяное
	13	Общественный ...	35	Восточный адм...	район Ивановс
	14	Общественный ...	36	Восточный адм...	район Ивановс
	15	Общественный ...	38	Восточный адм...	район Ивановс
	16	Общественный ...	37	Восточный адм...	район Ивановс
	17	Общественный ...	39	Восточный адм...	район Ивановс
	18	Общественный ...	40	Восточный адм...	район Ивановс

16) Нажать на надпись OPOPNumber в шапке таблицы.

Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

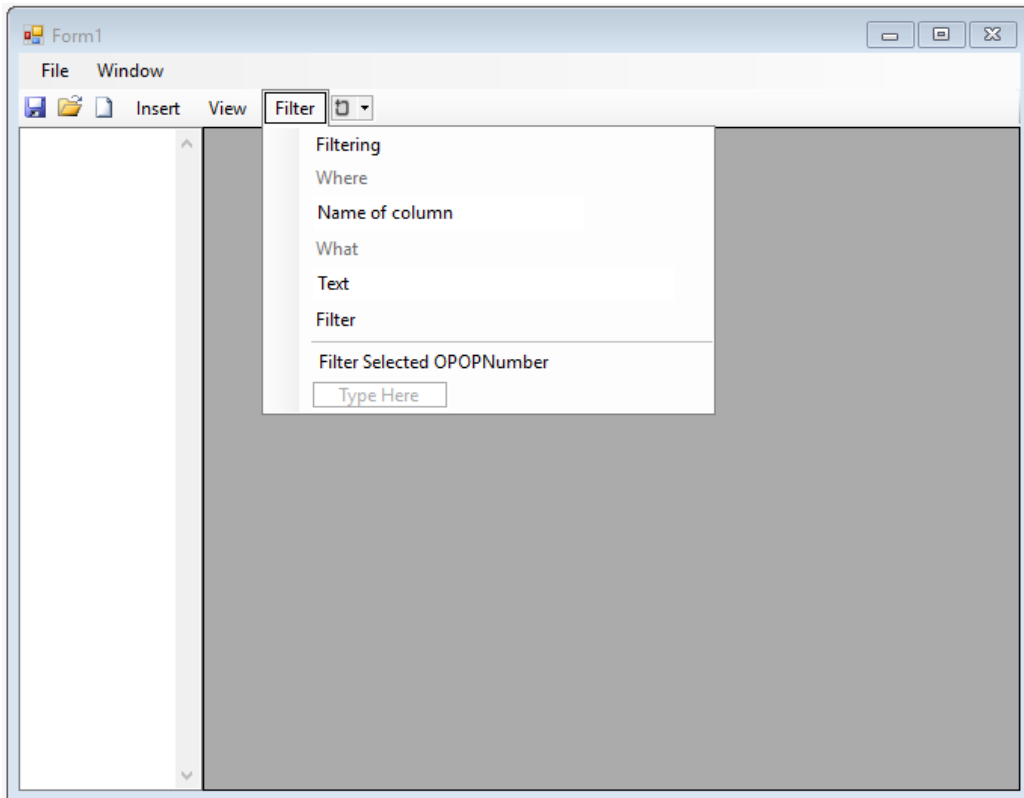
File Window

Insert View Filter

	ROWNUM	Name	OPOPNumber	AdmArea	District
	267	Общественный ...	1	Юго-Западный а...	Академический
	532	Общественный ...	1	Восточный адм...	район Богородс
	373	Общественный ...	1	Северный адми...	район Аэропорт
	607	Общественный ...	1	Западный адми...	район Дорогом
	754	Общественный ...	1	Зеленоградски...	район Матушки
	444	Общественный ...	1	Северо-Восточн...	Алексеевский р
	701	Общественный ...	1	Северо-Западн...	район Северное
	766	Общественный ...	1	Троицкий адми...	поселение Трои
	305	Общественный ...	1	Центральный а...	район Арбат
	78	Общественный ...	1	Юго-Восточный ...	район Выхино..
	304	Общественный ...	1	Южный админи...	район Бирюлёв
	306	Общественный ...	2	Центральный а...	район Арбат
	608	Общественный ...	2	Западный адми...	район Дорогом
	79	Общественный ...	2	Юго-Восточный ...	район Выхино..
	755	Общественный ...	2	Зеленоградски...	район Матушки
	702	Общественный ...	2	Северо-Западн...	район Северное
	533	Общественный ...	2	Восточный адм...	район Богородс
	445	Общественный ...	2	Северо-Восточн...	Алексеевский р



17) Нажать на Filter. Нажать на Filtering (появится галочка) и ввести под Where – District (вместо Name of column) под What – рай (вместо Text) и нажать кнопку Filter под полем для ввода (в котором был Text а теперь ввели рай).

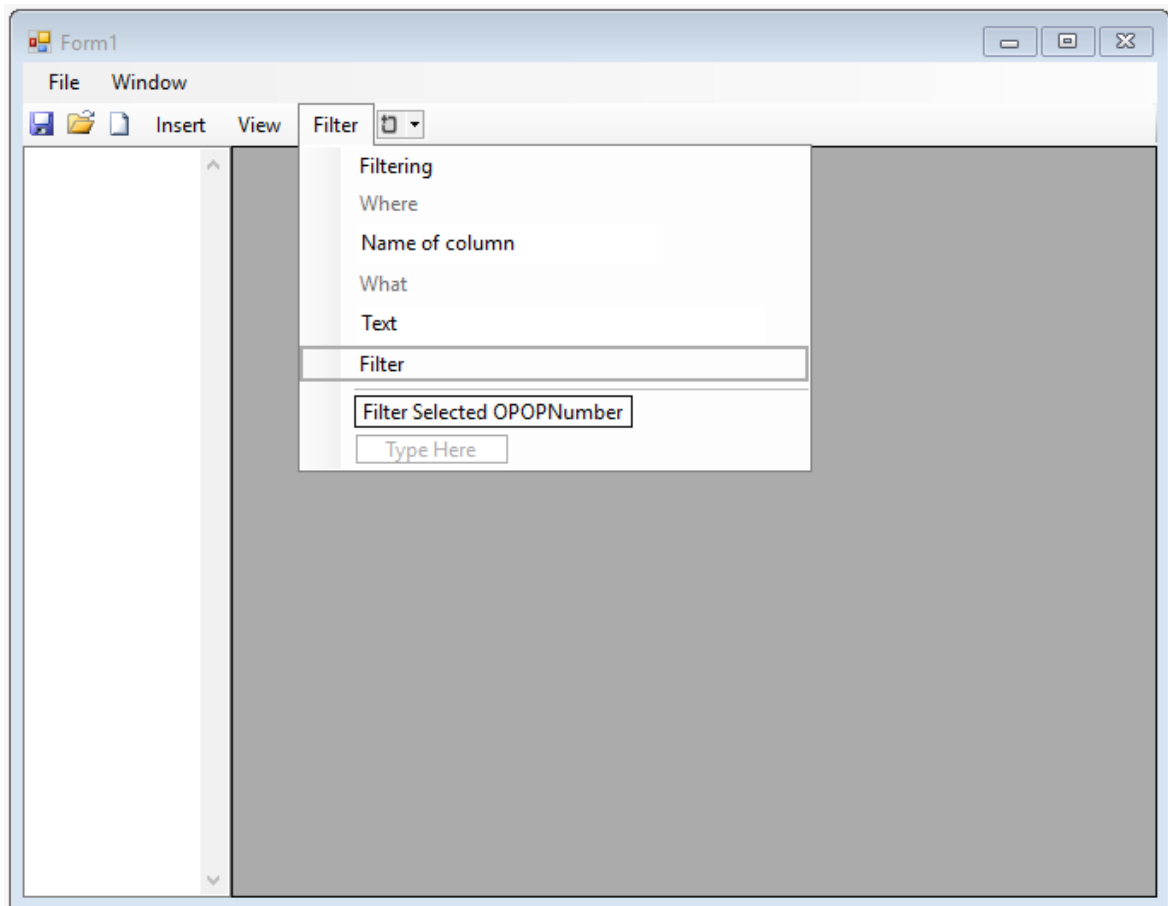


18) Результат.

The screenshot shows a software window titled 'Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv'. It has a menu bar with 'File' and 'Window'. Below the menu bar is a toolbar with icons for file operations and a 'Filter' button. The 'Filter' button is highlighted, and its dropdown menu is open. The menu contains the following items: 'Filtering' (with a checkmark), 'Where', 'Name of column', 'What', 'Text', 'Filter', and 'Filter Selected OPOPNumber'. At the bottom of the menu is a text input field labeled 'Type Here'.

ROWNUM	Name	OPOPNumber	AdmArea	District
267	Общественный ...	1	Юго-Западный а...	Академический
532	Общественный ...	1	Восточный адм...	район Богородс
373	Общественный ...	1	Северный адми...	район Аэропорт
607	Общественный ...	1	Западный адми...	район Дорогом
754	Общественный ...	1	Зеленоградски...	район Матушкин
444	Общественный ...	1	Северо-Восточн...	Алексеевский р
701	Общественный ...	1	Северо-Западн...	район Северное
305	Общественный ...	1	Центральный а...	район Арбат
78	Общественный ...	1	Юго-Восточный ...	район Выхино...
304	Общественный ...	1	Южный админи...	район Бирюлёв
306	Общественный ...	2	Центральный а...	район Арбат
608	Общественный ...	2	Западный адми...	район Дорогом
79	Общественный ...	2	Юго-Восточный ...	район Выхино...
755	Общественный ...	2	Зеленоградски...	район Матушкин
702	Общественный ...	2	Северо-Западн...	район Северное
533	Общественный ...	2	Восточный адм...	район Богородс
445	Общественный ...	2	Северо-Восточн...	Алексеевский р
268	Общественный ...	2	Юго-Западный а...	Академический

19) Выбрать строку с ОПОП и нажав на Filter выбрать Filter Selected OPOPNumber.

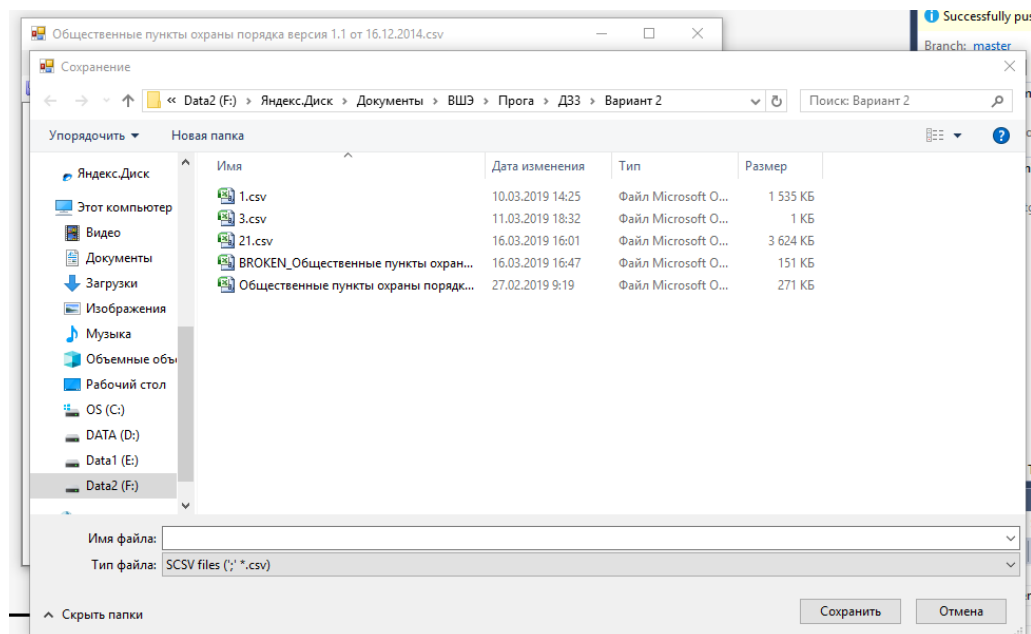


20) Результат.

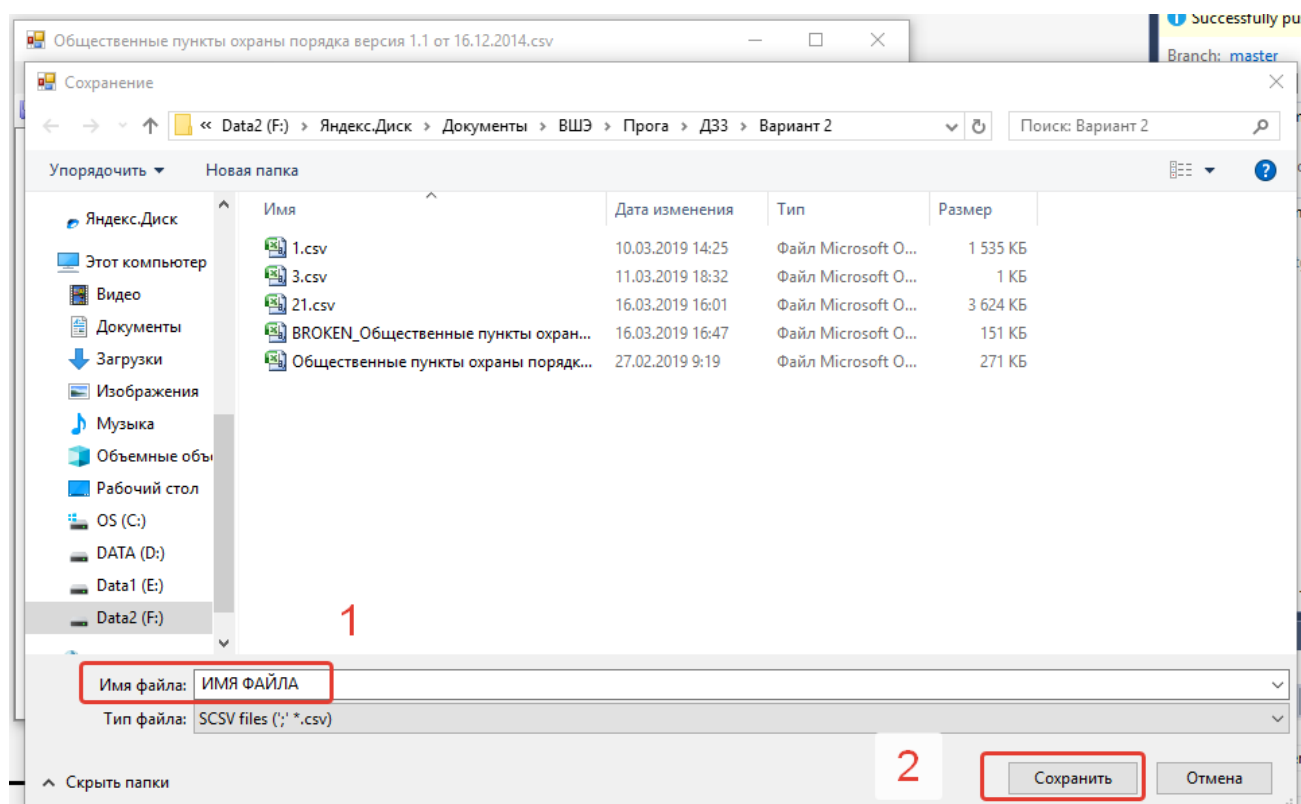
Общественные пункты охраны порядка версия 1.1 от 16.12.2014.csv

ROWNUM	Name	OPOPNumber	AdmArea	District
267	Общественный ...	1	Юго-Западный а...	Академический ...
532	Общественный ...	1	Восточный адм...	район Богородс...
373	Общественный ...	1	Северный адми...	район Аэропорт
607	Общественный ...	1	Западный адми...	район Дорогом...
754	Общественный ...	1	Зеленоградски...	район Матушкино
444	Общественный ...	1	Северо-Восточ...	Алексеевский р...
701	Общественный ...	1	Северо-Западн...	район Северное...
305	Общественный ...	1	Центральный а...	район Арбат
78	Общественный ...	1	Юго-Восточный ...	район Выхино-...
304	Общественный ...	1	Южный админи...	район Бирюлёв...
*				

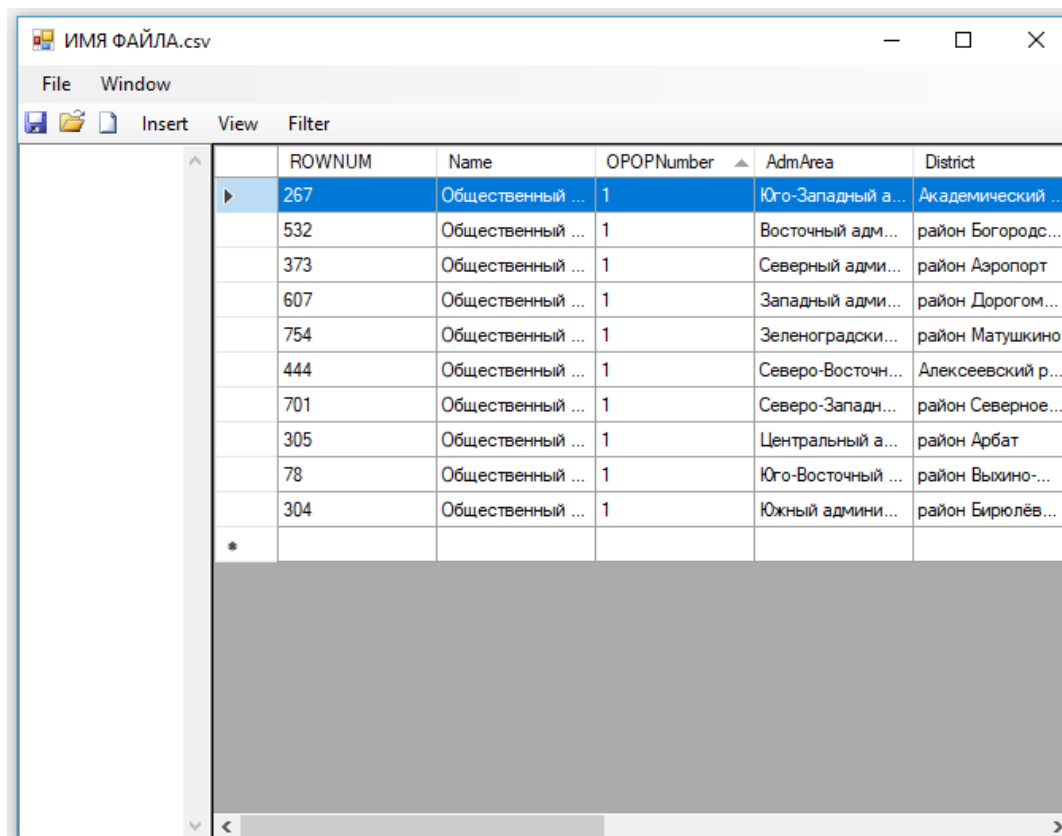
21) Открыть меню File, нажать на Override – теперь при сохранении файл перезапишется, открыть меню File, нажать Save As.



22) Ввести имя ИМЯ\_ФАЙЛА и нажать Сохранить.

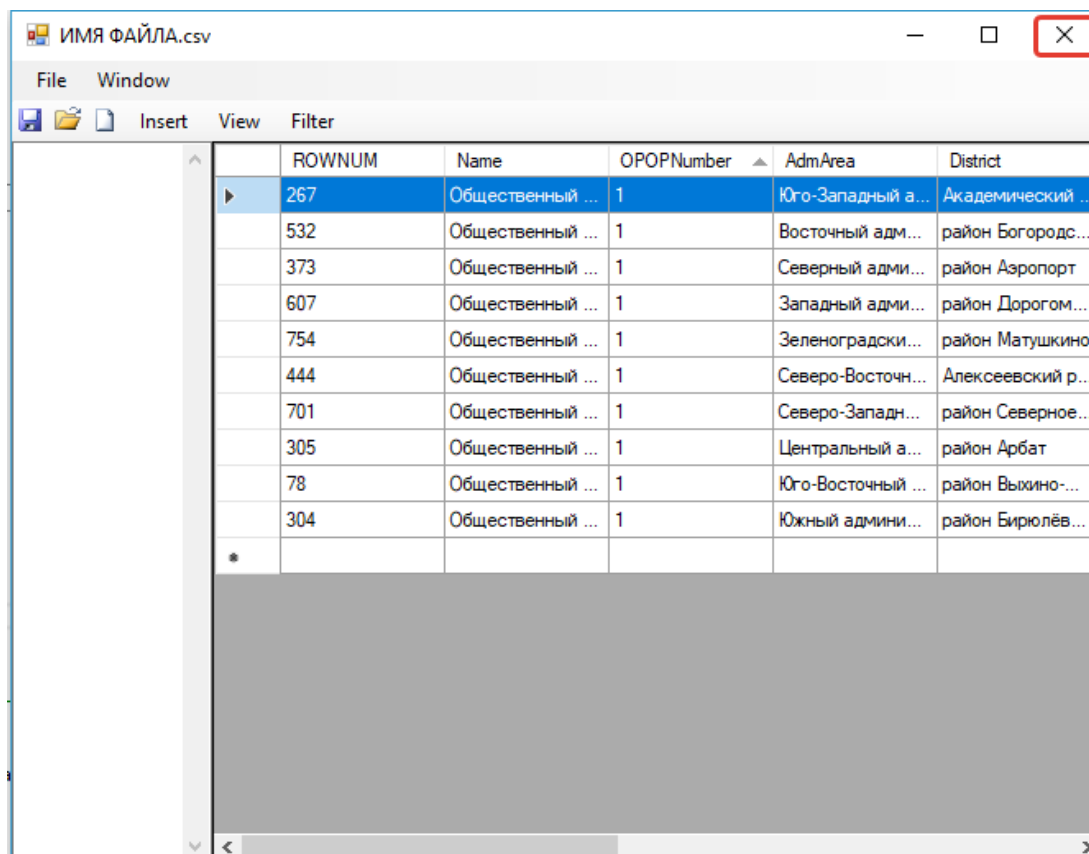


23) Результат.



ROWNUM	Name	OPOPNumber	AdmArea	District
267	Общественный ...	1	Юго-Западный а...	Академический ...
532	Общественный ...	1	Восточный адм...	район Богородс...
373	Общественный ...	1	Северный адми...	район Аэропорт
607	Общественный ...	1	Западный адми...	район Дорогом...
754	Общественный ...	1	Зеленоградски...	район Матушкино
444	Общественный ...	1	Северо-Восточн...	Алексеевский р...
701	Общественный ...	1	Северо-Западн...	район Северное...
305	Общественный ...	1	Центральный а...	район Арбат
78	Общественный ...	1	Юго-Восточный ...	район Выхино-...
304	Общественный ...	1	Южный админи...	район Бирюлёв...
*				

24) Закреть программу.



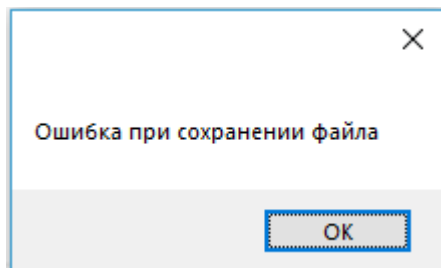
ROWNUM	Name	OPOPNumber	AdmArea	District
267	Общественный ...	1	Юго-Западный а...	Академический ...
532	Общественный ...	1	Восточный адм...	район Богородс...
373	Общественный ...	1	Северный адми...	район Аэропорт
607	Общественный ...	1	Западный адми...	район Дорогом...
754	Общественный ...	1	Зеленоградски...	район Матушкино
444	Общественный ...	1	Северо-Восточн...	Алексеевский р...
701	Общественный ...	1	Северо-Западн...	район Северное...
305	Общественный ...	1	Центральный а...	район Арбат
78	Общественный ...	1	Юго-Восточный ...	район Выхино-...
304	Общественный ...	1	Южный админи...	район Бирюлёв...
*				

Проверка завершена. Готово

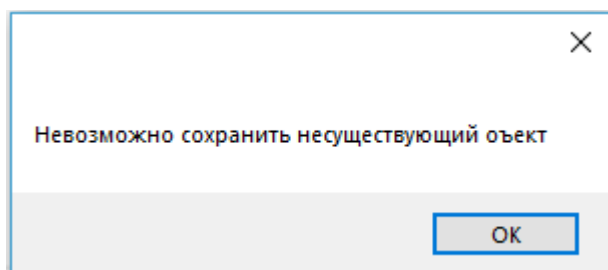
## 7. Сообщения пользователю

Все ошибки содержат в себе Строку-пояснение (представлена здесь) и Код ошибки (находится под Строкой пояснением и не представлена здесь).

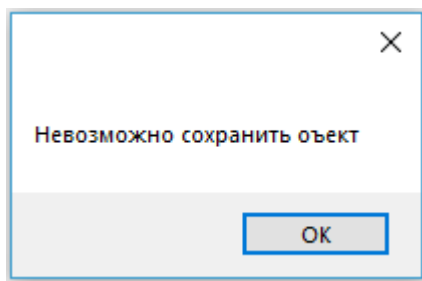
Необходимо предоставить программе права доступа к файлу



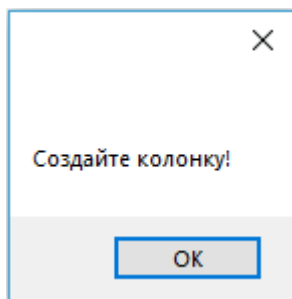
Необходимо создать или загрузить таблицу



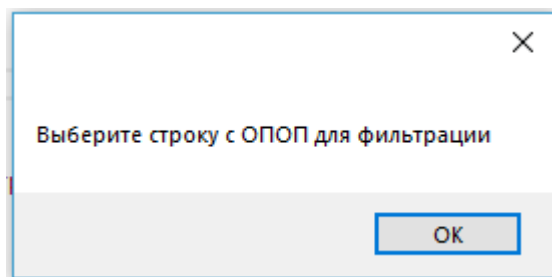
Ошибка файловой системы выберите другой путь или обратитесь к администратору



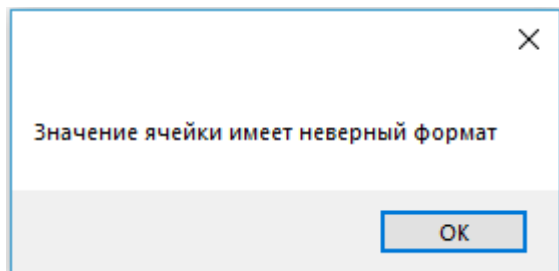
Попытка создать строку не создав столбец или таблицу создайте новый столбец или таблицу



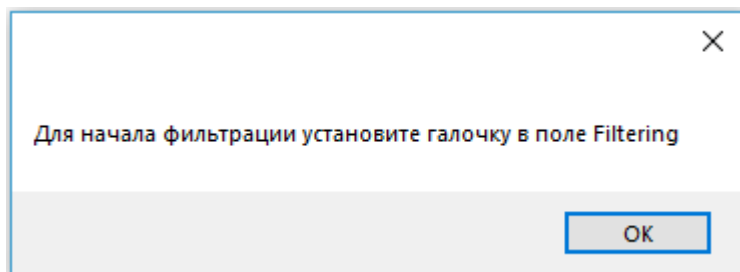
Необходимо выделить строку для того, чтобы программа поняла по какой строке фильтровать



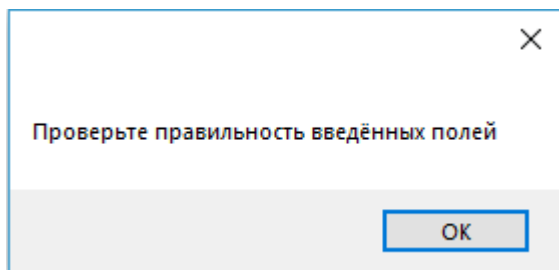
Неверный формат данных введённых в ячейку



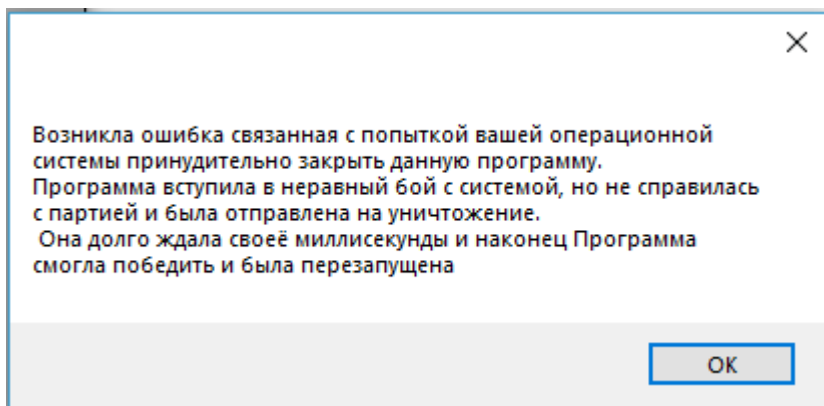
Попытка фильтровать без указания об данном намерении



Ошибка при вводе названия столбца для фильтрации



Поздравляю вас вы нашли новую ошибку пожалуйста отправьте сообщение с ошибкой на почту [ergerasimenko@edu.hse.ru](mailto:ergerasimenko@edu.hse.ru) и она будет в скором времени исправлена.



## 8. Текст (код) программы

### AhoKorasik.cs

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

namespace GerasimenkoER_KDZ3_v2
{
    class AhoCorasik
    {

        //int NMAX = 1000;

        public AhoCorasik() { }
        public AhoCorasik(IEnumerable which, int nmax = 1000)
        {
            init(nmax);
            foreach(string i in which)
            {
                add_string(i);
            }
        }

        private int findInString(string s, string[] w)
        {
            int i = 0;
            for (i = 0; i < w.Length; i++)
            {
```

```
        if (s == w[i]) return i;
    }
    return i;
}

/// <summary>
/// Find substring in string
/// </summary>
/// <param name="where"></param>
/// <returns></returns>
public vector<pair<vector<int>,string>> find(string where)
{
    int v = 0;
    vector<pair<vector<int>,string>> n = new
vector<pair<vector<int>, string>>();
    for(int i = 0; i < where.Length; i++)
    {
        v = go(v, where[i]);
        if (t[v].leaf)
        {
            int j = 0;
            bool flag = true;
            for(j = 0; j < n.size(); ++j)
            {
                if(n[j].second == t[v].leafs)
                {
                    n[j].first.append(i - t[v].leafs.Length + 1);
                    flag = false;
                }
            }
            if (flag) {
                n.append(new pair<vector<int>, string>());
                j = n.size() - 1;
                //n[j] = new pair<vector<int>, string>();
            }
        }
    }
}
```



```

        n[j].second=t[v].leafs;
        //n[j].setsecond(t[v].leafs);
        //vector<int> _ = new vector<int>();
        n[j].first=new vector<int>();//_;
        //n[j].setfirst(_);
        n[j].first.append(i - t[v].leafs.Length + 1);
    }
}

return n;
}

/// <summary>
/// Find substring in collection string
/// </summary>
/// <param name="ii">String</param>
/// <param name="isienumerable">Fictive parameter</param>
/// <returns></returns>
public IEnumerable<vector<pair<vector<int>, string>>>
find(IEnumerable ii, bool isienumerable)
{
    int v = 0;
    int iin = -1;
    vector<pair<vector<int>, string>> n = new
vector<pair<vector<int>, string>>();
    foreach (string where in ii)
    {

        ++iin;
        for (int i = 0; i < where.Length; i++)
        {
            v = go(v, where[i]);
            if (t[v].leaf)

```

```

    {
        int j = 0;
        bool flag = true;
        for (j = 0; j < n.size(); ++j)
        {
            if (n[j].second == t[v].leafs)
            {
                n[j].first.append(i - t[v].leafs.Length +
1);
            }
        }
        if (flag)
        {
            n.append(new pair<vector<int>, string>());
            j = n.size() - 1;
            //n[j] = new pair<vector<int>, string>();
            n[j].second=t[v].leafs;
            n[j].first = new vector<int>();
            n[j].first.append(i - t[v].leafs.Length + 1);
        }
    }
}
yield return n;
}
}
```

```
struct vertex
{
    public int[] next; //adress
    public bool leaf;
    public string leafs;
    public int p;
    public char pch;
    public int link;
    public int[] go; //adress
    //public int myadress;
};

void build(int nmax = 1000) {
    t = new vertex[nmax + 1];
    //for(int i = 0; i < nmax + 1; i++)
    //{
    //    t[i] = new vertex();
    //}
}

vertex[] t;
int sz;

void memset(out int[] p, int _, int[] __)
{
    p = new int[char.MaxValue];
    for (int i = 0; i < p.Length; ++i)
    {
        p[i] = -1;
    }
}

void init(int nmax = 1000)
{
    build(nmax);
    t[0].p = t[0].link = -1;
```

```
//t[0].myadress = 0;
memset(out t[0].next, 255, t[0].next);
memset(out t[0].go, 255, t[0].go);
sz = 1;
}

void add_string(string s) {
    int v = 0;
    for (int i=0; i<s.Length; ++i) {
        char c = s[i];
        if (t[v].next[c] == -1) {
            memset(out t[sz].next, 255, t[sz].next); //ERROR
            memset(out t[sz].go, 255, t[sz].go);
            t[sz].link = -1;
            t[sz].p = v;
            t[sz].pch = c;
            t[v].next[c] = sz++;
        }
        v = t[v].next[c];
    }
    t[v].leaf = true;
    t[v].leafs = s;
}

//int go(int v, char c);

int get_link(int v)
{
    if (t[v].link == -1)
        if (v == 0 || t[v].p == 0)
            t[v].link = 0;
        else
            t[v].link = go(get_link(t[v].p), t[v].pch);
    return t[v].link;
}
```

```
    }

    int go(int v, char c)
    {
        if (t[v].go[c] == -1)
            if (t[v].next[c] != -1)
                t[v].go[c] = t[v].next[c];
            else
                t[v].go[c] = v == 0 ? 0 : go(get_link(v), c);
        return t[v].go[c];
    }

}

}
CSVconv.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using GerasimenkoER_KDZ3_v2;

namespace GerasimenkoER_KDZ3_v2
{
```

```
    [System.Serializable]
    public class CSVException : ApplicationException
    {
        public CSVException() { }
        public CSVException(string message) : base(message) { }
        public CSVException(string message, Exception inner) :
base(message, inner) { }
        protected CSVException(
```

```
        System.Runtime.Serialization.SerializationInfo info,
        System.Runtime.Serialization.StreamingContext context) :
base(info, context) { }
    }

    public class CSVconv
    {

        #region f*f
        /// <summary>
        /// Read lines from file
        /// </summary>
        /// <param name="path">Path to file</param>
        /// <param name="n">Number of readed lines (read all -1)</param>
        /// <param name="encode">Encoding of file</param>
        /// <exception name="CSVException"></exception>
        /// <returns>Array of strings from file</returns>
        public static string[] fscanf(string path, Encoding encode = null,
int n = -1)
        {
            string str = $"Can't read all lines from file {path}";
            StreamReader f = null;
            if (encode == null) { encode = Encoding.Default; }
            try
            {
                //FileStream file = new FileStream(path, FileMode.Open);
                if (n == -1)
                {

                    return File.ReadAllLines(path, encode);
                }
                str = $"Can't read {n} lines from file {path}";
                f = File.OpenText(path);
                string[] s = new string[n];
```

```
        for (int i = 0; i < n; i++)
        {
            s[i] = f.ReadLine();
        }
        return s;
    }
    catch (ArgumentNullException e)
    {
        throw new CSVException(str + " because you give null string
instead path", e);
    }
    catch (ArgumentException e)
    {
        throw new CSVException(str + " because you give path in
wrong format", e);
    }
    catch (PathTooLongException e)
    {
        throw new CSVException(str + " because given path is too
long", e);
    }
    catch (DirectoryNotFoundException e)
    {
        throw new CSVException(str + " because you give path to
nonexistent directory", e);
    }
    catch (FileNotFoundException e)
    {
        throw new CSVException(str + " because you give path to
nonexistent file", e);
    }
    catch (IOException e)
    {
```

```
        throw new CSVException(str + " because program has error
when reading data from file", e);
    }
    catch (UnauthorizedAccessException e)
    {
        throw new CSVException(str + " because your permissions is
insufficient to open this file", e);
    }
    catch (NotSupportedException e)
    {
        throw new CSVException(str + " because stream does not
support invoked functionality", e);
    }
    catch (System.Security.SecurityException e)
    {
        throw new CSVException(str + " because program take
security error", e);
    }
    catch (Exception e)
    {
        throw new CSVException(str + "Unknown exception", e);
    }
    finally
    {
        f?.Close();
    }
    return null;
}

/// <summary>
/// Write lines to file
/// </summary>
/// <param name="path">Path to file</param>
```



```
    /// <param name="es">Array of strings which needed to write to this
file</param>

    /// <param name="rewrite">Rewrite file (true) or append data to the
end of file (false)</param>

    public static void fprintf(string path, string[] es, bool
rewrite=false, Encoding encode = null)
    {
        string str = $"Can't write all lines from file {path}";
        if (encode == null) { encode = Encoding.Default; }
        //StreamReader f = null;
        try
        {
            //FileStream file = new FileStream(path, FileMode.Open);
            if (!File.Exists(path)) { rewrite = true; }
            if (rewrite) {
                File.WriteAllLines(path, es, encode);
                return;
            }
            else
            {
                File.AppendAllLines(path, es, encode);
                return;
            }
        }
        catch (ArgumentNullException e)
        {
            throw new CSVException(str + " because you give null string
instead path", e);
        }
        catch (ArgumentException e)
        {
            throw new CSVException(str + " because you give path in
wrong format", e);
        }
    }
}
```

```
        catch (PathTooLongException e)
        {
            throw new CSVException(str + " because given path is too
long", e);
        }
        catch (DirectoryNotFoundException e)
        {
            throw new CSVException(str + " because you give path to
nonexistent directory", e);
        }
        catch (FileNotFoundException e)
        {
            throw new CSVException(str + " because you give path to
nonexistent file", e);
        }
        catch (IOException e)
        {
            throw new CSVException(str + " because program has error
when writing data to file", e);
        }
        catch (UnauthorizedAccessException e)
        {
            throw new CSVException(str + " because your permissions is
insufficient to open this file", e);
        }
        catch (NotSupportedException e)
        {
            throw new CSVException(str + " because stream does not
support invoked functionality", e);
        }
        catch (System.Security.SecurityException e)
        {
            throw new CSVException(str + " because program take
security error", e);
        }
    }
```

```
    }
    catch (Exception e)
    {
        throw new CSVException(str + "Unknown exception", e);
    }
    finally
    {
        //f.Close();
    }
    return;
}

#endregion

/// <summary>
/// Load CSV file and return List of List of str
/// </summary>
/// <param name="path">Path to file</param>
/// <param name="c">Separator</param>
/// <returns>Data for cells in table</returns>
public static List<List<string>> LoadCSVtoStr(string path, char
c=',', Encoding encode=null)
{
    List<List<string>> res = new List<List<string>>();
    string[] s;
    try
    {
        s = fscanf(path, encode);

    }
    catch (CSVException e) {
        throw e;
    }
    catch (Exception e)
```

```
{
    throw new CSVException(null, e);
}

for (int i = 0; i < s.Length; i++)
{
    res.Add(ConvertCSVlinetoListstr(s[i], c));
}
res.Add(new List<string>());
return res;
}

/// <summary>
/// Save data from table to CSV file
/// </summary>
/// <param name="path">Path to file</param>
/// <param name="s">Data from cells of table</param>
/// <param name="c">CSV or another separator</param>
/// <param name="rewrite">Rewrite or append</param>
public static void SaveStrtoCSV(string path, List<List<string>> s,
char c = ',', bool rewrite = false, Encoding encode = null)
{
    string[] se = new string[s.Count];
    for(int i = 0; i < s.Count; i++)
    {
        se[i] = ConvertListstrtoCSVline(s[i], c);
    }
    try
    {
        fprintf(path,se,rewrite,encode);
    }
    catch (CSVException e)
    {

```

```
        throw e;
    }
    catch (Exception e)
    {
        throw new CSVException(null, e);
    }
}

/// <summary>
/// Save data from table to CSV file
/// </summary>
/// <param name="path">Path to file</param>
/// <param name="s">Data from rows of table</param>
/// <param name="c">CSV or another separator</param>
/// <param name="rewrite">Rewrite or append</param>
public static void SaveStrtoCSV(string path,
System.Windows.Forms.DataGridViewRowCollection s, char c = ',', bool
rewrite = false, Encoding encode = null, string header= "")
{
    if (header == "")
    {
        header = "ROWNUM" + c + "Name" + c + "OPOPNumber" + c +
"AdmArea" + c + "District" + c + "Address" + c + "PublicPhone" + c +
"ExtraInfo" + c + "X_WGS" + c + "Y_WGS" + c + "GLOBALID";
    }
    List<string> se = new List<string>();
    se.Add(header);
    for (int i = 0; i < s.Count; i++)
    {
        if (s[i] == null) {
            continue;
        }
        if(s[i].Visible)
```

```
        se.Add(ConvertListstrtoCSVline(s[i].Cells, c));
    }
    try
    {
        fprintf(path, se.ToArray(), rewrite, encode);

    }
    catch (CSVException e)
    {
        throw e;
    }
    catch (Exception e)
    {
        throw new CSVException(null, e);
    }
}

#region Converters

/// <summary>
/// Convert form CSV line to String arr for rows
/// </summary>
/// <param name="s">Raw string</param>
/// <param name="c">Separator between columns</param>
/// <returns></returns>
public static List<string> ConvertCSVlinetoListstr(string s, char
c=',')
{
    List<string> l = new List<string>();
    bool isquote = false;
    bool istrim = true;
    bool fs = true;
    string sn = "";
```

```
for (int i = 0; i < s.Length; i++)
{
    if(s[i]=='"' && (fs || i+1==s.Length || s[i + 1] != '"'))
    {
        if (!isquote)
        {
            sn.TrimStart(' ');
            istrim = false;
        }
        isquote ^= true;
        if (fs) { fs = false; }
        continue;
    }
    if(s[i]=='"' && i+1<s.Length && s[i + 1] == '"')
    {
        sn += s[i];
        ++i;
        if (fs) { fs = false; }
        continue;
    }
    if (isquote)
    {
        sn += s[i];
        if (fs) { fs = false; }
        continue;
    }
    if (s[i] == c)
    {
        if (istrim) { sn = sn.Trim(' ', '\n', '\t', '\r'); }
        l.Add(sn);
        sn = "";
        isquote = false;
        istrim = true;
        fs = true;
    }
}
```

```
        continue;
    }
    if (!isquote)
    {
        sn += s[i];
        if (fs) { fs = false; }
        continue;
    }

}

if (sn.Length > 0) { l.Add(sn); }

return l;
}

/// <summary>
/// Convert form String arr from rows to CSV line
/// </summary>
/// <param name="s">List of string from columns</param>
/// <param name="c">Separator between columns</param>
/// <returns></returns>
public static string ConvertListstrtoCSVline(List<string> s, char c
= ',', bool always=false)
{
    bool isquote = false;
    string sn = "",sr = "";

    for(int j = 0;j < s.Count; j++)
    {
        if (s[j] == null) { isquote = true; }
        for (int i = 0;s[j]!=null && i < s[j].Length; i++)
        {
```



```
        if (s[j][i] == '"')
        {
            sn += '"';
            sn += s[j][i];
            isquote = true;
            continue;
        }
        if (isquote)
        {
            sn += s[j][i];
            continue;
        }
        if (s[j][i] == c)
        {
            sn += s[j][i];
            isquote = true;
            continue;
        }
        if (!isquote)
        {
            sn += s[j][i];
            continue;
        }
    }
    if (isquote || always) { sn = '"' + sn + '"'; }
    sr += sn;
    sr += c;
    sn = "";
    isquote = false;
}
sr = sr.Substring(0, Math.Max(0, sr.Length - 1));
return sr; //.TrimEnd(c);
}
```

```
/// <summary>
/// Convert form String arr from columns to CSV line
/// </summary>
/// <param name="s">List of string from columns</param>
/// <param name="c">Separator between columns</param>
/// <returns></returns>
public static string
ConvertListstrtoCSVline(System.Windows.Forms.DataGridViewCellCollection s,
char c = ',', bool always = false)
{
    bool isquote = false;
    string sn = "", sr = "";

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

        if (s[j].Value == null) { isquote = true; }
        for (int i = 0; s[j].Value!=null && i <
((string)(s[j].Value)).Length; i++)
        {
            if (((string)(s[j].Value))[i] == '"')
            {
                sn += '"';
                sn += ((string)(s[j].Value))[i];
                isquote = true;
                continue;
            }
            if (isquote)
            {
                sn += ((string)(s[j].Value))[i];
                continue;
            }
        }
    }
}
```

```
        if (((string)(s[j].Value))[i] == c)
        {
            sn += ((string)(s[j].Value))[i];
            isquote = true;
            continue;
        }
        if (!isquote)
        {
            sn += ((string)(s[j].Value))[i];
            continue;
        }
    }
    if (isquote || always) { sn = '"' + sn + '"'; }
    sr += sn;
    sr += c;
    sn = "";
    isquote = false;
}
sr = sr.Substring(0, Math.Max(0, sr.Length - 1));
return sr;//.TrimEnd(c);
}
```

#endregion

#region GetSeparatorType

```
/// <summary>
/// Return separator value of separator type
/// </summary>
/// <param name="s">Separator type</param>
/// <returns></returns>
public static char GetSeparType(string s)
{
```

```
        switch (s[0])
        {
            case ('C'):
                return ',';

            case ('T'):
                return '\t';

            case ('S'):
                return ';';

        }
        return ',';
    }

    /// <summary>
    /// Return separator value of separator type
    /// </summary>
    /// <param name="s">Separator type</param>
    /// <returns></returns>
    public static char GetSeparType(char s)
    {
        switch (s)
        {
            case ('C'):
                return ',';

            case ('T'):
                return '\t';

            case ('S'):
                return ';';

        }
        return ',';
    }
}
```

```
    }

    #endregion
}
}
Data.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GerasimenkoER_KDZ3_v2
{

    public class ОПОП
    {
        #region values
        public int _ROWNUM=0;

        int _OPOPNumber=0;
        Расположение _adress;
        Phone _PublicPhone = new Phone();
        int _GLOBALID=0;
        string Name="";
        string Address="";
        string ExtraInfo="";
        string X_WGS="", Y_WGS="";
        pair<string,string> Adress { get { return pair<string,
string>.makepair<string, string>(adress.AdmArea, adress.District); } set {
adress.AdmArea = value.first; adress.District = value.second; } }
        string ROWNUM { get { return "" + _ROWNUM; } set {
int.TryParse(value, out _ROWNUM); } }
    }
```

```
        string OPOPNumber { get { return "" + _OPOPNumber; } set {
int.TryParse(value, out _OPOPNumber); } }
        string AdmArea { get { return address.AdmArea; } set {
address.AdmArea = value; } }
        string District { get { return address.District; } set {
address.District = value; } }
        string PublicPhone{ get { return _PublicPhone.get(); } set {
_PublicPhone.set(value); } }
        string GLOBALID { get { return "" + _GLOBALID; } set {
int.TryParse(value, out _GLOBALID); } }

#endregion

public Расположение address { get { return _address; } set { _address
= value; } }

public ОПОП() { }
public ОПОП(Расположение a)
{
    address = a;
}

public ОПОП(IEnumerable<string> i)
{
    int n = 0;
    foreach (var s in i)
    {
        this[n++] = s;
    }
}

public ОПОП(IEnumerable<string> i, Расположение a):this(a)
{
    int n = 0;
    foreach (var s in i)
    {
```

```
        this[n++] = s;
    }
}

/// <summary>
/// GET AND SET
/// </summary>
/// <param name="n"></param>
/// <returns></returns>
public string this [int n]
{
    get {
        switch (n)
        {
            case (0): return ROWNUM;
            case (1): return Name;
            case (2): return OPOPNumber;
            case (3): return AdmArea;
            case (4): return District;
            case (5): return Address;
            case (6): return PublicPhone;
            case (7): return ExtraInfo;
            case (8): return X_WGS;
            case (9): return Y_WGS;
            case (10): return GLOBALID;
        }
        return "";
    }
    set {
        switch (n)
        {
            case (0): ROWNUM = value; break;
            case (1): Name = value; break;
            case (2): OPOPNumber = value; break;
```

```
        case (3): AdmArea = value; break;
        case (4): District = value; break;
        case (5): Address = value; break;
        case (6): PublicPhone = value; break;
        case (7): ExtraInfo = value; break;
        case (8): X_WGS = value; break;
        case (9): Y_WGS = value; break;
        case (10): GLOBALID = value; break;
    }
}
}

}

public class Расположение
{
    public string AdmArea="";
    public string District="";
}

public class Phone
{
    string number="";

    public string get()
    {
        return number;
    }

    public void set(string num) //(499) 367-49-82
    {
        //if (num.Length != 10) throw new ApplicationException("Not a
correct phone number");
        //int num2 = int.Parse(String.Join("",parce(num).Reverse()));
        string num2 = /*int.Parse*/(parce(num));
    }
}
```



```
        string strok = "";
        int i = -1;
        while(++i!=-1 && num2.Length>i)
        {
            strok += "" + (num2[num2.Length - 1 - i]); // % 10);
            //num2 /= 10;
            if (i == 1) { strok += "-"; }
            if (i == 3) { strok += "-"; }
            if (i == 6) { strok += " )"; }
            if (i == 9) { strok += "("; }
        }
        number = String.Join("", strok.Reverse());
    }
    public string parce(string s="")
    {
        if(s.Length==0) { s = number; }
        string strok = "";
        foreach(char c in s)
        {
            if(c>='0' && c <= '9')
            {
                strok += c;
            }
        }
        return strok; //(number.Trim(strok.ToCharArray()));
    }
}
```

```
class Data
{
    public bool flagmb = false;
    public bool rewrite = false;
```

```
        public bool issaved = true;
        public bool isaded = false;
        public bool sne = false, ene = false;
        public string name = "";
        public List<List<string>> data = null;
        public string[] datas = null;
        public char separ = ';';
        public Encoding encode = Encoding.Default;
        public int sn = 1;
        public int en = -1;
    }
```

```
}
```

Form1.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;
using GerasimenkoER_KDZ3_v2;

namespace GerasimenkoER_KDZ3_v2
{
    public partial class Form1 : Form
    {
        bool IS_AUTO_UPDATE_ES = false;

        Data d = new Data();
        List<ОПОП> opop = new List<ОПОП>();
    }
}
```

```
List<Расположение> adr = new List<Расположение>();
//bool flagmb = false;
//string name = "";
//List<List<string>> data = null;
//string[] datas = null;
//char separ = ';';
//bool rewrite = false;
//bool issaved = true;
//bool sne = false, ene = false;
//Encoding encode = Encoding.Default;
//int sn = -1;
//int en = -1;
#region MetoData
bool flagmb { get { return d.flagmb; } set { d.flagmb = value; } }
string name { get { return d.name; } set { d.name = value; } }
List<List<string>> data { get { return d.data; } set { d.data =
value; } }
string[] datas { get { return d.datas; } set { d.datas = value; } }
char separ { get { return d.separ; } set { d.separ = value; } }
bool rewrite { get { return d.rewrite; } set { d.rewrite = value; }
}
bool issaved { get { return d.issaved; } set { d.issaved = value; }
}
bool isadded { get { return d.isaded; } set { d.isaded = value; } }
bool sne { get { return d.sne; } set { d.sne = value; } }
bool ene { get { return d.ene; } set { d.ene = value; } }
Encoding encode { get { return d.encode; } set { d.encode = value;
} }

int sn { get { return d.sn; } set { d.sn = value; } }
int en { get { return d.en; } set { d.en = value; } }
#endregion
```

```
#region UI

bool flagcontrol = false;
bool flagshift = false;
/// <summary>
/// Отлов нажатий клавиш
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Form1_KeyDown(object sender, EventArgs e)
{
    if (e.Control) flagcontrol = true;
    if (!e.Control) flagcontrol = false;
    if (e.Shift) flagshift = true;
    if (!e.Shift) flagshift = false;
    if (e.KeyCode == Keys.O && flagcontrol)
    {
        openToolStripButton_Click(sender, e);
    }
    if (e.KeyCode == Keys.S && flagcontrol && !flagshift)
    {
        saveToolStripButton_Click(sender, e);
    }
    if (e.KeyCode == Keys.N && flagcontrol)
    {
        newToolStripMenuItem_Click(sender, e);
    }
    if (e.KeyCode == Keys.S && flagcontrol && flagshift)
    {
        saveAsToolStripMenuItem_Click_1(sender, e);
    }
    //Debug
    if(flagshift && flagcontrol && e.KeyCode == Keys.E)
    {
```

```
DropExWindow("Ошибка при сохранении файла\n");
DropExWindow("Невозможно сохранить несуществующий
объект\n");

DropExWindow("Невозможно сохранить объект\n");
DropExWindow("Невозможно сохранить объект\n");
DropExWindow("\nСоздайте колонку!");
DropExWindow("Значение ячейки имеет неверный формат\n");
DropExWindow("Для начала фильтрации установите галочку в
поле Filtering");

DropExWindow("Проверьте правильность введенных полей\n");
DropExWindow("Возникла ошибка связанная с попыткой вашей
операционной системы принудительно закрыть данную программу.\nПрограмма
вступила в неравный бой с системой, но не справилась с партией и была
отправлена на уничтожение.\nОна долго ждала своей миллисекунды и наконец
Программа смогла победить и была перезапущена");
}

#region c
//if (Frac == null || Frac.isdrawing) return;
//if (e.KeyCode == Keys.C)
//{
//    if (this.Frac.isdrawing) return;
//    Init();
//}
//if (e.KeyCode == Keys.E)
//{
//    if (this.Frac.isdrawing) return;
//    ZoomUp();
//    if (Frac == null)
//    {
//        this.textBox1.Text = "1";
//    }
//    else
//    {
```

```
//      this.textBox1.Text = $"{this.Frac.scale:f3}";
//    }
//}
//if (e.KeyCode == Keys.Q)
//{
//    if (this.Frac.isdrawing) return;
//    ZoomDown();
//    this.label5.Text = $"Масштаб: ";
//    if (Frac == null)
//    {
//        this.textBox1.Text = "1";
//    }
//    else
//    {
//        this.textBox1.Text = $"{this.Frac.scale:f3}";
//    }
//}
//if (e.KeyCode == Keys.B)
//{
//    if (this.Frac.isdrawing) return;
//    if (checkBox_buffer.Checked)
//    {
//        DoubleBuffered = false;
//        checkBox_buffer.Checked = false;
//    }
//    else
//    {
//        DoubleBuffered = true;
//        checkBox_buffer.Checked = true;
//    }
//}
//if (e.KeyCode == Keys.Enter)
//{
//    if (this.Frac.isdrawing) return;
```

```
// Rewrite();
//}
//if (e.KeyCode == Keys.L && this.textBox1.Text == "42" &&
this.textBox_max_depth_of_rec.Text == "42")
//{
// fenableformwhendrawing ^= true;
// DropExWindow("В чём заключается смысл Жизни: " +
(fenableformwhendrawing ? "42" : "I dont know"));
//}
#endregion
}

/// <summary>
/// Вывод сообщения об ошибке
/// </summary>
/// <param name="s"></param>
void DropExWindow(string s)
{
    if (flagmb) return;
    flagmb = true;
    if (MessageBox.Show(s) == DialogResult.OK)
    {
        flagmb = false;
    }
}

#region Save

/// <summary>
/// Изменение выбранности пункта меню Перезаписать и статуса
Перезаписать при записи данных в файл
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
```

```
private void toolStripMenuItem1_Click(object sender, EventArgs e)
{
    toolStripMenuItem1.Checked ^= true;
    rewrite = toolStripMenuItem1.Checked;
}

/// <summary>
/// Сохранение таблицы
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void saveToolStripMenuItem1_Click(object sender, EventArgs
e)
{
    try
    {
        if (name.Length == 0)
        {
            saveAsToolStripMenuItem_Click(sender, e);
        }
        else
        {
            if (isadded) CSVconv.SaveStrtoCSV("'" + name,
dataGridView1.Rows, separ, rewrite, this.encode);
            else CSVconv.SaveStrtoCSV("'" + name, data, separ,
rewrite, this.encode);
        }
        issaved = true;
    }
    catch (CSVException ex)
    {
        DropExWindow("Ошибка при сохранении файла\n" + ex.Message +
ex.InnerException?.Message);
    }
}
```



```
        catch (NullReferenceException ex)
        {
            DropExWindow("Невозможно сохранить несуществующий объект\n"
+ ex.Message);
        }
        catch (ArgumentNullException ex)
        {
            DropExWindow("Невозможно сохранить объект\n" + ex.Message);
        }
        catch (System.Runtime.InteropServices.ExternalException ex)
        {
            DropExWindow("Невозможно сохранить объект\n" + ex.Message);
        }
        catch (Exception ex)
        {
            DropExWindow("" + ex.Message);
        }
    }

    /// <summary>
    /// Сохранение таблицы как (старый интерфейс)
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void saveDialogToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        try
        {
            SaveFileDialogWithEncoding ofd = new
SaveFileDialogWithEncoding();
            ofd.DefaultExt = ".csv";
            ofd.EncodingType = EncodingType.UTF8;
```

```
ofd.Filter = "CSV files (';' *.csv)|*.csv|TSV files ('\t'
*.csv)|*.csv|SCSV files (';' *.csv)|*.csv|TSV files ('\t' *.tsv)|*.tsv|SCSV
files (';' *.scsv)|*.scsv|All files (*.*)|*.*";
    if (separ == ';') { ofd.FilterIndex = 3; }
    if (separ == ',') { ofd.FilterIndex = 1; }
    if (separ == '\t') { ofd.FilterIndex = 2; }
    //if (ofd.ShowDialog() == DialogResult.OK)
    //{
    //    MessageBox.Show(String.Format("Name={0},
Encoding={1}", ofd.FileName, ofd.EncodingType));
    //}
    //FolderBrowserDialog FBD = new FolderBrowserDialog();
    //Encoding encode = null;
    /*
    *
    *
    UTF8=65001, //Encoding.UTF8
    //UTF8WithPreamble,
    Unicode=1201, //Encoding.Unicode
    Ansi=1251 //Encoding.Default
    *
    */
    if (ofd.ShowDialog() == DialogResult.OK)
    {
        int enc = 1251;
        switch ((int)ofd.EncodingType)
        {
            case (0):
                enc = 65001; break;
            case (1):
                enc = 1201; break;
            case (2):
                enc = 1251; break;
        }
    }
```

```
        encode = Encoding.GetEncoding((int)enc);
        separ = ofd.FilterIndex-1 == 0 ? ',' : ofd.FilterIndex-
1 == 1 ? '\t' : ofd.FilterIndex-1 == 2 ? ';' : ofd.FilterIndex-1 == 3 ?
'\t' : ofd.FilterIndex-1 == 4 ? ';' :
','; //FBD.FileName[FBD.FileName.Length - 1];
        name = ofd.FileName; //.Remove(ofd.FileName.Length - 1);

        if (isadded) CSVconv.SaveStrtoCSV("" + name,
dataGridView1.Rows, separ, rewrite, this.encode);
        else CSVconv.SaveStrtoCSV("" + name, data, separ,
rewrite, this.encode/*,(char)ofd.EncodingType*/*,ofd.Rewrite*/);
        issaved = true;
    }
}
catch (CSVException ex)
{
    DropExWindow("Ошибка при сохранении файла\n" + ex.Message +
ex.InnerException?.Message);
}
catch (NullReferenceException ex)
{
    DropExWindow("Невозможно сохранить несуществующий объект\n"
+ ex.Message);
}
catch (ArgumentNullException ex)
{
    DropExWindow("Невозможно сохранить объект\n" + ex.Message);
}
catch (System.Runtime.InteropServices.ExternalException ex)
{
    DropExWindow("Невозможно сохранить объект\n" + ex.Message);
}
catch (Exception ex)
{

```

```
        DropExWindow("" + ex.Message);
    }
}

/// <summary>
/// Сохранение таблицы как текстовый файл (новый интерфейс)
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void saveAstxtToolStripMenuItem_Click(object sender,
EventArgs e)
{
    try
    {
        SaveFileDialog FBD = new SaveFileDialog();
        FBD.Filter = "CSV for EXEL files (';' *.csv)|*.csv|All
files (*.*)|*.*";
        //FBD.CreatePrompt = true;
        if (FBD.ShowDialog() == DialogResult.OK)
        {
            separ = FBD.FilterIndex-1 == 0 ? ';' :
',';//FBD.FileName[FBD.FileName.Length - 1];
            name = FBD.FileName;//.Remove(FBD.FileName.Length - 1);
            if (isadded) CSVconv.SaveStrtoCSV("" + name,
dataGridView1.Rows, separ, rewrite, this.encode);
            else CSVconv.SaveStrtoCSV("" + name, data, separ,
rewrite, Encoding.Default);
            issaved = true;
        }
    }
    catch (CSVException ex)
    {
        DropExWindow("Ошибка при сохранении файла\n" + ex.Message +
ex.InnerException?.Message);
    }
}
```

```
    }
    catch (NullReferenceException ex)
    {
        DropExWindow("Невозможно сохранить несуществующий объект\n"
+ ex.Message);
    }
    catch (ArgumentNullException ex)
    {
        DropExWindow("Невозможно сохранить объект\n" + ex.Message);
    }
    catch (System.Runtime.InteropServices.ExternalException ex)
    {
        DropExWindow("Невозможно сохранить объект\n" + ex.Message);
    }
    catch (ArgumentException ex)
    {
        DropExWindow("Невозможно сохранить объект\n" + ex.Message);
    }
    catch (Exception ex)
    {
        DropExWindow("" + ex.Message);
    }
}

/// <summary>
/// Сохранение таблицы как (новый интерфейс)
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void saveAsToolStripMenuItem_Click(object sender, EventArgs
e)
{
    try
    {
```

```
SaveFileDialog FBD = new SaveFileDialog();
FBD.Filter = "CSV files (';' *.csv)|*.csv|TSV files ('\t'
*.csv)|*.csv|SCSV files (';' *.csv)|*.csv|TSV files ('\t' *.tsv)|*.tsv|SCSV
files (';' *.scsv)|*.scsv|All files (*.*)|*.*";
if (separ == ';') { FBD.FilterIndex = 3; }
if (separ == ',') { FBD.FilterIndex = 1; }
if (separ == '\t') { FBD.FilterIndex = 2; }
if (FBD.ShowDialog() == DialogResult.OK)
{
    separ = FBD.FilterIndex - 1 == 0 ? ',' :
FBD.FilterIndex - 1 == 1 ? '\t' : FBD.FilterIndex - 1 == 2 ? ';' :
FBD.FilterIndex - 1 == 3 ? '\t' : FBD.FilterIndex - 1 == 4 ? ';' :
','; //FBD.FileName[FBD.FileName.Length - 1];
    name = FBD.FileName; //.Remove(FBD.FileName.Length - 1);
    if (isadded) CSVconv.SaveStrtoCSV("" + name,
dataGridView1.Rows, separ, rewrite, this.encode);
    else CSVconv.SaveStrtoCSV("" + name, data, separ,
rewrite, this.encode);
    issaved = true;
}
}
catch (CSVException ex)
{
    DropExWindow("Ошибка при сохранении файла\n" + ex.Message +
ex.InnerException?.Message);
}
catch (NullReferenceException ex)
{
    DropExWindow("Невозможно сохранить несуществующий объект\n"
+ ex.Message);
}
catch (ArgumentNullException ex)
{
    DropExWindow("Невозможно сохранить объект\n" + ex.Message);
}
```

```
    }
    catch (System.Runtime.InteropServices.ExternalException ex)
    {
        DropExWindow("Невозможно сохранить объект\n" + ex.Message);
    }
    catch (ArgumentException ex)
    {
        DropExWindow("Невозможно сохранить объект\n" + ex.Message);
    }
    catch (Exception ex)
    {
        DropExWindow("" + ex.Message);
    }
}

/// <summary>
/// Сохранение таблицы как (новый интерфейс)
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void saveAsToolStripMenuItem_Click_1(object sender,
EventArgs e)
{
    fileToolStripMenuItem.HideDropDown();
    saveAsToolStripMenuItem_Click(sender, e);
}

/// <summary>
/// Analog of save
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void saveToolStripButton_Click(object sender, EventArgs e)
```

```
{
    saveToolStripMenuItem1_Click(sender, e);
}

#endregion

#region Load

/// <summary>
/// Load data from file
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void loadToolStripMenuItem_Click(object sender, EventArgs
e)
{
    if (!SaveorLose(issaved))
    {
        return;
    }
    if (encodingToolStripComboBox1.Text != "Encoding Type")
    {
        //encodingToolStripComboBox1.Name
        encode =
Encoding.GetEncoding((int)int.Parse(encodingToolStripComboBox1.Text.Split('
')[0]));
    }
    if (typeToolStripMenuItem.Text != "Separator Type")
    { //typeToolStripMenuItem.Name
        separ =
CSVconv.GetSeparType(typeToolStripMenuItem.Text[0]);
    }
}
```



```

    }
    //contextMenuStrip1.Show();
    try
    {
        OpenFileDialog FBD = new OpenFileDialog();
        FBD.AddExtension = false;
        if (name.Length > 0) { FBD.FileName = name; }
        FBD.Filter = "CSV files (';' *.csv)|*.csv|TSV files ('\t'
*.csv)|*.csv|SCSV files (';' *.csv)|*.csv|TSV files ('\t' *.tsv)|*.tsv|SCSV
files (';' *.scsv)|*.scsv|All files (*.*)|*.*";
        if (separ == ';') { FBD.FilterIndex = 3; }
        if (separ == ',') { FBD.FilterIndex = 1; }
        if (separ == '\t') { FBD.FilterIndex = 2; }

        if (FBD.ShowDialog() == DialogResult.OK)
        {
            if (isadded) { toolStripMenuItem5_Click(sender, e); }
            separ = FBD.FilterIndex-1 == 0 ? ',' : FBD.FilterIndex
- 1 == 1 ? '\t' : FBD.FilterIndex - 1 == 2 ? ';' : FBD.FilterIndex - 1 == 3
? '\t' : FBD.FilterIndex - 1 == 4 ? ';' :
','; //FBD.FileName[FBD.FileName.Length - 1];
            name = FBD.FileName; //.Remove(FBD.FileName.Length - 1);
            datas = CSVconv.fscanf(name, this.encode);
            data=CSVconv.LoadCSVtoStr("" + name, separ,
this.encode);

            UpdateData(data, out opop, out adr);
            UpdateGrid();
            issaved = true;
        }
    }
    catch (CSVException ex)
    {
        DropExWindow("Ошибка при загрузке данных из файла\n" +
ex.Message + ex.InnerException?.Message);
    }
}

```

```
    }
    catch (NullReferenceException ex)
    {
        DropExWindow("Невозможно загрузить несуществующий объект\n"
+ ex.Message);
    }
    catch (ArgumentNullException ex)
    {
        DropExWindow("Невозможно загрузить объект\n" + ex.Message);
    }
    catch (System.Runtime.InteropServices.ExternalException ex)
    {
        DropExWindow("Невозможно загрузить объект\n" + ex.Message);
    }
    catch (Exception ex)
    {
        DropExWindow("" + ex.Message);
    }
}

/// <summary>
/// Load data from file
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void loadToolStripMenuItem_Click(object sender, EventArgs
e, bool f = true)
{
    if (!SaveorLose(issaved))
    {
        return;
    }
    //contextMenuStrip1.Show(); c
    try
```

```

    {
        if (f || datas == null)
        {
            #region comment
            //SaveFileDialog FBD = new SaveFileDialog();
            //FBD.Filter = "CSV files (';' *.csv)|*.csv|TSV files
('\'t' *.csv)|*.csv|SCSV files (';' *.csv)|*.csv|TSV files ('\'t'
*.tsv)|*.tsv|SCSV files (';' *.scsv)|*.scsv|All files (*.*)|*.*";
            //if (FBD.ShowDialog() == DialogResult.OK)
            //{
                //    separ = FBD.FilterIndex-1 == 0 ? ',' :
FBD.FilterIndex-1 == 1 ? '\t' : FBD.FilterIndex-1 == 2 ? ';' :
FBD.FilterIndex-1 == 3 ? '\t' : FBD.FilterIndex-1 == 4 ? ';' :
','; //FBD.FileName[FBD.FileName.Length - 1];
                //    name = FBD.FileName; //.Remove(FBD.FileName.Length
- 1);

                //    datas = CSVconv.fscanf(name);
                //    data = CSVconv.LoadCSVtoStr("'" + name, separ);
                //}
            #endregion
            loadToolStripMenuItem_Click(sender, e);
        }
        else
        {
            List<List<string>> res = new List<List<string>>();
            for (int i = 0; i < datas.Length; i++)
            {
                res.Add(CSVconv.ConvertCSVlinetoListstr(datas[i],
separ));
            }
            data = res;
        }
        UpdateGrid();
        issaved = true;
    }

```

```
    }
    catch (CSVException ex)
    {
        DropExWindow("Ошибка при загрузке файла\n" + ex.Message +
ex.InnerException?.Message);
    }
    catch (NullReferenceException ex)
    {
        DropExWindow("Невозможно загрузить несуществующий объект\n"
+ ex.Message);
    }
    catch (ArgumentNullException ex)
    {
        DropExWindow("Невозможно загрузить объект\n" + ex.Message);
    }
    catch (System.Runtime.InteropServices.ExternalException ex)
    {
        DropExWindow("Невозможно загрузить объект\n" + ex.Message);
    }
    catch (Exception ex)
    {
        DropExWindow("" + ex.Message);
    }
}

/// <summary>
/// Reopen file with encoding
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void ReopenToolStripMenuItem_Click(object sender, EventArgs
e)
{
```

```
        if (!SaveorLose(issaved))
        {
            return;
        }
        //if (ReopenToolStripMenuItem.DropDownItems[0] != null) {
//encodingToolStripComboBox1.Name
        //encode =
Encoding.GetEncoding((int)int.Parse(ReopenToolStripMenuItem.DropDownItems[0
].Text.Split(' ')[0]));}
        //separ =
CSVconv.GetSeparType(ReopenToolStripMenuItem.DropDownItems[1].Text[0]);
//typeToolStripMenuItem.Name
        bool reopen = false;
        if (encodingToolStripComboBox1.Text != "Encoding Type") {
reopen = encode !=
Encoding.GetEncoding((int)int.Parse(encodingToolStripComboBox1.Text.Split('
')[0]));
        //encodingToolStripComboBox1.Name
        encode =
Encoding.GetEncoding((int)int.Parse(encodingToolStripComboBox1.Text.Split('
')[0]));}
        if (typeToolStripMenuItem.Text != "Separator Type") {
//typeToolStripMenuItem.Name
        separ = CSVconv.GetSeparType(typeToolStripMenuItem.Text[0]);}
        loadToolStripMenuItem_Click(sender, e, reopen);
        issaved = false;
    }

    /// <summary>
    /// Quick load
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void openToolStripButton_Click(object sender, EventArgs e)
```

```
{
    loadToolStripMenuItem_Click(sender, e);
}
```

```
#endregion
```

```
/// <summary>
```

```
/// Новая таблица
```

```
/// </summary>
```

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

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

```
private void newToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
    if (!SaveorLose(issaved))
    {
        return;
    }
    //if (this.Frac != null && this.Frac.isdrawing) return;
    //(new Form1()).ShowDialog(new Form1());
    dataGridView1.Rows.Clear();
    data = new List<List<string>>();
    data.Add(new List<string>());
    if (dataGridView1.Columns.Count > 0)
    {
        for (int i = 0; i < dataGridView1.Columns.Count; ++i)
        {
            data[0].Add(((string)(dataGridView1.Columns[i].HeaderText)));
        }
        data.Add(new List<string>());
        for (int i = 0; i < dataGridView1.Columns.Count; ++i)
        {
```

```
        data[1].Add(((string)("")));
    }
}
else
{
    int n = 1;
    for (int i = 0; i < n; ++i)
    {
        data[0].Add(((string)("")));
    }
    data.Add(new List<string>());
    for (int i = 0; i < n; ++i)
    {
        data[1].Add(((string)("")));
    }
}
UpdateData(data, out opop, out adr);
UpdateGrid();
issaved = true;
}

#region table_edit

/// <summary>
/// Event New table
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void newToolStripButton_Click(object sender, EventArgs e)
{
    newToolStripMenuItem_Click(sender, e);
}

/// <summary>
```

```
    /// Add row
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void rowToolStripMenuItem_Click(object sender, EventArgs e)
    {
        try {
            //
            if (dataGridView1.SelectedRows.Count == 0 && data != null)
{ dataGridView1.Rows.Add(); }
            else {
dataGridView1.Rows.Insert(dataGridView1.SelectedRows[dataGridView1.Selected
Rows.Count - 1].Index); }
            return;

            if (dataGridView1.Rows.Count == 0 ||
dataGridView1.SelectedRows.Count == 0) {
                dataGridView1.Rows.Add();
                if (datas == null)
                {
                    data = new List<List<string>>>();
                    datas = new string[0];
                }
                data.Add(new List<string>(data[0].Count));
                for(int i = 0; i < data[data.Count - 1].Count; i++)
                {
                    data[data.Count - 1][i] = "";
                }
                string[] datas2 = datas;
                Array.Resize(ref datas2, datas.Length + 1);
                datas = datas2;
                datas[datas.Length - 1] = "";
            }
        }
    }
}
```



```
        }
        else {

data.Insert(dataGridView1.SelectedRows[dataGridView1.SelectedRows.Count -
1].Index, new List<string>(data[0].Count));
            for (int i = 0; i <
data[dataGridView1.SelectedRows[dataGridView1.SelectedRows.Count -
1].Index].Count; i++)
            {

data[dataGridView1.SelectedRows[dataGridView1.SelectedRows.Count -
1].Index][i] = "";
                }
                string[] datas2 = datas;
                Array.Resize(ref datas2, datas.Length + 1);
                datas = datas2;
                for (int i = 0; i < data.Count; i++) {

datas[i]=CSVconv.ConvertListstrtoCSVline(data[i],separ);
                }

dataGridView1.Rows.Insert(dataGridView1.SelectedRows[dataGridView1.Selected
Rows.Count-1].Index);

        }

    }
    catch(System.InvalidOperationException ex)
    {
        DropExWindow(ex.Message + "\nСоздайте колонку");
    }
    catch(Exception ex)
    {
        DropExWindow("\nСоздайте колонку!");
    }
}
```

```
    }
}

/// <summary>
/// Added rows event
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void dataGridView1_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
{
    //e.RowIndex;
    if (isadded) { return; }
    for (int k = 0; k < e.RowCount; k++) {
        data.Insert(e.RowIndex+sn, new
List<string>(data[0].Count));
        for (int i = 0; i < data[0].Count; i++)
        {
            data[e.RowIndex + sn].Add("");
            //data[e.RowIndex+sn][i] = "";
        }
        string[] datas2 = datas;
        Array.Resize(ref datas2, data.Count + 1);
        datas = datas2;
        for (int i = 0; i < data.Count; i++)
        {
            datas[i] = CSVconv.ConvertListstrtoCSVline(data[i],
separ);
        }

//dataGridView1.Rows.Insert(dataGridView1.SelectedRows[dataGridView1.Select
edRows.Count - 1].Index);
    }
    UpdateData(data, out opop, out adr);
}
```

```
        int _ = 0;
    }

    /// <summary>
    /// Remove row from data
    /// </summary>
    /// <param name="d"></param>
    int removeRowInData(DataGridViewRow d)
    {
        int i = 0;
        for(i = 0; i < data.Count; ++i)
        {
            bool flag = true;
            for(int j = 0; j < Math.Min(data[i].Count, d.Cells.Count);
++j)
            {
                if(data[i][j]!=(string)(d.Cells[j].Value)) { flag =
false; break; }
            }
            if (flag) {
                data.RemoveAt(i);
                List<string> datasp = datas.ToList();
                datasp.RemoveAt(i);
                datas = datasp.ToArray();
                return i;
            }
        }
        return -1;
    }

    /// <summary>
    /// Deleted rows event
    /// </summary>
    /// <param name="sender"></param>
```

```
    /// <param name="e"></param>
    private void dataGridView1_DeleteRow(object sender,
DataGridViewRowEventArgs e)
    {
        if(!SaveorLose(false,"Вы точно хотите удалить ячейку?")) {
UpdateGrid(); return; }
        if (removeRowInData(e.Row) == -1) { DropExWindow("Not a string
in data.\n Please reopen table"); }
        UpdateData(data, out opop, out adr);
    }

    /// <summary>
    /// Add column
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void columnToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        if (datas == null)
        {
            data = new List<List<string>>();
            data.Add(new List<string>());
            //data.Add(new List<string>());
            datas = new string[1];
        }
        string str = columnNameToolStripMenuItem.Text;
        var column = new DataGridViewTextBoxColumn();
        column.HeaderText = str;
        column.Name = str;
        column.CellTemplate = new DataGridViewTextBoxCell();
        dataGridView1.Columns.Add(column);
        //dataGridView1.Columns[dataGridView1.Columns.Count-1].SortMode
= DataGridViewColumnSortMode.Programmatic;
```

```
//data[0].Add("");
for (int i = 0; i < data.Count-1; i++)
{
    data[i].Add("");
    datas[i] = CSVconv.ConvertListstrtoCSVline(data[i], separ);
}
UpdateData(data, out opop, out adr);
}

/// <summary>
/// Edit data of some cell eventHandler
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void editCellData(object sender,
System.Windows.Forms.DataGridViewCellEventArgs e) {
    if(isadded) { return; }
    int ci = e.ColumnIndex;
    int ri = e.RowIndex;
    data[ri+sn][ci] =
(string)dataGridView1.Rows[ri].Cells[ci].Value;
    datas[ri+sn] = CSVconv.ConvertListstrtoCSVline(data[ri+sn],
separ);
    UpdateData(data, out opop, out adr);
}

#endregion

#region Find

/// <summary>
/// Create AhoCorasic exemplar
/// </summary>
/// <param name="which"></param>
```

```
    /// <returns></returns>
    private AhoCorasik FindSubstring(params string[] which) //string
where, vector<pair<vector<int>, string>>
    {
        int max = 0;
        foreach(var i in which) { max = Math.Max(max, i.Length); }
        AhoCorasik act = new AhoCorasik(which, which.Length*max);

        return act; //.find(where);
    }

    /// <summary>
    /// Find equal string
    /// </summary>
    /// <param name="s"></param>
    /// <param name="w"></param>
    /// <returns></returns>
    private int findInString(string s, string[] w)
    {
        int i = 0;
        for(i = 0; i < w.Length; i++)
        {
            if (s == w[i]) return i;
        }
        return i;
    }

    /// <summary>
    /// Only one of equals string
    /// </summary>
    /// <param name="s"></param>
    /// <returns></returns>
    private string[] deleteCopy(string[] s)
```

```
{
    List<string> sr = new List<string>();
    for(int i = 0; i < s.Length; ++i)
    {
        bool flag = true;
        foreach (string j in sr)
        {
            if (j == s[i])
            {
                flag = false;
                break;
            }
        }
        if (flag) sr.Add(s[i]);
    }
    return sr.ToArray();
}
```

```
/// <summary>
```

```
/// Find substrings in string
```

```
/// </summary>
```

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

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

```
private void findToolStripMenuItem1_Click(object sender, EventArgs
```

e)

```
{
    if (datas == null) { return; }
    string[] ws=ssesepToolStripMenuItem.Text.Split(new char[] { ';'
}, StringSplitOptions.RemoveEmptyEntries);
    ws = deleteCopy(ws);
    List<string>[] str = new List<string>[ws.Length];
    for(int i = 0; i < ws.Length; ++i)
    {
        str[i] = new List<string>();
    }
}
```

```

        str[i].Add(ws[i] + ": \n\r");
    }
    AhoCorasik act = FindSubstring(ws);
    for (int di = 0; di <
(isadded?dataGridView1.Rows.Count:data.Count); ++di){
        int colnum=0;
        if (isadded)
        {
            if(!dataGridView1.Rows[di].Visible) { continue; }
            try
            {
                for (int ii = 0; ii <
dataGridView1.Rows[di].Cells.Count; ++ii)
                {
                    string i =
(string)dataGridView1.Rows[di].Cells[ii].Value;
                    if (i == null) { continue; }
                    vector<pair<vector<int>, string>> v =
act.find(i);

                    for (int j = 0; j < v.size(); ++j)
                    { //ws
                        if (v[j].first.size() == 0) { continue; }
                        int n = findInString(v[j].second, ws);
                        str[n].Add("(R:C)=( " +
(string)dataGridView1.Rows[di].Cells[0].Value + ":" + (colnum + 1) + "): "
+ v[j].first.toString(",") + '\n' + '\r');
                    }
                    colnum++;
                }
            }
            catch(Exception ex) { }
        }
        else
        {

```



```

        foreach (string i in data[di])
        {
            vector<pair<vector<int>, string>> v = act.find(i);
            for (int j = 0; j < v.size(); ++j)
            { //ws
                if (v[j].first.size() == 0) { continue; }
                int n = findInString(v[j].second, ws);
                str[n].Add("(R:C)=( " +
(string)dataGridView1.Rows[di].Cells[0].Value + ":" + (colnum + 1) + "): "
+ v[j].first.toString(",") + '\n' + '\r');
            }
            colnum++;
        }
    }
    vector<string> outs = new vector<string>();
    for(int i = 0; i < str.Length; ++i)
    {
        bool flag = true;
        foreach(var j in str[i]) {
            outs.append(j);
            if (flag) { outs.append("Count: " + (str[i].Count -
1)); flag = false; }
        }

    }
    //DropExWindow(outs.toString("\n"));
    this.textBox1.Lines = outs.ToArray();
    //textBox1.Text=outs.toString("\n\r");
}

#endregion

```

```
#region Sort
```

```
//private Button sortButton = new Button();
```

```
private void dataGridView1_SortCompare(object sender,  
DataGridViewSortCompareEventArgs e)
```

```
{  
    // Try to sort based on the cells in the current column.  
    bool flag = true;  
    try  
    {  
        string e1 = (string)e.CellValue1;  
        string e2 = (string)e.CellValue2;  
        if (e.CellValue1 == null)  
        {  
            e1 = "";  
        }  
        if (e.CellValue2 == null)  
        {  
            e2 = "";  
        }  
        if (flag && (e.Column.Name == "OPOPNumber" || e.Column.Name  
== "GLOBALID"))  
        {  
            flag = false;  
            int n = Math.Max(e1.Length, e2.Length);  
            e.SortResult =  
String.Compare((int.Parse(e1)).ToString("D" + n),  
(int.Parse(e2)).ToString("D" + n));  
        }  
        if (flag && e.Column.Name == "ROWNUM")  
        {  
            flag = false;  
            int n = Math.Max(e1.Length, e2.Length);
```

```
        e.SortResult =
String.Compare((int.Parse(e1)).ToString("D" + n),
(int.Parse(e2)).ToString("D" + n));
    }

    if (flag) { e.SortResult = System.String.Compare(e1, e2); }
    // If the cells are equal, sort based on the ID column.
    if (e.SortResult == 0 && e.Column.Name != "ROWNUM")
    {
        int n = Math.Max(e1.Length, e2.Length);
        e.SortResult =
String.Compare((int.Parse(e1)).ToString("D" + n),
(int.Parse(e2)).ToString("D" + n));
    }
    e.Handled = true;
}
catch (ArgumentNullException ex)
{
    DropExWindow("Значение ячейки имеет неверный формат\n" +
ex.Message);
}
catch (FormatException ex)
{
    DropExWindow("Значение ячейки имеет неверный формат\n" +
ex.Message);
}
catch (OverflowException ex)
{
    DropExWindow("Значение ячейки имеет неверный формат\n" +
ex.Message);
}
catch (Exception ex)
{

```

```
                DropExWindow("Значение ячейки имеет неверный формат\n" +
ex.Message);
            }
        }

#endregion

#region Filter

/// <summary>
/// Start filtering
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void filterToolStripMenuItem_Click(object sender, EventArgs
e)
{
    issaved = false;
    bool flag = true;
    if (!isadded) {
        DropExWindow("Для начала фильтрации установите галочку в
поле Filtering");
        return;
    }
    try
    {
        string colname = nameOfColumnToolStripMenuItem.Text;
        string[] filtert = textToolStripMenuItem.Text.Split(';');
        //BindingSource filter = new BindingSource();
        //filter.DataSource = dataGridView1.Columns[colname];
    }
}
```

```
        filter.Filter = colname + " Like " + filtert;
    //if (filtert == "")
    //{
    //    filter.Filter = String.Empty;
    //}
    //else
    //{
    //    filter.Filter = colname + " Like '%" + filtert +
"%'";

    //}

    ///DataSet ds = new DataSet();
    ///ds = ((DataSet)(dataGridView1.DataSource));
    ///DataSet deT = (DataSet)(dataGridView1.DataSource);
    ///DataView dv = ds.Tables[0].DefaultView;
    ///dv.RowFilter = string.Format("country LIKE '{0}%',
filtert);

    ///dataGridView1.DataSource = dv;

    ///((dataGridView1.DataSource as
DataTable).DefaultView.RowFilter = string.Format(""+colname+" = '{0}'",
filtert);

    dataGridView1.DataSource = filter;
    AhoCorasik act = FindSubstring(filtert);
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        if (filtert!=null &&
act.find((string)(dataGridView1.Rows[i].Cells[dataGridView1.Columns[colname
].Index.Value)).size() == 0)
        {
            dataGridView1.Rows[i].Visible = false;
            flag = false;
            continue;
        }
    }
}
```

```
        }
        dataGridView1.Rows[i].Visible = true;
        flag = false;

    }
}
catch(Exception ex)
{
    if(flag) DropExWindow("Проверьте правильность введенных
полей\n" + ex.Message);
}
}

/// <summary>
/// Change filtering state
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void toolStripMenuItem5_Click(object sender, EventArgs e)
{
    issaved = false;
    isadded ^= true;
    toolStripMenuItem5.Checked = isadded;
    if (!isadded)
    {
        dataGridView1.DataSource = null;
    }
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        dataGridView1.Rows[i].Visible = true;
    }
}
```

```
    /// <summary>
    /// Filter for selected opop's number
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void toolStripButton1_Click(object sender, EventArgs e)
    {
        if(!isadded)
        {
            DropExWindow("Для начала фильтрации установите галочку в
поле Filtering");
            return;
        }
        issaved = false;
        if (!(dataGridView1.SelectedRows.Count != 0 && data != null)) {
DropExWindow("Выберите строку с ОПОП для фильтрации"); return; }
        for(int i = 0; i < dataGridView1.Rows.Count; ++i)
        {
            try
            {
                if (!dataGridView1.Rows[i].Visible) { continue; }
                bool flag = true;
                foreach (System.Windows.Forms.DataGridViewRow j in
dataGridView1.SelectedRows)
{ //dataGridView1.SelectedRows[dataGridView1.SelectedRows.Count-1]
                    if
(((string)(dataGridView1.Rows[i].Cells["OPOPNumber"].Value)) ==
((string)(j.Cells["OPOPNumber"].Value))) { flag = false; break; }
                }
                if (flag) { dataGridView1.Rows[i].Visible = false; }
            }
            catch(Exception ex) { }
        }
    }
}
```

```
#endregion
```

```
#endregion
```

```
#region backend
```

```
/// <summary>
```

```
/// Конструктор по умолчанию
```

```
/// </summary>
```

```
public Form1()
```

```
{
```

```
    InitializeComponent();
```

```
}
```

```
/// <summary>
```

```
/// Конструктор от строки
```

```
/// </summary>
```

```
public Form1(string s):this()
```

```
{
```

```
    if (s != "")
```

```
    {
```

```
        DropExWindow(s);
```

```
    }
```

```
}
```

```
/// <summary>
```

```
/// Load Form1
```

```
/// </summary>
```



```
/// <param name="sender"></param>
/// <param name="e"></param>
private void Form1_Load(object sender, EventArgs e)
{

}

/// <summary>
/// Update data of ОПОП
/// </summary>
/// <param name="data"></param>
/// <param name="opop"></param>
/// <param name="adr"></param>
private void UpdateData(List<List<string>> data, out List<ОПОП>
opop, out List<Расположение> adr)
{
    issaved = false;
    opop = new List<ОПОП>();
    adr = new List<Расположение>();
    if (data[0].Count < 11) { return; }
    bool flage = true;
    foreach (var i in data)
    {
        if (flage)
        {
            flage = false;
            continue;
        }
        if (i.Count < 11) { continue; }
        bool flag = true;
        int k = 0;
        for (; k < adr.Count; ++k)
        {
```

```
        if (adr[k].AdmArea == i[3] && adr[k].District == i[4])
        {
            flag = false;
            break;
        }
    }
    if (flag)
    {
        adr.Add(new Расположение());
    }
    опор.Add(new ОПОП(i, adr[k]));
}

}

/// <summary>
/// Изменение выбранности пункта меню Поверх остальных окон и
статуса Поверх остальных окон основного окна
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void overAllWindowsToolStripMenuItem_Click(object sender,
EventArgs e)
{
    overAllWindowsToolStripMenuItem.Checked ^= true;
    TopMost = overAllWindowsToolStripMenuItem.Checked;
}

///// <summary>
///// Загрузка файла .CSV
///// </summary>
///// <param name="sender"></param>
///// <param name="e"></param>
```

```
    /// <summary>
    /// Обновление таблицы
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    public void UpdateGrid(object sender = null, EventArgs e = null)
    {
        //isadded = true;
        bool flag = false;
        if (!isadded) { toolStripMenuItem5_Click(sender, e); flag =
true; }

        sn = en = -1;
        int ssn=0;
        if (sne) { if (!int.TryParse(toolStripMenuItem3.Text, out ssn)
|| ssn < 1) { sn = 1; DropExWindow("Uncorrect format of value \"From\"
(needed>=1)"); return; } }
        sn = ssn;
        if (ene) { if (!int.TryParse(toolStripMenuItem4.Text, out ssn)
|| ssn <= sn || ssn>=data.Count) { en = data.Count-1;
DropExWindow("Uncorrect format of value \"To\" (Count of
rows>needed>From)"); return; } }
        en = ssn;
        sn = Math.Max(sn,1);
        if (en <= sn) { en = data.Count-1; }
        if(data==null) { return; }
        int maxlen = data[0].Count;
        for(int i = sn; i < en; ++i)
        {
            maxlen = Math.Max(maxlen, data[i].Count);
        }
        dataGridView1.Columns.Clear();
        for (int i = 0; i < maxlen; ++i)
        {
            string str = "";
```

```
        if (i < data[0].Count)
        {
            str = data[0][i];
        }
        if (data[0].Count > i) { str = data[0][i]; }
        var column = new DataGridViewTextBoxColumn();
        column.HeaderText = str;
        column.Name = str;
        column.CellTemplate = new DataGridViewTextBoxCell();
        dataGridView1.Columns.Add(column);
    }
    for (int j = sn; j < en; ++j) {
        dataGridView1.Rows.Add(data[j].ToArray());
    }
    //isadded = false;
    if (flag) { toolStripMenuItem5_Click(sender, e); flag = false;
}

    Invalidate();
}

/// <summary>
/// Закрытие приложения
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void Form1Closed(object sender, EventArgs e)
{

    this.Close();
}

/// <summary>
/// Хотите ли вы продолжить и потерять изменения?
```

```
/// </summary>
/// <param name="s">Предупреждающая строка</param>
/// <returns>Заккрыть (false) OK (true)</returns>
public static bool SaveorLose(bool f=false, string s="При
продолжении несохранённые данные будут потеряны.\nВы точно хотите
продолжить?\n") {///Для отмены закройте это окно.
    if (f) { return f; }
    bool flag =
    MessageBox.Show(text:s,caption:"WARNING",buttons:MessageBoxButtons.OKCancel
,icon:MessageBoxIcon.Warning) == DialogResult.OK;
    return flag;
    //return f || MessageBox.Show(s) == DialogResult.OK;
}
```

```
/// <summary>
/// Отлов события Закрытие программы и предложение сохранить данные
или выйти из программы
/// </summary>
/// <param name="e"></param>
protected override void OnFormClosing(FormClosingEventArgs e)
{
    if (datas == null || issaved)
    {
        Dispose();
    }
    else
    {
        if (SaveorLose(issaved))
        {
            Dispose();
        }
        else
        {

```

```
        e.Cancel = true;
    }
}

/// <summary>
/// Переопределение перерисовки окна
/// </summary>
/// <param name="e"></param>
protected override void OnPaint(PaintEventArgs e)
{
    try
    {
        this.Text = name.Split('\\')[name.Split('\\').Length-1];
        //if (datas != null) { UpdateGrid(); }
        base.OnPaint(e);
    }
    catch (Exception ex)
    {
        DropExWindow("" + ex.Message);
    }
}

/// <summary>
/// Заккрытие контекстного меню с
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void ContextMenuItem_Closing(object sender, CancelEventArgs
e)
{
    if (IS_AUTO_UPDATE_ES) {
        if (encodingToolStripComboBox1.Text != "Encoding Type")
        {
```

```
        //encodingToolStripComboBox1.Name
        encode =
Encoding.GetEncoding((int)int.Parse(encodingToolStripComboBox1.Text.Split('
')[0]));

        }
        if (typeToolStripMenuItem.Text != "Separator Type")
        { //typeToolStripMenuItem.Name
            separ =
CSVconv.GetSeparType(typeToolStripMenuItem.Text[0]);
        }
    }
    //Invalidate();
}

/// <summary>
/// Обновление номера начальной колонки
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void fromToolStripMenuItem_Click(object sender, EventArgs
e)
{
    sne^= true;
    fromToolStripMenuItem.Checked = sne;
}

/// <summary>
/// Обновление номера конечной колонки
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void toToolStripMenuItem_Click(object sender, EventArgs e)
{
    ene ^= true;
```

```
        toToolStripMenuItem.Checked = ene;
    }
```

```
#endregion
```

```
}
```

```
}
```

Program.cs

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Threading.Tasks;
```

```
using System.Windows.Forms;
```

```
namespace GerasimenkoER_KDZ3_v2
```

```
{
```

```
    static class Program
```

```
    {
```

```
        /// <summary>
```

```
        /// The main entry point for the application.
```

```
        /// </summary>
```

```
        [STAThread]
```

```
        static void Main()
```

```
        {
```

```
            //Комменты, так как иначе она не убивается (в самом теле также  
потёр все Catche для эксперимента)
```

```
            start: bool flag = false;
```

```
            string s = "";
```

```
            try
```

```
            {
```

```
                flag = false;
```

```
                Application.EnableVisualStyles();
```



```
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new Form1(s));

}
catch (Exception ex) { flag = true; s = "Возникла ошибка
связанная с попыткой вашей операционной системы принудительно закрыть
данную программу.\nПрограмма вступила в неравный бой с системой, но не
справилась с партией и была отправлена на уничтожение.\nОна долго ждала
своей миллисекунды и наконец Программа смогла победить и была
перезапущена\n" + ex.Message; }
finally { }
if (flag) { goto start; }
//Application.Run(new Find());
}
}
}
}
SaveFileDialog.cs
//www.codeproject.com/Articles/8086/Extending-the-save-file-dialog-class-
in-NET

#region usings

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Runtime.InteropServices;
using System.IO;
using System.Text;

public partial class Form1 : System.Windows.Forms.Form
{
```

```
//private void button1_Click(object sender, System.EventArgs e)
//{
//    SaveFileDialogWithEncoding ofd=new SaveFileDialogWithEncoding();
//    ofd.DefaultExt="sql";
//    ofd.EncodingType=EncodingType.UTF8;
//    ofd.Filter="SQL files (*.sql)|*.sql|All files (*.*)|*.*";
//    if (ofd.ShowDialog()==DialogResult.OK)
//    {
//        MessageBox.Show(String.Format("Name={0}, Encoding={1}",
ofd.FileName, ofd.EncodingType));
//    }
//}
}
```

#endregion

//The interesting bit starts here

//note the order of these is important

```
public enum EncodingType
{
    UTF8=0, //Encoding.UTF8
    //UTF8WithPreamble,
    Unicode, //Encoding.Unicode
    Ansi //Encoding.Default
}
```

public class SaveFileDialogWithEncoding

```
{
    private delegate int OFNHookProcDelegate(int hdlg, int msg, int
wParam, int lParam);
    #region metodata
```

```
private int m_LabelHandle=0;
private int m_ComboHandle=0;
private int m_FilterIndex=0;

private string m_Filter="";
private string m_DefaultExt="";

private string m_FileName="";

private EncodingType m_EncodingType;
private Screen m_ActiveScreen;

[StructLayout(LayoutKind.Sequential, CharSet=CharSet.Auto)]
private struct OPENFILENAME
{
    public int lStructSize;
    public IntPtr hwndOwner;
    public int hInstance;
    [MarshalAs(UnmanagedType.LPCTSTR)] public string lpstrFilter;
    [MarshalAs(UnmanagedType.LPCTSTR)] public string
lpstrCustomFilter;
    public int nMaxCustFilter;
    public int nFilterIndex;
    [MarshalAs(UnmanagedType.LPCTSTR)] public string lpstrFile;
    public int nMaxFile;
    [MarshalAs(UnmanagedType.LPCTSTR)] public string lpstrFileName;
    public int nMaxFileName;
    [MarshalAs(UnmanagedType.LPCTSTR)] public string lpstrInitialDir;
    [MarshalAs(UnmanagedType.LPCTSTR)] public string lpstrTitle;
    public int Flags;
    public short nFileOffset;
    public short nFileExtension;
    [MarshalAs(UnmanagedType.LPCTSTR)] public string lpstrDefExt;
```

```
    public int lCustData;
    public OFNHookProcDelegate lpfnHook;
    [MarshalAs(UnmanagedType.LPTStr)] public string lpTemplateName;
    //only if on nt 5.0 or higher
    public int pvReserved;
    public int dwReserved;
    public int FlagsEx;
}
```

```
[DllImport("Comdlg32.dll", CharSet=CharSet.Auto, SetLastError=true)]
private static extern bool GetSaveFileName(ref OPENFILENAME lpofn);
```

```
[DllImport("Comdlg32.dll")]
private static extern int CommDlgExtendedError();
```

```
[DllImport("user32.dll")]
private static extern bool SetWindowPos(int hWnd, int
hWndInsertAfter, int X, int Y, int cx, int cy, uint uFlags);
```

```
private struct RECT
{
    public int Left;
    public int Top;
    public int Right;
    public int Bottom;
}
```

```
private struct POINT
{
    public int X;
    public int Y;
}
```

```
private struct NMHDR
```

```
{
    public int HwndFrom;
    public int IdFrom;
    public int Code;
}

[DllImport("user32.dll")]
private static extern bool GetWindowRect(int hWnd, ref RECT lpRect);

[DllImport("user32.dll")]
private static extern int GetParent(int hWnd);

[DllImport("user32.dll", CharSet=CharSet.Auto)]
private static extern bool SetWindowText(int hWnd, string lpString);

[DllImport("user32.dll")]
private static extern int SendMessage(int hWnd, int Msg, int wParam,
int lParam);

[DllImport("user32.dll", CharSet=CharSet.Auto)]
private static extern int SendMessage(int hWnd, int Msg, int wParam,
string lParam);

[DllImport("user32.dll")]
private static extern bool DestroyWindow(int hwnd);

private const int OFN_ENABLEHOOK=0x00000020;
private const int OFN_EXPLORER=0x00080000;
private const int OFN_FILEMUSTEXIST=0x00001000;
private const int OFN_HIDEREADONLY=0x00000004;
private const int OFN_CREATEPROMPT=0x00002000;
private const int OFN_NOTESTFILECREATE=0x00010000;
private const int OFN_OVERWRITEPROMPT=0x00000002;
private const int OFN_PATHMUSTEXIST=0x00000800;
```

```
private const int SWP_NOSIZE=0x0001;
private const int SWP_NOMOVE=0x0002;
private const int SWP_NOZORDER=0x0004;
```

```
private const int WM_INITDIALOG=0x110;
private const int WM_DESTROY=0x2;
private const int WM_SETFONT=0x0030;
private const int WM_GETFONT=0x0031;
```

```
private const int CBS_DROPDOWNLIST=0x0003;
private const int CBS_HASSTRINGS=0x0200;
private const int CB_ADDSTRING=0x0143;
private const int CB_SETCURSEL=0x014E;
private const int CB_GETCURSEL=0x0147;
```

```
private const uint WS_VISIBLE=0x10000000;
private const uint WS_CHILD=0x40000000;
private const uint WS_TABSTOP=0x00010000;
```

```
private const int CDN_FILEOK=-606;
private const int WM_NOTIFY=0x004E;
```

```
[DllImport("user32.dll", CharSet=CharSet.Auto)]
private static extern int GetDlgItem(int hDlg, int nIDDlgItem);
```

```
[DllImport("user32.dll", CharSet=CharSet.Auto)]
private static extern int CreateWindowEx(int dwExStyle, string
lpClassName, string lpWindowName, uint dwStyle, int x, int y, int nWidth,
int nHeight, int hWndParent, int hMenu, int hInstance, int lpParam);
```

```
[DllImport("user32.dll")]
private static extern bool ScreenToClient(int hWnd, ref POINT
lpPoint);
```

```
#endregion
private int HookProc(int hdlg, int msg, int wParam, int lParam)
{
    switch (msg)
    {
        case WM_INITDIALOG:

            //we need to centre the dialog
            Rectangle sr=m_ActiveScreen.Bounds;
            RECT cr=new RECT();
            int parent=GetParent(hdlg);
            GetWindowRect(parent, ref cr);

            int x=(sr.Right + sr.Left - (cr.Right-cr.Left))/2;
            int y=(sr.Bottom + sr.Top - (cr.Bottom-cr.Top))/2;

            SetWindowPos(parent, 0, x, y, cr.Right-cr.Left,
cr.Bottom - cr.Top + 32, SWP_NOZORDER);

            //we need to find the label to position our new label
under

            int fileTypeWindow=GetDlgItem(parent, 0x441);

            RECT aboveRect=new RECT();
            GetWindowRect(fileTypeWindow, ref aboveRect);

            //now convert the label's screen co-ordinates to
client co-ordinates

            POINT point=new POINT();
            point.X=aboveRect.Left;
            point.Y=aboveRect.Bottom;
```

```
ScreenToClient(parent, ref point);

//create the label
int labelHandle=CreateWindowEx(0, "STATIC",
"mylabel", WS_VISIBLE | WS_CHILD | WS_TABSTOP, point.X, point.Y + 12, 200,
100, parent, 0, 0, 0);
SetWindowText(labelHandle, "&Encoding:");

int fontHandle=SendMessage(fileTypeWindow,
WM_GETFONT, 0, 0);
SendMessage(labelHandle, WM_SETFONT, fontHandle, 0);

//we now need to find the combo-box to position the
new combo-box under

int fileComboWindow=GetDlgItem(parent, 0x470);
aboveRect=new RECT();
GetWindowRect(fileComboWindow, ref aboveRect);

point=new POINT();
point.X=aboveRect.Left;
point.Y=aboveRect.Bottom;
ScreenToClient(parent, ref point);

POINT rightPoint=new POINT();
rightPoint.X=aboveRect.Right;
rightPoint.Y=aboveRect.Top;

ScreenToClient(parent, ref rightPoint);

//we create the new combobox

int comboHandle=CreateWindowEx(0, "ComboBox",
"mycombobox", WS_VISIBLE | WS_CHILD | CBS_HASSTRINGS | CBS_DROPDOWNLIST |
```



```
WS_TABSTOP, point.X, point.Y + 8, rightPoint.X-point.X, 100, parent, 0, 0,  
0);
```

```
SendMessage(comboHandle, WM_SETFONT, fontHandle, 0);
```

```
//and add the encodings we want to offer
```

```
SendMessage(comboHandle, CB_ADDSTRING, 0, "UTF-8");
```

```
//SendMessage(comboHandle, CB_ADDSTRING, 0, "UTF-8  
with preamble");
```

```
SendMessage(comboHandle, CB_ADDSTRING, 0, "Unicode");
```

```
SendMessage(comboHandle, CB_ADDSTRING, 0, "ANSI");
```

```
SendMessage(comboHandle, CB_SETCURSEL,  
(int)m_EncodingType, 0);
```

```
////we need to find the label to position our new label  
under
```

```
//fileTypeWindow = GetDlgItem(parent, 0x441);
```

```
//aboveRect = new RECT();
```

```
//GetWindowRect(fileTypeWindow, ref aboveRect);
```

```
////now convert the label's screen co-ordinates to client  
co-ordinates
```

```
//point = new POINT();
```

```
//point.X = aboveRect.Left;
```

```
//point.Y = aboveRect.Bottom;
```

```
//ScreenToClient(parent, ref point);
```

```
////create the label
```

```
//labelHandle = CreateWindowEx(0, "STATIC", "mylabel",  
WS_VISIBLE | WS_CHILD | WS_TABSTOP, point.X, point.Y + 12, 200, 100,  
parent, 0, 0, 0);
```

```
//SetWindowText(labelHandle, "&Override?");

//fontHandle = SendMessage(fileTypeWindow, WM_GETFONT, 0,
0);

//SendMessage(labelHandle, WM_SETFONT, fontHandle, 0);

////we now need to find the combo-box to position the new
combo-box under

//fileComboWindow = GetDlgItem(parent, 0x470);
//aboveRect = new RECT();
//GetWindowRect(fileComboWindow, ref aboveRect);

//point = new POINT();
//point.X = aboveRect.Left + (int)((aboveRect.Right -
aboveRect.Left) * 0.9);
//point.Y = aboveRect.Bottom;
//ScreenToClient(parent, ref point);

//rightPoint = new POINT();
//rightPoint.X = aboveRect.Right;
//rightPoint.Y = aboveRect.Top;

//ScreenToClient(parent, ref rightPoint);

////we create the new checkbox

//int checkHandle = CreateWindowEx(0, "CheckBox",
"mycheckbox", WS_VISIBLE | WS_CHILD | CBS_HASSTRINGS | CBS_DROPDOWNLIST |
WS_TABSTOP, point.X, point.Y + 8, rightPoint.X - point.X, 100, parent, 0,
0, 0);

//SendMessage(checkHandle, WM_SETFONT, fontHandle, 0);

////and add the encodings we want to offer
```

```
//SendMessage(checkHandle, CB_ADDSTRING, 0, "UTF-8");
//SendMessage(checkHandle, CB_ADDSTRING, 0, "UTF-8 with
preamble");

//SendMessage(checkHandle, CB_ADDSTRING, 0, "Unicode");
//SendMessage(checkHandle, CB_ADDSTRING, 0, "ANSI");

//SendMessage(checkHandle, CB_SETCURSEL,
(int)m_EncodingType, 0);

//remember the handles of the controls we have created so
we can destroy them after
m_LabelHandle =labelHandle;
    m_ComboHandle=comboHandle;
//m_CheckHandle = checkHandle;

break;
case WM_DESTROY:
    //destroy the handles we have created
    if (m_ComboHandle!=0)
    {
        DestroyWindow(m_ComboHandle);
    }

    if (m_LabelHandle!=0)
    {
        DestroyWindow(m_LabelHandle);
    }
    break;
case WM_NOTIFY:

    //we need to intercept the CDN_FILEOK message
    //which is sent when the user selects a filename
```

```
        NMHDR nmhdr=(NMHDR)Marshal.PtrToStructure(new
IntPtr(1Param), typeof(NMHDR));

        if (nmhdr.Code==CDN_FILEOK)
        {
            //a file has been selected
            //we need to get the encoding

            m_EncodingType=(EncodingType)SendMessage(m_ComboHandle, CB_GETCURSEL,
0, 0);

            }
            break;

        }
        return 0;
    }

    public string DefaultExt
    {
        get
        {
            return m_DefaultExt;
        }
        set
        {
            m_DefaultExt=value;
        }
    }

    public string Filter
    {
        get
        {
```

```
        return m_Filter;
    }
    set
    {
        m_Filter=value;
    }
}

public int FilterIndex
{
    get
    {
        return m_FilterIndex;
    }
    set
    {
        m_FilterIndex = value;
    }
}

public string FileName
{
    get
    {
        return m_FileName;
    }
    set
    {
        m_FileName=value;
    }
}

public EncodingType EncodingType
{
```

```
get
{
    return m_EncodingType;
}
set
{
    m_EncodingType=value;
}
}

public DialogResult ShowDialog()
{

    //set up the struct and populate it

    OPENFILENAME ofn=new OPENFILENAME();

    ofn.lStructSize= Marshal.SizeOf( ofn );
    ofn.lpstrFilter= m_Filter.Replace('|', '\\0') + '\\0';

    ofn.lpstrFile = m_FileName + new string(' ', 512);
    ofn.nMaxFile= ofn.lpstrFile.Length;
    ofn.lpstrFileTitle= System.IO.Path.GetFileName(m_FileName) + new
string(' ', 512);
    ofn.nMaxFileTitle = ofn.lpstrFileTitle.Length;
    ofn.lpstrTitle= "Save file as";
    ofn.lpstrDefExt=m_DefaultExt;

    //position the dialog above the active window
    ofn.hwndOwner=Form.ActiveForm.Handle;

    //we need to find out the active screen so the dialog box is
    //centred on the correct display
```

```
m_ActiveScreen=Screen.FromControl(Form.ActiveForm);

//set up some sensible flags
ofn.Flags=OFN_EXPLORER | OFN_PATHMUSTEXIST |
OFN_NOTESTFILECREATE | OFN_ENABLEHOOK | OFN_HIDEREADONLY |
OFN_OVERWRITEPROMPT;

//this is where the hook is set. Note that we can use a C#
delegate in place of a C function pointer
ofn.lpfnHook=new OFNHookProcDelegate(HookProc);

//if we're running on Windows 98/ME then the struct is smaller
if (System.Environment.OSVersion.Platform!=PlatformID.Win32NT)
{
    ofn.lStructSize-=12;
}

//show the dialog

if (!GetSaveFileName(ref ofn))
{
    int ret=CommDlgExtendedError();

    if (ret!=0)
    {
        throw new ApplicationException("Couldn't show file
open dialog - " + ret.ToString());
    }

    return DialogResult.Cancel;
}

m_FileName=ofn.lpstrFile;
m_FilterIndex = ofn.nFilterIndex;
```

```
        return DialogResult.OK;
    }
}

//public class OpenFileDialogWithEncoding
//{
//    private delegate int OFNHookProcDelegate(int hdlg, int msg, int
wParam, int lParam);

//    private int m_LabelHandle = 0;
//    private int m_ComboHandle = 0;
//    private int m_FilterIndex = 0;

//    private string m_Filter = "";
//    private string m_DefaultExt = "";

//    private string m_FileName = "";

//    private EncodingType m_EncodingType;
//    private Screen m_ActiveScreen;

//    [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Auto)]
//    private struct OPENFILENAME
//    {
//        public int lStructSize;
//        public IntPtr hwndOwner;
//        public int hInstance;
//        [MarshalAs(UnmanagedType.LPStr)]
//        public string lpstrFilter;
//        [MarshalAs(UnmanagedType.LPStr)]
//        public string lpstrCustomFilter;
//        public int nMaxCustFilter;
```



```
//      public int nFilterIndex;
//      [MarshalAs(UnmanagedType.LPTStr)]
//      public string lpstrFile;
//      public int nMaxFile;
//      [MarshalAs(UnmanagedType.LPTStr)]
//      public string lpstrFileTitle;
//      public int nMaxFileTitle;
//      [MarshalAs(UnmanagedType.LPTStr)]
//      public string lpstrInitialDir;
//      [MarshalAs(UnmanagedType.LPTStr)]
//      public string lpstrTitle;
//      public int Flags;
//      public short nFileOffset;
//      public short nFileExtension;
//      [MarshalAs(UnmanagedType.LPTStr)]
//      public string lpstrDefExt;
//      public int lCustData;
//      public OFNHookProcDelegate lpfnHook;
//      [MarshalAs(UnmanagedType.LPTStr)]
//      public string lpTemplateName;
//      //only if on nt 5.0 or higher
//      public int pvReserved;
//      public int dwReserved;
//      public int FlagsEx;
//  }

//  [DllImport("Comdlg32.dll", CharSet = CharSet.Auto, SetLastError =
true)]
//  private static extern bool GetSaveFileName(ref OPENFILENAME lpofn);

//  [DllImport("Comdlg32.dll")]
//  private static extern int CommDlgExtendedError();

//  [DllImport("user32.dll")]
```

```
// private static extern bool SetWindowPos(int hWnd, int
hWndInsertAfter, int X, int Y, int cx, int cy, uint uFlags);

// private struct RECT
// {
//     public int Left;
//     public int Top;
//     public int Right;
//     public int Bottom;
// }

// private struct POINT
// {
//     public int X;
//     public int Y;
// }

// private struct NMHDR
// {
//     public int HwndFrom;
//     public int IdFrom;
//     public int Code;
// }

// [DllImport("user32.dll")]
// private static extern bool GetWindowRect(int hWnd, ref RECT lpRect);

// [DllImport("user32.dll")]
// private static extern int GetParent(int hWnd);

// [DllImport("user32.dll", CharSet = CharSet.Auto)]
// private static extern bool SetWindowText(int hWnd, string lpString);

// [DllImport("user32.dll")]
```

```
// private static extern int SendMessage(int hWnd, int Msg, int wParam,
int lParam);

// [DllImport("user32.dll", CharSet = CharSet.Auto)]
// private static extern int SendMessage(int hWnd, int Msg, int wParam,
string lParam);

// [DllImport("user32.dll")]
// private static extern bool DestroyWindow(int hwnd);

// private const int OFN_ENABLEHOOK = 0x00000020;
// private const int OFN_EXPLORER = 0x00080000;
// private const int OFN_FILEMUSTEXIST = 0x00001000;
// private const int OFN_HIDEREADONLY = 0x00000004;
// private const int OFN_CREATEPROMPT = 0x00002000;
// private const int OFN_NOTESTFILECREATE = 0x00010000;
// private const int OFN_OVERWRITEPROMPT = 0x00000002;
// private const int OFN_PATHMUSTEXIST = 0x00000800;

// private const int SWP_NOSIZE = 0x0001;
// private const int SWP_NOMOVE = 0x0002;
// private const int SWP_NOZORDER = 0x0004;

// private const int WM_INITDIALOG = 0x110;
// private const int WM_DESTROY = 0x2;
// private const int WM_SETFONT = 0x0030;
// private const int WM_GETFONT = 0x0031;

// private const int CBS_DROPDOWNLIST = 0x0003;
// private const int CBS_HASSTRINGS = 0x0200;
// private const int CB_ADDSTRING = 0x0143;
// private const int CB_SETCURSEL = 0x014E;
// private const int CB_GETCURSEL = 0x0147;
```

```
// private const uint WS_VISIBLE = 0x10000000;
// private const uint WS_CHILD = 0x40000000;
// private const uint WS_TABSTOP = 0x00010000;

// private const int CDN_FILEOK = -606;
// private const int WM_NOTIFY = 0x004E;

// [DllImport("user32.dll", CharSet = CharSet.Auto)]
// private static extern int GetDlgItem(int hDlg, int nIDDlgItem);

// [DllImport("user32.dll", CharSet = CharSet.Auto)]
// private static extern int CreateWindowEx(int dwExStyle, string
lpClassName, string lpWindowName, uint dwStyle, int x, int y, int nWidth,
int nHeight, int hWndParent, int hMenu, int hInstance, int lpParam);

// [DllImport("user32.dll")]
// private static extern bool ScreenToClient(int hWnd, ref POINT
lpPoint);

// private int HookProc(int hdlg, int msg, int wParam, int lParam)
// {
//     switch (msg)
//     {
//         case WM_INITDIALOG:

//             //we need to centre the dialog
//             Rectangle sr = m_ActiveScreen.Bounds;
//             RECT cr = new RECT();
//             int parent = GetParent(hdlg);
//             GetWindowRect(parent, ref cr);

//             int x = (sr.Right + sr.Left - (cr.Right - cr.Left)) / 2;
//             int y = (sr.Bottom + sr.Top - (cr.Bottom - cr.Top)) / 2;
```

```
//          SetWindowPos(parent, 0, x, y, cr.Right - cr.Left,
cr.Bottom - cr.Top + 32, SWP_NOZORDER);

//          //we need to find the label to position our new label
under

//          int fileTypeWindow = GetDlgItem(parent, 0x441);

//          RECT aboveRect = new RECT();
//          GetWindowRect(fileTypeWindow, ref aboveRect);

//          //now convert the label's screen co-ordinates to client
co-ordinates
//          POINT point = new POINT();
//          point.X = aboveRect.Left;
//          point.Y = aboveRect.Bottom;

//          ScreenToClient(parent, ref point);

//          //create the label
//          int labelHandle = CreateWindowEx(0, "STATIC", "Load
dialog", WS_VISIBLE | WS_CHILD | WS_TABSTOP, point.X, point.Y + 12, 200,
100, parent, 0, 0, 0);
//          SetWindowText(labelHandle, "&Encoding:");

//          int fontHandle = SendMessage(fileTypeWindow, WM_GETFONT,
0, 0);
//          SendMessage(labelHandle, WM_SETFONT, fontHandle, 0);

//          //we now need to find the combo-box to position the new
combo-box under

//          int fileComboWindow = GetDlgItem(parent, 0x470);
```

```
//          aboveRect = new RECT();
//          GetWindowRect(fileComboWindow, ref aboveRect);

//          point = new POINT();
//          point.X = aboveRect.Left;
//          point.Y = aboveRect.Bottom;
//          ScreenToClient(parent, ref point);

//          POINT rightPoint = new POINT();
//          rightPoint.X = aboveRect.Right;
//          rightPoint.Y = aboveRect.Top;

//          ScreenToClient(parent, ref rightPoint);

//          //we create the new combobox

//          int comboHandle = CreateWindowEx(0, "ComboBox",
"mycombobox", WS_VISIBLE | WS_CHILD | CBS_HASSTRINGS | CBS_DROPDOWNLIST |
WS_TABSTOP, point.X, point.Y + 8, rightPoint.X - point.X, 100, parent, 0,
0, 0);
//          SendMessage(comboHandle, WM_SETFONT, fontHandle, 0);

//          //and add the encodings we want to offer
//          SendMessage(comboHandle, CB_ADDSTRING, 0, "UTF-8");
//          //SendMessage(comboHandle, CB_ADDSTRING, 0, "UTF-8 with
preamble");
//          SendMessage(comboHandle, CB_ADDSTRING, 0, "Unicode");
//          SendMessage(comboHandle, CB_ADDSTRING, 0, "ANSI");

//          SendMessage(comboHandle, CB_SETCURSEL,
(int)m_EncodingType, 0);

//          ////we need to find the label to position our new label
under
```

```
//          //fileTypeWindow = GetDlgItem(parent, 0x441);

//          //aboveRect = new RECT();
//          //GetWindowRect(fileTypeWindow, ref aboveRect);

//          ////now convert the label's screen co-ordinates to client
co-ordinates
//          //point = new POINT();
//          //point.X = aboveRect.Left;
//          //point.Y = aboveRect.Bottom;

//          //ScreenToClient(parent, ref point);

//          ////create the label
//          //labelHandle = CreateWindowEx(0, "STATIC", "mylabel",
WS_VISIBLE | WS_CHILD | WS_TABSTOP, point.X, point.Y + 12, 200, 100,
parent, 0, 0, 0);
//          //SetWindowText(labelHandle, "&Override?");

//          //fontHandle = SendMessage(fileTypeWindow, WM_GETFONT, 0,
0);
//          //SendMessage(labelHandle, WM_SETFONT, fontHandle, 0);

//          ////we now need to find the combo-box to position the new
combo-box under

//          //fileComboWindow = GetDlgItem(parent, 0x470);
//          //aboveRect = new RECT();
//          //GetWindowRect(fileComboWindow, ref aboveRect);

//          //point = new POINT();
//          //point.X = aboveRect.Left + (int)((aboveRect.Right -
aboveRect.Left) * 0.9);
```

```
//          //point.Y = aboveRect.Bottom;
//          //ScreenToClient(parent, ref point);

//          //rightPoint = new POINT();
//          //rightPoint.X = aboveRect.Right;
//          //rightPoint.Y = aboveRect.Top;

//          //ScreenToClient(parent, ref rightPoint);

//          ////we create the new checkbox

//          //int checkHandle = CreateWindowEx(0, "CheckBox",
"mycheckbox", WS_VISIBLE | WS_CHILD | CBS_HASSTRINGS | CBS_DROPDOWNLIST |
WS_TABSTOP, point.X, point.Y + 8, rightPoint.X - point.X, 100, parent, 0,
0, 0);
//          //SendMessage(checkHandle, WM_SETFONT, fontHandle, 0);

//          ////and add the encodings we want to offer
//          //SendMessage(checkHandle, CB_ADDSTRING, 0, "UTF-8");
//          //SendMessage(checkHandle, CB_ADDSTRING, 0, "UTF-8 with
preamble");
//          //SendMessage(checkHandle, CB_ADDSTRING, 0, "Unicode");
//          //SendMessage(checkHandle, CB_ADDSTRING, 0, "ANSI");

//          //SendMessage(checkHandle, CB_SETCURSEL,
(int)m_EncodingType, 0);

//          //remember the handles of the controls we have created so
we can destroy them after
//          m_LabelHandle = labelHandle;
//          m_ComboHandle = comboHandle;
//          //m_CheckHandle = checkHandle;

//          break;
```



```
//          case WM_DESTROY:
//              //destroy the handles we have created
//              if (m_ComboHandle != 0)
//              {
//                  DestroyWindow(m_ComboHandle);
//              }

//              if (m_LabelHandle != 0)
//              {
//                  DestroyWindow(m_LabelHandle);
//              }
//              break;
//          case WM_NOTIFY:

//              //we need to intercept the CDN_FILEOK message
//              //which is sent when the user selects a filename

//              NMHDR nmhdr = (NMHDR)Marshal.PtrToStructure(new
IntPtr(1Param), typeof(NMHDR));

//              if (nmhdr.Code == CDN_FILEOK)
//              {
//                  //a file has been selected
//                  //we need to get the encoding

//                  m_EncodingType =
(EncodingType)SendMessage(m_ComboHandle, CB_GETCURSEL, 0, 0);
//              }
//              break;

//          }
//          return 0;
//      }
```

```
// public string DefaultExt
// {
//     get
//     {
//         return m_DefaultExt;
//     }
//     set
//     {
//         m_DefaultExt = value;
//     }
// }

// public string Filter
// {
//     get
//     {
//         return m_Filter;
//     }
//     set
//     {
//         m_Filter = value;
//     }
// }

// public int FilterIndex
// {
//     get
//     {
//         return m_FilterIndex;
//     }
//     set
//     {
//         m_FilterIndex = value;
//     }
// }
```

```
//    }

//    public string FileName
//    {
//        get
//        {
//            return m_FileName;
//        }
//        set
//        {
//            m_FileName = value;
//        }
//    }

//    public EncodingType EncodingType
//    {
//        get
//        {
//            return m_EncodingType;
//        }
//        set
//        {
//            m_EncodingType = value;
//        }
//    }

//    public DialogResult ShowDialog()
//    {

//        //set up the struct and populate it

//        OPENFILENAME ofn = new OPENFILENAME();

//        ofn.lStructSize = Marshal.SizeOf(ofn);
```

```
//      ofn.lpstrFilter = m_Filter.Replace('|', '\\0') + '\\0';

//      ofn.lpstrFile = m_FileName + new string(' ', 512);
//      ofn.nMaxFile = ofn.lpstrFile.Length;
//      ofn.lpstrFileTitle = System.IO.Path.GetFileName(m_FileName) + new
string(' ', 512);
//      ofn.nMaxFileTitle = ofn.lpstrFileTitle.Length;
//      ofn.lpstrTitle = "Load file";
//      ofn.lpstrDefExt = m_DefaultExt;

//      //position the dialog above the active window
//      ofn.hwndOwner = Form.ActiveForm.Handle;

//      //we need to find out the active screen so the dialog box is
//      //centred on the correct display

//      m_ActiveScreen = Screen.FromControl(Form.ActiveForm);

//      //set up some sensible flags
//      ofn.Flags = OFN_EXPLORER | OFN_PATHMUSTEXIST |
OFN_NOTESTFILECREATE | OFN_ENABLEHOOK | OFN_HIDEREADONLY |
OFN_OVERWRITEPROMPT;

//      //this is where the hook is set. Note that we can use a C#
delegate in place of a C function pointer
//      ofn.lpfHook = new OFNHookProcDelegate(HookProc);

//      //if we're running on Windows 98/ME then the struct is smaller
//      if (System.Environment.OSVersion.Platform != PlatformID.Win32NT)
//      {
//          ofn.lStructSize -= 12;
//      }

//      //show the dialog
```

```
//      if (!GetSaveFileName(ref ofn))
//      {
//          int ret = CommDlgExtendedError();

//          if (ret != 0)
//          {
//              throw new ApplicationException("Couldn't show file open
dialog - " + ret.ToString());
//          }

//          return DialogResult.Cancel;
//      }

//      m_FileName = ofn.lpstrFile;
//      m_FilterIndex = ofn.nFilterIndex;

//      return DialogResult.OK;
//  }
//}
STL.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GerasimenkoER_KDZ3_v2
{
    //class STL
    //{
    //{
    //{

    class pair<T1, T2>
```

```
//where T1 : IComparable
//where T2 : IComparable
{
    public T1 first;
    public T2 second;
    public pair() { }
    public pair(T1 firste, T2 seconde){
        first = firste;
        second = seconde;
    }

    public static pair<TT1,TT2> makepair<TT1,TT2>(TT1 first, TT2
second)
    {
        return new pair<TT1, TT2>(first, second);
    }

    public void setfirst(T1 w)
    {
        first = w;
    }
    public void setsecond(T2 w)
    {
        second = w;
    }

    //public static bool operator ==(pair<T1, T2> a, pair<T1, T2> b)
    //{
    //    return a.first.CompareTo(b.first)==0 &&
a.second.CompareTo(b.second)==0;
    //}
    //public static bool operator !=(pair<T1, T2> a, pair<T1, T2> b)
    //{
    //    return !(a==b);
    //}
```

```
//}

//public static bool operator ==(pair<string, string> a,
pair<string, string> b)
//{
//    return a.first.CompareTo(b.first)==0 &&
a.second.CompareTo(b.second)==0;
//}

//public static bool operator !=(pair<string, string> a,
pair<string, string> b)
//{
//    return !(a==b);
//}

}

//struct pair
//{
//    public int first;
//    public string second;
//    public void setsecond(string w)
//    {
//        second = w;
//    }
//}

struct vector<T>
{
    List<T> e;
    int length;
    public void fill(List<T> c)
    {
        e = c;
        length = c.Count;
    }
}
```

```
    }  
    public int size()  
    {  
        return length;  
    }  
    public T this [int n]  
    {  
        get  
        {  
            return e[n];  
        }  
        set  
        {  
            e[n] = value;  
        }  
    }  
}  
public void append(T v)  
{  
    if (e == null) { e = new List<T>(); }  
    e.Add(v);  
    ++length;  
}  
public void remove(int n)  
{  
    e.RemoveAt(n);  
    --length;  
}  
public void reverse()  
{  
    e.Reverse();  
}  
public string toString(string separator = "")  
{  
    string s = "";
```



```
        for(int i = 0; i < length; ++i)
        {
            s += e[i] + separator;
        }
        return s;
    }
    public T[] toarray()
    {
        return e.ToArray();
    }
}
```

## **9. Список литературы**

1. Шилдт Г. C# 4.0: полное руководство. Москва: ООО "И.Д. Вильямс", 2014.
2. «Руководство по программированию на C# | Microsoft Docs»[Электронный ресурс] //URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/> (Дата обращения 26.02.19)