Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Программное средство

«Электронный каталог автомобилей»

БГУИР КП I–40 01 01 360 ПЗ

Выполнил

студент гр. 051004 Яковлев В.Ю.

Проверил: Фадеева Е.П.

Минск 2021

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

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

(подпись)

Лапицкая Н.В. 2021г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Яковлеву Вадиму Юрьевичу*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Программное средство «Электронный каталог автомобилей»\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

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

2. Срок сдачи законченной работы *11.06.2021г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Среда программирования Delphi. Чтение списка автомобилей и их характеристик из типизированного файла. Возможность просмотра раздела «о программе». Реализация функций добавления, удаления, редактирования автомобилей в динамической структуре данных односвязный список с дисциплиной обслуживания «очередь». Реализация функций сортировки автомобилей по цене, мощности и году выпуска. Фильтрация автомобилей по цене, марке, году выпуска, приводу, мощности и другим характеристикам автомобилей. Возможность добавления автомобилей в избранное. Возможность добавления автомобилей в корзину. Возможность сравнения автомобилей в корзине. Список автомобилей в избранном и корзине записываются в типизированный файл \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

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

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 Руководство по установке и использованию программного средства\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Фадеева Е.П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *16.02.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 28.02.2021г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.03.2021г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.04.2021г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 10.05.2021г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5.Заключение. Приложения к 20.05.2021г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 01.06.2021г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 02.06.2021г. по 11.06.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Фадеева Е.П.*

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Яковлев В.Ю. 16.02.2021г.*

*(дата и подпись студента)*

Содержание

[Введение 6](#_Toc72906734)

[1 Анализ литературных источников 7](#_Toc72906735)

[2 Постановка задачи 8](#_Toc72906736)

[3 Разработка программного средства 9](#_Toc72906737)

[3.1 Описание алгоритмов 9](#_Toc72906738)

[3.2 Структура данных 12](#_Toc72906739)

[3.2.1 Структура типов основного алгоритма 12](#_Toc72906740)

[3.2.2 Структура данных алгоритма AddCarToList 13](#_Toc72906741)

[3.2.3 Структура данных алгоритма DeleteCarFromList 13](#_Toc72906742)

[3.2.4 Структура данных алгоритма SetCarOnPanel 13](#_Toc72906743)

[3.2.5 Структура данных алгоритма ShowCars 14](#_Toc72906744)

[3.2.6 Структура данных алгоритма WriteListToFile 14](#_Toc72906745)

[3.2.7 Структура данных алгоритма ToShiftArrLeft 15](#_Toc72906746)

[3.2.8 Структура данных алгоритма GetPhotosAmount 15](#_Toc72906747)

[3.2.9 Структура данных алгоритма btnPhotoInpClick 15](#_Toc72906748)

[3.3 Описание графических компонентов 17](#_Toc72906749)

[3.3.1 Взаимосвязь графического интерфейса 17](#_Toc72906750)

[3.3.2 Описание графических компонентов формы MainForm 17](#_Toc72906751)

[3.3.3 Описание графических компонентов формы FormHelp 18](#_Toc72906752)

[3.3.4 Описание графического компонента формы FormCarCreate 19](#_Toc72906753)

[3.3.5 Описание графических компонентов формы FormFavs 20](#_Toc72906754)

[3.3.6 Описание графических компонентов формы FormCart 21](#_Toc72906755)

[3.4 Схема алгоритмов решения задачи по ГОСТ 19.701-90 23](#_Toc72906756)

[3.4.1 Схема алгоритма AddToCarList 23](#_Toc72906757)

[3.4.2 Схема алгоритма SetCarOnPanel 24](#_Toc72906758)

[3.4.3 Схема алгоритма ShowCars 25](#_Toc72906759)

[3.4.4 Схема алгоритма WriteListToFile 26](#_Toc72906760)

[3.4.5 Схема алгоритма ToShiftArrLeft 27](#_Toc72906761)

[3.4.6 Схема алгоритма GetPhotosAmount 28](#_Toc72906762)

[3.4.7 Схема алгоритма btnPhotoInpClick 29](#_Toc72906763)

[4 Тестирование и проверка работоспособности программного средства 30](#_Toc72906764)

[4.1 Корректность фильтрации и сортировки 30](#_Toc72906765)

[4.1.1 Тест 1 30](#_Toc72906766)

[4.1.2 Тест 2 30](#_Toc72906767)

[4.2 Добавление в избранное 32](#_Toc72906768)

[4.2.1 Тест 3 32](#_Toc72906769)

[4.2.2 Тест 5 33](#_Toc72906770)

[4.3 Редактирование информации об автомобиле 34](#_Toc72906771)

[4.3.1 Тест 6 34](#_Toc72906772)

[4.4 Сохранение автомобилей в файл 35](#_Toc72906773)

[4.4.1 Тест 7 35](#_Toc72906774)

[4.4.2 Тест 13 36](#_Toc72906775)

[5 Руководство по установке и использованию программного средства 37](#_Toc72906776)

[5.1 Установка 37](#_Toc72906777)

[5.2 Фильтрация и сортировка 38](#_Toc72906778)

[5.3 Просмотр корзины 40](#_Toc72906779)

[5.4 Просмотр справки 42](#_Toc72906780)

[5.5 Созданием объявлений 43](#_Toc72906781)

[Заключение 45](#_Toc72906782)

[Список использованной литературы 46](#_Toc72906783)

[Приложение А 47](#_Toc72906784)

Введение

С появлением людей земли начали активно заселяться. Однако, в то давнее время пустой оставалась очень обширная часть земной коры. Это было обусловлено отсутствием транспорта, а значит невозможностью легко преодолевать большие расстояния.

После понимания этого, люди изобрели первое колесо. Оно и дало начало для всех видов нынешнего перемещения по суше. Так первым видом колёсного транспорта стала телега. Некто из первых в истории фермеров мог передвигаться по полям в двухколесных тележках, которые тянулись быками и верблюдами - это показывает новый анализ крошечных моделей этих повозок, которые относятся к 6-5 тысячелетиям до н.э.[7]

С долгим течением времени телеги превратились в автомобили, хотя телеги и сегодня не исчезли совсем и используются в сельской местности. Важный шаг, который привёл к такому превращению – изобретение двигателя внутреннего сгорани.

Двигатель внутреннего сгорания (ДВС) – тепловой двигатель, в котором химическая энергия топлива, сгорающего в его рабочей полости (камере сгорания), преобразуется в механическую работу. Первый работоспособный газовый ДВС сконструирован Э. Ленуаром в 1860 году.[6]

Сегодня же автомобили стали на любой вкус и цвет. Множество двигателей разных мощностей, различные цвета и размеры, разный уровень комфорта, материал изготовления, эксперементальные технологии. Но за всё это разнообразие нужно платить.

Целью данного проекта является создание такого продукта, с помощью которого можно подбирать подходящие автомобили не только по цене, но и по характеристикам за оптимальную цену.

В реализации проекта попутно решались следующие задачи:

* создание динамической структуры данных;
* работа с файлами (текстовыми, типизированными);
* запись /чтение данных в/из файл(а);
* разработка пользовательского интерфейса для реализации перечисленных функций.

В качестве языка программирования был выбран язык Delphi, а средой разработки – Embarcadero Delphi 10, которые изучаются в рамках предмета «Основы алгоритмизации и программирования».

# Анализ литературных источников

Электронный каталог автомобилей – это информационная система, содержащая данные, описывающие автомобили, их характеристики и цену. Информация в электронном каталоге представлена в четкой иерархической структуре. Система электронного каталога с помощью специальной программы позволяет управлять всеми данными и обеспечивает удобство работы с ними (поиск, представление, выбор и т.д.).[4]

Люди на протяжении всей истории были заинтересованы в облегчении своей жизни. Живым доказательством этого является сегодняшний мир. Вокруг нас везде новые технологии. Одним из таких облегчений было изобретение автомобиля. Его создание стало возможным из-за человеческой необходимости в передвижении на большие расстояния. Поэтому технологии не обошли стороной и автомобили. Они позволили улучшить многие характеристики «железного коня». Теперь автолюбители соревнуются друг с другом мощностями, оборотами, объёмом двигателей.[5]

Однако покупка автомобиля весьма сложное решение, ведь стоят они немалых денег. Однако, стоит отметить их разнообразие. Даже опытный, разбирающийся в этом деле человек может запутаться. Для этого и был создан электронный каталог автомобилей. Он позволяет людям сделать лучший выбор, ввиду того, что вы можете фильтровать автомобили как вам угодно и найти для себя «тот самый». Так вы можете подобрать нужные вам характеристики, заплатив при этом оптимальную цену.

Сейчас же все крупные компании-производители автомобилей имеют собственные электронные каталоги своей продукции, а сайты в сети интернет имеют каталоги, которые содержат в себе автомобили от разных производителей.

# Постановка задачи

Реализовать программное средство «Электронный каталог автомобилей».

В программе доступны следующие возможности:

* просмотр раздела «о программе»;
* просмотр автомобилей, добавленных в программу;
* фильтрация автомобилей по характеристикам;
* сортировка автомобилей по возрастанию и убыванию цены;
* сортировка автомобилей по возрастанию и убыванию мощности;
* сортировка автомобилей по году выпуска;
* добавление автомобилей в избранное;
* добавление автомобилей в корзину (отображается итоговая цена);
* возможность сравнения автомобилей из корзины;
* добавление новых автомобилей в каталог;
* редактирование информации об автомобилях;
* удаление автомобилей из каталога.

Использовать типизированный файл и динамическую структуру данных типа однонаправленного списка с дисциплиной обслуживания «очередь» для хранения данных об автомобилях, избранного и содержимого корзины.

Информация, представленная в программе о каждом автомобиле: марка, модель, год выпуска, цвет, пробег, мощность и объём двигателя, разгон до 100 км/ч, средний расход топлива, тип топлива, коробка передач, привод, масса, габариты, цена. Также присутствуют имя владельца, его номер мобильного телефона и электронная почта.

# Разработка программного средства

## Описание алгоритмов

Таблица - Описание основных алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предлагаемый тип реализации |
| 1 | Инициализация главной формы | Создание типизированного файла с автомобилями, избранными автомобилями и корзины. Возможность создать автомобиль, добавить в избранное и корзину, редактировать, удалить, сортировать и фильтровать автомобили. Вызов следующих подпрограмм: AddCarToList,  DeleteCarFromList,  SetCarOnPanel,  ShowCars,  WriteListToFile |  |  |
| 2 | AddCarToList  (CarLst,  CurrCar) | Добавляет элемент CurrCar в хвост однонаправленного списка CarLst | CarLst – получает адрес;  CurrCar – получает адрес с защитой | Процедура |
| 3 | DeleteCarFromList  (CarLst,  CurrCar) | Удаляет элемент CurrCar из | CarLst – получает адрес; | Процедура |
| Продолжение Таблицы 1 | | | | |
|  |  | однонаправленного списка CarLst | CurrCar – получает адрес с защитой |  |
| 4 | SetCarOnPanel (CarPanel,  CurrPanel,  Car,  PanelOwner,  Skip) | Создаёт панель CarPanel на компоненте PanelOwner с отступом Skip сверху. Устанавливает значения автомобиля Car на панели и увеличивает число панелей CurrPanel | CarPanel, CurrPanel – получают адреса;  Car, Skip, PanelOwner – получают значения | Процедура |
| 5 | ShowCars  (CarLst,  CurrPanel, CarPanels,  PanelOwner,  Skip) | Выводит все автомобили из списка CarLst на панели CarPanels на компоненте PanelOwner с отступом Skip. Количество панелей пишется в CurrPanel. Вызывает подпрограмму SetCarOnPanel | CarPanels, CurrPanel – получают адреса;  Skip, PanelOwner – получают значения;  CarLst – получает адрес с защитой | Процедура |
| 6 | WriteListToFile (Lst,  TypeFile) | Записывает односвязный список Lst в TypeFile | Lst, TypeFile – получают адрес | Процедура |
| 7 | ToShiftArrLeft  (ImageArr,  PicturesAm, indDeletedEl) | Смещает массив путей на картинки ImageArr с количеством картинок PicturesAm, до удалённой картинки с индексом indDeletedEl | ImageArr,  PicturesAm, indDeletedEl – получают адреса | Процедура |
| Продолжение Таблицы 1 | | | | |
| 8 | GetPhotosAmount (ImageArr,  Amount) | Возвращает реальное количество путей картинок в массиве ImageArr в Amount | ImageArr –  получает адрес;  Amount –  получает адрес  возвращаемое значение | Процедура |
| 9 | btnPhotoInpClick  (Sender) | Загружает картинку, по пути пользователя для просмотра | Sender – получает | Процедура |

## 

## Структура данных

### Структура типов основного алгоритма

При разработке программного средства была использована динамическая структура данных линейный однонаправленный список.

Таблица - Структура типов программы

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | | Рекомендуемый тип | Назначение |
| TCar | | packed Record  Brand: string[20];  Model: string[20];  Year: Integer;  Color: string[20];  Mileage: Integer;  Power: Integer;  EngineVol: real;  Acceler100:real;  AverFuelRate: real;  Fuel: string[10];  Transm: string[10];  DriveUnit: string[20];  Weight: Integer;  Dimensions:  TCarSizeParms;  Price: Integer;  HolderName:  string[40];  Contacts: string[17];  Email: string[50];  Images: TImageArr;  End | Тип, предназначенный для построения линейного однонаправленного списка, содержащего каталог автомобилей, избранного и корзины.  Brand – марка машины;  Model – модель машины;  Year – год выпуска;  Color – цвет машины;  Mileage – пробег;  Power – мощность;  EngineVol – объём двигателя;  Acceler100 – разгон 0 – 100км/ч;  AverFuelRate – средний расход топлива;  Fuel – тип топлива;  Transm – трансмиссия;  DriveUnit – ;  Weight – масса машины;  Dimensions – габариты машины;  Price – цена машины;  HolderName – имя владельца;  Contacts – мобильный телефон владельца; |
| TCarSizeParms | | packed Record  Length: Integer;  Width: Integer;  Height: Integer; | Email – электронаая почта владельца;  Images – фотографии автомобиля;  PNextCar – следующий автомобиль;  PFirstCar – первый автомобиль; |
| TImageArr | | array[1..20] of string[255] | PLastCar – последний автомобиль |
| Продолжение Таблицы 2 | | | |
| TPCar | ^TCarListElem | |  |
| TCarListElem | record  CarInfo: TCar;  PNextCar: TPCar;  end | |  |
| TCarList | record  PFirstCar,  PLastCar: TPCar;  end | |  |
| TCarFile | file of TCar | | Файл для хранения автомобилей |

### Структура данных алгоритма AddCarToList

Таблица 3 - Структура данных алгоритма AddCarToList(CarLst, CurrCar)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| CarLst | TCarList | Список автомобилей | Формальный |
| CurrCar | TCar | Автомобиль для добавления | Формальный |
| PCurrCar | TPCar | Указатель на добавляемую машину | Локальный |

### Структура данных алгоритма DeleteCarFromList

Таблица - Структура данных алгоритма DeleteCarFromList(CarLst, CurrCar)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| CarLst | TCarList | Список автомобилей | Формальный |
| CurrCar | TCar | Автомобиль для удаления | Формальный |
| PCar | TPCar | Временный указатель на список, для удаления элемента | Локальный |

### Структура данных алгоритма SetCarOnPanel

Таблица - Структура данных алгоритма SetCarOnPanel(CarPanel, CurrPanel, Car, PanelOwner,Skip)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
|  | | | |

Продолжение таблицы 5

|  |  |  |  |
| --- | --- | --- | --- |
| CarPanel | TCarPanel | Панель с информацией об автомобиле | Формальный |
| CurrPanel | Integer | Текущая панель | Формальный |
| Car | TPCar | Указатель на автомобиль с информацией, которую необходимо разместить | Формальный |
| PanelOwner | TComponent | Владелец панели | Формальный |
| Skip | Integer | Отступ | Формальный |

### Структура данных алгоритма ShowCars

Таблица - Структура данных алгоритма ShowCars (CarLst, CurrPanel, CarPanels, PanelOwner, Skip)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| CarPanels | Array of TCarPanel | Панель с информацией об автомобиле | Формальный |
| CurrPanel | Integer | Текущая панель | Формальный |
| CarLst | TCarList | Указатель на автомобиль с информацией, которую неоюходимо разместить | Формальный |
| PanelOwner | TComponent | Владелец панели | Формальный |
| Skip | Integer | Отступ | Формальный |
| PCar | TPCar | Указатель на автомобиль с информацией, которую необходимо разместить | Локальный |

### Структура данных алгоритма WriteListToFile

Таблица - Структура данных алгоритма WriteListToFile (Lst, TypeFile)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Lst | TCarList | Список для записи | Формальный |

Продолжение таблицы 7

|  |  |  |  |
| --- | --- | --- | --- |
| TypeFile | TCarFile | Файл, в который записывается список | Формальный |
| PCar | TPCar | Указатель на автомобиль с информацией, которую необходимо записать | Локальный |

### Структура данных алгоритма ToShiftArrLeft

Таблица - Структура данных алгоритма ToShiftArrLeft(ImageArr,

PicturesAm, indDeletedEl)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| ImageArr | TImageArr | Массив картинок | Формальный |
| PicturesAm | Integer | Реальное количество картинок в массиве | Формальный |
| IndDeletedEl | Integer | Индекс удалённой картинки | Формальный |
| i | Integer | Параметр цикла | Локальный |

### Структура данных алгоритма GetPhotosAmount

Таблица - Структура данных алгоритма GetPhotosAmount (ImageArr,

Amount)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| ImageArr | TImageArr | Массив картинок | Формальный |
| Amount | Integer | Реальное количество картинок в массиве | Формальный |
| i | Integer | Параметр цикла | Локальный |

### Структура данных алгоритма btnPhotoInpClick

Таблица - Структура данных алгоритма btnPhotoInpClick(Sender)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Sender | TObject | Имеет значение  объекта –  источника  события, в  обработчике | Формальный |

Продолжение таблицы 10

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | которого он  используется |  |
| IsPhotoLoad | Boolean | Хранит значение была ли загружена картинка | Локальный |

## 

## Описание графических компонентов

### Взаимосвязь графического интерфейса

Для организации графического интерфейса программного средства для реализации вычислений типа «Длинная арифметика» было использовано 5 форм: FormMain, FormHelp, FormCarCreate, FormFavs, FormCart. Взаимосвязь между формами представлена на рисунке 3.1

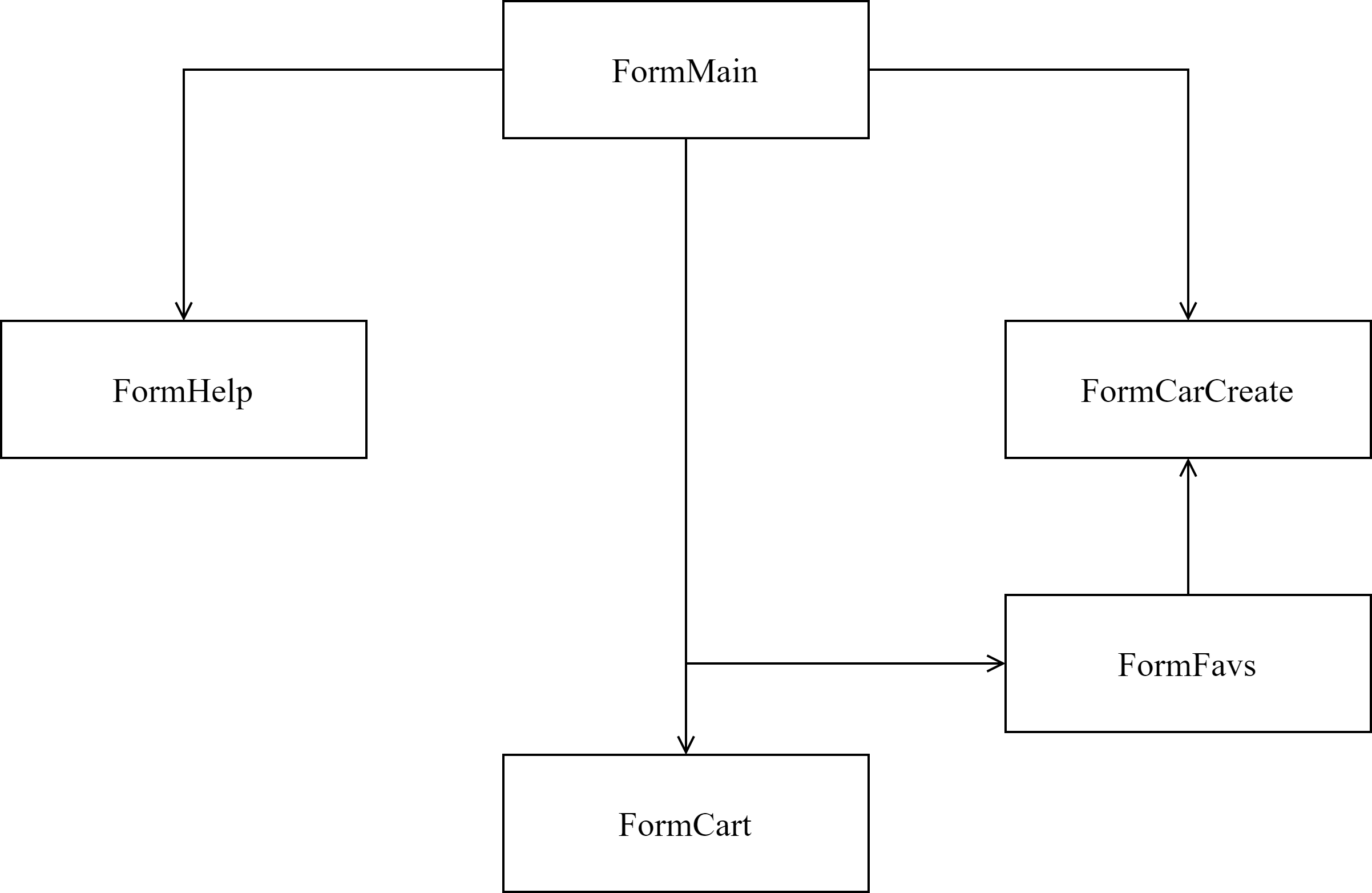


Рисунок 3.1 – Взаимосвязь графических форм

### Описание графических компонентов формы MainForm

Форма главного меню программы MainForm дает доступ к основным функциям программного средства, таким как открытие формы создания машины, открытие избранного, открытие корзины, добавление в избранное и корзину, открытие формы редактирования машины, фильтрация и сортировка автомобилей, обращение к истории, обращение к справке и имеет вид, представленный на рисунке 3.2

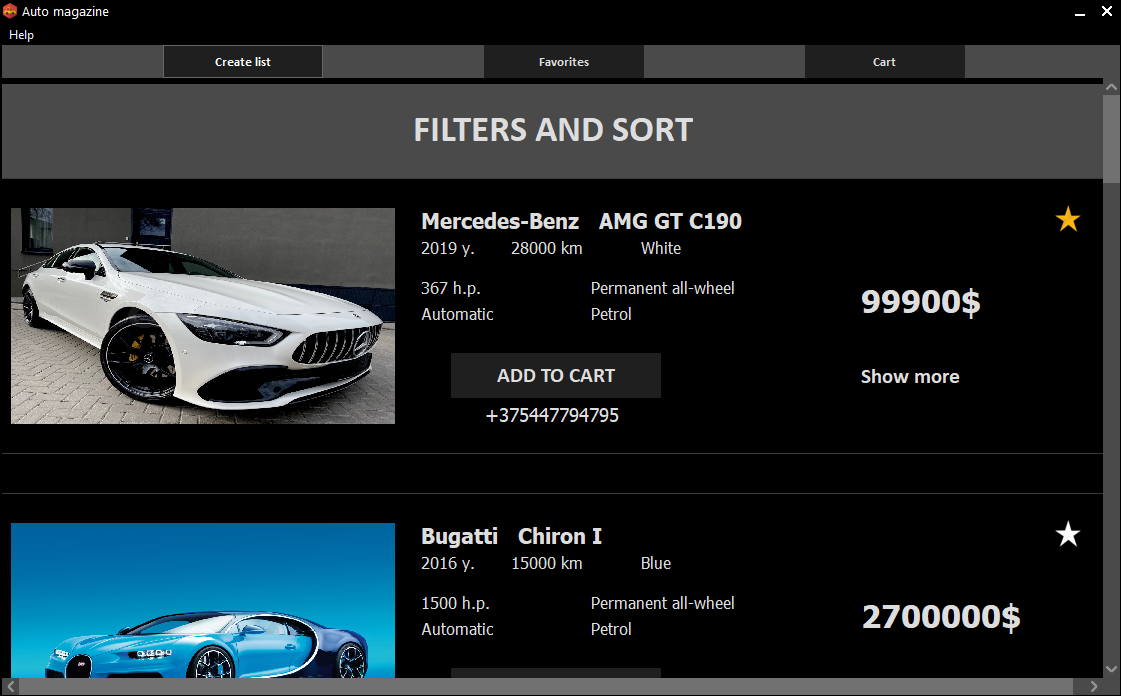


Рисунок 3.2 – Вид формы MainForm

Составляющие главной формы MainForm:

* панели с информацией об автомобилях;
* кнопки добавления в корзину;
* изображения «добавить в избранное», при нажатии добаляет в избранное;
* текст «Show more», при нажатии предоставляет доступ к форме FormCarCreate для просмотра полной информации об автомобиле, её редактированию или удалению;
* кнопка «Create list» предоставляет доступ к форме FormCarCreate для создания машины;
* кнопка «Favorites», предоставляет доступ к форме FormFavs для просмотра избранного;
* кнопка «Cart», предоставляет доступ к форме FormCart для просмотра корзины;
* вкладка меню «Help», предоставляет доступ к форме FormHelp для получения информации о программном средстве;
* панель фильтрации, наведение на которую курсором мыши позволяет использовать фильтрацию и сортировку.

### Описание графических компонентов формы FormHelp

Вспомогательная форма программы FormHelp предоставляет возможность пользователю просмотра информации о программном средстве и о правилах работы с ним и имеет вид, представленный на рисунке 3.3. Доступ к вспомогательной форме осуществляется по нажатию вкладки меню «Help».

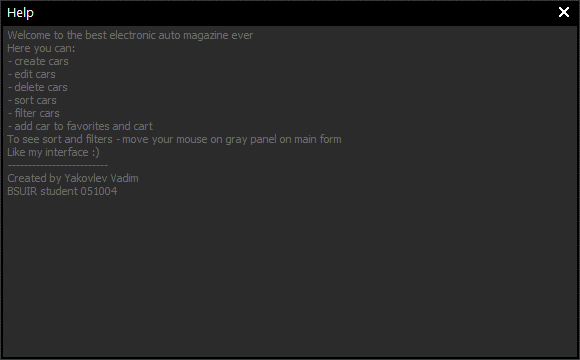


Рисунок 3.3 – Вид формы FormHelp

Составляющие вспомогательной формы HelpForm:

* компонент для отображения справки.

### Описание графического компонента формы FormCarCreate

Вспомогательная форма программы FormCarCreate предоставляет возможность изменять настройки вычислений и имеет вид, показанный на рисунке 3.4. Доступ к вспомогательной форме осуществляется по нажатию кнопки «Create list» или «Show more».



Рисунок 3.4 – Вид формы FormCarCreate

Составляющие вспомогательной формы FormCarCreate

* поля ввода информации об автомобиле;
* кнопка «Create ad» для создания объявления;
* кнопка «Сancel» для отмены создания объявления;
* кнопки загрузки и сохранения картинок;
* кнопки для пролистывания загруженных картинок;
* изображение «крест» по нажатию на который, картинка удаляется;
* при открытии через «Show more» появляются кнопки «изменить объявление» и «удалить объявление».

### Описание графических компонентов формы FormFavs

Вспомогательная форма программы Favorites предоставляет возможность просмотра избранных автомобилей (характерная черта – золотая звезда в углу панели) и имеет вид, показанный на рисунке 3.5. Доступ к вспомогательной форме осуществляется по нажатию кнопки «Favorites».

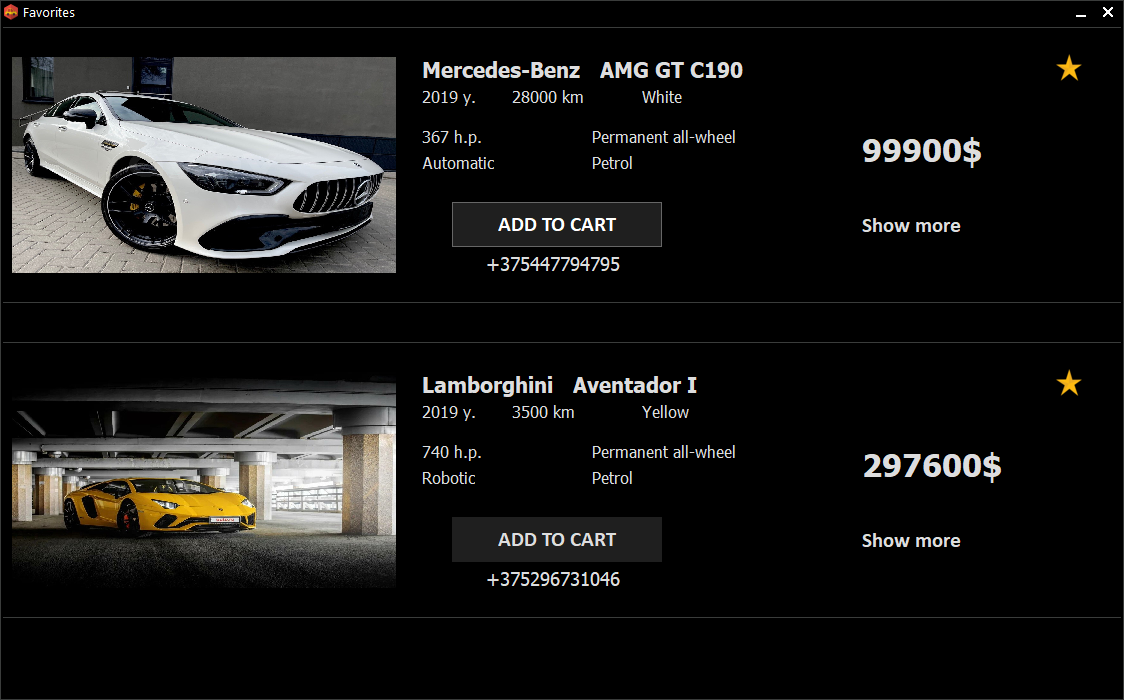


Рисунок 3.5 – Вид формы FormFavs

Составляющие вспомогательной формы FormFavs:

* панели с любимыми автомобилями;
* кнопка добавления в корзину;
* текст «ShowMore» при нажатии можно изменять данные об автомобиле или удалить его;
* изображения «звезда» при нажатии можно исключать/добавлять в избранное.

### Описание графических компонентов формы FormCart

Вспомогательная форма программы FormCart предоставляет доступ к корзине и имеет вид, показанный на рисунке 3.6. Доступ к вспомогательной форме осуществляется по нажатию кнопки «Cart».



Рисунок 3.6 – Вид формы FormCart

Составляющие вспомогательной формы FormСart:

* сравнение автомобилей;
* общая цена.

## Схема алгоритмов решения задачи по ГОСТ 19.701-90

### Схема алгоритма AddToCarList

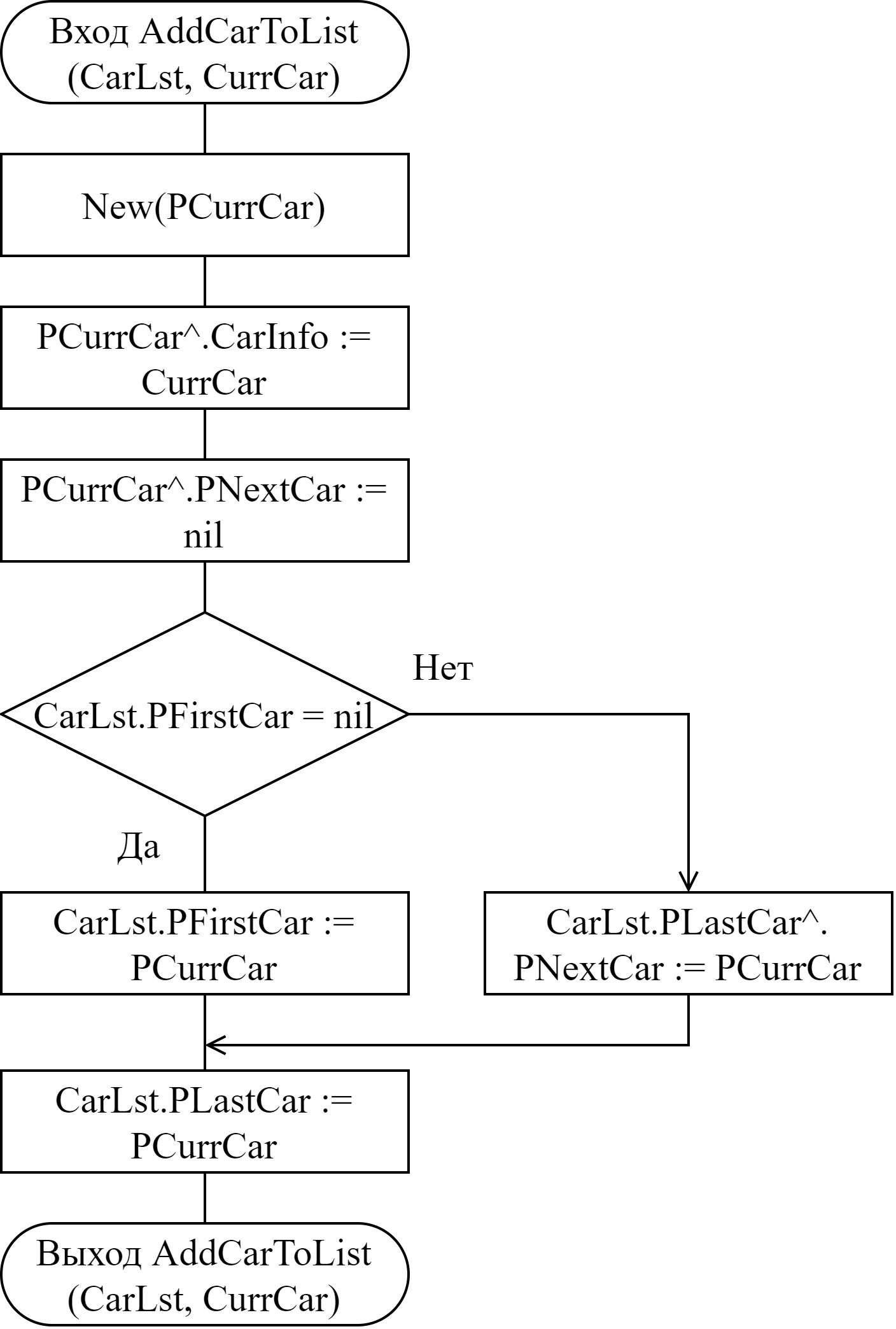


Рисунок 3.7 – Схема алгоритма AddCarToList

### Схема алгоритма SetCarOnPanel

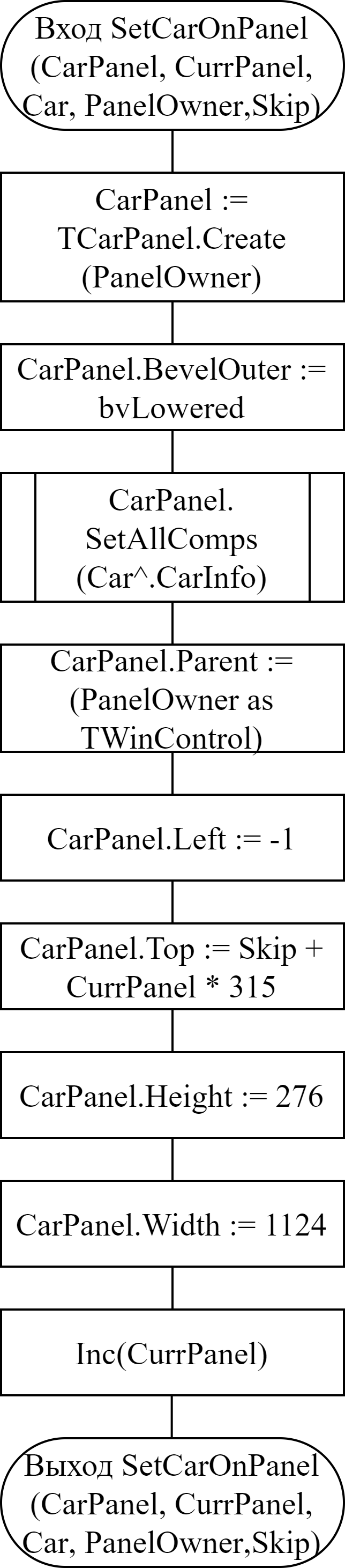


Рисунок 3.8 – Схема алгоритма SetCarOnPanel

### Схема алгоритма ShowCars

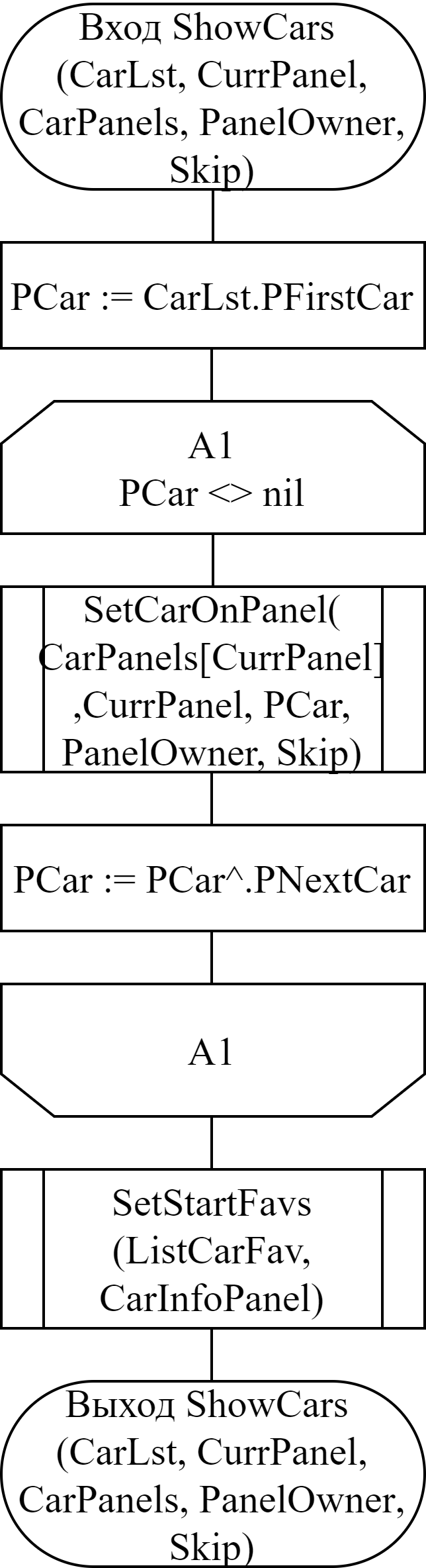


Рисунок 3.9 – Схема алгоритма ShowCars

### Схема алгоритма WriteListToFile

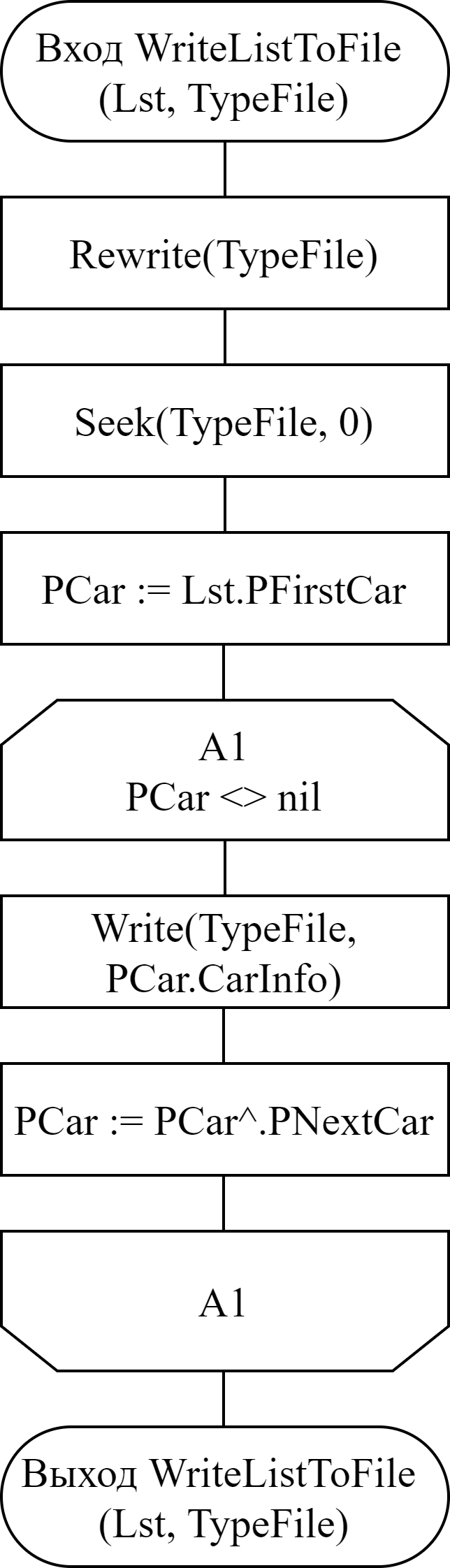


Рисунок 3.10 – Схема алгоритма WriteListToFile

### Схема алгоритма ToShiftArrLeft

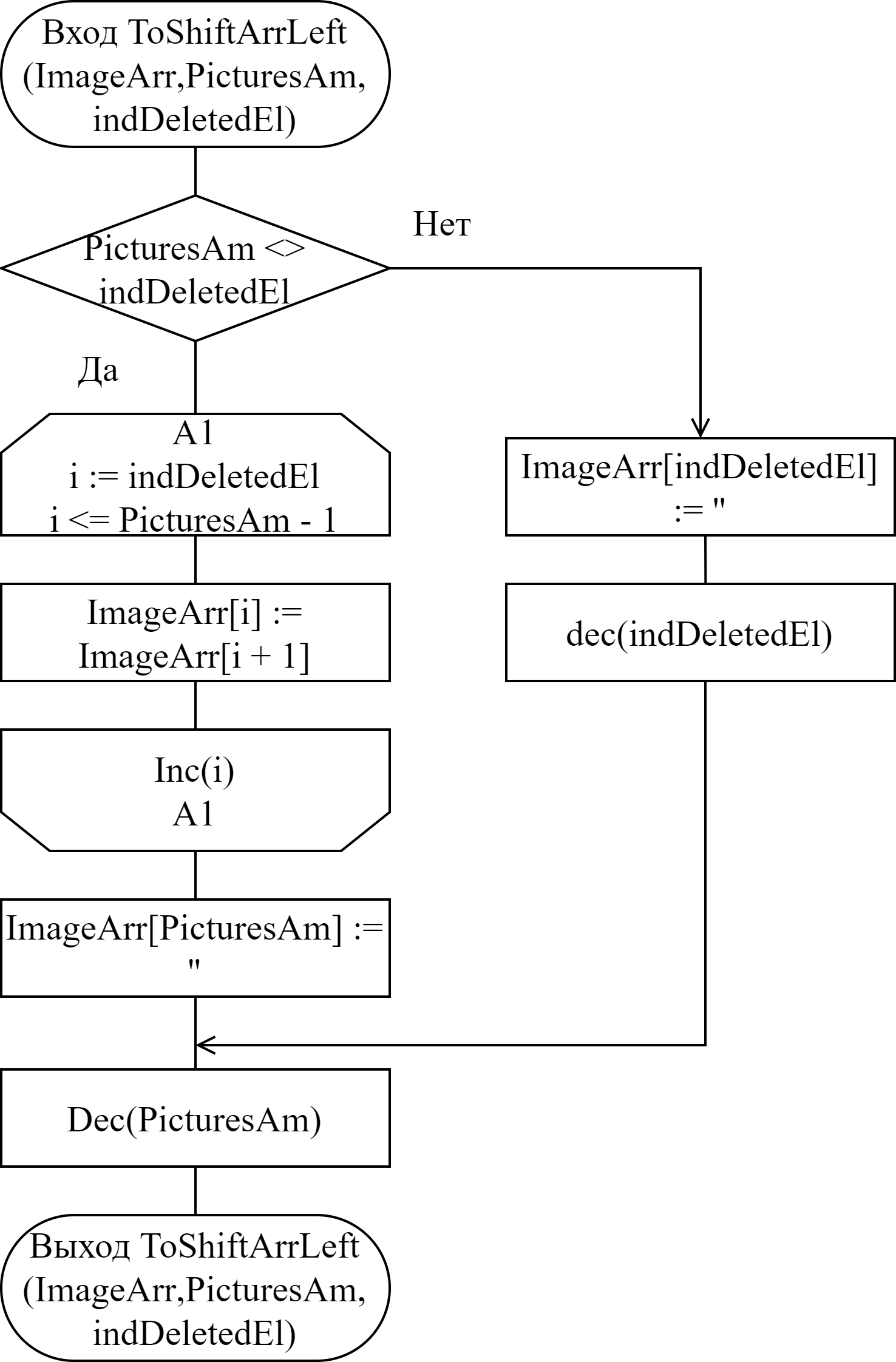


Рисунок 3.11 – Схема алгоритма ToShiftArrLeft

### Схема алгоритма GetPhotosAmount

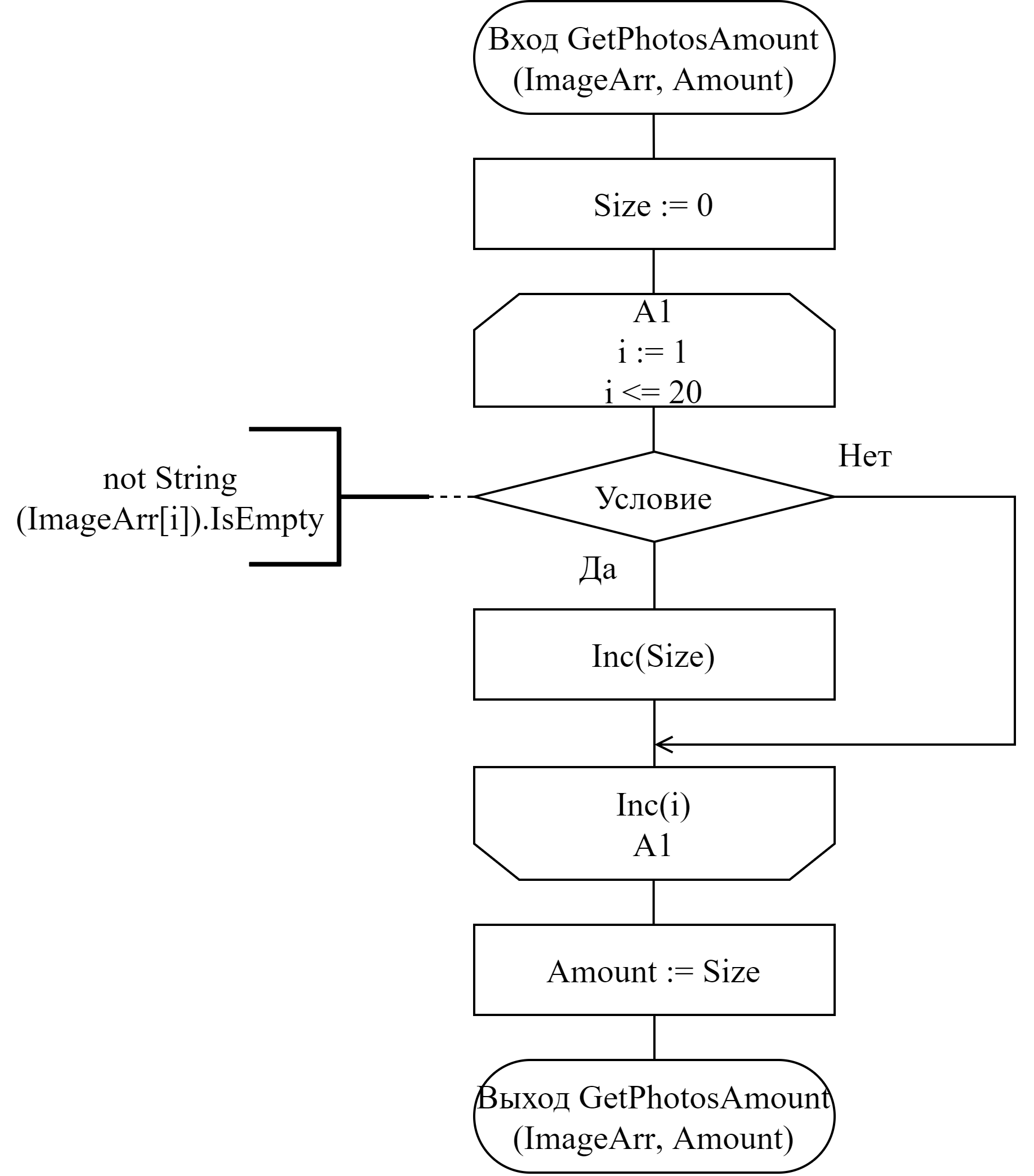


Рисунок 3.12 – Схема алгоритма GetPhotosAmount

### Схема алгоритма btnPhotoInpClick

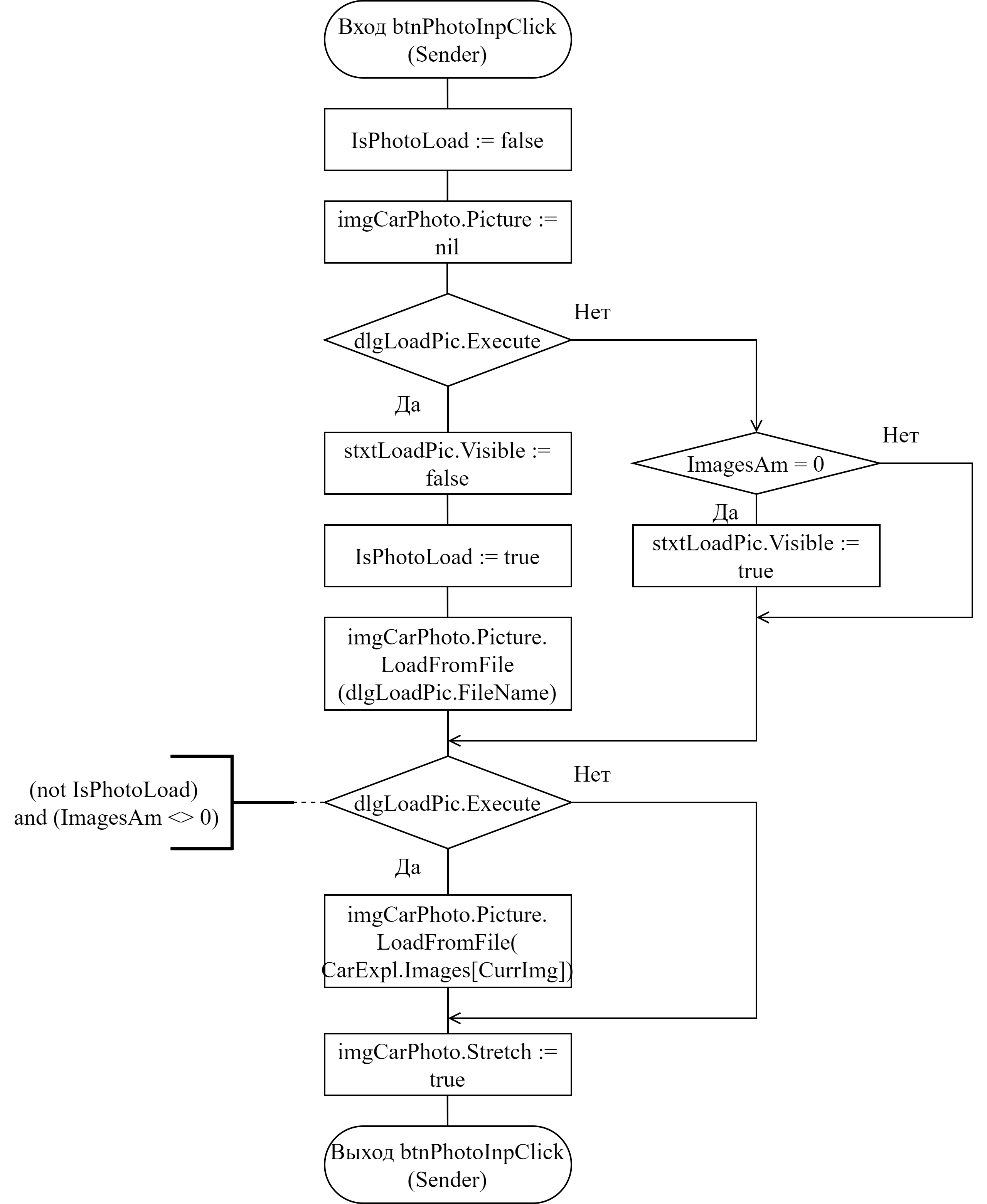


Рисунок 3.13 – Схема алгоритма btnPhotoInpClick

# Тестирование и проверка работоспособности программного средства

В ходе тестирования были рассмотрены в действие все основные возможности программного средства электронный каталог.

## Корректность фильтрации и сортировки

### Тест 1

Таблица – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка фильтрации и сортировки |
| Исходный набор данных: | Сортировка: цена по убыванию  Фильтрация: цена более 1000000 USD |
| Ожидаемый результат: | Bugatti |
| Полученный результат: | C:\Users\yakov\OneDrive\เอกสาร\ShareX\Screenshots\2021-05\ProjAuto_tVuZjLg2kk.png |

### Тест 2

Таблица – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка обнуления фильтров |
| Исходный набор данных: | Bugatti;  Сортировка: цена по убыванию  Фильтрация: цена более 1000000 USD |
| Продолжение Таблицы 12 | |
| Ожидаемый результат: | Выводится весь список |
| Полученный результат: |  |

## 

## Добавление в избранное

### Тест 3

Таблица – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация | Занесение автомобилей в избранное |
| Исходный набор данных: | C:\Users\yakov\OneDrive\เอกสาร\ShareX\Screenshots\2021-05\ProjAuto_OMIy7RmPl5.png |
| Ожидаемый результат: | Автомобили добавятся в избранное |
| Полученный результат: |  |

### Тест 5

Таблица – Тест 5

|  |  |
| --- | --- |
| Тестовая ситуация | Занесение автомобилей в корзину |
| Исходный набор данных: |  |
| Ожидаемый результат: | Автомобили добавятся в корзину |
| Полученный результат: |  |
|  | |
|  | |
|  | |

## Редактирование информации об автомобиле

### Тест 6

Таблица – Тест 6

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка редактирования информации о машине |
| Исходный набор данных: | C:\Users\yakov\OneDrive\เอกสาร\ShareX\Screenshots\2021-05\ProjAuto_GtL2XX550o.png |
| Ожидаемый результат: | Замена Permanent all-wheel на Front-wheeel |
| Полученный результат: | C:\Users\yakov\OneDrive\เอกสาร\ShareX\Screenshots\2021-05\ProjAuto_hFwHA7TVDl.png |

## Сохранение автомобилей в файл

### Тест 7

Таблица – Тест 7

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка сохранения основного списка после изменения |
| Исходный набор данных: |  |
| Ожидаемый результат: | Файл сохранился |
| Полученный результат: |  |

### Тест 13

Таблица – Тест 8

|  |  |
| --- | --- |
| Тестовая ситуация | Проверка сохранения избранного |
| Исходный набор данных: |  |
| Ожидаемый результат: | Файл сохранился |
| Полученный результат: |  |

### 

# Руководство по установке и использованию программного средства

## Установка

Для открытия программного средства необходимо распаковать архив и запустить файл ProjAuto.exe. Далее появится главное окно программы, представленное на рисунке 5.1.

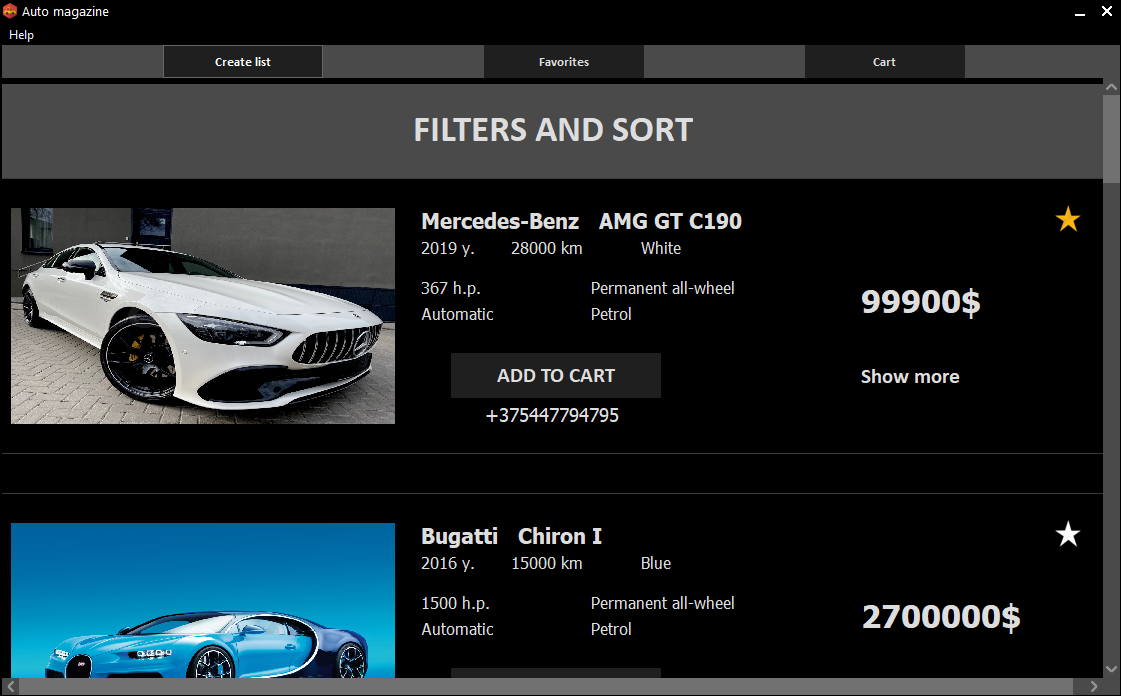


Рисунок 5.1 – Главное окно программы

## Фильтрация и сортировка

Для использования фильтрации и сортировки нужно поместить курсор мыши на серую панель в основной форме. После этого вы увидите все возможные сортировки и фильтрации и сможете их использовать, как на рисунке 5.2.

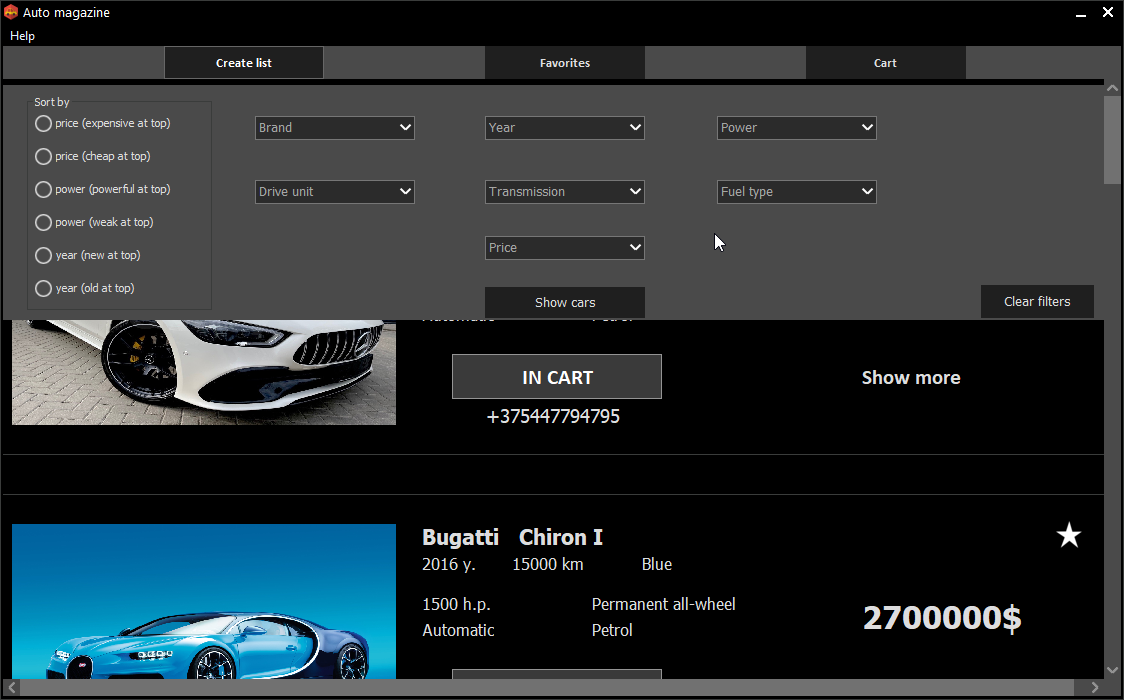


Рисунок 5.2 – Главное окно программы

После выбора сортировки и фильтрации нужно нажать кнопку «Show cars». Теперь сортировка и фильтрация применилась и вы видите результат, как на рисунке 5.3.

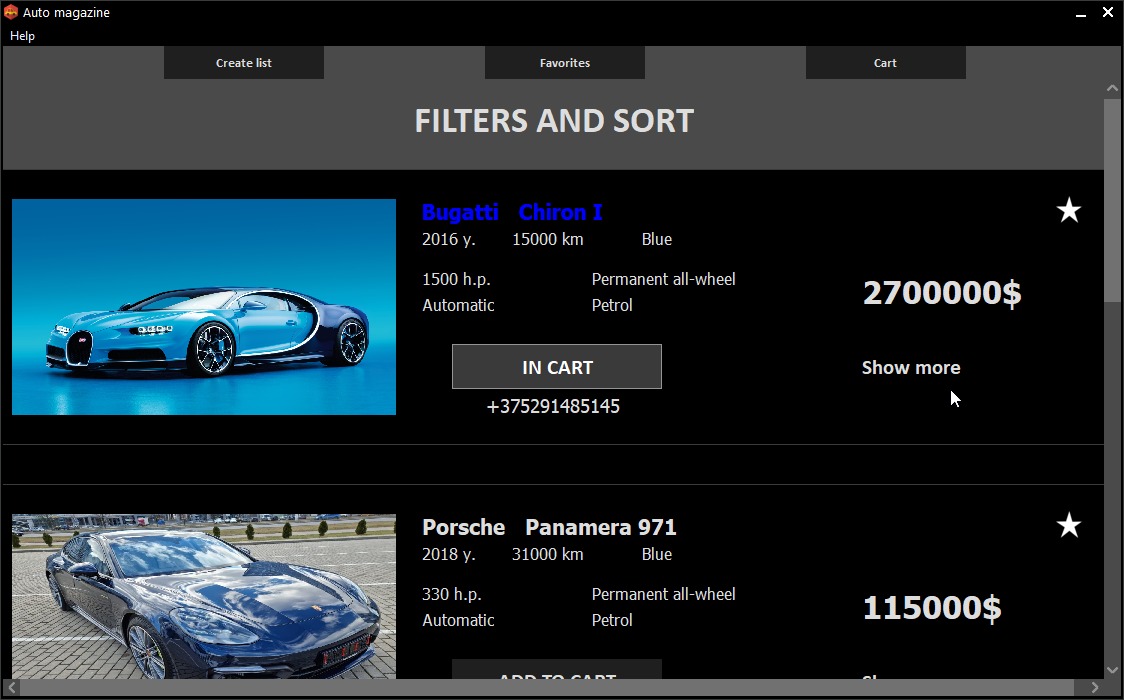


Рисунок 5.3 – Главное окно программы

Чтобы отменить сортировки и фильтрации нажмите «Clear filters».

## Просмотр корзины

Для просмотра корзины нужно в главном меню нажать кнопку «Cart». Если вы что-нибудь добавляли в корзину, то она откроется, как на рисунке 5.4, если же нет, то выведется сообщение о том, что корзина пуста, как на рисунке 5.5



Рисунок 5.4 – Вид формы FormCart

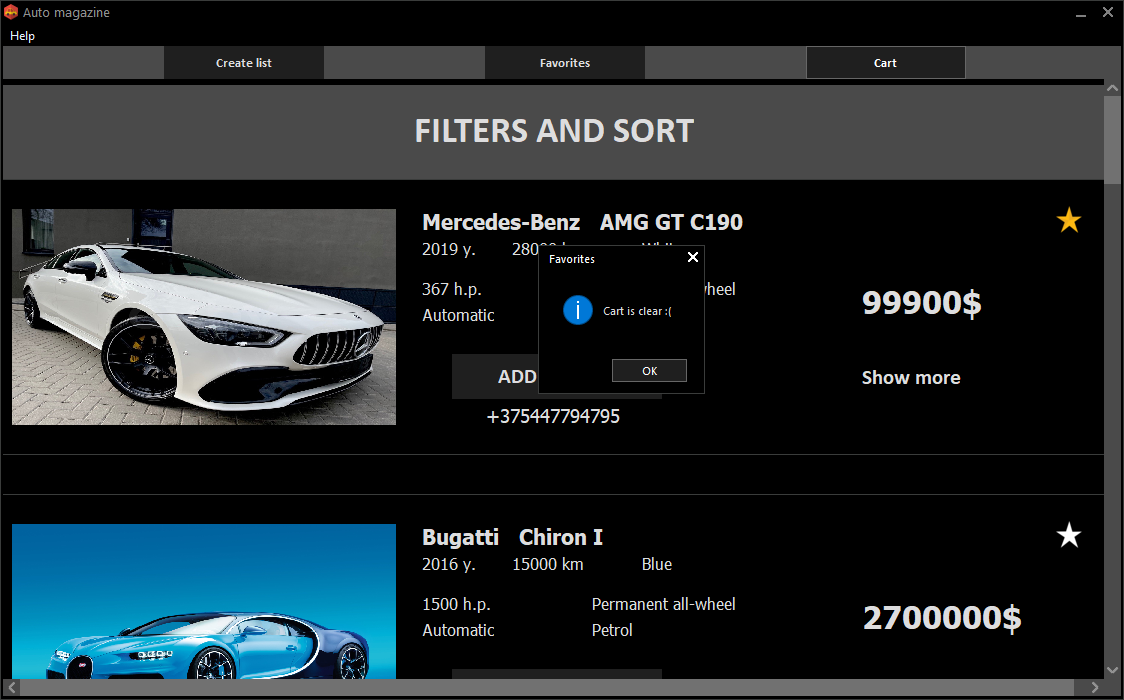


Рисунок 5.5 – Вид формы FormCart

## Просмотр справки

Для просмотра справки в пункте меню необходимо выбрать «Help» и затем откроется окно для просмотра справки, вид окна показан на рисунке 5.6.

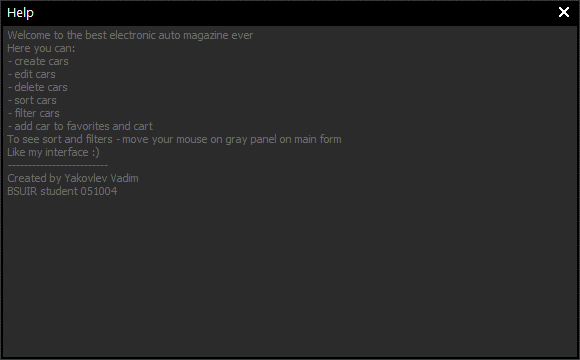


Рисунок 5.6 – Вид формы FormHelp

Справка предоставляет краткую информацию о программном средстве и информацию о правильном использовании программного средства.

## Созданием объявлений

Для создания объявления нужно нажать кнопку «Create list», после чего откроется следующее окно, где можно вводить все данные о машине, как на рисунке 5.7

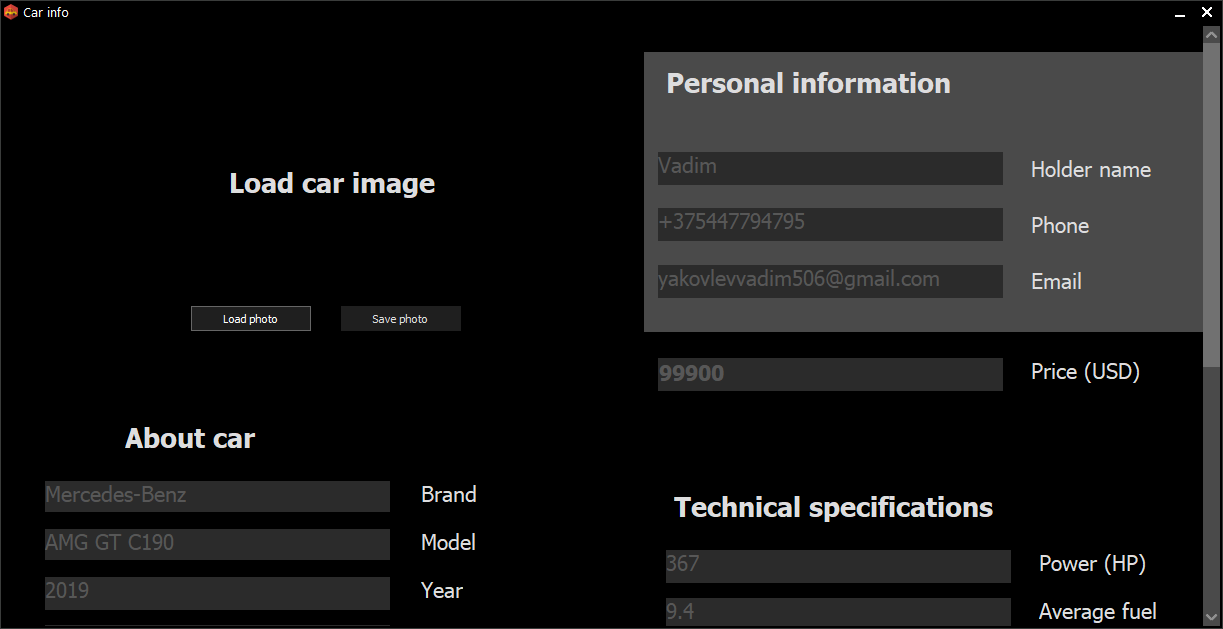


Рисунок 5.7 – Вид формы FormCarCreate

После ввода всех полей нужно нажать кнопку «Create ad». Если вы ввели всё корректно, то объявление будет создано, иначе – ошибка. Форму FormCarCreate можно открыть через текст «Show more» на главной форме или в избранном по нажатию, как на рисунке 5.8

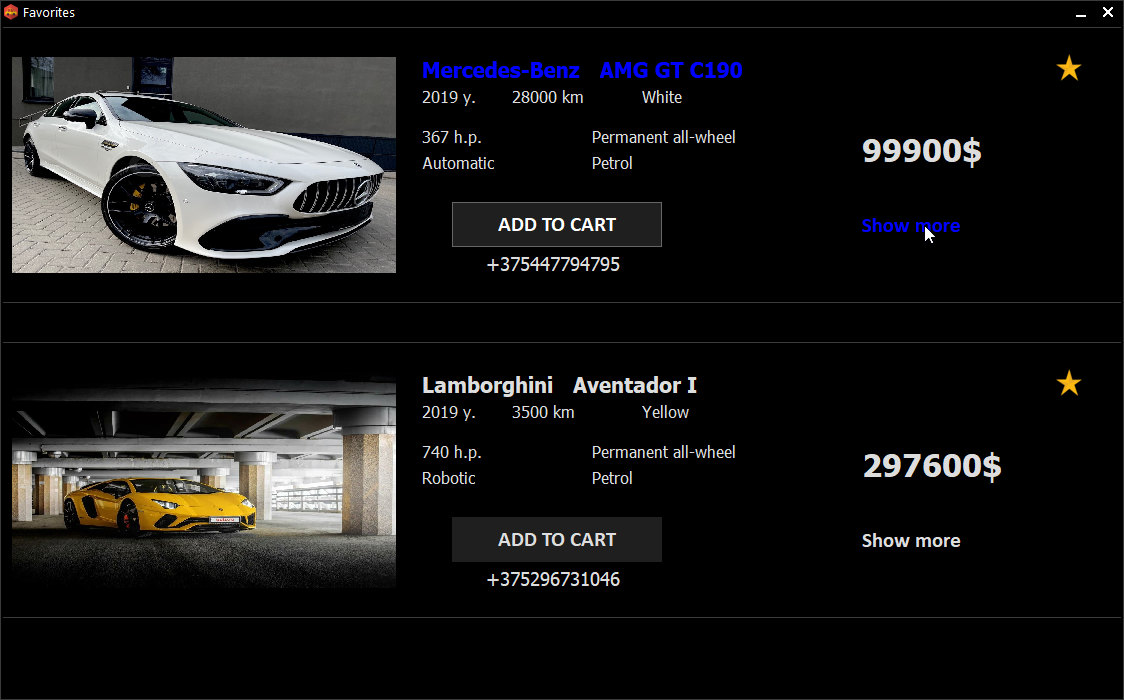


Рисунок 5.7 – Вид формы FormFavs

Заключение

По итогу работы над курсовым проектом было разработано программное средство электронный каталог автомобилей, обладающее графическим интерфейсом для взаимодействия с пользователем.

Приложение прошло все этапы тестирования, в результате которых были устранены все неполадки. Приложение имеет высокую скорость работы.

Данное программное средство, кроме простого просмотра предоставляет пользователю возможность создания, редактирования, удаления, фильтрации, сортировки машин, добавления в избранное и корзину, с возможностью сравнивать.

Бесценным результатом курсового проектирования является полученный опыт работы с динамической структурой данных и файлами. Был получено опыт работы с графическим интерфейсом.

При дальнейшей доработке программы, возможно появление различных аккаунтов, на которых сохранялись бы своё избранное и т.д.

Список использованной литературы

[1] Глухова, Л.А. Основы алгоритмизации и программирования ч.2: учебное пособие / Л.А.Глухова – Минск: БГУИР, 2006 – 146 с.

[2] Вирт Н. Алгоритмы и структуры данных: учебное пособие / Н.Вирт – М.: Мир, 1989 – 126 с.

[3] Ахо А. Структура данных и алгоритмы: учебное пособие / А. Ахо, Дж. Хопкрофт, Ульман Дж. – М.: Вильямс, 2003 – 45с.

[4] Виды электронных каталогов [Электронный ресурс]. – Режим доступа: https://www.amnt.ru/services/electronic/catalogs-types/. – Дата доступа: 21.03.2021.

[5] Технические характеристики автомобилей [Электронный ресурс]. – Режим доступа: http://www.drivernotes.net/tehnicheskie\_harakteristiki. – Дата доступа: 21.05.2021.

[6] Двигатель внутреннего сгорания [Электронный ресурс]. – Режим доступа: https://bigenc.ru/technology\_and\_technique/text/4341616. – Дата доступа: 21.03.2021.

[7] Первые автомобили [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/456810/. – Дата доступа: 21.03.2021.

Приложение А

(обязательное)

Исходный код программы

program ProjAuto;

uses

Vcl.Forms,

AutoMag in 'AutoMag.pas' {FormMain},

Help in 'Help.pas' {FormHelp},

Magazine in 'Win32\Debug\Magazine.pas',

CarCreate in 'CarCreate.pas' {FormCarCreate},

CarPanel in 'Win32\Debug\CarPanel.pas',

Favorites in 'Favorites.pas' {FormFavs},

Cart in 'Cart.pas' {FormCart};

{$R \*.res}

begin

Application.Initialize;

Application.MainFormOnTaskbar := True;

Application.CreateForm(TFormMain, FormMain);

Application.CreateForm(TFormHelp, FormHelp);

Application.Run;

end.

unit AutoMag;

{

==============[SOFTWARE IMPLEMENTATION TOOL]=======

===================[AUTO MAGAZINE]=================

Author: Yakovlev Vadim gr.051004

}

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils,

System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus,

Help, CarCreate, Vcl.StdCtrls,

Vcl.Buttons, Vcl.ExtCtrls, sSkinManager, Vcl.Mask,

sMaskEdit,

sCustomComboEdit, sComboBox, sComboBoxes,

sSkinProvider, Vcl.Grids, Data.DB,

Vcl.DBGrids, acDBGrid, CarPanel, sLabel,

System.ImageList, Vcl.ImgList,

acAlphaImageList, sButton, sBitBtn, sScrollBox,

sPanel, sGroupBox;

type

TFormMain = class(TForm)

MainMenu1: TMainMenu;

trlHelp: TMenuItem;

PanelControl: TPanel;

btnCarCreate: TButton;

btnCartEnter: TButton;

btnFavEnter: TButton;

SkinChange: TsSkinManager;

sSkinProvider1: TsSkinProvider;

lstImgCartnFav: TsAlphaImageList;

slblNoCars: TsLabel;

scrlboxCars: TsScrollBox;

spnlSortFilt: TsPanel;

srgrSort: TsRadioGroup;

slblSortFilt: TsLabel;

sbtnShow: TsButton;

scmbBrand: TsComboBox;

scmbYear: TsComboBox;

scmbPower: TsComboBox;

scmbDriveUnit: TsComboBox;

scmbTransm: TsComboBox;

scmbFuelType: TsComboBox;

sbtnClearFilt: TsButton;

scmbPrice: TsComboBox;

procedure trlHelpClick(Sender: TObject);

procedure btnCarCreateClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure btnFavEnterClick(Sender: TObject);

procedure FormClose(Sender: TObject;

var Action: TCloseAction);

procedure btnCartEnterClick(Sender: TObject);

procedure spnlSortFiltMouseMove(Sender: TObject;

Shift: TShiftState;

X,Y: Integer);

procedure sbtnShowClick(Sender: TObject);

procedure sbtnClearFiltClick(Sender: TObject);

private

{ Private declarations }

public

end;

TPCar = ^TCarListElem;

TCarListElem = record

CarInfo: TCar;

PNextCar: TPCar;

end;

TCarList = record

PFirstCar, PLastCar: TPCar;

end;

TCarArray = array of TCar;

Function IsCarEqual(Car1, Car2: TCar): Boolean;

procedure OneAddListOnShow(Car: TCar);

procedure SetCarOnPanel(var CarPanel: TCarPanel;

var CurrPanel: Integer;

Car: TPCar;

PanelOwner: TComponent;

Skip: Integer);

procedure ShowCars(const CarLst: TCarList;

var CurrPanel: Integer;

var CarPanels: array of TCarPanel;

PanelOwner: TComponent;

Skip: Integer);

procedure AddCarToList(var CarLst: TCarList;

const CurrCar: TCar);

procedure DeleteCarFromList(var CarLst: TCarList;

const CurrCar: TCar);

procedure ClearPanels(var Panels: array of TCarPanel;

var CurrPnl,

AmountPnl: Integer);

var

FormMain: TFormMain;

CarInfoPanel: array of TCarPanel;

CurrPanel: Integer = 0;

CurrPanelFilt: Integer = 0;

FileCar, FileFav, FileCart: TCarFile;

ListCar, ListCarFav, ListCarCart,

ListFilter: TCarList;

CarsAmount: Integer;

FiltCarsAmount: Integer;

FavsAmount: Integer = 0;

CartAmount: Integer = 0;

implementation

{$R \*.dfm}

uses Favorites, Cart;

procedure SetcmbBrands();

var

PCar: TPCar;

begin

FormMain.scmbBrand.Items.Clear;

PCar := ListCar.PFirstCar;

while PCar <> nil do

begin

if FormMain.scmbBrand.Items.IndexOf(PCar^.CarInfo.Brand) = -1 then

FormMain.scmbBrand.Items.Add(PCar^.CarInfo.Brand);

PCar := PCar^.PNextCar;

end;

end;

procedure ClearPanels(var Panels: array of TCarPanel;

var CurrPnl, AmountPnl: Integer);

var

i: Integer;

begin

for i := 0 to AmountPnl - 1 do

Panels[i].Destroy;

CurrPnl := 0;

end;

procedure SetStartFavs(FavsLst: TCarList;

var Panels: array of TCarPanel);

var

PCarFav: TPCar;

i: Integer;

begin

PCarFav := FavsLst.PFirstCar;

while PCarFav <> nil do

begin

for i := 0 to CarsAmount - 1 do

if IsCarEqual(PCarFav^.CarInfo,

Panels[i].LinkedCar) then

Panels[i].imgFav.ImageIndex := 2;

PCarFav := PCarFav^.PNextCar;

end;

end;

procedure SetStartCarts(CartLst: TCarList;

var Panels: array of TCarPanel);

var

PCarCart: TPCar;

i: Integer;

begin

PCarCart := CartLst.PFirstCar;

while PCarCart <> nil do

begin

for i := 0 to CarsAmount - 1 do

if IsCarEqual(PCarCart^.CarInfo, Panels[i].LinkedCar) then

begin

Panels[i].sbtnBuyCar.Down := true;

Panels[i].sbtnBuyCar.Caption := 'IN CART'

end;

PCarCart := PCarCart^.PNextCar;

end;

end;

procedure WriteListToFile(var Lst: TCarList; var TypeFile: TCarFile);

var

PCar: TPCar;

begin

Rewrite(TypeFile);

Seek(TypeFile, 0);

PCar := Lst.PFirstCar;

while PCar <> nil do

Begin

Write(TypeFile, PCar.CarInfo);

PCar := PCar^.PNextCar;

End;

end;

Function IsCarEqual(Car1, Car2: TCar): Boolean;

Begin

if (Car1.Brand = Car2.Brand)

and (Car1.Model = Car2.Model)

and (Car1.Year = Car2.Year)

and (Car1.Color = Car2.Color)

and (Car1.Mileage = Car2.Mileage)

and (Car1.Power = Car2.Power)

and (Car1.EngineVol = Car2.EngineVol)

and (Car1.AverFuelRate = Car2.AverFuelRate)

and (Car1.Acceler100 = Car2.Acceler100)

and (Car1.Dimensions.Length = Car2.Dimensions.Length)

and (Car1.Weight = Car2.Weight)

and (Car1.Price = Car2.Price)

and (Car1.Dimensions.Width = Car2.Dimensions.Width)

and (Car1.Dimensions.Height = Car2.Dimensions.Height)

and (Car1.HolderName = Car2.HolderName)

and (Car1.Contacts = Car2.Contacts)

and (Car1.Email = Car2.Email)

and (Car1.Fuel = Car2.Fuel)

and (Car1.Transm = Car2.Transm)

and (Car1.DriveUnit = Car2.DriveUnit) then

Result := true

else

Result := false;

End;

procedure InitList(var CarLst: TCarList);

begin

CarLst.PFirstCar := nil;

CarLst.PLastCar := nil;

end;

procedure AddCarToList(var CarLst: TCarList; const

CurrCar: TCar);

var

PCurrCar: TPCar;

begin

New(PCurrCar);

PCurrCar^.CarInfo := CurrCar;

PCurrCar^.PNextCar := nil;

if CarLst.PFirstCar = nil then

CarLst.PFirstCar := PCurrCar

else

CarLst.PLastCar^.PNextCar := PCurrCar;

CarLst.PLastCar := PCurrCar;

end;

procedure DeleteCarFromList(var CarLst: TCarList;

const CurrCar: TCar);

var

PCar: TPCar;

begin

PCar := CarLst.PFirstCar;

if IsCarEqual(PCar^.CarInfo, CurrCar) then

Begin

CarLst.PFirstCar := CarLst.PFirstCar.PNextCar;

Dispose(PCar);

End

Else

Begin

while not IsCarEqual(PCar^.PNextCar^.CarInfo,

CurrCar) do

Begin

PCar := PCar^.PNextCar;

End;

if PCar^.PNextCar = CarLst.PLastCar then

CarLst.PLastCar := PCar;

Dispose(PCar^.PNextCar);

PCar^.PNextCar := PCar^.PNextCar^.PNextCar;

End;

end;

procedure ReadFileToList(const FileOfCars: TCarFile;

var CarLst: TCarList);

var

Car: TCar;

Begin

while not EOF(FileOfCars) do

Begin

Read(FileOfCars, Car);

AddCarToList(CarLst, Car);

End;

End;

procedure SetCarOnPanel(var CarPanel: TCarPanel;

var CurrPanel: Integer;

Car: TPCar;

PanelOwner: TComponent;

Skip: Integer);

begin

CarPanel := TCarPanel.Create(PanelOwner);

CarPanel.BevelOuter := bvLowered;

CarPanel.SetAllComps(Car^.CarInfo);

with CarPanel do

Begin

Parent := (PanelOwner as TWinControl);

Left := -1;

Top := Skip + CurrPanel \* 315;

Height := 276;

Width := 1124;

End;

Inc(CurrPanel);

end;

procedure ShowCars(const CarLst: TCarList; var CurrPanel: Integer; var CarPanels: array of TCarPanel; PanelOwner: TComponent; Skip: Integer);

var

PCar: TPCar;

begin

PCar := CarLst.PFirstCar;

while PCar <> nil do

begin

SetCarOnPanel(CarPanels[CurrPanel], CurrPanel, PCar, PanelOwner, Skip);

PCar := PCar^.PNextCar;

end;

SetStartFavs(ListCarFav, CarInfoPanel);

SetStartCarts(ListCarCart, CarInfoPanel);

end;

procedure OneAddListOnShow(Car: TCar);

begin

FormMain.scrlboxCars.VertScrollBar.Range := FormMain.scrlboxCars.VertScrollBar.Range + 315;

Inc(CarsAmount);

SetLength(CarInfoPanel, CarsAmount);

AddCarToList(ListCar, Car);

SetCarOnPanel(CarInfoPanel[CurrPanel], CurrPanel, ListCar.PLastCar, FormMain.scrlboxCars, 100);

end;

procedure TFormMain.btnCarCreateClick(Sender: TObject);

begin

Application.CreateForm(TFormCarCreate, FormCarCreate);

FormCarCreate.Position := poScreenCenter;

FormCarCreate.ActiveControl := nil;

FormCarCreate.ShowModal;

end;

procedure TFormMain.btnCartEnterClick(Sender: TObject);

begin

if CartAmount <> 0 then

Begin

Application.CreateForm(TFormCart, FormCart);

FormCart.Position := poScreenCenter;

FormCart.DoubleBuffered := true;

FormCart.ShowModal;

End

Else

MessageBox(handle, PChar('Cart is clear :('), PChar('Favorites'), MB\_OK + MB\_ICONINFORMATION);

end;

procedure TFormMain.btnFavEnterClick(Sender: TObject);

begin

if FavsAmount <> 0 then

Begin

Application.CreateForm(TFormFavs, FormFavs);

FormFavs.Position := poScreenCenter;

FormFavs.DoubleBuffered := true;

FormFavs.ShowModal;

End

Else

MessageBox(handle, PChar('You have no favorites :('), PChar('Favorites'), MB\_OK + MB\_ICONINFORMATION);

end;

procedure TFormMain.FormClose(Sender: TObject; var Action: TCloseAction);

begin

WriteListToFile(ListCarFav, FileFav);

WriteListToFile(ListCar, FileCar);

WriteListToFile(ListCarCart, FileCart);

CloseFile(FileCar);

CloseFile(FileFav);

CloseFile(FileCart);

end;

procedure TFormMain.FormCreate(Sender: TObject);

begin

PanelControl.BringToFront;

InitList(ListCar);

InitList(ListCarFav);

InitList(ListCarCart);

DoubleBuffered := true;

AssignFile(FileCar, '../../CarPh/CarList');

if FileExists('../../CarPh/CarList') then

Reset(FileCar)

else

Rewrite(FileCar);

AssignFile(FileFav, '../../CarPh/FavList');

if FileExists('../../CarPh/FavList') then

Reset(FileFav)

else

Rewrite(FileFav);

AssignFile(FileCart, '../../CarPh/CartList');

if FileExists('../../CarPh/CartList') then

Reset(FileCart)

else

Rewrite(FileCart);

ReadFileToList(FileCart,ListCarCart);

ReadFileToList(FileCar, ListCar);

ReadFileToList(FileFav, ListCarFav);

CarsAmount := FileSize(FileCar);

FavsAmount := FileSize(FileFav);

CartAmount := FileSize(FileCart);

SetLength(CarInfoPanel, CarsAmount);

scrlboxCars.VertScrollBar.Range := 315 \* CarsAmount + 100;

if CarsAmount <> 0 then

ShowCars(ListCar, CurrPanel, CarInfoPanel, FormMain.scrlboxCars, 100)

else

begin

slblNoCars.BringToFront;

slblNoCars.Visible := true;

end;

end;

Function CheckFilters(): Boolean;

begin

Result := true;

with FormMain do

if (scmbBrand.ItemIndex = -1)

and (scmbYear.ItemIndex = -1)

and (scmbPower.ItemIndex = -1)

and (scmbDriveUnit.ItemIndex = -1)

and (scmbTransm.ItemIndex = -1)

and (scmbFuelType.ItemIndex = -1)

and (scmbPrice.ItemIndex = -1)

and (srgrSort.ItemIndex = -1) then

Result := false;

end;

procedure FilteringList(ListToFilter: TCarList);

procedure FiltModel(Brand: string);

var

PCar: TPCar;

begin

PCar := ListCar.PFirstCar;

if not Brand.IsEmpty then

begin

while PCar <> nil do

begin

if PCar^.CarInfo.Brand = Brand then

begin

AddCarToList(ListFilter, PCar^.CarInfo);

Inc(FiltCarsAmount);

end;

PCar := PCar^.PNextCar;

end;

end

else

while PCar <> nil do

begin

AddCarToList(ListFilter, PCar^.CarInfo);

Inc(FiltCarsAmount);

PCar := PCar^.PNextCar;

end;

end;

procedure FiltYear(LeftYear, RightYear: Integer);

var

PCar: TPCar;

begin

PCar := ListFilter.PFirstCar;

while PCar <> nil do

begin

if (PCar^.CarInfo.Year > RightYear) or (PCar^.CarInfo.Year < LeftYear) then

begin

DeleteCarFromList(ListFilter, PCar^.CarInfo);

Dec(FiltCarsAmount);

end;

PCar := PCar^.PNextCar;

end;

end;

procedure FiltPower(LeftPower, RightPower: Integer);

var

PCar: TPCar;

begin

PCar := ListFilter.PFirstCar;

while PCar <> nil do

begin

if (PCar^.CarInfo.Power > RightPower) or (PCar^.CarInfo.Power < LeftPower) then

begin

DeleteCarFromList(ListFilter, PCar^.CarInfo);

Dec(FiltCarsAmount);

end;

PCar := PCar^.PNextCar;

end;

end;

procedure FiltDriveUnit(DriveUnit: string);

var

PCar: TPCar;

begin

PCar := ListFilter.PFirstCar;

while PCar <> nil do

begin

if PCar^.CarInfo.DriveUnit <> DriveUnit then

begin

DeleteCarFromList(ListFilter, PCar^.CarInfo);

Dec(FiltCarsAmount);

end;

PCar := PCar^.PNextCar;

end;

end;

procedure FiltTransm(Transm: string);

var

PCar: TPCar;

begin

PCar := ListFilter.PFirstCar;

while PCar <> nil do

begin

if PCar^.CarInfo.Transm <> Transm then

begin

DeleteCarFromList(ListFilter, PCar^.CarInfo);

Dec(FiltCarsAmount);

end;

PCar := PCar^.PNextCar;

end;

end;

procedure FiltFuelType(Fuel: string);

var

PCar: TPCar;

begin

PCar := ListFilter.PFirstCar;

while PCar <> nil do

begin

if PCar^.CarInfo.Fuel <> Fuel then

begin

DeleteCarFromList(ListFilter, PCar^.CarInfo);

Dec(FiltCarsAmount);

end;

PCar := PCar^.PNextCar;

end;

end;

procedure FiltPrice(LeftPrice, RightPrice: Integer);

var

PCar: TPCar;

begin

PCar := ListFilter.PFirstCar;

while PCar <> nil do

begin

if (PCar^.CarInfo.Price > RightPrice) or (PCar^.CarInfo.Price < LeftPrice) then

begin

DeleteCarFromList(ListFilter, PCar^.CarInfo);

Dec(FiltCarsAmount);

end;

PCar := PCar^.PNextCar;

end;

end;

var

PCar: TPCar;

i: Integer;

begin

with FormMain do

begin

if scmbBrand.ItemIndex <> -1 then

begin

FiltModel(scmbBrand.Text);

end

else

FiltModel('');

if scmbYear.ItemIndex <> -1 then

begin

case scmbYear.ItemIndex of

0: FiltYear(2019, 2021);

1: FiltYear(2015, 2018);

2: FiltYear(2010, 2014);

3: FiltYear(2005, 2009);

4: FiltYear(2000, 2004);

5: FiltYear(1995, 1999);

6: FiltYear(1980, 1994);

7: FiltYear(1800, 1979);

end;

end;

if scmbPower.ItemIndex <> -1 then

begin

case scmbPower.ItemIndex of

0: FiltPower(1001, 10000);

1: FiltPower(750, 1000);

2: FiltPower(500, 749);

3: FiltPower(350, 499);

4: FiltPower(250, 349);

5: FiltPower(150, 249);

6: FiltPower(0, 150);

end;

end;

if scmbDriveUnit.ItemIndex <> -1 then

begin

FiltDriveUnit(scmbDriveUnit.Text);

end;

if scmbTransm.ItemIndex <> -1 then

begin

FiltTransm(scmbTransm.Text);

end;

if scmbFuelType.ItemIndex <> -1 then

begin

FiltFuelType(scmbFuelType.Text);

end;

if scmbPrice.ItemIndex <> -1 then

begin

case scmbPrice.ItemIndex of

0: FiltPrice(1000001, 99999999);

1: FiltPrice(750000, 1000000);

2: FiltPrice(500000, 749999);

3: FiltPrice(250000, 499999);

4: FiltPrice(150000, 249999);

5: FiltPrice(90000, 149999);

6: FiltPrice(60000, 89999);

7: FiltPrice(30000, 59999);

8: FiltPrice(10000, 29999);

9: FiltPrice(0, 9999);

end;

end;

end;

end;

procedure SortingList(var ListToSort: TCarList; SortInd: Integer);

var

PCar, PCar2: TPCar;

Arr: TCarArray;

TempList: TCarList;

i, j: Integer;

begin

PCar := ListToSort.PFirstCar;

SetLength(Arr, FiltCarsAmount + 1);

i := 1;

while PCar <> nil do

begin

case SortInd of

0: Arr[i] := PCar^.CarInfo;

1: Arr[i] := PCar^.CarInfo;

2: Arr[i] := PCar^.CarInfo;

3: Arr[i] := PCar^.CarInfo;

4: Arr[i] := PCar^.CarInfo;

5: Arr[i] := PCar^.CarInfo;

end;

Inc(i);

PCar := PCar^.PNextCar

end;

For i := 2 to Length(Arr) - 1 do

Begin

Arr[0] := Arr[i];

j := i - 1;

case SortInd of

0:

begin

While (Arr[j].Price < Arr[0].Price) and (j >= 0) do

Begin

Arr[j + 1] := Arr[j];

j := j - 1;

End;

end;

1:

begin

While (Arr[j].Price < Arr[0].Price) and (j >= 0) do

Begin

Arr[j + 1] := Arr[j];

j := j - 1;

End;

end;

2:

begin

While (Arr[j].Power < Arr[0].Power) and (j >= 0) do

Begin

Arr[j + 1] := Arr[j];

j := j - 1;

End;

end;

3:

begin

While (Arr[j].Power < Arr[0].Power) and (j >= 0) do

Begin

Arr[j + 1] := Arr[j];

j := j - 1;

End;

end;

4:

begin

While (Arr[j].Year < Arr[0].Year) and (j >= 0) do

Begin

Arr[j + 1] := Arr[j];

j := j - 1;

End;

end;

5:

begin

While (Arr[j].Year < Arr[0].Year) and (j >= 0) do

Begin

Arr[j + 1] := Arr[j];

j := j - 1;

End;

end;

end;

Arr[j + 1] := Arr[0];

End;

if SortInd mod 2 = 0 then

for i := 1 to Length(Arr) - 1 do

AddCarToList(TempList, Arr[i])

else

for i := Length(Arr) - 1 downto 1 do

AddCarToList(TempList, Arr[i]);

ListToSort.PFirstCar := TempList.PFirstCar;

end;

procedure TFormMain.sbtnShowClick(Sender: TObject);

begin

if CheckFilters() then

begin

if CurrPanel <> 0 then

ClearPanels(CarInfoPanel, CurrPanel, CarsAmount)

else

ClearPanels(CarInfoPanel, CurrPanelFilt, FiltCarsAmount);

FiltCarsAmount := 0;

InitList(ListFilter);

FilteringList(ListFilter);

if srgrSort.ItemIndex <> -1 then

SortingList(ListFilter, srgrSort.ItemIndex);

ShowCars(ListFilter, CurrPanelFilt, CarInfoPanel, FormMain.scrlboxCars, 100);

if FiltCarsAmount = 0 then

slblNoCars.Visible := true

else

slblNoCars.Visible := false;

scrlboxCars.VertScrollBar.Range := 100 + 315 \* FiltCarsAmount;

if FiltCarsAmount <> 0 then

begin

slblSortFilt.Visible := true;

scmbBrand.Visible := false;

scmbYear.Visible := false;

scmbPower.Visible := false;

scmbTransm.Visible := false;

scmbDriveUnit.Visible := false;

scmbFuelType.Visible := false;

scmbPrice.Visible := false;

srgrSort.Visible := false;

end

else

begin

slblSortFilt.Visible := false;

scmbBrand.Visible := true;

scmbYear.Visible := true;

scmbPower.Visible := true;

scmbTransm.Visible := true;

scmbDriveUnit.Visible := true;

scmbFuelType.Visible := true;

scmbPrice.Visible := true;

srgrSort.Visible := true;

end;

end;

end;

procedure TFormMain.sbtnClearFiltClick(Sender: TObject);

var

i: Integer;

begin

scmbBrand.ItemIndex := -1;

scmbYear.ItemIndex := -1;

scmbPower.ItemIndex := -1;

scmbDriveUnit.ItemIndex := -1;

scmbTransm.ItemIndex := -1;

scmbFuelType.ItemIndex := -1;

scmbPrice.ItemIndex := -1;

srgrSort.ItemIndex := -1;

if (FiltCarsAmount <> 0) or ((FiltCarsAmount = 0) and (CurrPanel = 0)) then

begin

ClearPanels(CarInfoPanel, CurrPanelFilt, FiltCarsAmount);

FiltCarsAmount := 0;

ShowCars(ListCar, CurrPanel, CarInfoPanel, FormMain.scrlboxCars, 100);

if FiltCarsAmount = 0 then

slblNoCars.Visible := true

else

slblNoCars.Visible := false;

scrlboxCars.VertScrollBar.Range := 100 + 315 \* CarsAmount;

slblSortFilt.Visible := true;

scmbBrand.Visible := false;

scmbYear.Visible := false;

scmbPower.Visible := false;

scmbTransm.Visible := false;

scmbDriveUnit.Visible := false;

scmbFuelType.Visible := false;

scmbPrice.Visible := false;

srgrSort.Visible := false;

end;

end;

procedure TFormMain.spnlSortFiltMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

if (X >= 0) and (Y >= 0) and (X < spnlSortFilt.Width) and (Y < spnlSortFilt.Height) then

begin

SetcmbBrands();

spnlSortFilt.BringToFront;

slblSortFilt.Visible := false;

scmbBrand.Visible := true;

scmbYear.Visible := true;

scmbPower.Visible := true;

scmbTransm.Visible := true;

scmbDriveUnit.Visible := true;

scmbFuelType.Visible := true;

scmbPrice.Visible := true;

srgrSort.Visible := true;

if GetCapture <> spnlSortFilt.Handle then

begin

SetCapture(spnlSortFilt.Handle);

end;

end

else

begin

slblSortFilt.Visible := true;

srgrSort.Visible := false;

scmbBrand.Visible := false;

scmbYear.Visible := false;

scmbPower.Visible := false;

scmbTransm.Visible := false;

scmbDriveUnit.Visible := false;

scmbFuelType.Visible := false;

scmbPrice.Visible := false;

spnlSortFilt.SendToBack;

ReleaseCapture;

end;

end;

procedure TFormMain.trlHelpClick(Sender: TObject);

begin

FormHelp.Position := poScreenCenter;

FormHelp.ShowModal;

end;

end.

//-------------------------------------

unit CarCreate;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,

Vcl.Imaging.pngimage, Vcl.Grids, Vcl.ValEdit, Vcl.ExtDlgs, JPEG, Magazine, sPanel,

acSlider, sButton, acAlphaImageList, Vcl.VirtualImageList,

Vcl.BaseImageCollection, Vcl.ImageCollection, System.ImageList, Vcl.ImgList,

Vcl.Buttons, sSpeedButton, acImage, acPNG, sGroupBox, Vcl.ComCtrls,

sRadioButton, Vcl.Mask, sMaskEdit, sCustomComboEdit, sComboBox, sComboBoxes,

sScrollBar, sBitBtn, IWVCLBaseControl, IWBaseControl, IWBaseHTMLControl,

IWControl, IWCompText, sSkinProvider, sLabel;

type

TCarSizeParms = packed Record

Length: Integer;

Width: Integer;

Height: Integer;

End;

TCar = packed Record

Brand: string[20];

Model: string[20];

Year: Integer;

Color: string[20];

Mileage: Integer;

Power: Integer;

EngineVol: extended;

Acceler100: extended;

AverFuelRate: extended;

Fuel: string[10];

Transm: string[10];

DriveUnit: string[20];

Weight: Integer;

Dimensions: TCarSizeParms;

Price: Integer;

HolderName: string[40];

Contacts: string[17];

Email: string[50];

Images: TImageArr;

End;

TFormCarCreate = class(TForm)

btnPhotoInp: TButton;

edtBrand: TEdit;

edtModel: TEdit;

edtYear: TEdit;

edtCarColor: TEdit;

edtMileage: TEdit;

edtPower: TEdit;

edtEngineVol: TEdit;

edtAverFuel: TEdit;

edtAcceler100: TEdit;

edtLength: TEdit;

edtWeight: TEdit;

edtPrice: TEdit;

edtWidth: TEdit;

edtHeight: TEdit;

edtHolderName: TEdit;

edtContacts: TEdit;

edtEmail: TEdit;

btnPhotoSave: TButton;

dlgLoadPic: TOpenPictureDialog;

cmbFuelType: TsComboBox;

cmbTransm: TsComboBox;

cmbDriveUnit: TsComboBox;

btnCancelCreate: TButton;

pnlControlCreate: TPanel;

btnConfirmCreate: TButton;

pnlImageLoad: TPanel;

imgCarPhoto: TsImage;

imgDeletePhoto: TImage;

stxtPersInf: TStaticText;

pnlPersInfo: TPanel;

pnlPrice: TPanel;

pnlDescription: TPanel;

stxtAboutCar: TStaticText;

stxtTechChars: TStaticText;

stxtDimensions: TStaticText;

stxtLoadPic: TStaticText;

imglstButtons: TsAlphaImageList;

bbtnSlideRight: TsBitBtn;

bbtnSlideLeft: TsBitBtn;

slblContacts: TsLabel;

slblEmail: TsLabel;

slblHolderName: TsLabel;

slblPrice: TsLabel;

slblModel: TsLabel;

slblYear: TsLabel;

slblMileage: TsLabel;

slblLength: TsLabel;

slblBrand: TsLabel;

slblWidth: TsLabel;

slblHeight: TsLabel;

slblWeight: TsLabel;

slblCarColor: TsLabel;

slblPower: TsLabel;

slblAverFuel: TsLabel;

slblAcceler100: TsLabel;

slblEngineVol: TsLabel;

slblTransm: TsLabel;

slblDriveUnit: TsLabel;

slblFuelType: TsLabel;

sbtnSaveChange: TsButton;

sbtnCancelChange: TsButton;

sbtnEditAd: TsButton;

sbtnDeleteCar: TsButton;

procedure btnPhotoInpClick(Sender: TObject);

procedure btnPhotoSaveClick(Sender: TObject);

procedure FormMouseWheelUp(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: Boolean);

procedure FormMouseWheelDown(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: Boolean);

procedure btnConfirmCreateClick(Sender: TObject);

procedure btnCancelCreateClick(Sender: TObject);

procedure bbtnSlideRightClick(Sender: TObject);

procedure bbtnSlideLeftClick(Sender: TObject);

procedure imgDeletePhotoMouseEnter(Sender: TObject);

procedure imgDeletePhotoClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure sbtnEditAdClick(Sender: TObject);

procedure sbtnCancelChangeClick(Sender: TObject);

procedure sbtnSaveChangeClick(Sender: TObject);

procedure sbtnDeleteCarClick(Sender: TObject);

private

{ Private declarations }

public

function IsAllCorrect(): Boolean;

function IsCarCreated(): Boolean;

procedure SetInfo(Car: TCar);

end;

TCarFile = file of TCar;

var

FormCarCreate: TFormCarCreate;

CarExpl: TCar;

CarOnPanel: TCar;

FileCarList: TCarFile;

ImagesAm: Integer = 0;

CurrImg: Integer;

implementation

{$R \*.dfm}

uses AutoMag, Favorites;

procedure ToClearCar();

begin

ImagesAm := 0;

CurrImg := 0;

ZeroMemory(@CarExpl, SizeOf(CarExpl));

end;

function TFormCarCreate.IsCarCreated(): Boolean;

var

PCar: TPCar;

begin

Result := false;

PCar := ListCar.PFirstCar;

while (PCar <> nil) and (not Result) do

begin

if IsCarEqual(PCar^.CarInfo, CarExpl) then

Begin

Result := true;

MessageBox(handle, PChar('Ad already created. Create other.'), PChar('Ad info'), MB\_OK + MB\_ICONINFORMATION);

End;

PCar := PCar^.PNextCar;

end;

end;

function TFormCarCreate.IsAllCorrect(): Boolean;

procedure GetBrandCorrect();

Begin

if string(edtBrand.Text).IsEmpty then

slblBrand.UseSkinColor := false

Else

Begin

slblBrand.UseSkinColor := true;

CarExpl.Brand := edtBrand.Text;

End;

End;

procedure GetModelCorrect();

Begin

if string(edtModel.Text).IsEmpty then

slblModel.UseSkinColor := false

Else

Begin

slblModel.UseSkinColor := true;

CarExpl.Model := edtModel.Text;

End;

End;

procedure GetYearCorrect();

Var

EnteredNum: Integer;

begin

if String(edtYear.Text).IsEmpty then

Begin

slblYear.UseSkinColor := false;

exit;

End;

ValidInp(edtYear.Text, EnteredNum);

if (EnteredNum < 1800) or (EnteredNum > 2021) then

slblYear.UseSkinColor := false

Else

Begin

slblYear.UseSkinColor := true;

CarExpl.Year := EnteredNum;

End;

end;

procedure GetMileageCorrect();

begin

if string(edtMileage.Text).IsEmpty then

Begin

slblMileage.UseSkinColor := false;

exit;

End;

slblMileage.UseSkinColor := true;

CarExpl.Mileage := StrToInt(edtMileage.Text);

end;

procedure GetLengthCorrect();

Var

EnteredNum: Integer;

begin

if string(edtLength.Text).IsEmpty then

Begin

slblLength.UseSkinColor := false;

exit;

End;

ValidInp(String(edtLength.Text), EnteredNum);

if (EnteredNum < 1000) then

slblLength.UseSkinColor := false

Else

Begin

slblLength.UseSkinColor := true;

CarExpl.Dimensions.Length := EnteredNum;

End;

end;

procedure GetWidthCorrect();

Var

EnteredNum: Integer;

begin

if string(edtWidth.Text).IsEmpty then

Begin

slblWidth.UseSkinColor := false;

exit;

End;

ValidInp(String(edtWidth.Text), EnteredNum);

if (EnteredNum < 1000) or (EnteredNum > 5000) then

slblWidth.UseSkinColor := false

Else

Begin

slblWidth.UseSkinColor := true;

CarExpl.Dimensions.Width := EnteredNum;

End;

end;

procedure GetHeightCorrect();

Var

EnteredNum: Integer;

begin

if string(edtHeight.Text).IsEmpty then

Begin

slblHeight.UseSkinColor := false;

exit;

End;

ValidInp(String(edtHeight.Text), EnteredNum);

if (EnteredNum < 1100) or (EnteredNum > 4000) then

slblHeight.UseSkinColor := false

Else

Begin

slblHeight.UseSkinColor := true;

CarExpl.Dimensions.Height := EnteredNum;

End;

end;

procedure GetWeightCorrect();

Var

EnteredNum: Integer;

begin

if string(edtWeight.Text).IsEmpty then

Begin

slblWeight.UseSkinColor := false;

exit;

End;

ValidInp(String(edtWeight.Text), EnteredNum);

if (EnteredNum < 300) or (EnteredNum > 15000) then

slblWeight.UseSkinColor := false

Else

Begin

slblWeight.UseSkinColor := true;

CarExpl.Weight := EnteredNum;

End;

end;

procedure GetCarColorCorrect();

var

ColorChars: set of char;

Each: char;

begin

ColorChars := ['A'..'z', '-'];

if string(edtCarColor.Text).IsEmpty then

Begin

slblCarColor.UseSkinColor := false;

exit;

End;

for Each in edtCarColor.Text do

if not (Each in ColorChars)then

Begin

slblCarColor.UseSkinColor := false;

exit;

End;

slblCarColor.UseSkinColor := true;

CarExpl.Color := edtCarColor.Text;

end;

procedure GetPowerCorrect();

Var

EnteredNum: Integer;

begin

if string(edtPower.Text).IsEmpty then

Begin

slblPower.UseSkinColor := false;

exit;

End;

if ValidInp(String(edtPower.Text), EnteredNum) then

Begin

if (EnteredNum < 1) or (EnteredNum > 2000) then

slblPower.UseSkinColor := false

Else

Begin

slblPower.UseSkinColor := true;

CarExpl.Power := EnteredNum;

End;

End

Else

slblPower.UseSkinColor := false;

end;

procedure GetAverFuelCorrect();

Var

EnteredNum: Real;

begin

if string(edtAverFuel.Text).IsEmpty then

Begin

slblAverFuel.UseSkinColor := false;

exit;

End;

if ValidInp(String(edtAverFuel.Text), EnteredNum) then

Begin

if (EnteredNum <= 0) or (EnteredNum > 200) then

slblAverFuel.UseSkinColor := false

Else

Begin

slblAverFuel.UseSkinColor := true;

CarExpl.AverFuelRate := EnteredNum;

End;

End

Else

slblAverFuel.UseSkinColor := false;

end;

procedure GetAcceler100Correct();

Var

EnteredNum: Real;

begin

if string(edtAcceler100.Text).IsEmpty then

Begin

slblAcceler100.UseSkinColor := false;

exit;

End;

if ValidInp(String(edtAcceler100.Text), EnteredNum) then

Begin

if (EnteredNum < 0.1) or (EnteredNum >= 200) then

slblAcceler100.UseSkinColor := false

Else

Begin

slblAcceler100.UseSkinColor := true;

CarExpl.Acceler100 := EnteredNum;

End;

End

Else

slblAcceler100.UseSkinColor := false;

end;

procedure GetEngineVolCorrect();

Var

EnteredNum: Real;

begin

if string(edtEngineVol.Text).IsEmpty then

Begin

slblEngineVol.UseSkinColor := false;

exit;

End;

if ValidInp(String(edtEngineVol.Text), EnteredNum) then

Begin

if (EnteredNum < 0) or (EnteredNum >= 10) then

slblEngineVol.UseSkinColor := false

Else

Begin

slblEngineVol.UseSkinColor := true;

CarExpl.EngineVol := EnteredNum;

End;

End

Else

slblEngineVol.UseSkinColor := false;

end;

procedure GetTransmCorrect();

begin

if cmbTransm.ItemIndex = -1 then

slblTransm.UseSkinColor := false

Else

Begin

slblTransm.UseSkinColor := true;

CarExpl.Transm := cmbTransm.Text;

End;

end;

procedure GetDriveUnitCorrect();

begin

if cmbDriveUnit.ItemIndex = -1 then

slblDriveUnit.UseSkinColor := false

Else

Begin

slblDriveUnit.UseSkinColor := true;

CarExpl.DriveUnit := cmbDriveUnit.Text;

End;

end;

procedure GetFuelTypeCorrect();

begin

if cmbFuelType.ItemIndex = -1 then

slblFuelType.UseSkinColor := false

Else

Begin

slblFuelType.UseSkinColor := true;

CarExpl.Fuel := cmbFuelType.Text;

End;

end;

procedure GetHolderNameCorrect();

var

Alphabet: set of char;

Each: char;

begin

Alphabet := ['A'..'z', ' '];

if string.IsNullOrWhiteSpace(edtHolderName.Text) then

Begin

slblHolderName.UseSkinColor := false;

exit;

End;

for Each in edtHolderName.Text do

if not (Each in Alphabet)then

Begin

slblHolderName.UseSkinColor := false;

exit;

End;

slblHolderName.UseSkinColor := true;

CarExpl.HolderName := edtHolderName.Text;

end;

procedure GetContactsCorrect();

var

NumberChars: set of char;

Each: Integer;

begin

NumberChars := ['0' .. '9'];

if string.IsNullOrWhiteSpace(edtContacts.Text) then

Begin

slblContacts.UseSkinColor := false;

exit;

End;

for Each := 2 to edtContacts.GetTextLen do

if (not (edtContacts.Text[Each] in NumberChars)) or (edtContacts.Text[1] <> '+') or (edtContacts.GetTextLen < 10) then

Begin

slblContacts.UseSkinColor := false;

exit;

End;

slblContacts.UseSkinColor := true;

CarExpl.Contacts := edtContacts.Text;

end;

procedure GetEmailCorrect();

var

NumberChars: set of char;

Each: Integer;

DogCount: Integer;

begin

NumberChars := ['A' .. 'z', '0' .. '9', '\_', '-', '.', '@'];

if string.IsNullOrWhiteSpace(edtEmail.Text)then

Begin

slblEmail.UseSkinColor := false;

exit;

End;

DogCount := 0;

for Each := 1 to edtEmail.GetTextLen do

if edtEmail.Text[Each] = '@' then

Inc(DogCount);

for Each := 1 to edtEmail.GetTextLen do

Begin

if (not (edtEmail.Text[Each] in NumberChars)) or (edtEmail.Text[1] = '.') or (edtEmail.Text[edtEmail.GetTextLen] = '.')

or (DogCount <> 1) then

Begin

slblEmail.UseSkinColor := false;

exit;

End;

End;

slblEmail.UseSkinColor := true;

CarExpl.Email := edtEmail.Text;

end;

procedure GetPriceCorrect();

Var

EnteredNum: Integer;

begin

if string(edtPrice.Text).IsEmpty then

Begin

slblPrice.UseSkinColor := false;

exit;

End;

ValidInp(String(edtPrice.Text), EnteredNum);

slblPrice.UseSkinColor := true;

CarExpl.Price := EnteredNum;

end;

var

CompIndslbl: Integer;

begin

Result := true;

GetBrandCorrect();

GetModelCorrect();

GetYearCorrect();

GetMileageCorrect();

GetLengthCorrect();

GetWidthCorrect();

GetHeightCorrect();

GetWeightCorrect();

GetCarColorCorrect();

GetPowerCorrect();

GetAverFuelCorrect();

GetAcceler100Correct();

GetEngineVolCorrect();

GetTransmCorrect();

GetDriveUnitCorrect();

GetFuelTypeCorrect();

GetHolderNameCorrect();

GetContactsCorrect();

GetEmailCorrect();

GetPriceCorrect();

for CompIndslbl := 0 to FormCarCreate.ComponentCount - 1 do

if FormCarCreate.Components[CompIndslbl] is TsLabel then

if (FormCarCreate.Components[CompIndslbl] as TsLabel).UseSkinColor = false then

Begin

Result := false;

exit;

End;

end;

procedure TFormCarCreate.sbtnCancelChangeClick(Sender: TObject);

var

i: Integer;

begin

SetInfo(CarExpl);

for i := 0 to FormCarCreate.ComponentCount - 1 do

Begin

if FormCarCreate.Components[i] is TEdit then

(FormCarCreate.Components[i] as TEdit).Enabled := false;

if FormCarCreate.Components[i] is TsComboBox then

(FormCarCreate.Components[i] as TsComboBox).Enabled := false;

End;

btnPhotoInp.Visible := false;

btnPhotoSave.Visible := false;

imgDeletePhoto.Visible := false;

sbtnCancelChange.Enabled := false;

sbtnEditAd.Enabled := true;

sbtnSaveChange.Enabled := false;

sbtnDeleteCar.Enabled := true;

end;

procedure TFormCarCreate.sbtnDeleteCarClick(Sender: TObject);

var

btnSelected: Integer;

PCar: TPCar;

begin

btnSelected := MessageDlg('Do you want to delete ad?', mtWarning, mbOKCancel, 0);

if btnSelected = mrOk then

begin

DeleteCarFromList(ListCar, CarOnPanel);

PCar := ListCarFav.PFirstCar;

while PCar <> nil do

begin

if IsCarEqual(PCar^.CarInfo, CarOnPanel) then

begin

DeleteCarFromList(ListCarFav, CarOnPanel);

Dec(FavsAmount);

end;

PCar := PCar^.PNextCar;

end;

ClearPanels(CarInfoPanel, CurrPanel, CarsAmount);

Dec(CarsAmount);

FormMain.scrlboxCars.VertScrollBar.Range := FormMain.scrlboxCars.VertScrollBar.Range - 315;

ShowCars(ListCar, CurrPanel, CarInfoPanel, FormMain.scrlboxCars, 100);

FormCarCreate.Close;

end;

end;

procedure TFormCarCreate.sbtnEditAdClick(Sender: TObject);

var

i: Integer;

begin

for i := 0 to FormCarCreate.ComponentCount - 1 do

begin

if FormCarCreate.Components[i] is TEdit then

(FormCarCreate.Components[i] as TEdit).Enabled := true;

if FormCarCreate.Components[i] is TsComboBox then

(FormCarCreate.Components[i] as TsComboBox).Enabled := true;

imgDeletePhoto.Enabled := true;

imgDeletePhoto.Visible := true;

btnPhotoInp.Visible := true;

btnPhotoSave.Visible := true;

sbtnEditAd.Enabled := false;

sbtnCancelChange.Enabled := true;

sbtnSaveChange.Enabled := true;

cmbTransm.ItemIndex := TransmConvert(cmbTransm.Text);

cmbFuelType.ItemIndex := FuelTypeConvert(cmbFuelType.Text);

cmbDriveUnit.ItemIndex := DriveUnitConvert(cmbDriveUnit.Text);

cmbFuelType.Style := csDropDownList;

cmbTransm.Style := csDropDownList;

cmbDriveUnit.Style := csDropDownList;

sbtnDeleteCar.Enabled := false;

end;

end;

procedure TFormCarCreate.sbtnSaveChangeClick(Sender: TObject);

var

btnSelected, i, j: Integer;

PCar: TPCar;

begin

if not IsAllCorrect then

Begin

if ImagesAm <> 0 then

MessageBox(handle, PChar('Enter all fields correctly!'), PChar('Ad creating'), MB\_OK + MB\_ICONERROR)

else

MessageBox(handle, PChar('Enter all fields correctly and load photo!'), PChar('Ad creating'), MB\_OK + MB\_ICONERROR);

End

Else

if ImagesAm <> 0 then

Begin

btnSelected := MessageDlg('Do you want to edit ad?', mtWarning, mbOKCancel, 0);

if (btnSelected = mrOk) and (not IsCarCreated()) then

begin

PCar := ListCar.PFirstCar;

while PCar <> nil do

begin

if IsCarEqual(PCar^.CarInfo, CarOnPanel) then

PCar^.CarInfo := CarExpl;

PCar := PCar^.PNextCar;

end;

PCar := ListCarFav.PFirstCar;

while PCar <> nil do

begin

if IsCarEqual(PCar^.CarInfo, CarOnPanel) then

PCar^.CarInfo := CarExpl;

PCar := PCar^.PNextCar;

end;

for i := 0 to CarsAmount - 1 do

if IsCarEqual(CarInfoPanel[i].LinkedCar, CarOnPanel) then

Begin

CarInfoPanel[i].SetInfo(CarExpl);

CarInfoPanel[i].LinkedCar := CarExpl;

End;

with Screen do

for i := 0 to FormCount - 1 do

begin

if Forms[i] = FormFavs then

for j := 0 to FavsAmount - 1 do

if IsCarEqual(FavsInfoPanel[j].LinkedCar, CarOnPanel) then

Begin

FavsInfoPanel[j].SetInfo(CarExpl);

FavsInfoPanel[j].LinkedCar := CarExpl;

End;

end;

sbtnCancelChange.Click;

end;

End

Else

MessageBox(handle, PChar('Load photo!'), PChar('Ad creating'), MB\_OK + MB\_ICONERROR);

end;

procedure TFormCarCreate.btnPhotoInpClick(Sender: TObject);

Var

IsPhotoLoad: Boolean;

begin

IsPhotoLoad := false;

imgCarPhoto.Picture := nil;

if dlgLoadPic.Execute then

Begin

stxtLoadPic.Visible := false;

IsPhotoLoad := true;

imgCarPhoto.Picture.LoadFromFile(dlgLoadPic.FileName);

End

Else

if ImagesAm = 0 then

stxtLoadPic.Visible := true;

if (not IsPhotoLoad) and (ImagesAm <> 0) then

imgCarPhoto.Picture.LoadFromFile(CarExpl.Images[CurrImg]);

imgCarPhoto.Stretch := true;

end;

procedure TFormCarCreate.btnPhotoSaveClick(Sender: TObject);

Var

i: Integer;

begin

if ImagesAm + 1 > 20 then

Begin

MessageBox(handle, PChar('There is a limit: 20 photos. You can delete some previous.'), PChar('Car photo'), MB\_OK + MB\_ICONWARNING);

exit;

End;

for i := 1 to ImagesAm do

if CarExpl.Images[i] = dlgLoadPic.FileName then

Begin

MessageBox(handle, PChar('The picture already loaded, choose other!'), PChar('Car photo'), MB\_OK + MB\_ICONWARNING);

btnPhotoInp.Click;

exit;

End;

if dlgLoadPic.FileName <> '' then

Begin

inc(ImagesAm);

imgDeletePhoto.Visible := true;

imgDeletePhoto.BringToFront;

CarExpl.Images[ImagesAm] := dlgLoadPic.FileName;

CurrImg := ImagesAm;

End

Else

MessageBox(handle, PChar('Load photo at first!'), PChar('Car photo'), MB\_OK + MB\_ICONWARNING);

if ImagesAm >= 2 then

begin

bbtnSlideRight.Visible := true;

bbtnSlideLeft.Visible := true;

end;

dlgLoadPic.FileName := '';

end;

procedure TFormCarCreate.bbtnSlideLeftClick(Sender: TObject);

begin

if CurrImg <> 1 then

dec(CurrImg)

else

CurrImg := ImagesAm;

imgCarPhoto.Picture.LoadFromFile(CarExpl.Images[CurrImg]);

end;

procedure TFormCarCreate.bbtnSlideRightClick(Sender: TObject);

begin

if CurrImg <> ImagesAm then

inc(CurrImg)

else

CurrImg := 1;

imgCarPhoto.Picture.LoadFromFile(CarExpl.Images[CurrImg]);

end;

procedure TFormCarCreate.btnCancelCreateClick(Sender: TObject);

begin

FormCarCreate.Close;

end;

procedure TFormCarCreate.btnConfirmCreateClick(Sender: TObject);

begin

Seek(FileCarList, FileSize(FileCarList));

if not IsAllCorrect then

Begin

if ImagesAm <> 0 then

MessageBox(handle, PChar('Enter all fields correctly!'), PChar('Ad creating'), MB\_OK + MB\_ICONERROR)

else

MessageBox(handle, PChar('Enter all fields correctly and load photo!'), PChar('Ad creating'), MB\_OK + MB\_ICONERROR);

End

Else

if ImagesAm <> 0 then

Begin

if not IsCarCreated() then

begin

OneAddListOnShow(CarExpl);

FormCarCreate.Close;

end;

End

Else

MessageBox(handle, PChar('Load photo!'), PChar('Ad creating'), MB\_OK + MB\_ICONERROR);

end;

procedure TFormCarCreate.FormClose(Sender: TObject; var Action: TCloseAction);

begin

ToClearCar();

end;

procedure TFormCarCreate.FormMouseWheelDown(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: Boolean);

begin

with FormCarCreate.VertScrollBar do

Position := Position + Increment;

end;

procedure TFormCarCreate.FormMouseWheelUp(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: Boolean);

begin

with FormCarCreate.VertScrollBar do

Position := Position - Increment;

end;

procedure TFormCarCreate.imgDeletePhotoClick(Sender: TObject);

begin

ToShiftArrLeft(CarExpl.Images, ImagesAm, CurrImg);

if ImagesAm = 1 then

begin

bbtnSlideRight.Visible := false;

bbtnSlideLeft.Visible := false;

end;

if ImagesAm = 0 then

begin

imgCarPhoto.Picture := nil;

stxtLoadPic.Visible := true;

imgDeletePhoto.Visible := false;

end

else

imgCarPhoto.Picture.LoadFromFile(CarExpl.Images[CurrImg]);

end;

procedure TFormCarCreate.imgDeletePhotoMouseEnter(Sender: TObject);

begin

imgDeletePhoto.Enabled := true;

end;

procedure TFormCarCreate.SetInfo(Car: TCar);

begin

CarExpl := Car;

CarOnPanel := Car;

edtBrand.Text := Car.Brand;

edtModel.Text := Car.Model;

edtYear.Text := IntToStr(Car.Year);

edtCarColor.Text := Car.Color;

edtMileage.Text := IntToStr(Car.Mileage);

edtPower.Text := IntToStr(Car.Power);

edtEngineVol.Text := RealToStr(Car.EngineVol);

edtAverFuel.Text := RealToStr(Car.AverFuelRate);

edtAcceler100.Text := RealToStr(Car.Acceler100);

edtLength.Text := IntToStr(Car.Dimensions.Length);

edtWeight.Text := IntToStr(Car.Weight);

edtPrice.Text := IntToStr(Car.Price);

edtWidth.Text := IntToStr(Car.Dimensions.Width);

edtHeight.Text := IntToStr(Car.Dimensions.Height);

edtHolderName.Text := Car.HolderName;

edtContacts.Text := Car.Contacts;

edtEmail.Text := Car.Email;

cmbFuelType.Style := csDropDown;

cmbTransm.Style := csDropDown;

cmbDriveUnit.Style := csDropDown;

cmbFuelType.Text := Car.Fuel;

cmbTransm.Text := Car.Transm;

cmbDriveUnit.Text := Car.DriveUnit;

GetPhotosAmount(CarExpl.Images, ImagesAm);

CurrImg := 1;

imgCarPhoto.Picture.LoadFromFile(CarExpl.Images[CurrImg]);

end;

end.

unit Cart;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.DBCGrids, acDBCtrlGrid, Vcl.Grids,

Vcl.StdCtrls, sLabel;

type

TFormCart = class(TForm)

strgridCart: TStringGrid;

slblTotalPrice: TsLabel;

stxtTotal: TStaticText;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FormCart: TFormCart;

implementation

{$R \*.dfm}

uses AutoMag;

procedure SetCars();

var

PCar: TPCar;

CurrCol: Integer;

Total: Integer;

img: TPicture;

begin

Total := 0;

PCar := ListCarCart.PFirstCar;

For CurrCol := 1 to CartAmount do

begin

with FormCart.strgridCart do

begin

Cells[CurrCol,1] := PCar^.CarInfo.Brand;

Cells[CurrCol,2] := PCar^.CarInfo.Model;

Cells[CurrCol,3] := IntToStr(PCar^.CarInfo.Year);

Cells[CurrCol,4] := IntToStr(PCar^.CarInfo.Mileage);

Cells[CurrCol,5] := IntToStr(PCar^.CarInfo.Power);

Cells[CurrCol,6] := FloatToStr(PCar^.CarInfo.AverFuelRate);

Cells[CurrCol,7] := FloatToStr(PCar^.CarInfo.EngineVol);

Cells[CurrCol,8] := FloatToStr(PCar^.CarInfo.Acceler100);

Cells[CurrCol,9] := PCar^.CarInfo.Transm;

Cells[CurrCol,10] := PCar^.CarInfo.Fuel;

Cells[CurrCol,11] := PCar^.CarInfo.DriveUnit;

Cells[CurrCol,12] := IntToStr(PCar^.CarInfo.Price);

Cells[CurrCol,13] := IntToStr(PCar^.CarInfo.Dimensions.Length);

Cells[CurrCol,14] := IntToStr(PCar^.CarInfo.Dimensions.Width);

Cells[CurrCol,15] := IntToStr(PCar^.CarInfo.Dimensions.Height);

Cells[CurrCol,16] := IntToStr(PCar^.CarInfo.Weight);

Cells[CurrCol,17] := PCar^.CarInfo.Color;

end;

Inc(Total, PCar^.CarInfo.Price);

PCar := PCar^.PNextCar;

end;

FormCart.slblTotalPrice.Caption := IntToStr(Total) + '$';

end;

procedure TFormCart.FormCreate(Sender: TObject);

begin

strGridCart.ColCount := CartAmount + 1;

with strgridCart do

begin

Cells[0,1] := 'Brand';

Cells[0,2] := 'Model';

Cells[0,3] := 'Year';

Cells[0,4] := 'Mileage';

Cells[0,5] := 'Power';

Cells[0,6] := 'Average fuel';

Cells[0,7] := 'Engine volume';

Cells[0,8] := '0 - 100 km/h';

Cells[0,9] := 'Transmission';

Cells[0,10] := 'Fuel type';

Cells[0,11] := 'Drive unit';

Cells[0,12] := 'Price';

Cells[0,13] := 'Length';

Cells[0,14] := 'Width';

Cells[0,15] := 'Height';

Cells[0,16] := 'Weight';

Cells[0,17] := 'Color';

end;

SetCars();

end;

end.

unit Favorites;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, sGroupBox, Vcl.ExtCtrls,

sSplitter, Vcl.ComCtrls, sTrackBar, Vcl.Mask, sMaskEdit, sCustomComboEdit,

sCurrEdit, Vcl.OleCtrls, SHDocVw, acWebBrowser, sScrollBox, CarPanel, sButton;

type

TFormFavs = class(TForm)

scrlboxFavs: TsScrollBox;

procedure FormCreate(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FormFavs: TFormFavs;

FavsInfoPanel: array of TCarPanel;

FavCurrPanel: integer;

implementation

{$R \*.dfm}

uses AutoMag;

procedure TFormFavs.FormClose(Sender: TObject; var Action: TCloseAction);

var

pnlFavsCount, pnlMainCount: Integer;

begin

for pnlFavsCount := 0 to High(FavsInfoPanel) do

begin

if FavsInfoPanel[pnlFavsCount].imgFav.ImageIndex = 1 then

for pnlMainCount := 0 to CarsAmount - 1 do

if IsCarEqual(FavsInfoPanel[pnlFavsCount].LinkedCar, CarInfoPanel[pnlMainCount].LinkedCar) then

CarInfoPanel[pnlMainCount].imgFav.ImageIndex := 1;

end;

end;

procedure TFormFavs.FormCreate(Sender: TObject);

var

i: Integer;

begin

FavCurrPanel := 0;

scrlboxFavs.VertScrollBar.Range := FavsAmount \* 315;

SetLength(FavsInfoPanel, FavsAmount);

ShowCars(ListCarFav, FavCurrPanel, FavsInfoPanel, FormFavs.scrlboxFavs, 0);

for i := 0 to FavsAmount - 1 do

FavsInfoPanel[i].imgFav.ImageIndex := 2;

end;

end.

unit Help;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.DBCtrls, Vcl.ToolWin,

Vcl.ComCtrls, Vcl.ExtCtrls, acTitleBar, sButton, sMemo;

type

TFormHelp = class(TForm)

sMemo1: TsMemo;

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

FormHelp: TFormHelp;

implementation

{$R \*.dfm}

procedure TFormHelp.FormCreate(Sender: TObject);

var

HelpFile: textfile;

StrToWrite, Tmp: string;

begin

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

AssignFile(HelpFile, '../../CarPh/ProgramInfo.txt');

Reset(HelpFile);

ReadLn(HelpFile, Tmp);

sMemo1.Text := Tmp;

StrToWrite := Tmp;

while not EOF(HelpFile) do

Begin

ReadLn(HelpFile, Tmp);

sMemo1.Text := sMemo1.Text + #13#10;

sMemo1.Text := sMemo1.Text + Tmp;

End;

Readln(HelpFile, Tmp);

CloseFile(HelpFile);

end;

end.

unit CarPanel;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,

Vcl.Imaging.pngimage, Vcl.Grids, Vcl.ValEdit, Vcl.ExtDlgs, JPEG, Magazine, sPanel,

acSlider, sButton, acAlphaImageList, Vcl.VirtualImageList,

Vcl.BaseImageCollection, Vcl.ImageCollection, System.ImageList, Vcl.ImgList,

Vcl.Buttons, sSpeedButton, acImage, acPNG, sGroupBox, Vcl.ComCtrls,

sRadioButton, Vcl.Mask, sMaskEdit, sCustomComboEdit, sComboBox, sComboBoxes,

sScrollBar, sBitBtn, IWVCLBaseControl, IWBaseControl, IWBaseHTMLControl,

IWControl, IWCompText, sSkinProvider, sLabel, CarCreate;

type

TCarPanel = class(TsPanel)

slblBrand: TsLabel;

slblModel: TsLabel;

slblYear: TsLabel;

slblCarColor: TsLabel;

slblMileage: TsLabel;

slblPrice: TsLabel;

slblPower: TsLabel;

slblTransm: TsLabel;

slblDriveUnit: TsLabel;

slblFuelType: TsLabel;

slblContacts: TsLabel;

imgCarPhoto: TsImage;

imgFav: TsImage;

sbtnBuyCar: TsButton;

slblShowMore: TsLabel;

procedure WhenClickedFav(Sender:Tobject);

procedure pnlOnMouseEnter(Sender: TObject);

procedure pnlOnMouseLeave(Sender: TObject);

procedure slblShowOnMouseEnter(Sender: TObject);

procedure slblShowOnMouseLeave(Sender: TObject);

procedure slblShowOnClick(Sender: TObject);

procedure btnAddOnClick(Sender: TObject);

procedure btnAddInCartOnEnter(Sender: TObject);

procedure btnAddInCartOnLeave(Sender: TObject);

procedure NothingToDo(Sender: TObject);

private

{ Private declarations }

protected

{ Protected declarations }

public

LinkedCar: TCar;

procedure SetAllComps(CarInfo: TCar);

procedure SetInfo(CarInfo: TCar);

procedure pnlOnCreate();

//procedure imgFavClick();

published

{ Published declarations }

end;

procedure Register;

implementation

Uses

AutoMag, Favorites;

procedure TCarPanel.pnlOnCreate();

begin

Self.slblBrand := TsLabel.Create(Self);

Self.slblBrand.Parent := Self;

Self.slblModel := TsLabel.Create(Self);

Self.slblModel.Parent := Self;

Self.slblYear := TsLabel.Create(Self);

Self.slblYear.Parent := Self;

Self.slblMileage := TsLabel.Create(Self);

Self.slblMileage.Parent := Self;

Self.slblCarColor := TsLabel.Create(Self);

Self.slblCarColor.Parent := Self;

Self.slblPrice := TsLabel.Create(Self);

Self.slblPrice.Parent := Self;

Self.slblPower := TsLabel.Create(Self);

Self.slblPower.Parent := Self;

Self.slblTransm := TsLabel.Create(Self);

Self.slblTransm.Parent := Self;

Self.slblTransm := TsLabel.Create(Self);

Self.slblTransm.Parent := Self;

Self.slblDriveUnit := TsLabel.Create(Self);

Self.slblDriveUnit.Parent := Self;

Self.slblFuelType := TsLabel.Create(Self);

Self.slblFuelType.Parent := Self;

Self.slblContacts := TsLabel.Create(Self);

Self.slblContacts.Parent := Self;

Self.imgFav := TsImage.Create(Self);

Self.imgFav.Parent := Self;

Self.imgCarPhoto := TsImage.Create(Self);

Self.imgCarPhoto.Parent := Self;

end;

procedure TCarPanel.SetAllComps(CarInfo: TCar);

procedure SetBrand();

Begin

with slblBrand do

Begin

Font.Style := [fsBold];

Font.Size := 16;

Font.Color := clBlue;

Left := 420;

Top := 30;

End;

End;

procedure SetModel();

Begin

with slblModel do

Begin

Font.Style := [fsBold];

Font.Size := 16;

Font.Color := clBlue;

Left := slblBrand.Left + slblBrand.Width + 20;

Top := 30;

End;

End;

procedure SetYear();

Begin

with slblYear do

Begin

Font.Size := 12;

Left := slblBrand.Left;

Top := slblBrand.Top + 30;

End;

End;

procedure SetMileage();

Begin

with slblMileage do

Begin

Font.Size := 12;

Left := slblYear.Left + 50 + 40;

Top := slblBrand.Top + 30;

End;

End;

procedure SetColor();

Begin

with slblCarColor do

Begin

Font.Size := 12;

Left := slblMileage.Left + 90 + 40;

Top := slblBrand.Top + 30;

End;

End;

procedure SetPrice();

Begin

with slblPrice do

Begin

Font.Style := [fsBold];

Font.Size := 24;

Left := 860;

Top := imgCarPhoto.Top + imgCarPhoto.Height div 2 - 35;

End;

End;

procedure SetPower();

Begin

with slblPower do

Begin

Font.Size := 12;

Left := slblBrand.Left;

Top := slblYear.Top + 40;

End;

End;

procedure SetTransm();

Begin

with slblTransm do

Begin

Font.Size := 12;

Left := slblPower.Left;

Top := slblPower.Top + 26;

End;

End;

procedure SetDriveUnit();

Begin

with slblDriveUnit do

Begin

Font.Size := 12;

Left := slblPower.Left + 70 + 100;

Top := slblPower.Top;

End;

End;

procedure SetFuelType();

Begin

with slblFuelType do

Begin

Font.Size := 12;

Left := slblDriveUnit.Left;

Top := slblTransm.Top;

End;

End;

procedure SetContacts();

Begin

with slblContacts do

Begin

Font.Size := 14;

Left := sbtnBuyCar.Left + sbtnBuyCar.Width div 2 - Width div 2 - 4;

Top := sbtnBuyCar.Top + sbtnBuyCar.Height + 5;

End;

End;

procedure SetCarPhoto();

Begin

with imgCarPhoto do

Begin

Stretch := true;

Height := 216;

Width := 384;

Left := 10;

Top := 30;

End;

End;

procedure SetimgFav();

Begin

Self.imgFav := TsImage.Create(Self);

Self.imgFav.Parent := Self;

with imgFav do

Begin

Stretch := true;

Height := 30;

Width := 33;

Left := 1050;

Top := slblBrand.Top - 5;

Images := FormMain.lstImgCartNFav;

ImageIndex := 1;

OnClick := WhenClickedFav;

End;

End;

procedure SetbtnBuyCar();

Begin

Self.sbtnBuyCar := TsButton.Create(Self);

Self.sbtnBuyCar.Parent := Self;

with sbtnBuyCar do

Begin

Caption := 'ADD TO CART';

Font.Size := 16;

Font.Name := 'Calibri';

Font.Style := [fsBold];

Height := 45;

Width := slblFuelType.Left - slblPower.Left + 40;

Left := slblBrand.Left + 30;

Top := 175;

OnClick := btnAddOnClick;

End;

End;

procedure PaintComps();

Var

i:Integer;

Begin

Self.OnMouseEnter := pnlOnMouseEnter;

Self.OnMouseLeave := pnlOnMouseLeave;

for i := 0 to Self.ComponentCount - 1 do

begin

if Self.Components[i] is TsLabel then

(Self.Components[i] as TsLabel).OnMouseEnter := pnlOnMouseEnter;

if Self.Components[i] is TsButton then

(Self.Components[i] as TsButton).OnMouseEnter := pnlOnMouseEnter;

if Self.Components[i] is TsImage then

(Self.Components[i] as TsImage).OnMouseEnter := pnlOnMouseEnter;

end;

End;

procedure SetShowMore();

Begin

Self.slblShowMore := TsLabel.Create(Self);

Self.slblShowMore.Parent := Self;

with slblShowMore do

Begin

Caption := 'Show more';

Font.Size := 16;

Font.Style := [fsBold];

Font.Color := clBlue;

Font.Name := 'Calibri';

Left := slblPrice.Left;

Top := sbtnBuyCar.Top + 10;

OnMouseEnter := slblShowOnMouseEnter;

OnMouseLeave := slblShowOnMouseLeave;

OnClick := slblShowOnClick;

End;

End;

Begin

pnlOnCreate();

SetInfo(CarInfo);

SetBrand();

SetModel();

SetYear();

SetMileage();

SetColor();

SetPower();

SetTransm();

SetDriveUnit();

SetFuelType();

SetCarPhoto();

SetPrice();

SetimgFav();

SetbtnBuyCar();

SetContacts();

PaintComps();

SetShowMore();

End;

procedure TCarPanel.WhenClickedFav(Sender: TObject);

Begin

if Self.imgFav.ImageIndex = 1 then

Begin

Self.imgFav.ImageIndex := 2;

AddCarToList(ListCarFav, LinkedCar);

Inc(FavsAmount);

End

Else

Begin

Self.imgFav.ImageIndex := 1;

DeleteCarFromList(ListCarFav, LinkedCar);

Dec(FavsAmount);

End;

End;

procedure TCarPanel.pnlOnMouseEnter(Sender: TObject);

Begin

slblBrand.UseSkinColor := false;

slblModel.UseSkinColor := false;

End;

procedure TCarPanel.pnlOnMouseLeave(Sender: TObject);

Begin

slblBrand.UseSkinColor := true;

slblModel.UseSkinColor := true;

End;

procedure TCarPanel.slblShowOnMouseEnter(Sender: TObject);

Begin

slblShowMore.UseSkinColor := false;

slblBrand.UseSkinColor := false;

slblModel.UseSkinColor := false;

End;

procedure TCarPanel.slblShowOnMouseLeave(Sender: TObject);

Begin

slblShowMore.UseSkinColor := true;

End;

//

procedure TCarPanel.slblShowOnClick(Sender: TObject);

var

i: Integer;

Begin

if slblShowMore.UseSkinColor = false then

Begin

Application.CreateForm(TFormCarCreate, FormCarCreate);

for i := 0 to FormCarCreate.ComponentCount - 1 do

Begin

if FormCarCreate.Components[i] is TEdit then

(FormCarCreate.Components[i] as TEdit).Enabled := false;

if FormCarCreate.Components[i] is TsComboBox then

(FormCarCreate.Components[i] as TsComboBox).Enabled := false;

End;

FormCarCreate.ActiveControl := nil;

with FormCarCreate do

begin

pnlControlCreate.Visible := false;

VertScrollBar.Range := VertScrollBar.Range - 55;

btnPhotoInp.Visible := false;

btnPhotoSave.Visible := false;

imgDeletePhoto.Enabled := false;

stxtLoadPic.Visible := false;

bbtnSlideRight.Visible := true;

bbtnSlideLeft.Visible := true;

sbtnEditAd.Visible := true;

sbtnCancelChange.Visible := true;

sbtnSaveChange.Visible := true;

sbtnCancelChange.Enabled := false;

sbtnSaveChange.Enabled := false;

if Self.Owner = FormMain.scrlboxCars then

sbtnDeleteCar.Visible := true;

SetInfo(LinkedCar);

end;

FormCarCreate.ShowModal;

End;

End;

//

procedure TCarPanel.btnAddOnClick(Sender: TObject);

Begin

if sbtnBuyCar.Caption = 'ADD TO CART' then

begin

sbtnBuyCar.Caption := 'IN CART';

sbtnBuyCar.Down := true;

AddCarToList(ListCarCart, LinkedCar);

Inc(CartAmount);

sbtnBuyCar.OnMouseEnter := btnAddInCartOnEnter;

sbtnBuyCar.OnMouseLeave := btnAddInCartOnLeave;

end

else

begin

sbtnBuyCar.Caption := 'ADD TO CART';

sbtnBuyCar.Down := false;

FormMain.ActiveControl := nil;

DeleteCarFromList(ListCarCart, LinkedCar);

Dec(CartAmount);

sbtnBuyCar.OnMouseEnter := NothingToDo;

sbtnBuyCar.OnMouseLeave := NothingToDo;

end;

End;

procedure TCarPanel.btnAddInCartOnEnter(Sender: TObject);

begin

slblBrand.UseSkinColor := false;

slblModel.UseSkinColor := false;

sbtnBuyCar.Down := false;

end;

procedure TCarPanel.btnAddInCartOnLeave(Sender: TObject);

begin

sbtnBuyCar.Down := true;

end;

procedure TCarPanel.NothingToDo(Sender: TObject);

begin

//Nothing :)

end;

procedure TCarPanel.SetInfo(CarInfo: TCar);

begin

Self.LinkedCar := CarInfo;

with Self do

Begin

slblBrand.Caption := CarInfo.Brand;

slblModel.Caption := CarInfo.Model;

slblYear.Caption := IntToStr(CarInfo.Year) + ' y.';

slblCarColor.Caption := CarInfo.Color;

slblMileage.Caption := IntToStr(CarInfo.Mileage) + ' km';

slblPrice.Caption := IntToStr(CarInfo.Price) + '$';

slblPower.Caption := FloatToStr(CarInfo.Power) + ' h.p.';

slblTransm.Caption := CarInfo.Transm;

slblDriveUnit.Caption := CarInfo.DriveUnit;

slblFuelType.Caption := CarInfo.Fuel;

slblContacts.Caption := CarInfo.Contacts;

imgCarPhoto.Picture.LoadFromFile(CarInfo.Images[1]);

End;

end;

procedure Register;

begin

RegisterComponents('Cars', [TCarPanel]);

end;

end.

unit Magazine;

interface

Type

TImageArr = array [1..20] of string[255];

Function RealToStr(Num: Extended): string;

Function ValidInp(StrToCheck: string; var NumRes: Integer): Boolean; overload;

Function ValidInp(StrToCheck: string; var NumRes: Real): Boolean; overload;

Procedure ToShiftArrLeft(var ImageArr: TImageArr; var PicturesAm, indDeletedEl: Integer);

Procedure GetPhotosAmount(var ImageArr: TImageArr; var Amount: Integer);

Function TransmConvert(str: string): Integer;

Function FuelTypeConvert(str: string): Integer;

Function DriveUnitConvert(str: string): Integer;

implementation

Uses

Dialogs, SysUtils;

Function TransmConvert(str: string): Integer;

Begin

if str = 'Automatic' then

Result := 0;

if str = 'Manual' then

Result := 1;

if str = 'Robotic' then

Result := 2;

End;

Function FuelTypeConvert(str: string): Integer;

Begin

if str = 'Petrol' then

Result := 0;

if str = 'Diesel' then

Result := 1;

if str = 'Electro' then

Result := 2;

End;

Function DriveUnitConvert(str: string): Integer;

Begin

if str = 'Front-wheel' then

Result := 0;

if str = 'Rear-wheel' then

Result := 1;

if str = 'Plug-in all-wheel' then

Result := 2;

if str = 'Permanent all-wheel' then

Result := 3;

End;

Function RealToStr(Num: Extended): string;

var

i: Integer;

Begin

Result := FloatToStr(Num);

for i := 1 to Length(Result) do

if Result[i] = ',' then

Result[i] := '.';

End;

Function ValidInp(StrToCheck: string; var NumRes: Integer): Boolean; overload;

Var

ErrorPos: Integer;

Begin

Val(StrToCheck, NumRes, ErrorPos);

if ErrorPos <> 0 then

Begin

Result := false;

End

Else

Result := true;

End;

Function ValidInp(StrToCheck: string; var NumRes: Real): Boolean; overload;

Var

ErrorPos: Integer;

Begin

Val(StrToCheck, NumRes, ErrorPos);

if ErrorPos <> 0 then

Begin

Result := false;

End

Else

Result := true;

End;

Procedure ToShiftArrLeft(var ImageArr: TImageArr; var PicturesAm, indDeletedEl: Integer);

Var

i: Integer;

Begin

if PicturesAm <> indDeletedEl then

Begin

for i := indDeletedEl to PicturesAm - 1 do

ImageArr[i] := ImageArr[i + 1];

ImageArr[PicturesAm] := '';

End

Else

Begin

ImageArr[indDeletedEl] := '';

dec(indDeletedEl);

End;

Dec(PicturesAm);

End;

Procedure GetPhotosAmount(var ImageArr: TImageArr; var Amount: Integer);

Var

i, Size: Integer;

Begin

Size := 0;

for i := 1 to 20 do

if not String(ImageArr[i]).IsEmpty then

Inc(Size);

Amount := Size;

End;

end.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | | *Дополнитель-ные сведения* | |
|  | | | | | Текстовые документы | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР КП 1–40 01 01 360 ПЗ | | | | | Пояснительная записка | | | | 104 с. | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | Графические документы | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.051004-30 СА | | | | | Удаление элемента из списка. | | | | Формат А1 | |
|  | | | | | Схема алгоритма | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  | БГУИР КП I-40 01 01 360 ПЗ | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм | Лист | № докум. | Подп. | Дата | Программное средство «Электронный каталог автомобилей»  Ведомость курсового  проекта | Литера | | | Лист | Листов |
| Разраб. | | Яковлев В.Ю. |  | 01.06 |  | Т |  | 104 | 104 |
| Провер. | | Фадеева Е.П. |  | 05.06 | Кафедра ПОИТ  гр.051004 | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |