Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

филиал «Минский радиотехнический колледж»

**Отчет по практике по программированию**

Выполнила: учащаяся группы

Руководитель практики: Лазицкас Е.А.

Минск 2018

Содержание

Введение.................................................................................................................................3

1 Постановка задачи..............................................................................................................4

1.1 Описание предметной области.......................................................................................4

1.2 Обзор существующих аналогов.....................................................................................4

1.3 Информационная база.....................................................................................................6

1.4 Формулировка задачи.....................................................................................................6

2 Проектирование задачи.....................................................................................................7

2.1 Обоснования выбора средств и среды разработки......................................................7

2.2 Разработка алгоритма решения.....................................................................................8

2.3 Проектирование интерфейса........................................................................................10

3 Реализация.........................................................................................................................11

3.1 Описание разработанных классов................................................................................11

3.2 Описание реализованных методов...............................................................................12

3.3 Компоненты Windows Forms........................................................................................13

4 Тестирование.....................................................................................................................14

5 Применение.......................................................................................................................16

5.1 Установка и условия использование приложения......................................................16

5.2 Требования к аппаратно-программным ресурсам ПК...............................................16

5.3 Демонстрационный пример работы............................................................................16

Заключение..........................................................................................................................25

Список использованных источников................................................................................26

Приложение А Текст программы......................................................................................27

Приложение Б Экранные формы.......................................................................................35

Приложение В Блок-схема алгоритма..............................................................................37

Введение

Для того, чтобы получить опыт, необходимо практиковаться. Это очень важный элемент учёбы, ведь теоретические знания без практических – ничто. Невозможно научиться говорить на иностранном языке, если не произносить слова, не общаться с преподавателем или носителем языка, невозможно научиться писать код, если не практиковаться писать его в среде разработки. Именно поэтому нужно сочетать эти два метода получения знаний для максимизации полученной информации. Но во всём следует ставить цели и задачи, ведь без стратегии никогда не выиграть. Точно так же цели и задачи стоит поставить конкретно, чтобы не отвлекаться на то, что не имеет большого значения. Я поставила для себя следующие цели и задачи:

- данная практика проходит в филиале БГУИР «МРК»;

- необходимо повысить свой уровень написания программ, для этого постоянно практиковаться в их написании;

- научиться работать как в коллективе, так и индивидуально.

1 Постановка задачи

* 1. Описание предметной области

Автоматизации учета принимаемых звонков предназначена для международного отделения Автоматической телефонной станции (далее АТС). Основным предназначением АТС является обеспечение связью своих внутренних абонентов. Выход на городские телефоны ограничивается входящими в АТС городскими линиями. Сама АТС не может увеличить их количество, однако, использование и монтаж АТС позволяет оптимизировать связь благодаря грамотному распределению доступных линий. Это значит, что линия и номер телефона не закреплены за каким-либо конкретным аппаратом, а могут быть задействованы с любого телефона. АТС автоматически определяет свободную линию и проводит звонок (входящий или исходящий) через нее. Если же АТС является одним из элементов некоторой телефонной сети — то данная АТС должна обеспечивать соединение и поддержание связи со всеми абонентами сети — как «своими», так и подключенными к другим АТС сети.

Пользователями являются заведующий отделением и оператор АТС. Используется справочник, в котором прописаны города, их код и тариф за одну минуту. Диапазон работы АТС – страны Европы, а именно, их столицы. Список доступных городов должен обновляться, тем самым давая возможность звонить не только в столицы государств.

При создании записи формируется квитанция. Ей присваивается идентификационный номер, формируемый в зависимости от количества уже созданных квитанций. Квитанция содержит:

* дату разговора;
* код и название города;
* время разговора;
* стоимость (рассчитывается по формуле [тариф города за одну минуту] \* [продолжительность звонка]);
* номер телефона в этом городе;
* номер телефона абонента (абонент – это тот, кто обладает правом постоянного пользования какой-либо услугой, в данной случаем, связи).

Запрашиваемый пользователем отчет содержит:

* общее время разговора для каждого города;
* сумма на указанную дату (вводится с клавиатуры);
* название города, с абонентами которого наибольшее время разговоров.

|  |  |
| --- | --- |
| Тарифы на услуги по предоставлению международных телефонных соединений абонентам сети электросвязи общего пользования, оказываемые юридическим лицам и индивидуальным предпринимателям  Вводятся с 01 сентября 2019 года | Тариф за каждую полную или неполную минуту телефонного соединения без учета налога на добавленную стоимость, белорусских рублей |
| 1 СНГ I | 0,320 |
| 2 СНГ II | 0,352 |
| 3 СНГ III | 0,448 |
| 4 Европа I | 0,328 |
| 5 Европа II | 0,352 |
| 6 Европа III | 0,440 |

|  |  |
| --- | --- |
| Тарифная зона | Название страны |
| СНГ I | Россия (Москва) |
| Украина (Киев) |
| СНГ II | Армения (Ереван) |
| Грузия (Тбилиси) |
| Казахстан (Астана) |
| Кыргызстан (Бишкек) |
| Молдова (Кишинев) |
| Таджикистан (Душанбе) |
| Туркменистан (Ашхабад) |
| Узбекистан (Ташкент) |
| СНГ III | Азербайджан (Баку) |
| ЕВРОПА I | Германия (Берлин) |
| Италия,(Рим) |
| Литва (Вильнюс) |
| Польша (Варшава) |
| ЕВРОПА II | Австрия (Вена) |
| Болгария (София) |
| Босния и Герцеговина (Сараево) |
| Венгрия (Будапешт) |
| Латвия (Рига) |
| Македония (Скопье) |
| Румыния (Бухарест) |
| Сербия (Белград) |
| Словакия (Братислава) |
| Словения (Любляна) |
| Хорватия (Загреб) |
| Черногория (Подгорица) |
| Чехия (Прага) |
| Эстония (Таллинн) |
|  |
| ЕВРОПА III | Албания (Тирана) |
| Андорра (Андора-ла-Велья) |
| Бельгия (Брюссель) |
| Великобритания (Лондон) |
| Гибралтар (Гибралтар) |
| Греция (Афины) |
| Дания (Копенгаген) |
| Ирландия (Дублин) |
| Исландия (Рейкьявик) |
| Испания (Мадрид), включая Канарские о-ва |
| Кипр (Никосия) |
| Лихтинштейн (Вадуц) |
| Люксембург (Люксембург) |
| Мальта (Валлетта) |
| Монако (Монако) |
| Нидерланды (Амстердам) |
| Норвегия (Осло) |
| Португалия (Лиссабон), включая Мадейру и Азорские о-ва |
| Сан-Марино (Сан-Марино) |
| Турция (Анкара), включая Северный Кипр |
| Фарерские о-ва, территория Дании (Торсхавн) |
| Финляндия, (Хельсинки) |
| Франция (Париж), включая о-в Корсика |
| Швейцария (Берн) |
| Швеция (Стокгольм) |

* 1. Обзор существующих аналогов

Аналог – это то, что представляет собою подобие чего-либо, соответствие какому-либо предмету, явлению, понятию.

У большинства программ существуют аналоги, а для того, чтобы не создавать то, что ужу существует необходимо ознакомиться с ними.

Phontage

Рисунок 1 – Окна Phontage

Разработчик: Ericsson-LG, Республика Корея

Статус программы: лицензия

Версия: E.3Df

Интерфейс: русский

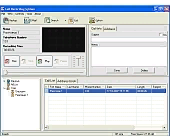
Система: Windows 2000/XP/7

Возможности приложения Phontage:

* авторизация абонента по id и паролю, без указания mac адреса;
* отображение номера звонящего при входящем вызове;
* удержание вызовов, перевод вызовов;
* программируемые клавиши;
* набор из записной книжки, меню быстрого вызова;
* всплывающее окно при входящем звонке;
* различные режимы отображения информации на дисплее;
* отображение списка вызовов;
* запись разговоров на жесткий диск компьютера;
* синхронизация с outlook, excel, goldmine, act;
* пользовательские профили;
* поддержка различных языков;
* регулировка громкости непосредственно из окна программы.

Недостаток: без лицензии допускается подключение к станции не более двух программных ip-терминалов, более 2 - необходима лицензия.

Call recording system

Рисунок 2 – Окно Call recording system

Разработчик: LG-Nortel, Ю.Корея

Статус программы: бесплатная

Версия: A.0Cb

Интерфейс: английский

Размер файла: 6.22 Mb zip-архив

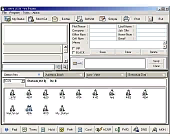
Система: Windows 2000/XP/Vista

Дата: добавлено: 24 марта 2010

Возможности приложения Call recording system:

* авторизация абонента по id и паролю, без указания mac адреса;
* разговоры записываются на жесткий диск компьютера;
* каждый разговор записывается в виде отдельного файла с автоматической фиксацией в базе данных программы номера звонящего, даты, времени и продолжительности;
* запись разговора включается оператором, принявшем вызов, нажатием кнопки на экране компьютера или путем использования горячих клавиш;
* встроенная возможность отправки записанных разговоров через электронную почту;
* поиск по базе данных записанных разговоров;
* прослушивание записанных разговоров;
* опции и настройки пользователя;
* резервное копирование записанных разговоров.

ez-Phone

Рисунок 3 – Окно ez-Phone

Разработчик: LG-Nortel, Ю.Корея

Статус программы: демо-версия на 30 дней

Версия: 3.8 Ab

Интерфейс: английский

Размер файла: 15.3 Mb zip-архив

Система: Windows 2000/XP/Vista

Дата: добавлено: 24 марта 2010

Возможности приложения ez-Phone:

* импорт и экспорт различных баз данных (outlook, gold mine, act!, access, excel);
* совместная работа со списком контактов outlook’a в реальном времени.;
* список зарезервированных наборов (номеров). (операция автоматического вызова);
* 3 вида всплывающих окон для извещения о входящем вызове;
* упрощенный вид рабочего окна (simple mode) и индивидуальный список напоминаний;
* осуществление исходящих вызовов, обработка входящих вызовов: постановка на удержание, перевод, парковка/снятие с парковки, конференция и т.д.;
* журнал вызовов для получения статистической информации;
* использование разнородных баз телефонных номеров;
* просмотр статуса обрабатываемого вызова;
* опции и настройки пользователя;
* отправка sms на мобильный телефон.

Недостаток: бесплатна только на 30 дней и только в демо-версии.

* 1. Информационная база

Информационная база данного ПО должна состоять из следующих входных данных:

* ­информация, которую вводит пользователь для заполнения квитанции звонка (дата, название города, код города, тариф (через запятую), стоимость, количество полных минут разговора, номер телефона звонящего, номер телефона абонента);
* текстовый файл с расширением txt;
* запросы пользователя при нажатии на различные элементы в процессе работы программы, для выполнения различных функций, таких как удаление, изменение, сохранение, определение города с наибольшем временем разговоров, вывод суммы на определенную дату.

А также из следующих выходных данных:

* информация, отображаемая на экране (интерфейс программы);
* стоимость звонка;
* сумма на запрашиваемую пользователем дату;
* название города наибольшим временем разговоров;
* текстовый файл с расширением docx (функция сохранение данных в файл).

К постоянным данным относятся:

* тарифы;
* города.
  1. Формулировка задачи

Создать MDI-приложение с использованием технологии Windows Forms на языке программирования C#, которое позволяет:

* сохранять вводимые данные (квитанции) в файле docx с названием, соответствующим идентификационному номеру квитанции;
* просматривать, изменять, удалять записи из файла;
* выводить результаты работы программы на экран и сохранять в другой текстовый файл;

Реализовать возможность переносить данные из одной квитанции в другую.

Добавить пункты меню для сохранения объектов в файл и загрузки данных из файла. При этом использовать стандартные диалоговые окна и механизм сериализации. В классе добавить поле дата создания объекта. Это поле не сериализовать, а при десериализации заново устанавливать по системной дате.

Основные требования к пользовательскому интерфейсу:

* приложение MDI-формата;
* функциональность (соответствие задачам пользователя);
* понятность и логичность (логичность расположения элементов на форме и удобство при эксплуатации программы);
* обеспечение защиты от человеческих ошибок;
* быстрое обучение пользователя (справка, в которой кратко и понятно описана работа с программой).

2. Проектирование задачи

2.1 Обоснования выбора средств и среды разработки

Выбор средств и среды разработки оказывает огромное влияние на процесс разработки и конечное ПС. Для создания ПС, которое будет удовлетворять всем вашим идеям и требованиям, необходимо выбирать средства и среду, исходя из их возможностей, но и реально смотреть на свои.

* среда разработки – Visual Studio 2019

Мой выбор пал на этот вариант, т.к. данная среда разработки имеет удобный интерфейс, является современной и обладает большими возможностями. В частности, встроенный редактор исходного кода с поддержкой технологии IntelliSense значительно упрощает процесс написания кода, ускоряя его. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Также встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. В среде существует довольно широкий выбор языков программирования, что также является значительным плюсом.

* интерфейс программирования приложений – Windows Forms

Windows Forms – интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework, упрощающий доступ к элементам интерфейса Microsoft Windows. Один из наиболее важных плюсов данного интерфейса – управляемый код – классы, реализующие API для Windows Forms, не зависят от языка разработки, следовательно, это дает возможность программисту использовать данный интерфейс не зависимо от выбора одного из предлагаемых языков средой разработки Visual Studio 2019.

* способ организации графического интерфейса пользователя – MDI

Multiple document interface (MDI) – способ организации графического интерфейса пользователя, предполагающий использование оконного интерфейса, в котором большинство окон (исключая, как правило, только модальные окна) расположены внутри одного общего окна. Этим он и отличается от SDI, в котором окна располагаются независимо друг от друга. У него много преимуществ, одно из которых: уменьшение загромождённости экрана элементами интерфейса, все окна приложения можно прятать/показывать, сворачивать/разворачивать и проводить с ними другие манипуляции, как с одним окном и т.д.;

* язык программирования – C#

C# - объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework. На данный момент C# — один из самых популярных языков программирования. Это важно для разработчиков, поскольку популярность языка прямо пропорциональна тому, насколько для него будут доступны онлайн-материалы. Чаще всего, все обращаются к Google или Stack Overflow для решения задач в разработке и чаще всего можно найти большое количество ответов по C#. Это сэкономит огромное количество времени новичкам при решении различных задач в разработке.

Гибкость языка C# является огромным преимуществом, по сравнению с Delphi. Разнообразие приложений, которые могут быть разработаны с помощью С#, .Net и Visual Studio практически безгранично:

* приложения для Windows;
* мобильные приложения;
* веб-приложения;
* игры;
* приложения для Android и iOS, которые разрабатываются с помощью дополнительных фреймворков, таких как Xamarin или Mono.

Вывод: разработка MDI-приложения с использованием технологии Windows Forms на языке программирования C# в Visual Studio 2019.

2.2 Разработка алгоритма решения

Начало работы программы предполагается с процедуры, которая заполняет таблицу данными, считанными из файла с расширением .xml. Дальнейшие действия пользователь выбирает сам (добавление, редактирование, удаление информации, поиск по критериям) (рисунок 2).

Предполагается, что при выборе элемента меню «Данные» можно будет осуществлять такие функции, как «Добавить», реализует добавление информации о новом участнике автоклуба, «Редактировать», реализует корректировку информации о текущих участниках автоклуба, «Удалить», реализует полное удаление информации о выбранном участнике автоклуба. В результате выбора какой-либо из данных функций подразумевается отображение измененной таблицы с данными. После этого существует возможность заново выполнить какую-либо функцию. Планируется возможность выбора элемента меню «Файл», предложенными вариантами которого являются: «Открыть», при выборе которого будет открыт файл, выбранный пользователем в диалоговом окне; «Сохранить», при выборе открывается диалоговое окно, в котором можно задать имя файла, данные из таблицы записываются в файл с расширением xml; «Сохранить как», при выборе открывается диалоговое окно, в котором можно задать имя файла с желаемым расширением, в который запишутся данные из таблицы.

Все перечисленные выше функции предполагается использовать неограниченное количество раз.

Алгоритм программы можно просмотреть в приложении В.

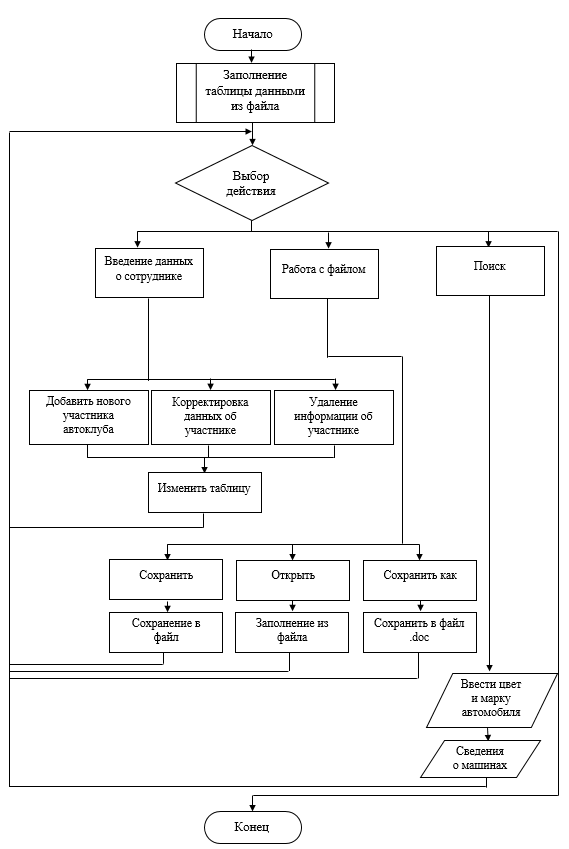


Рисунок 3 – Предполагаемый алгоритм работы программы

* 1. Проектирование интерфейса

В главной (родительской) форме планируется наличие одной формы с таблицей, где можно просматривать список сотрудников предприятия и данные о них. С помощью меню на главной форме можно осуществлять различные функции: добавление новых сотрудников в список, корректировать уже имеющуюся информацию о них и т.д. А также с помощью меню сверху можно будет вызывать другие формы, на которых будут присутствовать поля для заполнения или корректировки.

Примерный интерфейс приведен ниже на рисунке 4.

Меню

Главная форма

Добавить/Редактировать

Поиск

Рисунок 4 – Предполагаемый интерфейс

1. Реализация

3.1 Описание разработанных классов

Разработанная программа имеет 6 классов:

- Program – особый вид классов, в котором находится метод Main, представляющий собой точку входа в программу, содержащий в блоке инициализацию главной формы (Form1).

- Car – класс, необходимый для работы с данными. В нем прописаны поля, которые сериализуются: brand, year, num\_1, num\_2, color, number, owner, а также поле d\_ofcreate, которое не подлежит сериализации. Также в классе прописаны два конструктора, один базовый, а другой принимающий параметры.

- Form1 – класс главной формы под названием «Автоклуб», содержащий два экземпляра классов Add и Show, которые описываются ниже, список List<Car>,метод Form, который запускается первым при вызове этой формы. Также содержаться следующие методы: сохранитьToolStripMenuItem\_Click, в котором осуществляется XML сериализация даных, при помощи класса XmlSerializer, с последующей записью в файл с расширением .xml; открытьToolStripMenuItem\_Click – осуществляет десериализацию с использованием класса XmlSerializer, и вызывает метод updateTable, обновляющий данные в элементе управления dataGridView1; поМаркеИЦветуToolStripMenuItem\_Click и поМоментуВыпускаToolStripMenuItem\_Click - оба метода вызывают форму поиска Search, задают им определенные параметры и передают некоторые данные из главной формы; AddToolStripMenuItem1\_Click – вызывает форму под названием «Добавление»; EditToolStripMenuItem\_Click – вызывает форму под названием «Редактировать»; DelToolStripMenuItem\_Click – удаляет строку по выбранной пользователем ячейке; сохранитьКакToolStripMenuItem\_Click – осуществляет запись данных из элемента управления dataGridView1 в файл с расширением .txt; сохранитьВExcelToolStripMenuItem\_Click – вызывает метод SaveTable, принимающий параметром dataGridView1; SaveTable – осуществляет запись данных из элемента управления dataGridView1 в report.xlsx.

- Add – класс формы под названием «Добавить», содержащей экземпляр класса Form1 form и следующие методы: Add, set, обеспечивающий взаимодействие между этой формой и главной (Form1); button\_Click – записывает данные из элементов TextBox в dataGridView1, находящийся на главной форме, а также записывает эти же данные в список типа Car под названием cars.

- Edit – класс формы под названием «Редактировать», содержащий экземпляр класса Form1 form, а также переменные int row и int car\_id. В состав класса входят следующие методы: Edit, set – принимающий в качестве параметров экземпляр класса Form1 f, int row и int car\_id, осуществляющий взаимодействие формы с главной, инициализацию переменных int row и int car\_id и заполнение элементов TextBox полями объекта списка; button\_Click – перезаписывает данные из TextBox в поля объекта списка, из которого ранее записывались данные в TextBox.

- Search – класс формы под названием «Поиск», содержащий метод set и searches\_Click, осуществляющий проверку соответствия критериям поиска и вызывающий метод set, заполняющий элемент управления dataGridView1 подходящими данными.

* 1. Описание реализованных методов

- private void сохранитьToolStripMenuItem\_Click(object sender, EventArgs e) – метод класса Form1, который вызывается при выборе элемента меню «Файл» вариант «Сохранить». Осуществляет сериализацию данных при помощи класса XmlSerializer с последующей записью результата в файл с расширением .xml;

- private void открытьToolStripMenuItem\_Click(object sender, EventArgs e) – метод, который вызывается при выборе элемента меню «Файл» вариант «Открыть». Осуществляет десериализацию данных с последующей записью в список типа Car, а также вызывающий метод updateTable;

- public void updateTable() – метод, предназначенный для обновления содержимого элемента управления dataGridView1 после таких действий как «Открыть» и «Удалить»;

- private void поМаркеИЦветуToolStripMenuItem\_Click(object sender, EventArgs e) – метод, предназначенный для вызова формы под названием «Поиск» при выборе элемента меню «Поиск» вариант «1. по марке и цвету» задающий некоторые свойства элементов управления Label вызываемой формы;

- private void поМоментуВыпускаToolStripMenuItem\_Click(object sender, EventArgs e) – метод, предназначенный для подсчета значения среднего возраста всех автомобилей списка cars, проверки автомобилей списка на соответствие условию (срок службы больше среднего возраста всех автомобилей в списке), вызова формы под названием «Поиск» при выборе элемента меню «Поиск» вариант «2. по моменту выпуска», установки некоторых свойств элементов Label и вызова метода set класса Search;

- private void AddToolStripMenuItem1\_Click(object sender, EventArgs e) – метод вызывающий форму под названием «Добавление» при выборе элемента меню «Данные» варианта «Добавить», метод set класса Add, очищающий свойства Text всех TextBox формы «Добавление»;

- private void EditToolStripMenuItem\_Click(object sender, EventArgs e) – метод, вызывающий форму под названием «Редактировать» при выборе элемента меню «Данные» варианта «Редактировать» и передающий методу set этой формы в качестве параметров следующие значения: ссылку на главную форму, координаты выбранной ячейки и значение поля id выбранной строки;

- private void DelToolStripMenuItem\_Click(object sender, EventArgs e) – метод, осуществляющий удаление выбранного объекта списка cars с предшествующей проверкой на наличие в списке объектов, а также вызывает метод updateTable, который описывался выше при выборе элемента меню «Данные» варианта «Удаление»;

- private void сохранитьКакToolStripMenuItem\_Click(object sender, EventArgs e) – метод, осуществляющий запись данных из dataGridView1 главной формы в файл с расширением .txt при помощи экземпляров классов FileStream и StreamWriter при выборе элемента меню «Файл» варианта «Сохранить как .txt»;

- private void сохранитьВExcelToolStripMenuItem\_Click(object sender, EventArgs e) – метод, вызывающий метод SaveTable и передающий ему в качестве параметра dataGridView1 при выборе элемента меню «Файл» варианта «Сохранить в Excel»;

- void SaveTable(DataGridView What\_Save) – метод, осуществляющий запись данных из dataGridView1 в Save\_Channel.xlsx с помощью экземпляра класса Excel.Application;

- public void set (Form1 f) – метод класса Add, предназначенный для организации взаимодействия между формами под названием «Добавить» и «Автоклуб», принимающий в качестве параметра экземпляр класса Form1 f;

- private void button\_Click(object sender, EventArgs e) – метод, который вызывается при нажатии кнопки «Добавить» на форме под названием «Добавить», осуществляющий добавление новых данных в элемент dataGridView1 главной формы считывая то, что пользователь ввел в TextBox-ы и добавляя новый объект в список cars, после чего закрывает форму «Добавить»;

- public void set(Form1 f, int row, int car\_id) – метод класса Edit, осуществляющий связь формы «Редактировать» и «Автоклуб», а также считывает значения полей объекта списка cars в TextBox-ы формы «Редактировать»;

- private void button\_Click(object sender, EventArgs e) – метод, который вызывается при нажатии кнопки «Изменить» на форме «Редактировать», в котором создается новый экземпляр класса Car в поля которого записываются значения из TextBox-ов, после чего объект списка cars перезаписывается и изменяются данные в dataGridView1 формы «Автоклуб», после чего форма «Редактировать» закрывается;

- public void set(List<Car> cars) – метод класса Search, принимающий в качестве параметра список cars типа Car, выводящий в label под названием count на форме «Поиск» значение cars.count, осуществляющий проверку на cars.Count() > 0 и если условие верно, то элемент dataGridView1 заполняется объектами списка cars, подходящими по условиям, если нет то выводи сообщение об ошибке;

- private void searches\_Click(object sender, EventArgs e) – метод, вызываемый при нажатии на кнопку «Найти» на форме «Поиск», создающий новый список List<Car> ans и проверяющий, равно ли значение, которое ввел пользователь в TextBox-ы значениям полей brand и color всех объектов списка cars и если условие выполняется, то объект списка cars добавляется в новый список ans, который после проверки всех объектов списка cars передается в качестве параметра методу set, который описывался выше.

* 1. Компоненты Windows Forms

Были использованы следующие компоненты:

- menuStrip – элемент интерфейса пользователя, позволяющий выбрать одну из нескольких перечисленных опций программы;

- Label – элемент управления, предназначенный для отображения текста на форме;

- SaveFileDialog – элемент управления, вызывающий диалоговое окно, где можно выбрать куда сохранить файл и под каким именем;

- OpenFileDialog – элемент управления, вызывающий диалоговое окно, при помощи которого можно выбрать файл для открытия;

- DataGridView – элемент управления, который предназначен для отображения данных в виде таблицы;

- TextBox – элемент управления, в который можно вводить информацию и использовать ее в дальнейшем;

- Button – элемент управления, при клике на который происходит какое-либо программно- прописанное действие.

1. Тестирование

Тестирование – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом и выявление ситуаций, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Существующие на сегодня методы тестирования программного обеспечения не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого программного обеспечения.

Существует несколько признаков, по которым принято производить классификацию видов тестирования. Обычно выделяют следующие:

- по объекту тестирования;

- по знанию системы (тестирование чёрного ящика, тестирование белого ящика, тестирование серого ящика);

- по степени автоматизации;

- по степени изолированности компонентов;

- по времени проведения тестирования;

- по признаку позитивности сценариев;

- по степени подготовленности к тестированию.

На данный момент времени выделяют несколько видов тестирования по доступу к программному коду:

- тестирование «белого ящика» - тестирование программного кода с доступом к коду;

- тестирование «черного ящика» - тестирование без доступа к коду продукту;

- тестирование «серого ящика» - тестирование, основанное на ограниченном знании внутренней структуры программного продукта. Часть говорят, что это смесь тестирования «белого ящика» и «черного ящика», но это неверно. В данном случае тестировщик не работает с кодом программного продукта, но он знаком с внутренней структурой программы или приложения и взаимодействием между компонентами.

При тестировании разработанной программы использовалось тестирование «черного ящика» или поведенческое тестирование — метод тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве тестируемого объекта.

Стратегия поведенческого теста исходит из технических требований и их спецификаций (таблица 1).

Таблица 1 – Тестирование программного средства

|  |  |  |
| --- | --- | --- |
| Действие | Ожидание | Получено |
| Запуск программы | Открытие программы, где присутствует одно окно в другом, во внутреннем окне заполненная таблица | В результате тестов получено полное соответствие ожиданиям |
| Реакция на нажатие элемента меню Файл => Открыть | Открытие диалогового окна для выбора файла, заполнение таблицы данными из файла |
| Реакция на нажатие элемента меню Данные => Добавить | Открытие окна «Добавить», в котором присутствуют поля, которые необходимо заполнить, новая строка появляется в конце таблицы |
| Реакция на нажатие элемента меню Данные => Редактировать | Открытие окна «Редактирование» в котором присутствуют поля, заполненные данными из выбранной строки таблицы, обновленная строка остается на том же месте |
| Реакция на нажатие элемента меню Данные => Удалить | Удаление выбранной строки из таблицы |
| Реакция на нажатие элемента меню Файл => Сохранить | Открытие диалогового окна для выбора места сохранения файла и его имени, сохранение файл в указанном месте с расширением .xml |
| Реакция на нажатие элемента меню Файл => Сохранить как .txt | Сохранение данных из таблицы в файл report.txt |
| Реакция на нажатие элемента меню Файл => Сохранить в Excel | Сохранение данных из таблицы в файл Save\_Channel.xlsx |
| Реакция на нажатие элемента меню Поиск => 1. По марке и цвету | Открытие окна «Поиск» с полями, которые необходимо заполнить, после клика по кнопке отображение подходящих данных в таблице, при их отсутствии вывод сообщения об ошибке и просьбе проверить введенные данные |
| Реакция на нажатие элемента меню Поиск => 2. по моменту выпуска | Открытие окна «Поиск», отображение в таблице данных, подходящих по условию |

1. Применение
   1. Установка и условия использования приложения

Для работы в приложении необходимо установить его, для этого необходимо иметь файл Praza\_1.exe. Установка стандартная: двойной клик мышью по ярлыку и выбрать место сохранения приложения на жестком диске.

Данное ПО не нуждается в дополнительных файлах, но в течении работы приложения будут создаваться файлы, которые желательно располагать рядом с файлом приложения.

5.2 Требования к аппаратно-программным ресурсам ПК

Минимальные системные требования:

Microsoft Windows XP/Vista/7/8/8.1/10; Intel Pentium/Atom/Celeron 1,2 ГГц, ОЗУ 1 Гб, свободное место на жестком диске – 500 кб.

Рекомендуемые системные требования:

Intel Pentium/Atom/Celeron/Core i3 1,5 ГГц, ОЗУ 2 Гб, свободное место на жёстком диске – 1Мб.

5.3 Демонстрационный пример работы

Для того, чтобы открыть файл необходимо выбрать элемент меню «Файл» вариант «Открыть» (рисунок 5). После чего появляется диалоговое окно, в котором нужно выбрать файл, который надо открыть (рисунок 6), после чего данные из файла отобразятся в таблице (рисунок 7).

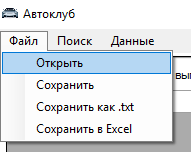


Рисунок 5 – Открытие файла с расширением .xml

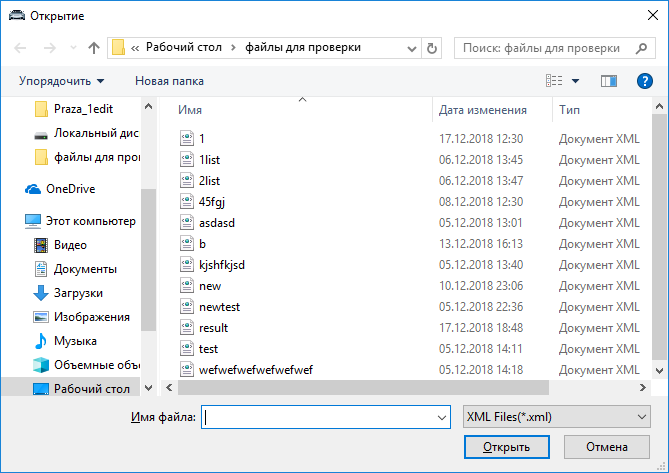


Рисунок 6 - Открытие диалогового окна

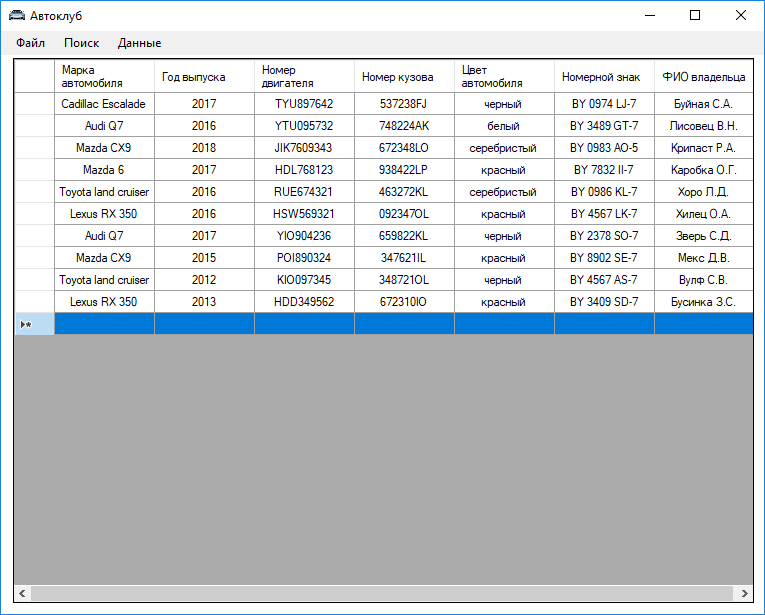


Рисунок 7 – Результат открытия файла

Для того, чтобы сохранить данные из таблицы в файл с расширением .txt необходимо в элементе меню «Файл» выбрать вариант «Сохранить как .txt» (рисунок 8). После чего данные появятся в файле report.txt

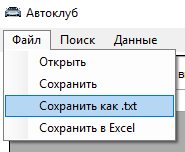


Рисунок 8 – сохранение в файл с расширением .txt

По умолчанию при выборе элемента меню «Файл» варианта «Сохранить» данные из таблицы записываются в файл с расширением .xml, при нажатии на меню открывается диалоговое окно, через которое можно указать место сохранения файла и его имя (рисунок 9).

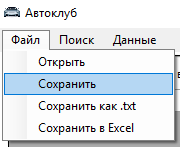


Рисунок 9 – сохранение в файл с расширением .xml

При выборе элемента меню «Файл» вариант «Сохранить в Excel» (рисунок 10) данные из таблицы записываются в файл Save\_Channel.xlsx.

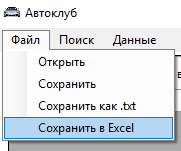
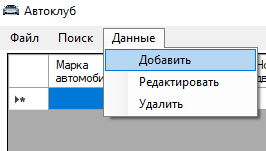


Рисунок 10 – сохранение в файл с расширением .xlsx

Для того, чтобы добавить информацию о новом участнике автоклуба нужно выбрать элемент меню «Данные» вариант «Добавить» (рисунок 11). После этого открывается форма под названием «Добавление» (рисунок 12), в которой после добавления данных нужно нажать кнопку «Добавить» после чего форма закроется, а данные в таблице на главной форме обновятся (рисунок 13).



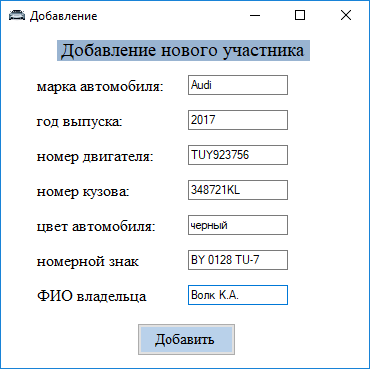
Рисунок 11 – добавление нового участника в список

Рисунок 12 – форма «Добавление»

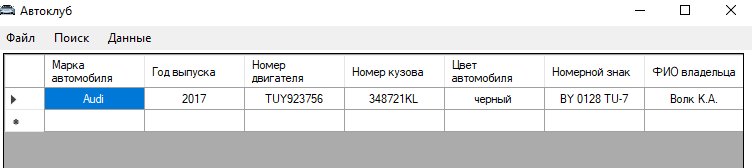


Рисунок 13 – демонстрация работы добавления

Для того, чтобы изменить информацию об участнике необходимо выделить ячейку или строку с информацией об участнике, данные о котором нужно изменить. После чего нужно выбрать элемент меню «Данные» вариант «Редактировать» (рисунок 14), после чего откроется форма под названием «Редактирование» (рисунок 15), в которой после изменения данных нужно нажать кнопку «Изменить» после чего изменяться данные в таблице на главной форме.

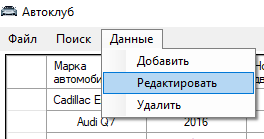


Рисунок 14 – редактирование информации об участнике

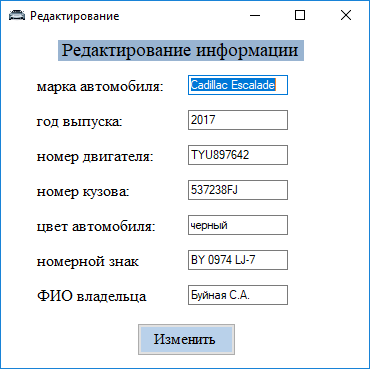
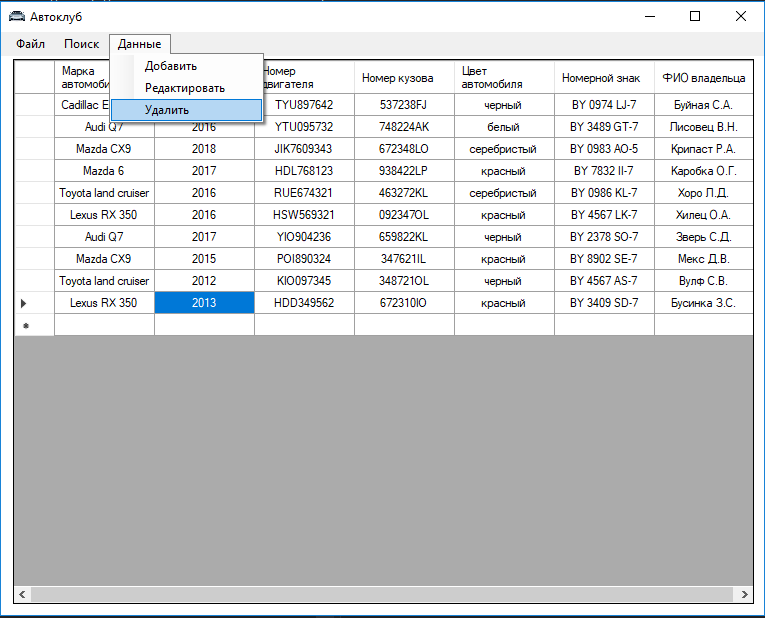


Рисунок 15 – форма «Редактирование»

Для того, чтобы удалить информацию об участнике клуба необходимо выделить ячейку в строке, где прописаны его данные и выбрать в элементе меню «Данные» вариант «Удалить» (рисунок 16) после чего в таблице удалится строка, которая была выбрана (рисунок 17).

Рисунок 16 – удаление информации об участнике

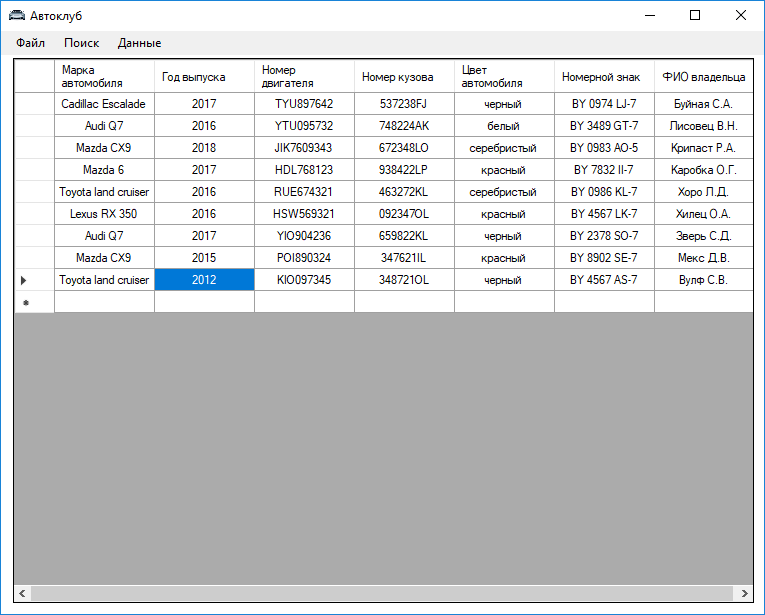


Рисунок 17 – демонстрация выполнения удаления

Для того, чтобы найти фамилии владельцев, которые имеют автомобили указанной марки одинакового цвета, которые вводятся пользователем необходимо выбрать пункт меню «Поиск» вариант «1. по марке и цвету» (рисунок 18). После чего будет открыта новая форма под названием «Поиск» с полями, в которые необходимо ввести критерии поиска (марку и цвет автомобиля) (рисунок 19), после нажать на кнопку «Найти» и если в списке членов клуба будут найдены совпадения, то в таблице на форме «Поиск» появится полная информация об автомобилях (рисунок 20). В противоположном случае появится сообщение с предупреждением (рисунок 21).

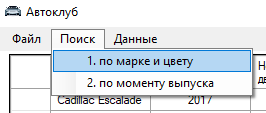


Рисунок 18 – поиск по марке и цвету

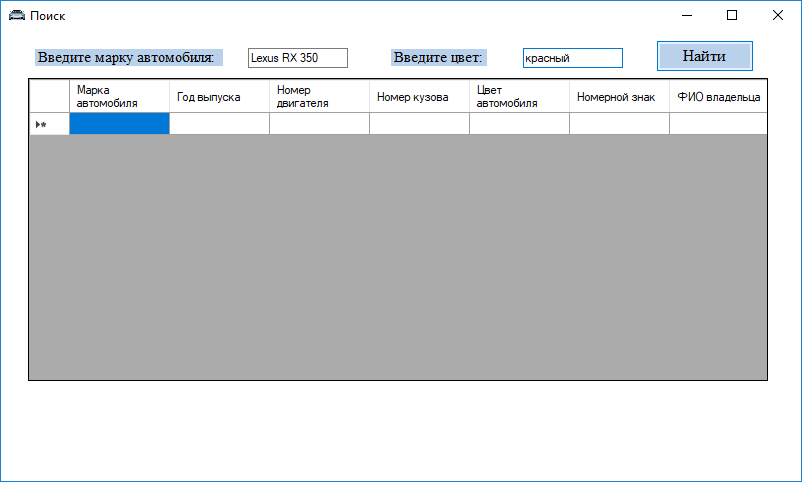
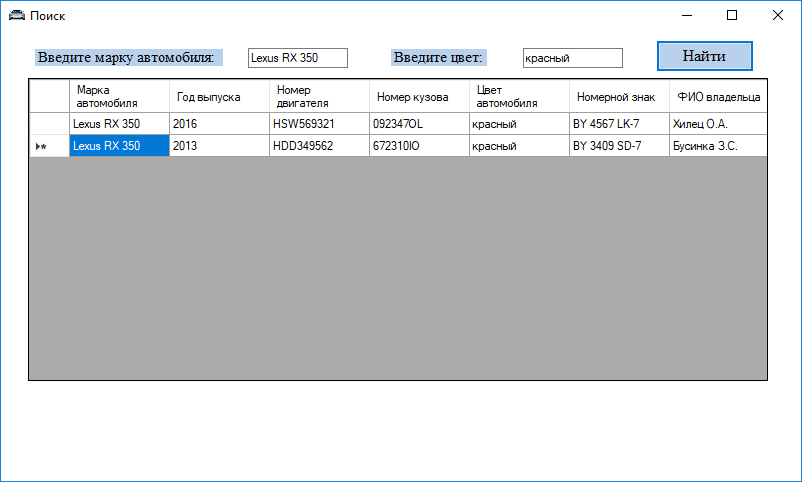


Рисунок 19 – форма «Поиск»

Рисунок 20 – демонстрация работы поиска по марке и цвету

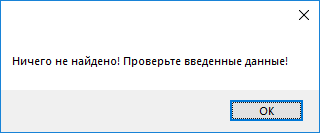


Рисунок 21 – сообщение с предупреждением

Также для того, чтобы подсчитать количество автомобилей и вывести фамилии их владельцев, с момента выпуска которых прошло более, чем средний возраст всех автомобилей в списке нужно, как и в предыдущем поиске выбрать элемент меню «Поиск» вариант «2. по моменту выпуска» (рисунок 22). После этого откроется форма «Поиск» с данными об автомобилях, которые удовлетворяют условию поиска (рисунок 23).

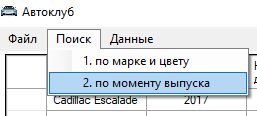


Рисунок 22 – поиск по моменту выпуска

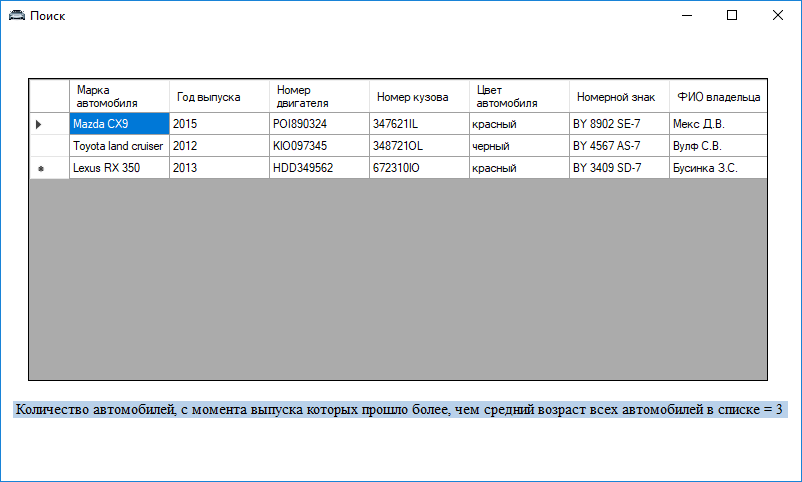


Рисунок 23 – демонстрация работы поиска по моменту выпуска

Заключение

Практика по программированию дала возможность получить дополнительный опыт в среде программирования Microsoft Visual Studio 2017, а также улучшить навыки в работе с языком программирования C#, что подразумевает под собой получение новых знаний, изучение предметной области и закрепление существующих умений.

Первым этапом разработки приложения после получения задания стало изучение предметной области, что дало ясное представление о том, какой должна быть программа, в частности с какими конкретными данными она должна работать и какие функции выполнять. Предметной областью являлись клубы автолюбителей, в следствии чего изучались их функции, структура и преимущества вхождения в клуб в качестве участника. Оперируя полученными данными был разработан простой и понятный интерфейс. Особое внимание уделялось формам работы с данными, т.к. они представляют собой полную информацию об автомобиле, входящем в клуб.

В дальнейшем изучалось создание MDI-приложения с использованием технологии WindowsForms на языке программирования C#.

Используя различные источники, было выполнено задание и улучшены навыки работы в среде разработки Visual Studio 2017 на языке программирования C#.

Список использованных источников

1. Албахарв, Джозеф. С# 6.0. Справочник. Полное описание языка, 6-е изд./ Албахарв, Джозеф : Пер. с англ. - М. : ООО
2. Стиллмен Э., Грин Дж. Изучаем C#. 3-е изд. — СПб.: Питер, 2014. — 816 с.: ил. — (Серия «Head First O’Reilly»).ISBN 978-5-496-00867-9
3. Шилдт, Герберт. C# 4.0: полное руководство.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011. —1056 с.: ил. — Парал. тит. англ.
4. Хейлсберг А., Торгерсен М., Вилтамут С., Голд П. Язык программирования C#. Классика Computers Science. 4-е изд./ — СПб.: Пи-тер, 2012. — 784 с.: ил.
5. C#. Введение в программирование. Учебное пособие/Марченко А.Л. Издательство Московского университета, 2005

Интернет источники

1. <https://www.bestprog.net/ru/2016/11/13/>
2. <https://metanit.com/sharp/tutorial/6.4.php>
3. <https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/controls/datagridview-control-windows-forms>

Приложение А

(обязательное)

Текст программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Praza\_1

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

public static List<Car> cars;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Praza\_1

{

[Serializable]

public class Car

{

public string brand;

public int year;

public string num\_1;

public string num\_2;

public string color;

public string number;

public string owner;

[NonSerialized]

public DateTime d\_ofcreate;

public Car() { }

public Car(string brand, int year, string num\_1, string num\_2, string color, string number, string owner)

{

this.brand = brand;

this.year = year;

this.num\_1 = num\_1;

this.num\_2 = num\_2;

this.color = color;

this.number = number;

this.owner = owner;

}

}

}

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 System.Xml.Serialization;

using System.IO;

using Excel = Microsoft.Office.Interop.Excel;

namespace Praza\_1

{

public partial class Form1 : Form

{

Add add\_f = new Add();

Edit edit\_f = new Edit();

public Form1()

{

InitializeComponent();

Program.cars = new List<Car>();

}

//фАЙЛ-Сохранение

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

{

XmlSerializer serializer = new XmlSerializer(typeof(List<Car>));

SaveFileDialog ofd = new SaveFileDialog();

ofd.Filter = "XML Files(\*.xml)|\*.xml";

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

{

FileStream fs = new FileStream(ofd.FileName, FileMode.OpenOrCreate);

serializer.Serialize(fs, Program.cars);

fs.Close();

}

}

//ФАЙЛ-Открыть

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

{

XmlSerializer serializer = new XmlSerializer(typeof(List<Car>));

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "XML Files(\*.xml)|\*.xml";

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

{

FileStream fs = new FileStream(ofd.FileName, FileMode.Open);

Program.cars = (List<Car>)serializer.Deserialize(fs);

fs.Close();

updateTable();

}

}

//Метод обновления DataGridView после закрытия окна Edit

public void updateTable()

{

dataGridView1.RowCount = Program.cars.Count + 1;

for (int i = 0; i < Program.cars.Count; i++)

{

dataGridView1["Brand", i].Value = Program.cars[i].brand;

dataGridView1["Year", i].Value = Program.cars[i].year;

dataGridView1["num1", i].Value = Program.cars[i].num\_1;

dataGridView1["num2", i].Value = Program.cars[i].num\_2;

dataGridView1["color", i].Value = Program.cars[i].color;

dataGridView1["Number", i].Value = Program.cars[i].number;

dataGridView1["owner", i].Value = Program.cars[i].owner;

dataGridView1[7, i].Value = i;

}

}

//Поиск 1 (по марке и цвету)

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

{

Search s = new Search();

s.brand.Text = "Введите марку автомобиля: ";

s.color.Text = "Введите цвет:";

s.count.Visible = false;

s.ShowDialog();

}

//Поиск 2 (по моменту выпуска)

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

{

float midage = 0;

foreach (Car c in Program.cars)

{

midage += c.year;

}

midage /= Program.cars.Count;

List<Car> ans = new List<Car>();

foreach (Car c in Program.cars)

{

if (midage >= c.year)

ans.Add(c);

}

Search s = new Search();

s.brand.Visible = false;

s.color.Visible = false;

s.brand\_text.Visible = false;

s.color\_text.Visible = false;

s.searches.Visible = false;

s.set(ans);

s.ShowDialog();

}

// ДАННЫЕ -- ДОБАВИТЬ

private void AddToolStripMenuItem1\_Click(object sender, EventArgs e)

{

add\_f.set(this);

//Очищение всех TextBox

add\_f.textBox1\_brand.Clear();

add\_f.textBox2\_year.Clear();

add\_f.textBox3\_num1.Clear();

add\_f.textBox4\_num2.Clear();

add\_f.textBox5\_color.Clear();

add\_f.textBox6\_Number.Clear();

add\_f.textBox7\_owner.Clear();

add\_f.ShowDialog();

}

// ДАННЫЕ -- РЕДАКТИРОВАТЬ

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

{

edit\_f.set(this, dataGridView1.CurrentCellAddress.Y, Convert.ToInt32(dataGridView1["id", dataGridView1.CurrentCellAddress.Y].Value));

edit\_f.ShowDialog();

}

// ДАННЫЕ -- УДАЛИТЬ

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

{

int row = dataGridView1.CurrentCellAddress.Y;

if (Program.cars.Count == 0)

return;

Program.cars.RemoveAt(Convert.ToInt32(dataGridView1["id", row].Value));

updateTable();

}

//ФАЙЛ -- СОХРАНИТЬ КАК

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

{

FileStream f1 = new FileStream("report.txt", FileMode.Create);

StreamWriter writer = new StreamWriter(f1);

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

{

for (int j = 0; j < dataGridView1.ColumnCount - 1; j++)

{

writer.Write(dataGridView1[j, i].Value + " ");

}

writer.Write("\r\n" + "\r\n");

}

writer.Close();

}

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

{

SaveTable(dataGridView1);

}

void SaveTable(DataGridView What\_Save)

{

string path = System.IO.Directory.GetCurrentDirectory() + @"\" + "Save\_Channel.xlsx";

Excel.Application excelapp = new Excel.Application();

Excel.Workbook workbook = excelapp.Workbooks.Add();

Excel.Worksheet worksheet = workbook.ActiveSheet;

for (int i = 1; i < What\_Save.RowCount + 1; i++)

{

for (int j = 1; j < What\_Save.ColumnCount + 1; j++)

{

worksheet.Rows[i].Columns[j] = What\_Save.Rows[i - 1].Cells[j - 1].Value;

}

}

excelapp.AlertBeforeOverwriting = false;

workbook.SaveAs(path);

excelapp.Quit();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Praza\_1

{

public partial class Add : Form

{

Form1 form;

public Add()

{

InitializeComponent();

}

public void set (Form1 f)

{

form = f;

}

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

{

form.dataGridView1.Rows.Add();

form.dataGridView1["Brand", form.dataGridView1.Rows.Count - 2].Value = textBox1\_brand.Text;

form.dataGridView1["Year", form.dataGridView1.Rows.Count - 2].Value = textBox2\_year.Text;

form.dataGridView1["num1", form.dataGridView1.Rows.Count - 2].Value = textBox3\_num1.Text;

form.dataGridView1["num2", form.dataGridView1.Rows.Count - 2].Value = textBox4\_num2.Text;

form.dataGridView1["color", form.dataGridView1.Rows.Count - 2].Value = textBox5\_color.Text;

form.dataGridView1["Number", form.dataGridView1.Rows.Count - 2].Value = textBox6\_Number.Text;

form.dataGridView1["owner", form.dataGridView1.Rows.Count - 2].Value = textBox7\_owner.Text;

form.dataGridView1["id", form.dataGridView1.Rows.Count - 2].Value = Program.cars.Count;

Program.cars.Add(new Car (textBox1\_brand.Text, Convert.ToInt32(textBox2\_year.Text), textBox3\_num1.Text, textBox4\_num2.Text, textBox5\_color.Text, textBox6\_Number.Text, textBox7\_owner.Text));

Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Praza\_1

{

public partial class Edit : Form

{

Form1 form;

int row;

int car\_id;

public Edit()

{

InitializeComponent();

}

public void set(Form1 f, int row, int car\_id)

{

form = f;

this.row = row;

this.car\_id = car\_id;

Car car = Program.cars[car\_id];

textBox1\_brand.Text = car.brand;

textBox2\_year.Text = car.year.ToString();

textBox3\_num1.Text = car.num\_1;

textBox4\_num2.Text = car.num\_2;

textBox5\_color.Text = car.color;

textBox6\_Number.Text = car.number;

textBox7\_owner.Text = car.owner;

}

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

{

Car car = new Car();

car.brand = textBox1\_brand.Text;

car.year = Convert.ToInt32(textBox2\_year.Text);

car.num\_1 = textBox3\_num1.Text;

car.num\_2 = textBox4\_num2.Text;

car.color = textBox5\_color.Text;

car.number = textBox6\_Number.Text;

car.owner = textBox7\_owner.Text;

Program.cars[car\_id] = car;

form.dataGridView1["Brand", row].Value = textBox1\_brand.Text;

form.dataGridView1["Year", row].Value = textBox2\_year.Text;

form.dataGridView1["num1", row].Value = textBox3\_num1.Text;

form.dataGridView1["num2", row].Value = textBox4\_num2.Text;

form.dataGridView1["color", row].Value = textBox5\_color.Text;

form.dataGridView1["Number", row].Value = textBox6\_Number.Text;

form.dataGridView1["owner", row].Value = textBox7\_owner.Text;

form.dataGridView1["id", row].Value = car\_id;

Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Praza\_1

{

public partial class Search : Form

{

public Search()

{

InitializeComponent();

}

public void set(List<Car> cars)

{

count.Text += "Количество автомобилей, с момента выпуска которых прошло более, чем средний возраст всех автомобилей в списке = " + cars.Count;

if (cars.Count() > 0)

{

dataGridView1.RowCount = cars.Count();

for (int i = 0; i < cars.Count; i++)

{

dataGridView1[0, i].Value = cars[i].brand;

dataGridView1[1, i].Value = cars[i].year;

dataGridView1[2, i].Value = cars[i].num\_1;

dataGridView1[3, i].Value = cars[i].num\_2;

dataGridView1[4, i].Value = cars[i].color;

dataGridView1[5, i].Value = cars[i].number;

dataGridView1[6, i].Value = cars[i].owner;

}

}

else

MessageBox.Show("Ничего не найдено! Проверьте введенные данные!");

}

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

{

List<Car> ans = new List<Car>();

foreach (Car c in Program.cars)

{

if (c.brand.ToLower().Equals(brand\_text.Text.ToLower()) && c.color.ToLower().Equals(color\_text.Text.ToLower()))

ans.Add(c);

}

set(ans);

}

}

Приложение Б

(справочное)

Экранные формы

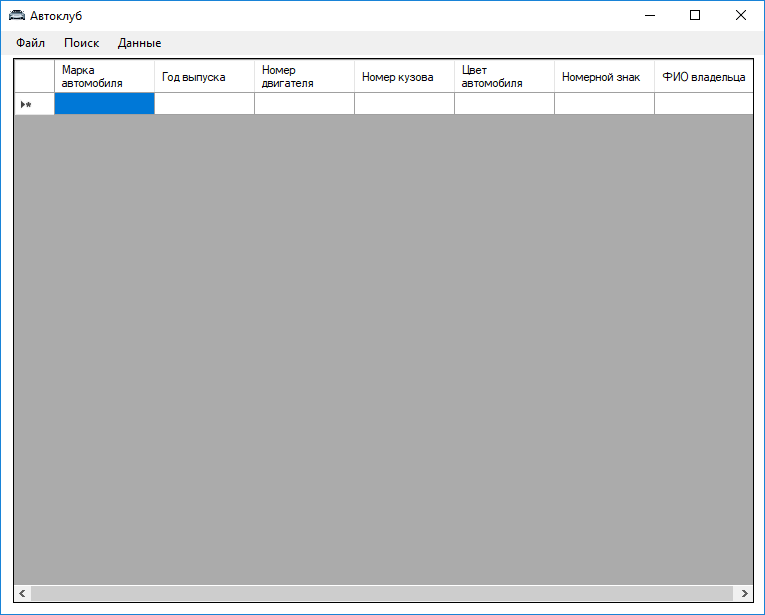


Рисунок Б.1 – Форма Автоклуб или Form1

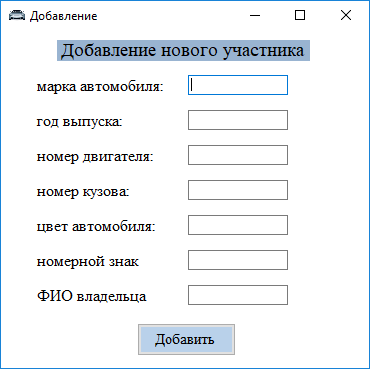


Рисунок Б.2 – Форма Добавление или Add

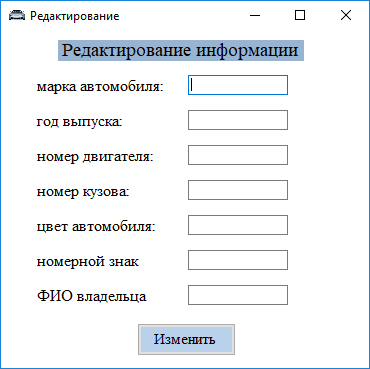


Рисунок Б.3 – Форма Редактирование или Edit



Рисунок Б.4 – Форма Поиск или Search

Приложение В

Алгоритм работы

Начало

Поиск по марке и цвету?

Нет

Да

foreach (Car c in Program.cars)

midage += c.year;

foreach (Car c in Program.cars)

Конец цикла

Совпадают ли значения полей brand, color и критериев?

Да

Находится средний возраст midage /= Program.cars.Count;

Нет

Конец

Конец цикла

Заполнение таблицы значениями

for (int i = 0; i < cars.Count; i++)

Да

Количество объектов списка ans >0

Вывод сообщения об ошибке

Нет

Конец цикла

Нет

Да

Добавление объекта в список ans.Add(c);

midage >= c.year

foreach (Car c in Program.cars)

Конец цикла

Добавление объекта в список ans.Add(c);