Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных,**

**принятия решений и финансовых технологий**

**Пояснительная записка к курсовой работе**

(по дисциплине «Современные технологии программирования»)

на тему:

**Информационно-справочная система**

**«Автомобили Toyota»**

Выполнил(а):

студент (ка) группы ПИ18-4 факультета

«Прикладная математика и

информационные технологии»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ефремов С. В.

(Подпись)

Тел: 8 (985) 252-0502

Научный руководитель:

профессор, Владова А. Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

2020

Введение...................................................................................................................3 Постановка задачи...................................................................................................4 Описание предметной области ..............................................................................5 Описание программы..............................................................................................6 Алгоритмические решения………………..……...................................................7 Главная форма…………………………………......................................................9 Связанная форма…...............................................................................................16 Адаптивность и сохранение параметров ….......................................................30 Заключение ............................................................................................................32 Список литературы ...............................................................................................32

**Введение**

В наши дни автоматизация различных производственных процессов занимает передовое место в мире. Результатом данного процесса являются: ускорение производства, сокращение ошибок при эксплуатации и многие другие.

Целью данной курсовой работы является ознакомление с работой в среде разработки Microsoft Visual Studio, посредством разработки приложения для работы под операционной системой Microsoft Windows.

**Постановка задачи**

Темой данной курсовой работы является разработка информационно- расчетной системы «Автомобили Toyota». В рамках этой задачи предполагается реализовать приложение для работы с существующей базой данных (БД), в которой хранится информация о моделях авто их комплектациях и характеристиках. Программа должна осуществлять отображение списка существующих на рынке моделей автомобилей Toyota, а также отображать список комплектаций, доступных у каждой модели. Осуществлять сортировку комплектации каждой модели по различным характеристикам. Программа также предоставляет возможность корректировки и добавления новых моделей и комплектаций.

**Описание предметной области**

В автосалоне осуществляется продажа автомобилей Toyota. В магазин обращаются клиенты с вопросом о наличии той или иной модели, доступной комплектации, характеристиках или их стоимости. Продавцу требуется быстро просмотреть наличие той или иной модели и комплектации, узнать стоимость модели, предложить клиенту близкие по стоимости или характеристикам модели. При обновлении ассортимента магазина, продавец должен иметь возможность актуализировать БД программы.

**Описание программы**

В рамках поставленной в курсовой работе задачи, была разработана программа, которая позволяет:

1. Отображать доступные модели Toyota, а также их комплектации;
2. Редактировать существующих и добавлять новые модели;
3. Отображать имеющиеся на комплектации для каждой из моделей;
4. Актуализировать информацию по каждой из имеющихся в наличии комплектаций. Добавлять новые и удалять отсутствующие комплектации;
5. Осуществлять фильтрацию, сортировку (в том числе многоуровневую) по различным параметрам.

**Алгоритмические решения**

**Изображение выглядит как текст, карта

Автоматически созданное описание**

**Краткое описание алгоритмического решения**

Пользователь запускает программу, на экране в сетке отображаются модели Toyota. Пользователь выбирает, надо ли добавлять новые модели, редактировать данные о существующих или удалить модель. Если в этом нет необходимости, пользователь может отобразить имеющиеся комплектации для этой модели. Далее пользователь может скорректировать информацию о комплектации (добавить, изменить, удалить) либо выбрать интересующую его комплектацию путем использования сортировки или фильтрации.

**Описательная часть**

**Главная форма**

При запуске пользователем программы открывается начальная форма:



В ней можно выделить несколько ключевых компонентов:

1. Краткая информация о компании

2. Основные данные

3. Кнопки для работы с формой

4. Эмблема

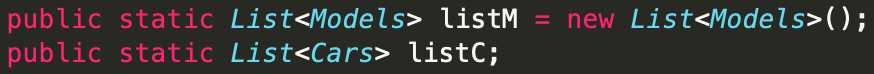
5. Окно параметров приложения

6. Кнопка информации о разработчике

Теперь подробнее про каждый из блоков:

1. В первом блоке собраны самые важные факты о программе, которые сразу могут быть интересны пользователю. Для создания данного блока было использовано несколько label’ов.

2. Во втором блоке собраны основные данные о моделях автомобилей, доступных для покупки. Для его создания было использовано DataGridView. Таблица заполнена через bindingSource из класса Models.







Класс Models:

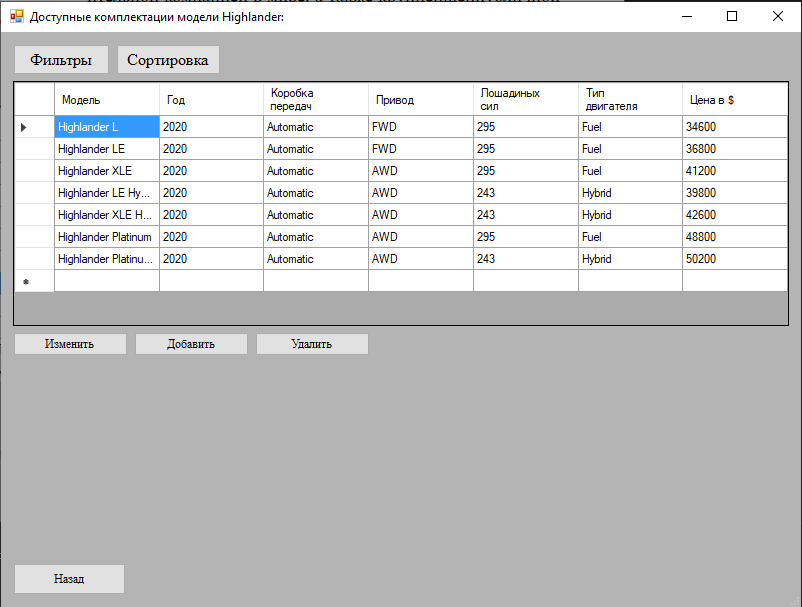


В таблице представлена информация в 5 основных столбцах: 1-й столбец содержит название модели, во 2-ом столбец содержит год выпуска доступной модели, 3-й столбец отвечает за тип кузова автомобиля, 4-й содержит минимальную цену на выбранную модель и в 5-ом находится вычисляемое свойство: количество доступных комплектаций выбранной модели.

При двойном нажатии на выбранную модель откроется вторая форма отображающая доступные комплектации выбранной модели.



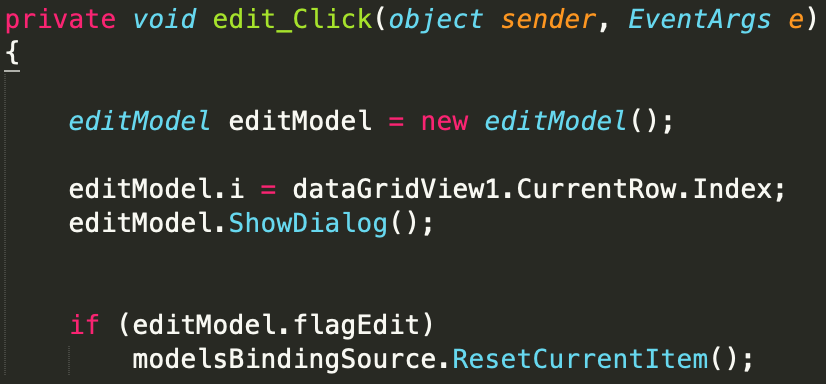
Вторая форма, отображающая доступные комплектации:



3. Снизу формы располагаются кнопки для возможности пользователя взаимодействовать с формой. Для создания элементов было использовано четыре button’а.

Самой первой идет кнопка «Изменить». При нажатии на нее появится форма для изменения конкретного поля. Изменяемым будет то поле, которое сделал активным пользователь.

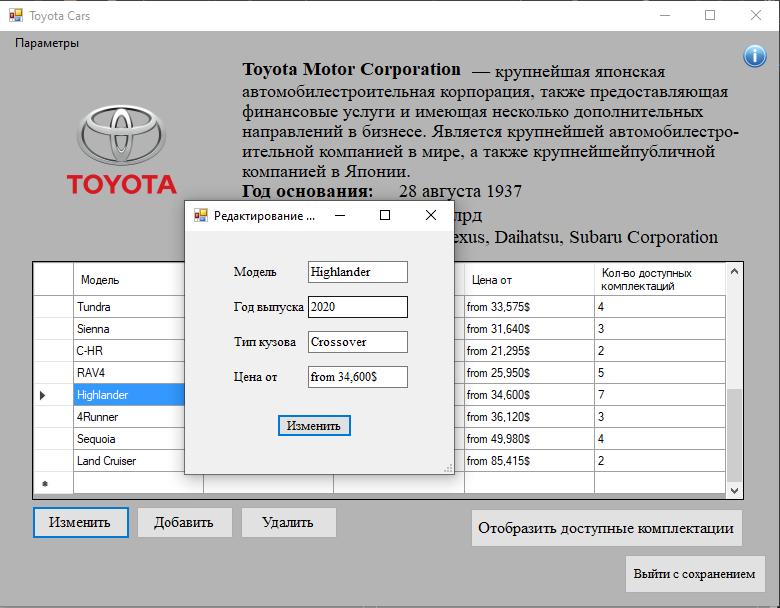
Программно это выглядит вот так:



Код формы:

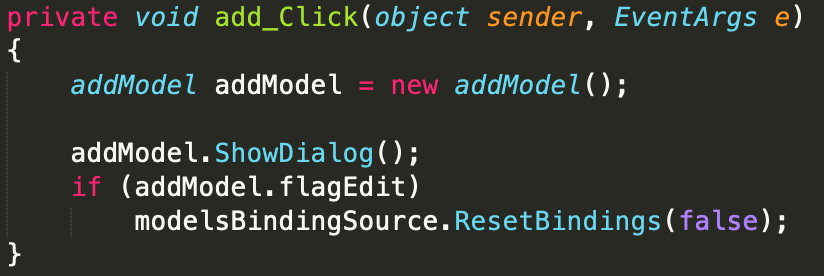


А сама форма будет выглядеть так:

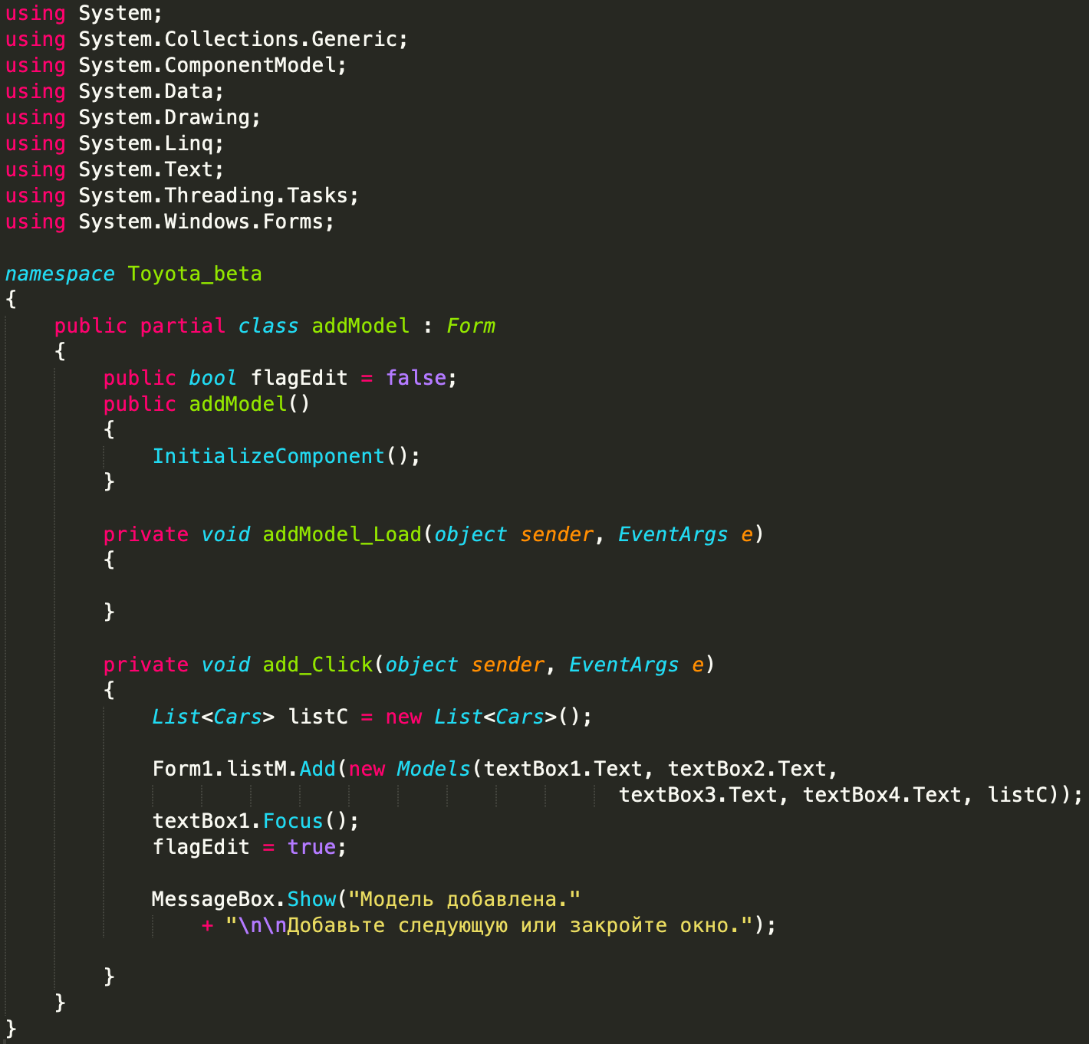


Справа от кнопки «Изменить» находится кнопка «Добавить», которая отвечает за добавление нового элемента в наш DataGridView.

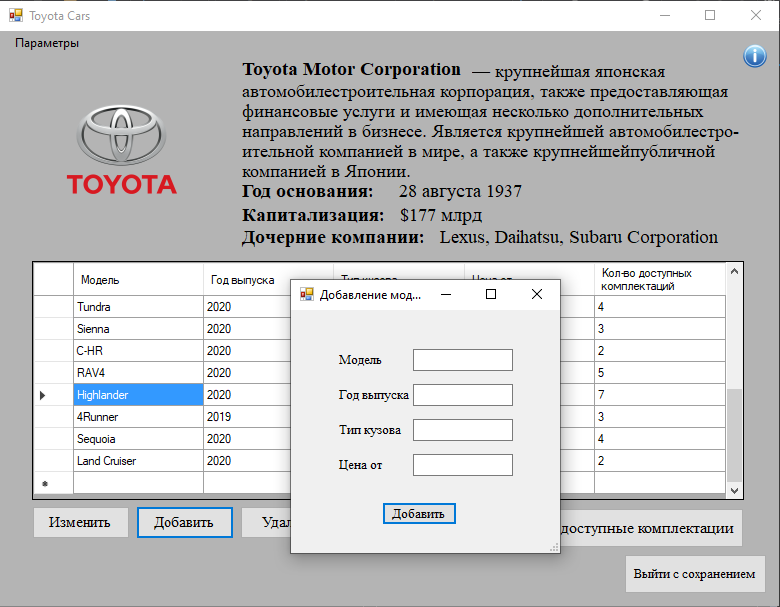
Реализация добавления нового элемента:



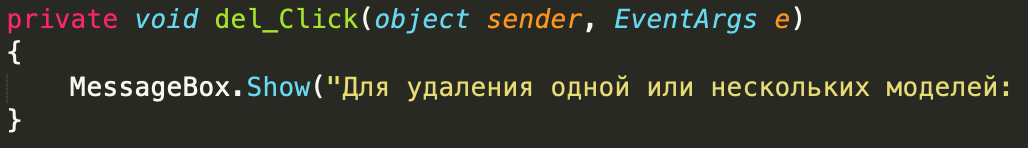
Код формы добавить:



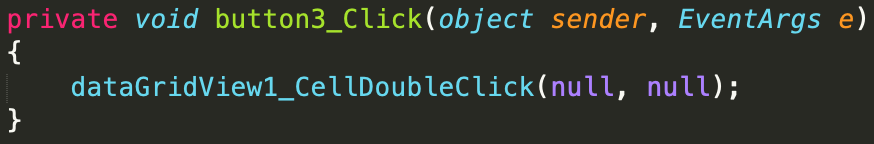
Внешний вид формы добавить:



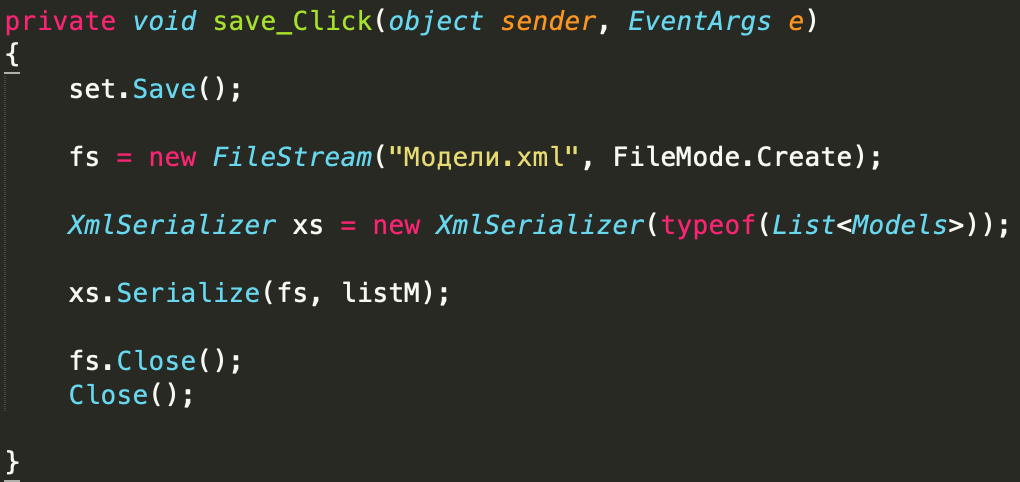
Правее кнопок «Изменить» и «Добавить» можно заметить кнопку «Удалить», которая отвечает за удаление элемента из DataGridView. При нажатии на данную кнопку пользователю будет предложено удалить ненужный ряд из таблицы путем нажатия на левый столбец нужной троки, а затем на клавишу «Delete» на клавиатуре.



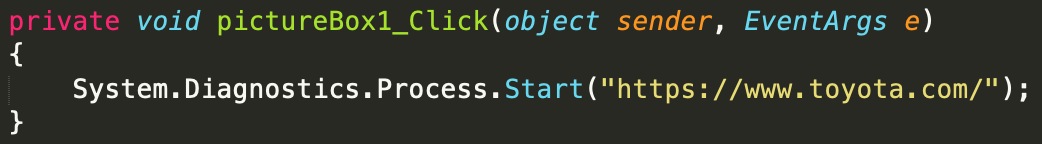
Снизу от правого угла таблицы располагается кнопка «Отобразить доступные комплектации». Ее функция полностью повторяет двойной клик по выбранной модели: она откроет вторую форму со всеми доступными комплектациями выбранной модели.



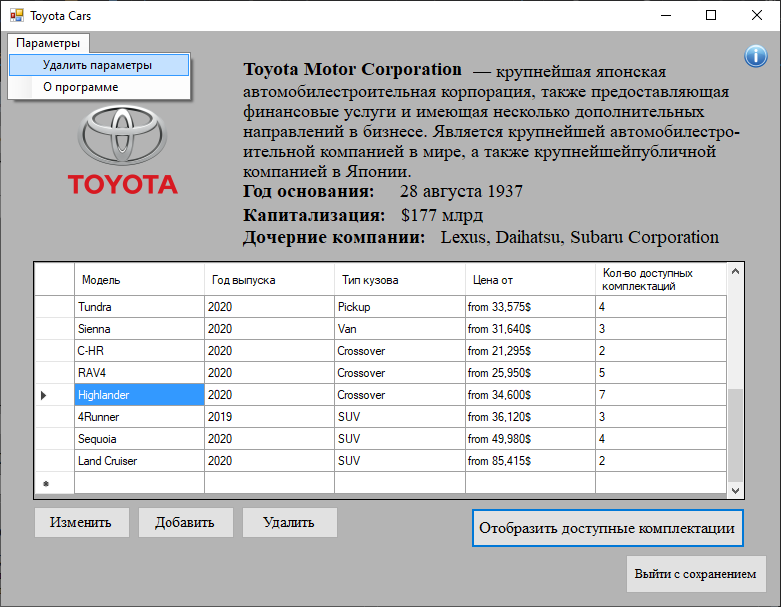
В нижнем правом углу нельзя не заметить кнопку выйти с сохранением, именно она отвечает за сохранение изменений, вносимых пользователем в приложении. Нажатие на кнопку запишет данные приложения в XML-файл.



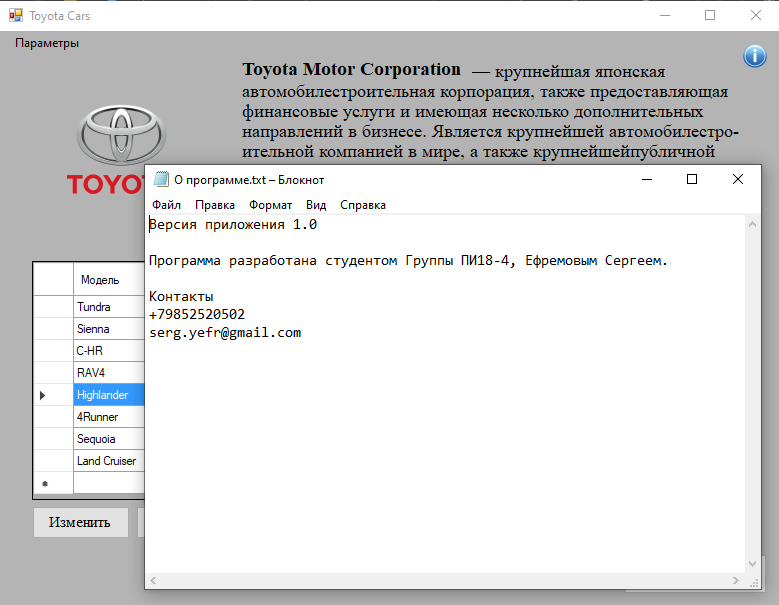
4. Логотип Toyota. При наведении на него пользователь заметит, что обычный курсор поменяется, а все потому что при нажатии на картинку пользователь сможет перейти на официальный сайт toyota.com для уточнения какой-либо информации. Для создания блока был использован pictureBox.



5. В левом верхнем углу находится неприметная кнопка параметры. При нажатии на нее пользователю будет предложено:



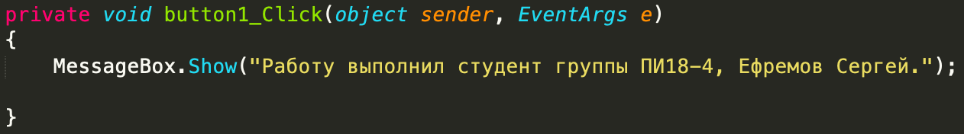
«Удаление параметров» полностью очистит сохраненные пользователем действия, а при нажатии на «О программе» пользователь сможет более подробно узнать о разработчике:



Программно это реализовано так:



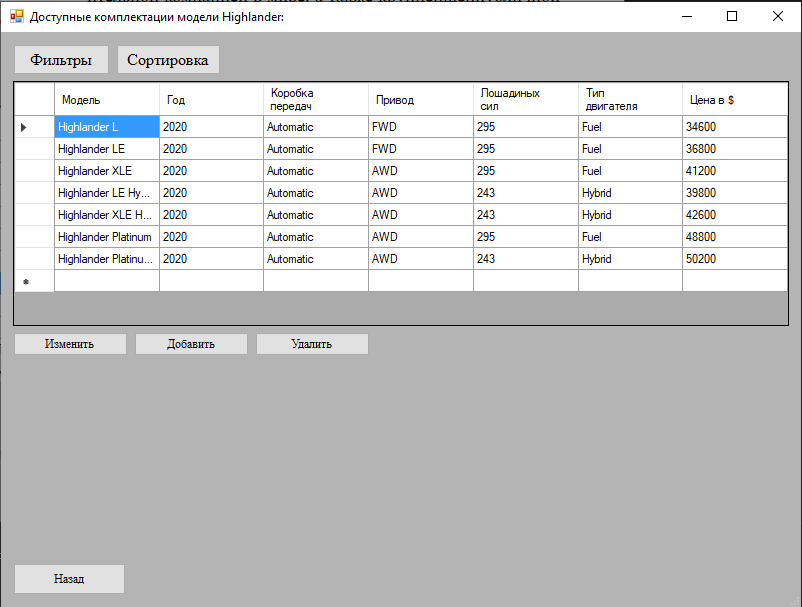
6. Маленькая кнопка информации, при нажатии на которую пользователь сможет быстрее узнать более краткую информацию о разрабочике.



Стоит также отметить, что в форме присутствуют toolTip’ы, которые помогают пользователю подсказками, для чего какие элементы предназначены.

**Связанная форма**

Через doubleClick или нажатию по кнопке, но пользователь доберется для второй формы, на которой он сможет выбрать комплектацию необходимой ему модели. Выглядит вторая форма так:



Следует заметить, что для удобства пользователя сверху в названии формы указывается комплектации какой модели сейчас отображены.

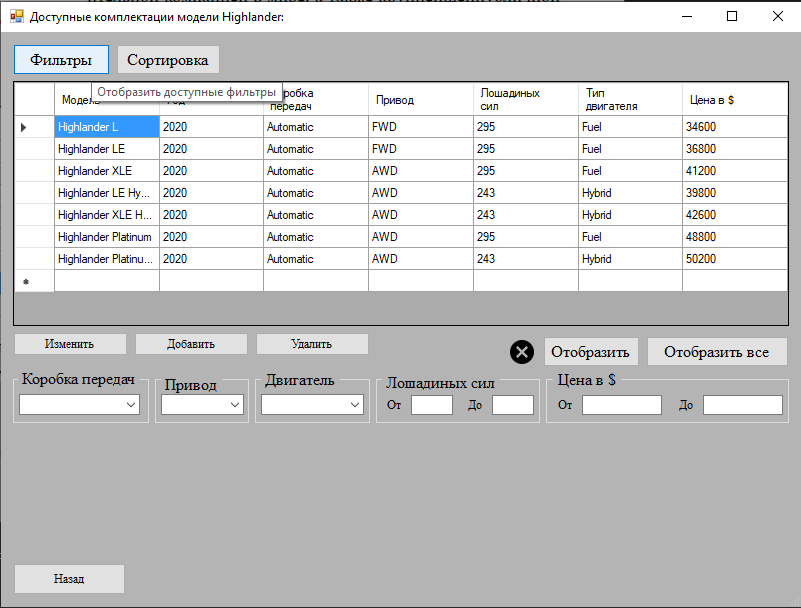
Основную часть формы занимает таблица DataGridView, которая через bindingSource заполняется из класса Cars:



Таблица состоит из семи столбцов: 1-й отвечает за название комплектации, во 2-ом хранится год выпуска данной комплектации, 3-й отображает тип коробки передач, 4-й привод текущей комплектации, в 5-ом находится кол-во лошадиных сил, 6-й отвечает за тип двигателя выбранной комплектации, а 7-й показывает конечную цены в долларах.

Сверху находятся две кнопки: «Фильтры» и «Сортировка» при нажатии на которые пользователем будут отображены дополнительные элементы.

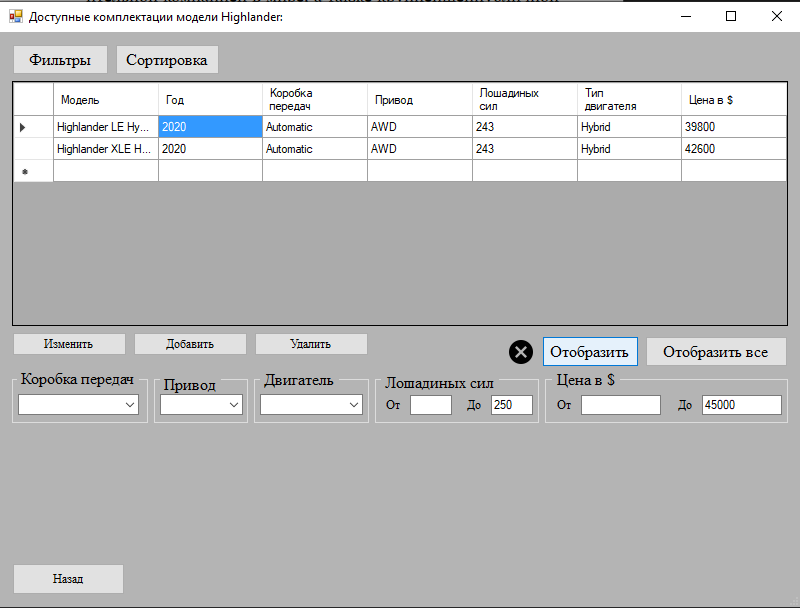
Вот что происходит при нажатии на кнопку «Фильтры»:



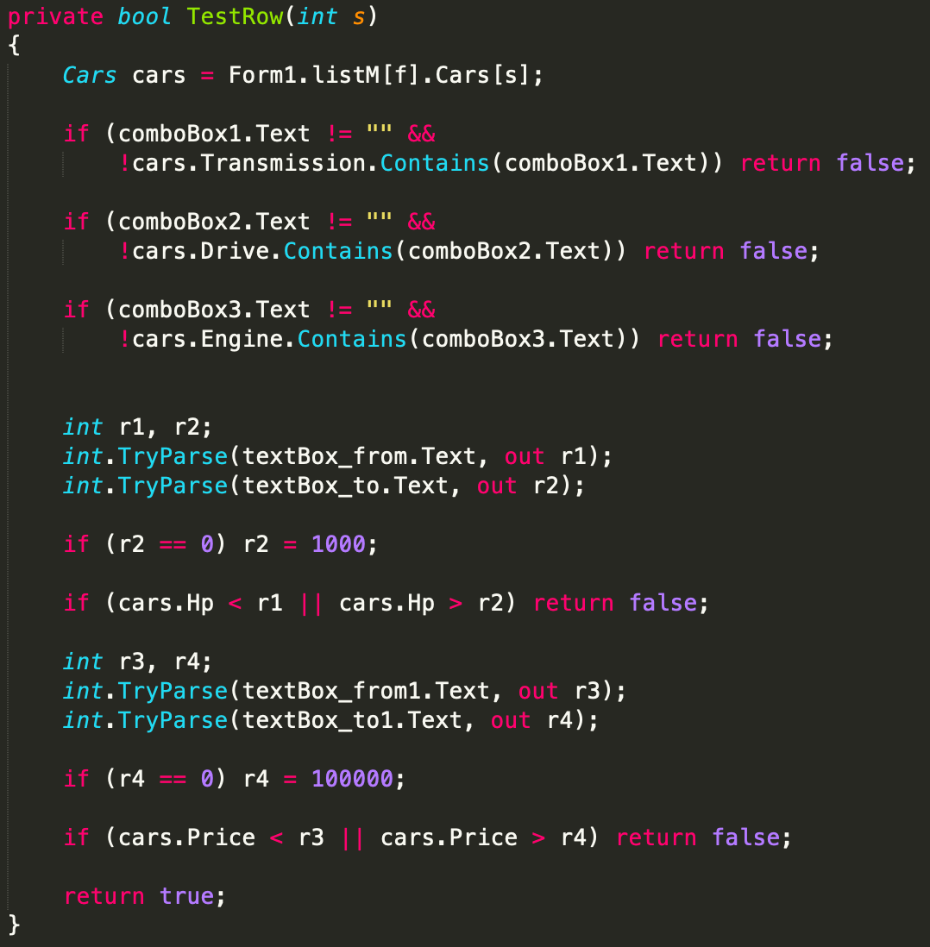
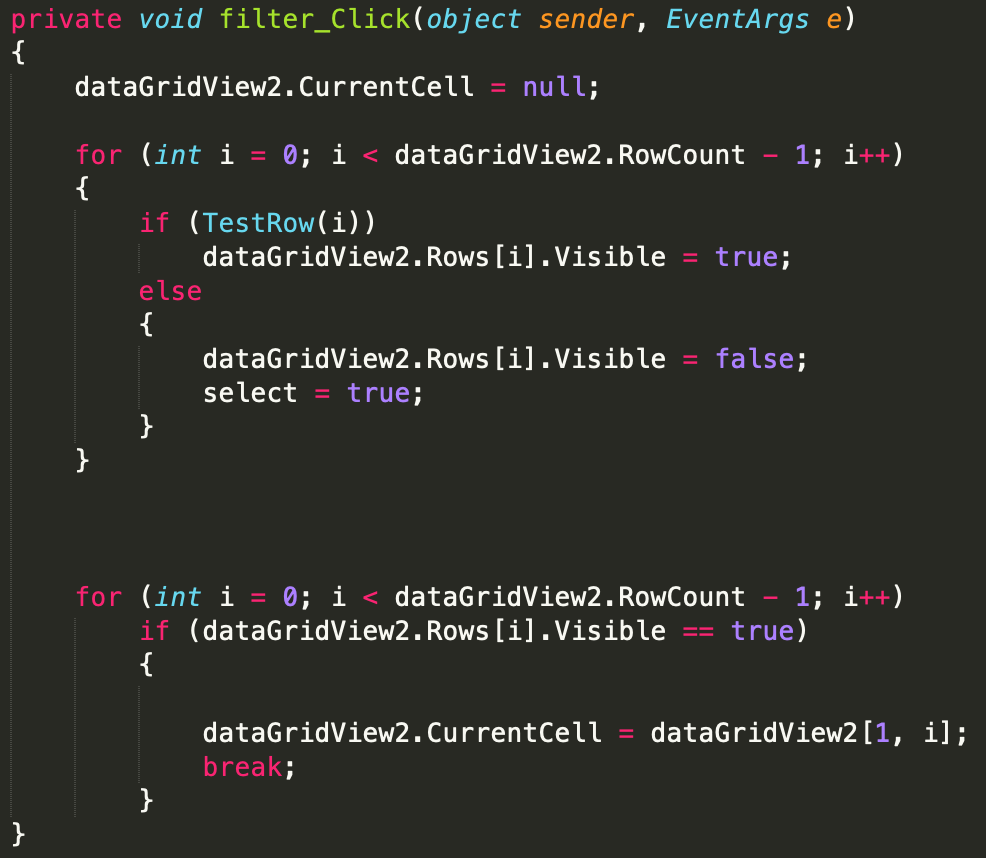
Изначально элементы для фильтрации данных скрыты. Для чего это сделано? Могу объяснить: например, у какой-то модели может быть одна или две комплектации, так что фильтровать банально нечего, следовательно, фильтры пользователю не понадобятся, а значит и не нужно лишний раз нагромождать форму.

В фильтрах можно указать любые параметры для того, чтобы выбрать наиболее подходящую комплектацию нужной нам модели.

Для примера отобразим только те комплектации, которые не мощнее 250 лошадиных сил, так как более мощные авто облагаются более высоким налогом в РФ, а также не дороже 45000 $. Интерфейс программы интуитивно понятен, так что, введя нужные нам данные и нажав кнопку отобразить получаем результат:

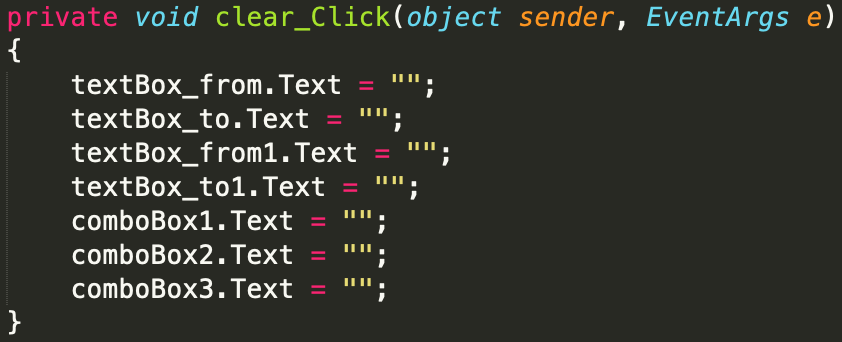


Как уже понятно из примера выше, кнопка отобразить выбирает из DataGridView только комплектации, удовлетворяющие пользовательским фильтрам. Для этого в программе реализована фильтрация:



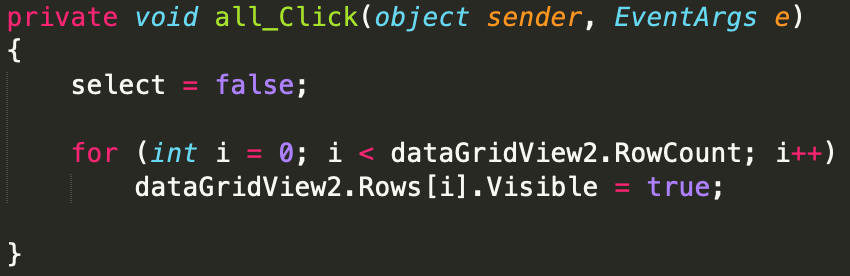
Слева от кнопки «Отобразить» располагается кнопка для очищения пользовательских фильтров. Для более комфортного использования программы она похожа на кнопку «Delete».

Её код выглядит следующим образом:



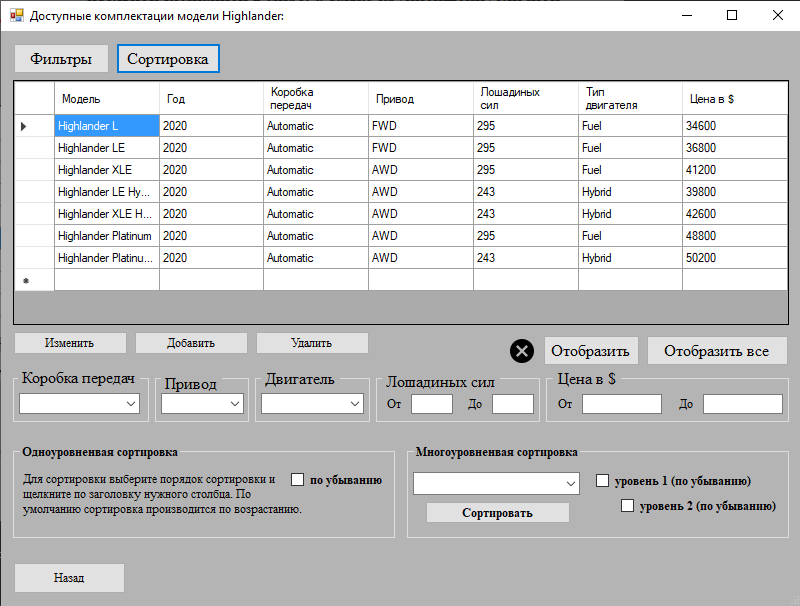
А справа от кнопки «Отобразить» находится кнопка «Отобразить все». Её задача экономит время пользователя, ведь в любой момент вместо сочетания кнопок «Очистить» и «Отобразить» можно использовать «Отобразить все» для вывода исходных данных.

Реализация этой кнопки:



Для реализации фильтров были использованы различные элементы: button’ы, combo box’ы, text box’ы.

Справа от кнопки «Фильтры» расположена кнопка «Сортировка». При нажатии на нее будут отображены дополнительные элементы, которые были скрыты, по причине обозначенной выше.

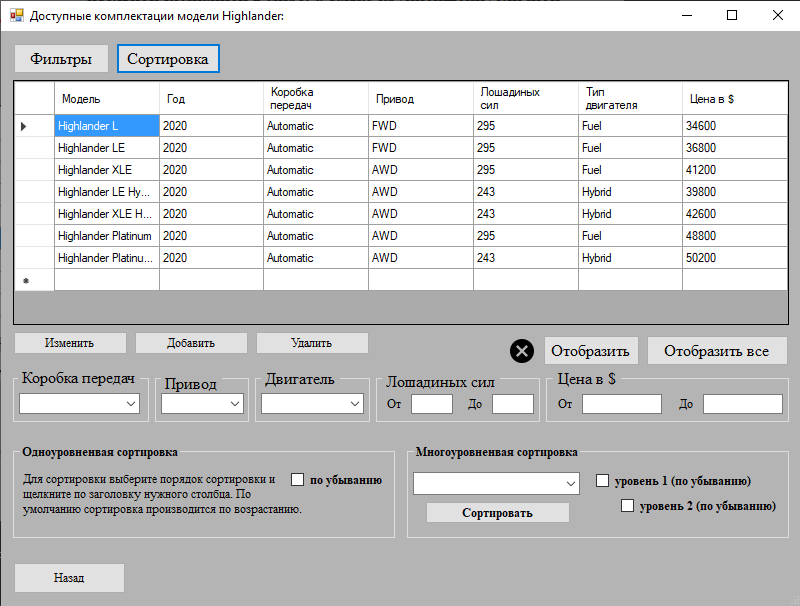


На выбор у пользователя будет сортировка одноуровневая и многоуровневая.

Лучше всего продемонстрировать работу одноуровневой на конкретном примере.

Допустим пользователь интересуется самой доступной комплектацией, для этого отсортируем данные по цене. Как пользоваться одноуровневой сортировкой подробно описано прямо в форме.

Результат получаем следующий:

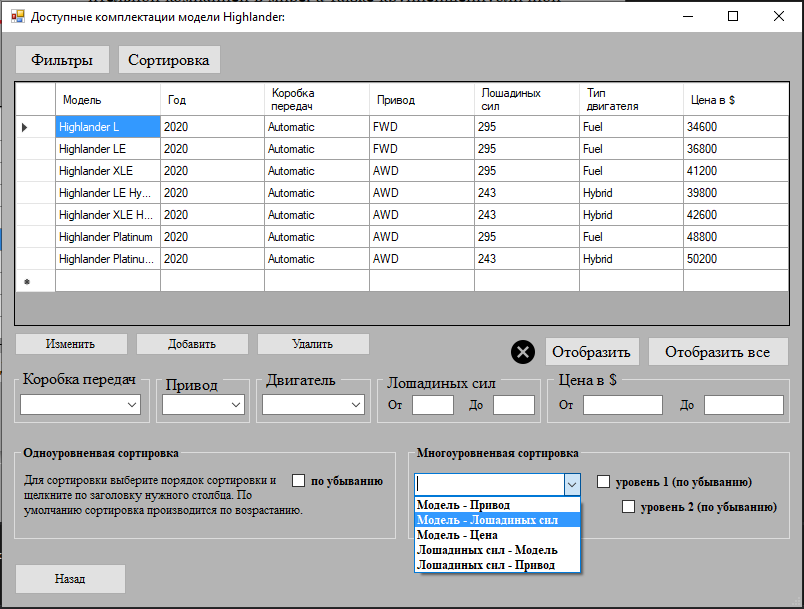


Если пользователю понадобится поменять способ одноуровневой сортировки с возрастания на убывание сделать это очень просто: для этого существует checkedBox, которые нужно использовать только в случае сортировки по убыванию.

Для сортировки одноуровневой используется событие columnHeaderMouseClick:



Справа от блока одноуровневой сортировки располагается блок многоуровневой сортировки. Тут уже пользователь может выбрать до двух критериев сортировки, которые интересны именно ему:



Для удобства пользователя также есть 2 checkedBox’а для переключения способа сортировки с возрастания на убывание. Многоуровневая сортировка срабатывает после нажатия пользователем кнопки «Сортировать».

Реализация этого действия происходит вот так:

private void button4\_Click\_1(object sender, EventArgs e)

{

List<Cars> list = Form1.listM[f].Cars;

if (!checkBox2.Checked && !checkBox3.Checked)

switch (comboBox4.SelectedIndex)

{

case 0:

list = (list.OrderBy(s => s.Model).ThenBy(s => s.Drive)).ToList();

break;

case 1:

list = (list.OrderBy(s => s.Model).ThenBy(s => s.Hp)).ToList();

break;

case 2:

list = (list.OrderBy(s => s.Model).ThenBy(s => s.Price)).ToList();

break;

case 3:

list = (list.OrderBy(s => s.Hp).ThenBy(s => s.Drive)).ToList();

break;

case 4:

list = (list.OrderBy(s => s.Hp).ThenBy(s => s.Model)).ToList();

break;

}

else if (!checkBox2.Checked && checkBox3.Checked)

switch (comboBox4.SelectedIndex)

{

case 0:

list = (list.OrderBy(s => s.Model).ThenByDescending(

s => s.Drive)).ToList();

break;

case 1:

list = (list.OrderBy(s => s.Model).ThenByDescending(

s => s.Hp)).ToList();

break;

case 2:

list = (list.OrderBy(s => s.Model).ThenByDescending(

s => s.Price)).ToList();

break;

case 3:

list = (list.OrderBy(s => s.Hp).ThenByDescending(

s => s.Drive)).ToList();

break;

case 4:

list = (list.OrderBy(s => s.Hp).ThenByDescending(

s => s.Model)).ToList();

break;

}

else if (checkBox2.Checked && checkBox3.Checked)

switch (comboBox4.SelectedIndex)

{

case 0:

list = (list.OrderByDescending(s => s.Model).ThenByDescending(

s => s.Drive)).ToList();

break;

case 1:

list = (list.OrderByDescending(s => s.Model).ThenByDescending(

s => s.Hp)).ToList();

break;

case 2:

list = (list.OrderByDescending(s => s.Model).ThenByDescending(

s => s.Price)).ToList();

break;

case 3:

list = (list.OrderByDescending(s => s.Hp).ThenByDescending(

s => s.Drive)).ToList();

break;

case 4:

list = (list.OrderByDescending(s => s.Hp).ThenByDescending(

s => s.Model)).ToList();

break;

}

else if (checkBox2.Checked && !checkBox3.Checked)

switch (comboBox4.SelectedIndex)

{

case 0:

list = (list.OrderByDescending(s => s.Model).ThenBy(

s => s.Drive)).ToList();

break;

case 1:

list = (list.OrderByDescending(s => s.Model).ThenBy(

s => s.Hp)).ToList();

break;

case 2:

list = (list.OrderByDescending(s => s.Model).ThenBy(

s => s.Price)).ToList();

break;

case 3:

list = (list.OrderByDescending(s => s.Hp).ThenBy(

s => s.Drive)).ToList();

break;

case 4:

list = (list.OrderByDescending(s => s.Hp).ThenBy(

s => s.Model)).ToList();

break;

}

Form1.listM[f].Cars = list;

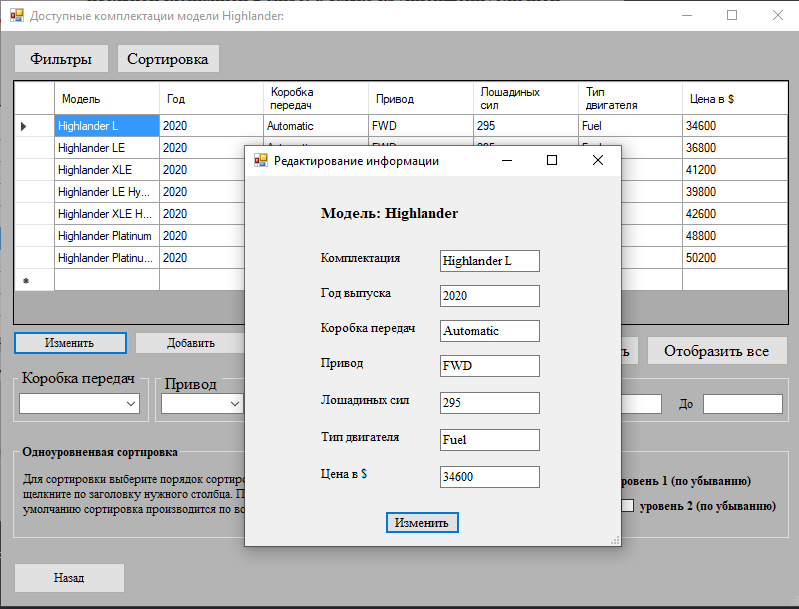
carsBindingSource.DataSource = list;

if (select) filter\_Click(null, null);

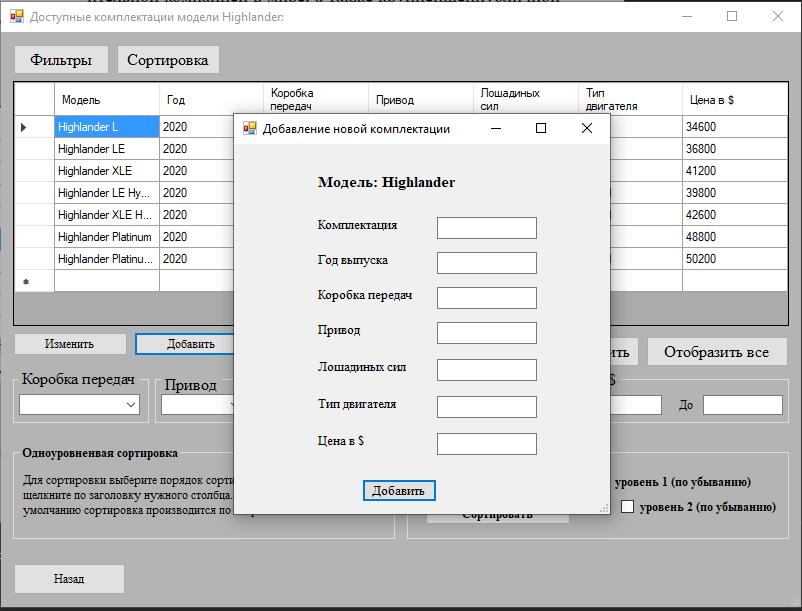
}

Выше сортировки и фильтров располагаются уже привычные нам кнопки: «Изменить», «Добавить» и «Удалить». Они отвечают за вызов форм, для работы с данными DataGridView.

Нажатие на кнопку «Изменить»:

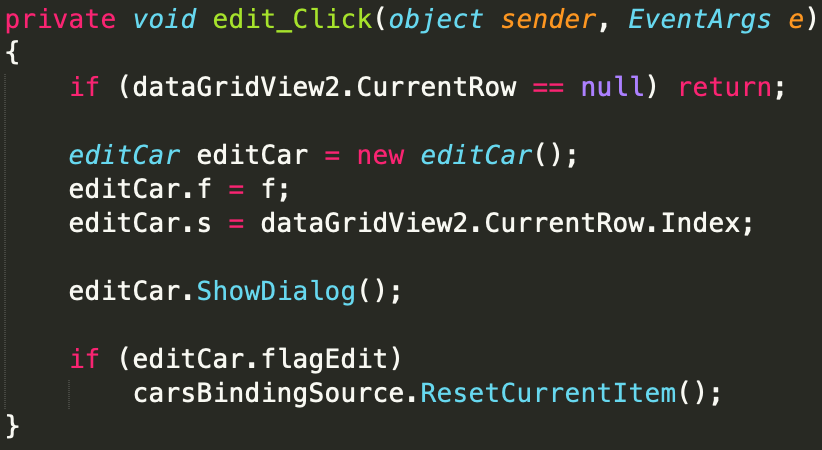


Нажатие на кнопку «Добавить»:

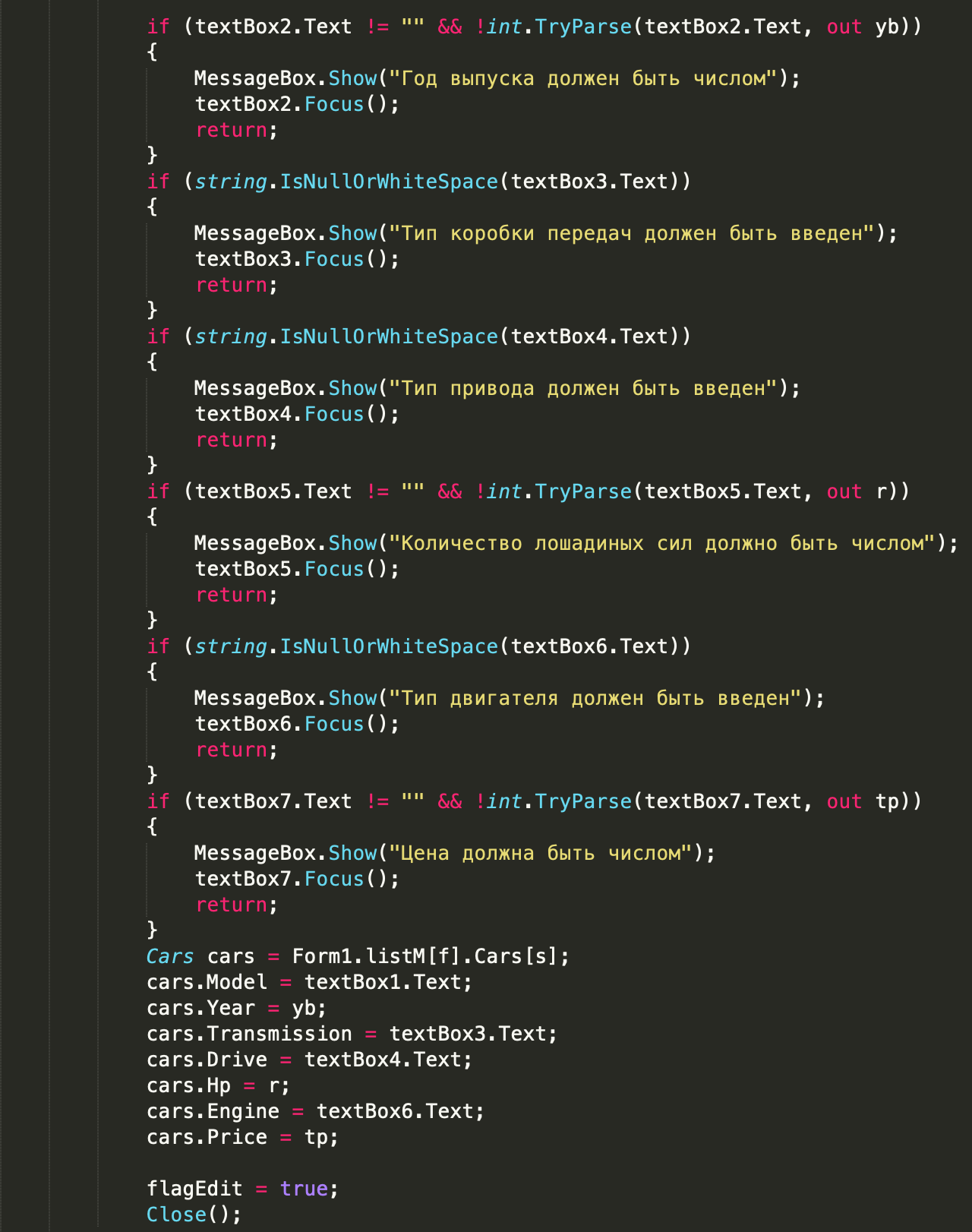


Нажатие на кнопку «Удалить» покажет пользователю инструкцию о том, как удалить данные из таблиц при помощи клавиши «Delete» на клавиатуре подобно тому, как это сделано на первой форме.

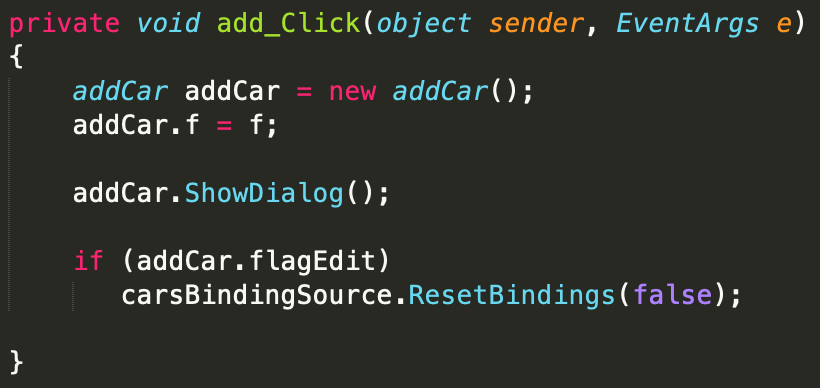
Нажатие на кнопку и код открывающейся формы «Изменить»:

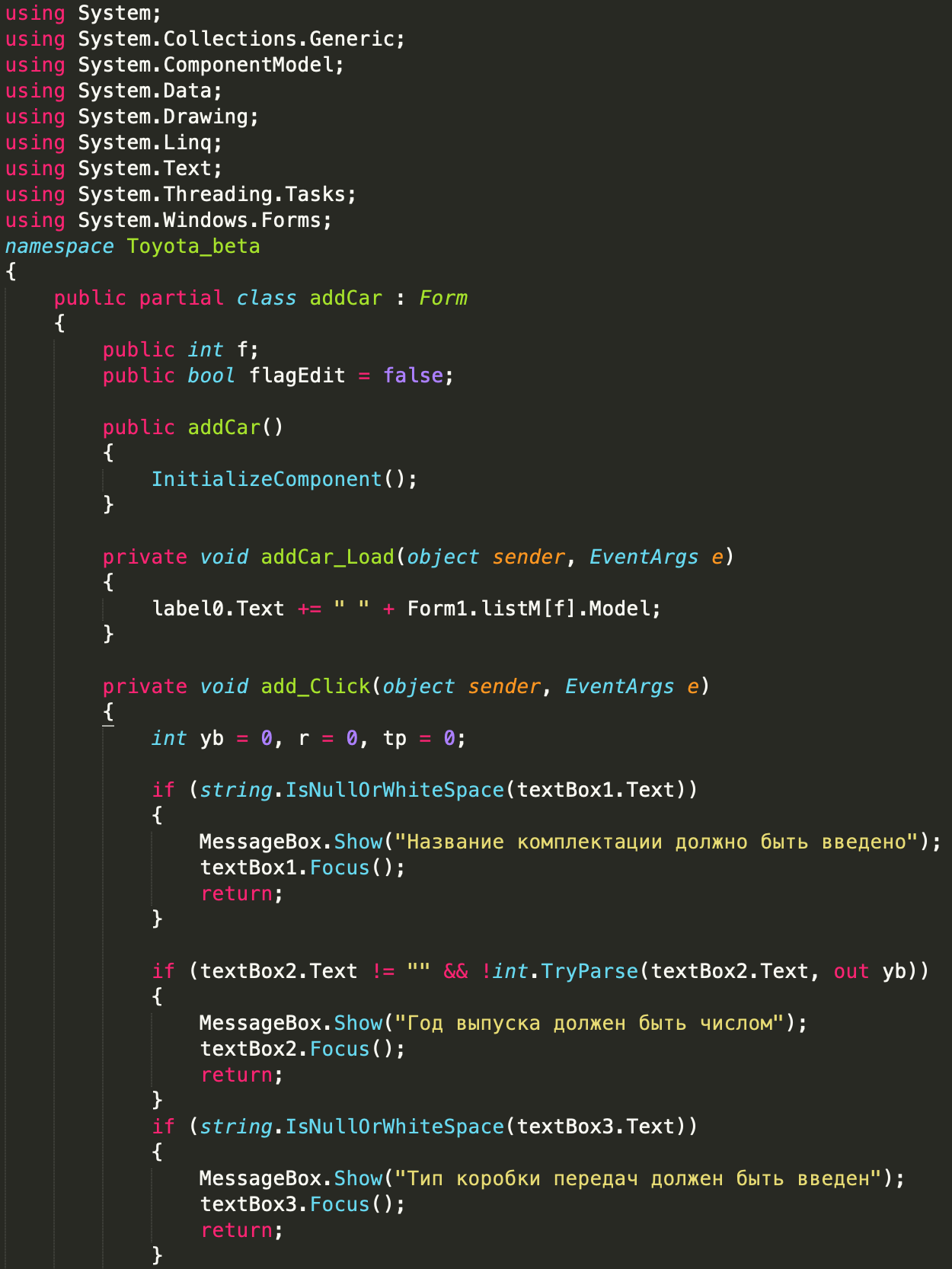




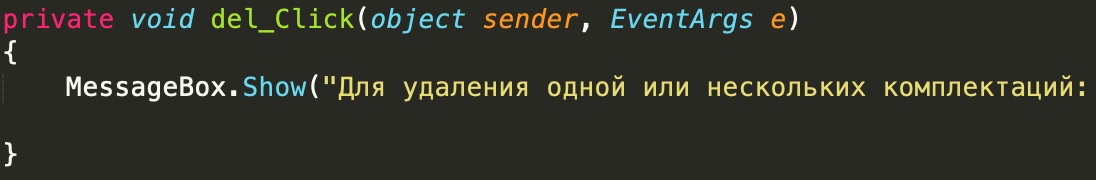


Нажатие на кнопку и код открывающейся формы «Добавить»:





Нажатие на кнопку «Удалить»:



В левом нижнем углу формы находится кнопка «Назад», которая схожа по своему действию с крестиком в верхнем углу формы. Она существует для более логического перехода пользователя обратно к главной форме и сохранению параметров приложения.

В текущей форме также используется toolTip, который при наведении на любые элементы подскажет пользователю как взаимодействовать с ней.

**Адаптивность и сохранение параметров**

Для сохранения параметров приложения были использованы:

В главной форме:



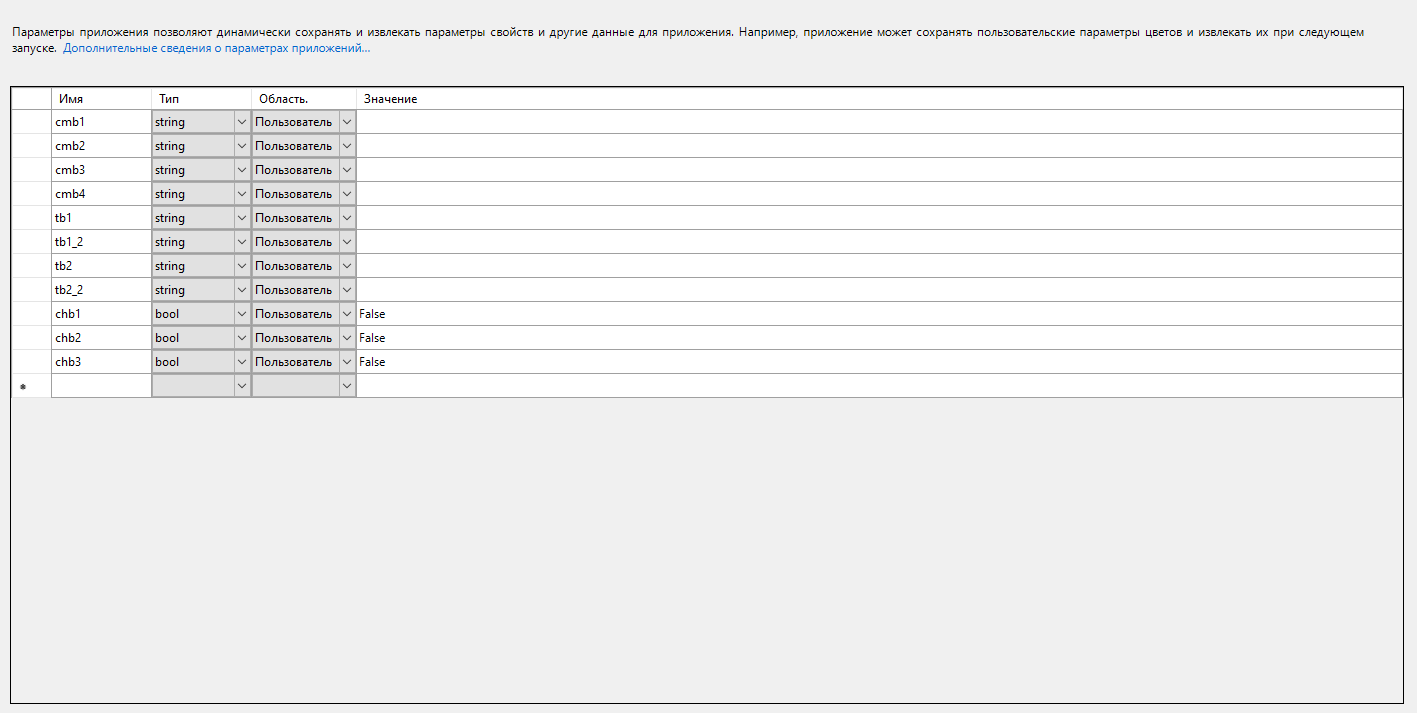
По нажатию на кнопку «Выйти с сохранением»



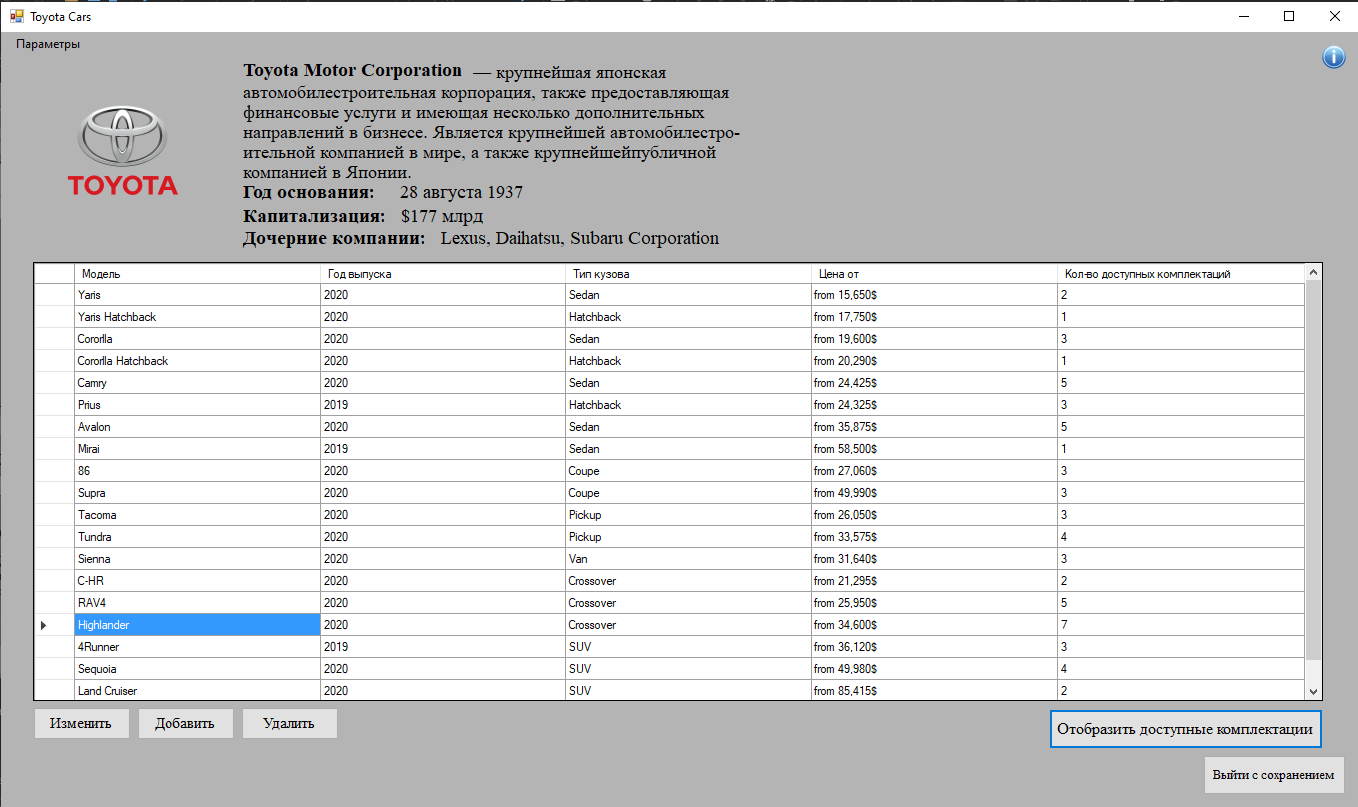
Во торой форме:



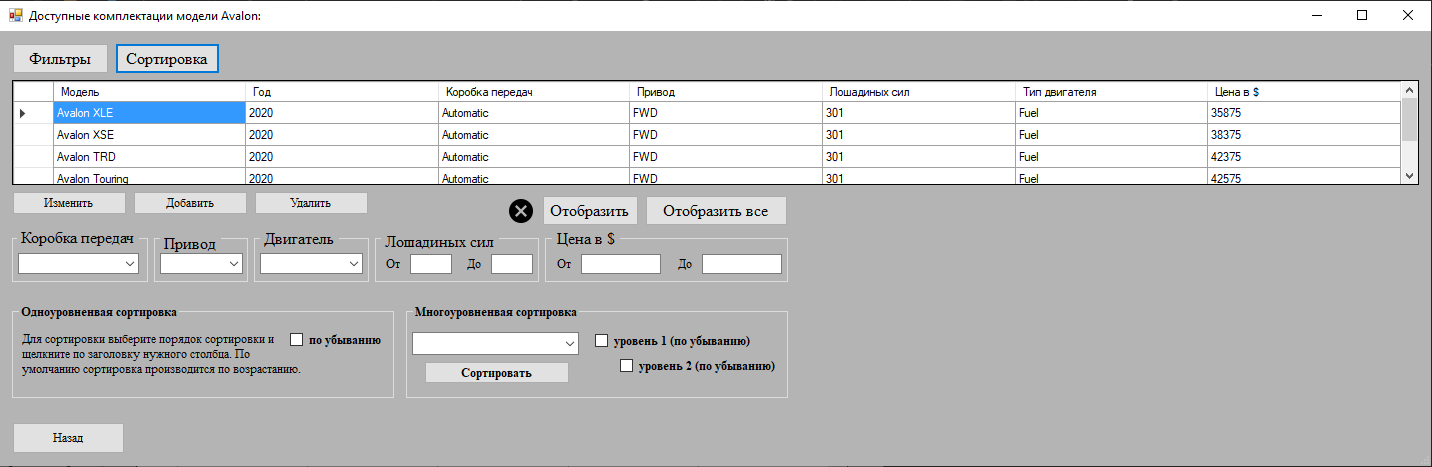
Сохраняемые настройки:



Приложение также адаптивно к любым размерам, которые необходимы пользователю и сохраняет размеры формы по нажатию кнопки сохранения.

Приложение растянуто во весь экран:

Приложение вытянуто в ширину:



**Вывод**

В процессе выполнения курсовой работы были изучены основные методы работы в среде разработки Microsoft Visual Studio. Разработана программа для автоматизации учета наличия моделей и их комплектаций. Функционал разработанной программы соответствует постановке задачи. В процессе работы над реализацией приложения были сделаны выводы о функциональных возможностях среды разработки, которые позволяют разрабатывать широкий спектр прикладного программного обеспечения для всех сфер автоматизации производства. В качестве наполнения были использованы автомобили Toyota, доступные для покупки в Соединенных Штатах Америки. Решение использовать именно эти данные было обусловлено тем, что ассортимент моделей и комплектаций доступных к покупке в данном регионе обширнее чем в России.

**Список литературы**

1. Горелов С.В. Современные технологии программирования: разработка Windows-приложений на языке C#: Учебник. В 2т.Т.1 – М.: Прометей. 2019. – 362 с.
2. Горелов С.В. Современные технологии программирования: разработка Windows-приложений на языке C#: Учебник. В 2т.Т.2 – М.: Прометей. 2019. – 378 с.
3. Горелов С.В., Волков А.Г. Учебное пособие по дисциплине «Современные языки программирования». Разработка приложений на языке С# для работы с базами данных. – М.: Финансовый университет, 2016. – 224 c.
4. https://www.toyota.com
5. https://auto.ru
6. https://www.youtube.com/channel/UCWIWPn-FoQREs95n7o3gU-Q
7. https://docs.microsoft.com/ru-ru/dotnet/csharp/
8. https://www.w3schools.com/cs/
9. https://www.bestprog.net/ru/sitemap\_ru/c-2/
10. https://www.youtube.com/channel/UCtqA2-drzzThgJ7ih0DHi5w