|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**КУРСОВОЙ ПРОЕКТ (РАБОТА)**

по дисциплине: Объектно-ориентированное программирование

по профилю: Прикладная информатика

направления профессиональной подготовки: Прикладная информатика, бакалавриат

Тема: «Программа для управления продажей автомобилей.»

Студент (ф.и.о. полностью): Качура Иван Андреевич

Группа: ИНБО-04-18

Работа представлена к защите\_\_\_\_\_\_\_\_\_\_\_(дата)\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись и ф.и.о. студента)

Руководитель: Ассистент кафедры ИиППО Хлебникова В.Л

Работа допущена к защите\_\_\_\_\_\_\_\_\_\_\_(дата)\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись ф.и.о. руководителя)

Оценка по итогам защиты: \_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. МИРЭА. 2019г.

Оглавление

[Глава 1. Техническое задание 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682224)

[1.1. Введение 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682225)

[1.1.1. Наименование программы 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682226)

[1.1.2. Назначение и область применения 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682227)

[1.2. Требования к программе или программному изделию 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682228)

[1.2.1. Требования к функциональным характеристикам 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682229)

[1.3. Условия эксплуатации 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682231)

[1.3.1. Требования к составу и параметрам технических средств 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682232)

[1.3.2. Требования к информационной и программной совместимости 5](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682233)

[1.3.2.1. Требования к исходным кодам и языкам программирования 6](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682234)

[1.3.2.2. Требования к программным средствам, используемым программой 6](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682235)

[1.4. Требования к программной документации 6](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682237)

[1.4.1. Предварительный состав программной документации 6](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682238)

[1.5. Стадии и этапы разработки 6](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682239)

[1.5.1. Стадии разработки 6](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682240)

[1.5.2. Этапы разработки 6](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682241)

[1.5.3. Содержание работ по этапам 7](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682242)

[1.6. Порядок контроля и приемки 8](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682243)

[1.6.1. Виды испытаний 8](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682244)

[1.6.2. Общие требования к приемке работы 8](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682245)

[Глава 2. Проектирование и разработка программы 8](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682246)

[2.1. Проектирование программы 8](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682247)

[2.1.1. Функциональные требования к программе 8](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682248)

[2.1.2. Проектирование классов 8](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682249)

[2.2. Разработка программы 8](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682250)

[2.2.1. Описание проектного решения 9](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682251)

[2.2.2. Тестирование программы 14](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682252)

[Глава 3. Руководство пользователя 15](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682253)

[Заключение 18](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682254)

[Список литературы 19](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682255)

[Приложение №1. Исходный код программы 20](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682256)

[Приложение №2. UML-диаграмма 44](file:///D:\USER\Desktop\Отчёт.docx#_Toc470682257)

**Цель работы:**

Целью работы является разработать программу, для заказа автомобиля в Автосалоне, с использованием функции поиска по заданным критериям. Реализовать функцию бронирования автомобиля и гибкую систему скидок для постоянных покупателей.

# Введение:

В ходе выполнения Курсовой работы должна быть реализована программа для покупки, бронирования автомобилей, а так же система лояльности для постоянных покупателей с помощью языка Объективно-Ориентированного Программирования C++, интегрированной среды разработки CLION. Язык C++ — компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование.

Программное предложение «Программа для управления продажей автомобилей» представляет собой систему, с помощью которой можно приобрести автомобиль, осуществить предварительное бронирование понравившегося автомобиля, а так же учесть скидку постоянного покупателя.

# Глава 1. Техническое задание

## Введение

### Наименование программы

Наименование программы – «Программа для управления продажей автомобилей».

### Назначение и область применения

«Программа для управления продажами автомобилей» заказа автомобиля в Автосалоне, с использованием функции поиска по заданным критериям поиска с гибкой системой скидок.

Требования к программе или программному изделию

### Требования к функциональным характеристикам

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

1. Функцию выбора действий, которые нужно выполнить.
2. Функции показа денежных показателей и характеристик автомобиля.
3. Функции запоминания клиентов и прошлых покупок.
4. Функции выхода из программы после завершения работы.

## Условия эксплуатации

### Требования к составу и параметрам технических средств

В состав технических средств должен входить IВМ-совместимый персональный компьютер (ПЭВМ), включающий в себя:   
1. Процессор Pentium-2.0Hz, не менее;   
2. Оперативную память объемом, 1 Гигабайт, не менее;   
3. Любую операционную систему;   
4. Стандартный пакет С++;

### Требования к информационной и программной совместимости

#### Требования к исходным кодам и языкам программирования

Исходные коды программы должны быть реализованы на языке C++. В качестве интегрированной среды разработки программы должна быть использована среда Qt.

#### Требования к программным средствам, используемым программой

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией любой операционной системы и ПО C++.

## Требования к программной документации

### Предварительный состав программной документации

Состав программной документации должен включать в себя:

1. Техническое задание;
2. Программу и методики испытаний;
3. Руководство пользователя.

## Стадии и этапы разработки

### Стадии разработки

Разработка должна быть проведена в три стадии:

1. Разработка технического задания;
2. Исследование предметной области;
3. Рабочее проектирование;
4. Внедрение.

### Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. Разработка программы;
2. Разработка программной документации;
3. Испытания программы.

На стадии внедрения должен быть выполнен этап разработки - подготовка и передача программы.

### Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. Постановка задачи;
2. Определение и уточнение требований к техническим средствам;
3. Определение требований к программе;
4. Определение стадий, этапов и сроков разработки программы и документации на неё;
5. Согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями к составу документации.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1. Разработка, согласование и утверждение программы и методики испытаний;
2. Проведение приемо-сдаточных испытаний;
3. корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию.

## Порядок контроля и приемки

### Виды испытаний

Демонстрация работы программы и ее функционала.

### Общие требования к приемке работы

Программа должна соответствовать всем вышесказанным требованиям.

# Глава 2. Проектирование и разработка программы

## Проектирование программы (Полещук)

### Функциональные требования к программе

Функциональные требования к программе описаны в пункте № 1.2 Технического задания.

## Проектирование классов (Швецов)

Для реализации проекта будет использован язык программирования C++. Данный язык является одним из самых наиболее распространённых объектно-ориентированных языков программирования (ООП).

Целью данной работы является написание приложения игры в “Симулятор студента”. Для удобства пользования все действия будут реализованы через кнопки, находящиеся в графическом окне, разработанном при помощи фреймворка Qt. При нажатии одной из кнопок, будет выполняться написанное на ней действие и изменения характеристик игрока и времени будут выводиться в этом же окне.

## Разработка программы

## Описание проектного решения

Avto – класс, содержащий название, цену, максимальную скорость, год выпуска автомобиля, баланс, наличие бронирования, скидку, а также функции поиска по названию, году выпуска, максимальной скорости, функции бронирования, отмены бронирования, бронирования со скидкой, получения баланса.

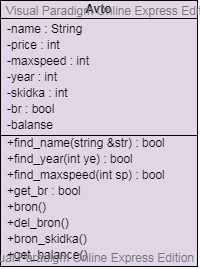


Рис. 1.1 UML диаграмма класса Avto

# Глава 3. Руководство пользователя

Запуск программы осуществляется через исполняемый файл .exe.

После запуска программы откроется главное меню, в котором можно выбрать одну из функций.

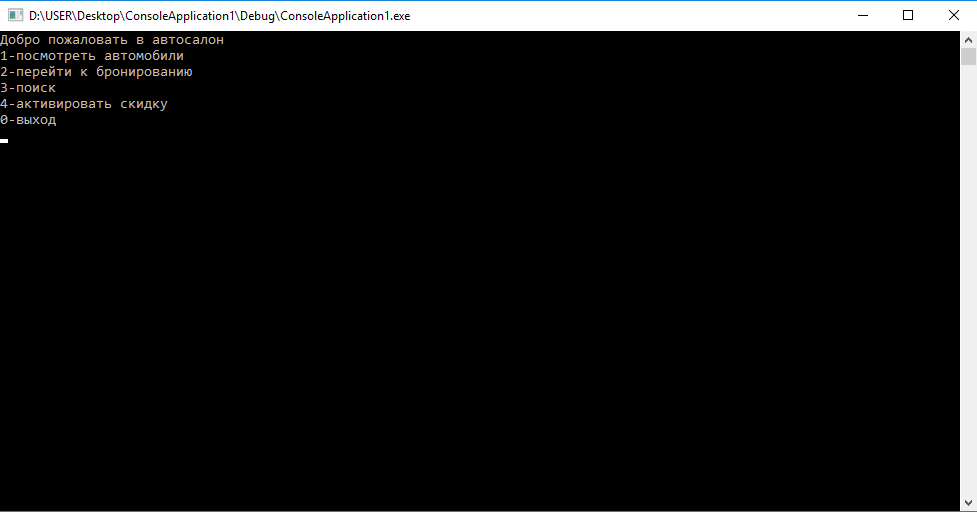


Рис.2.1 Скриншот окна “Основного меню”.

Если требуется посмотреть Список Автомобилей, то необходимо нажать клавишу на клавиатуре “1”.

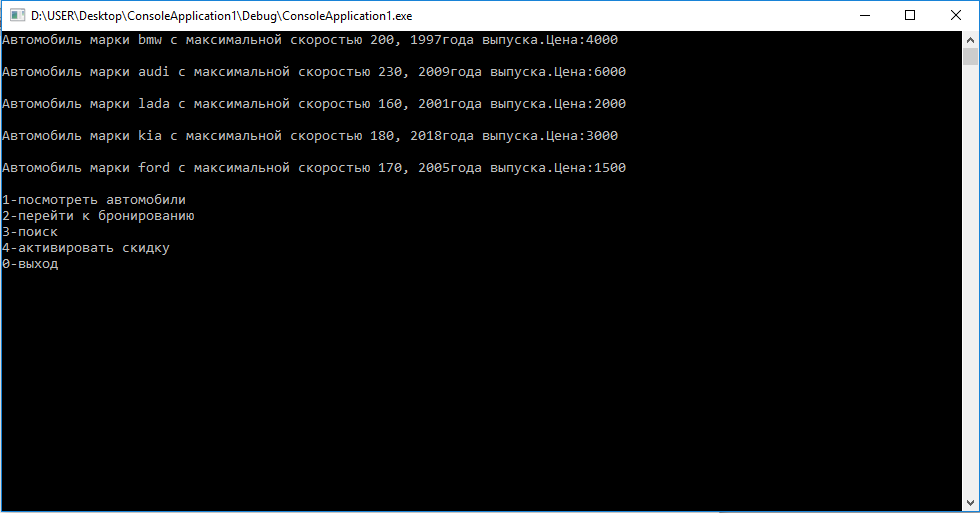


Рис.2.2 Скриншот окна “Список автомобилей ”.

Если вы захотели забронировать автомобиль для покупки, то потребуется нажать клавишу на клавиатуре “2”.

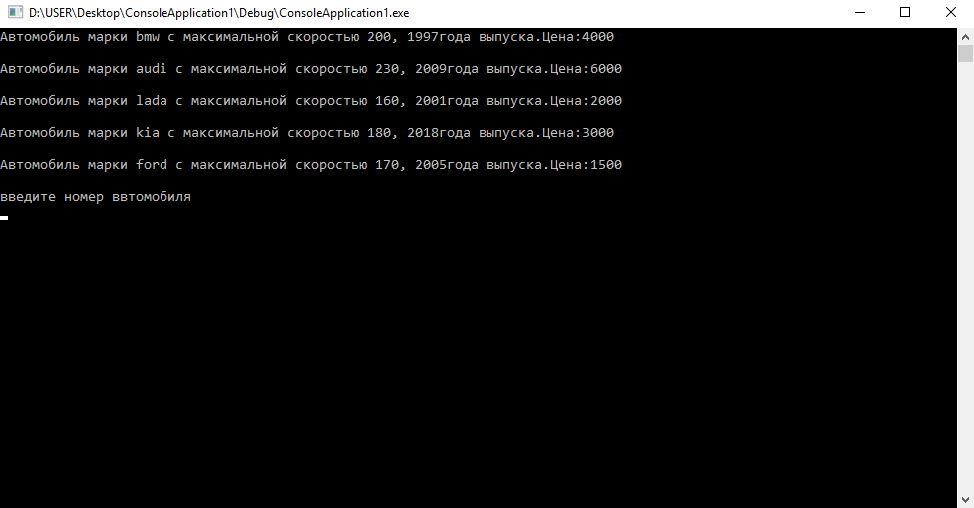


Рис.2.3 Скриншот окна “Список автомобилей для бронирования”.

После бронирования Автомобиля, в левом верхнем углу покажет ваш баланс.

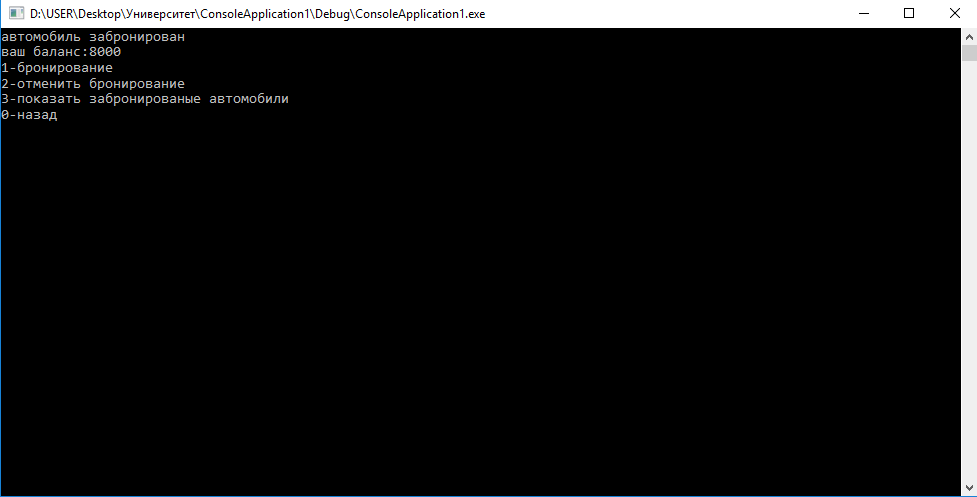


Рис.2.4 Скриншот окна “Список забронированных Автомобилей”.

Нажав клавишу “3” на клавиатуре, вы сможете посмотреть список ваших забронированных автомобилей.

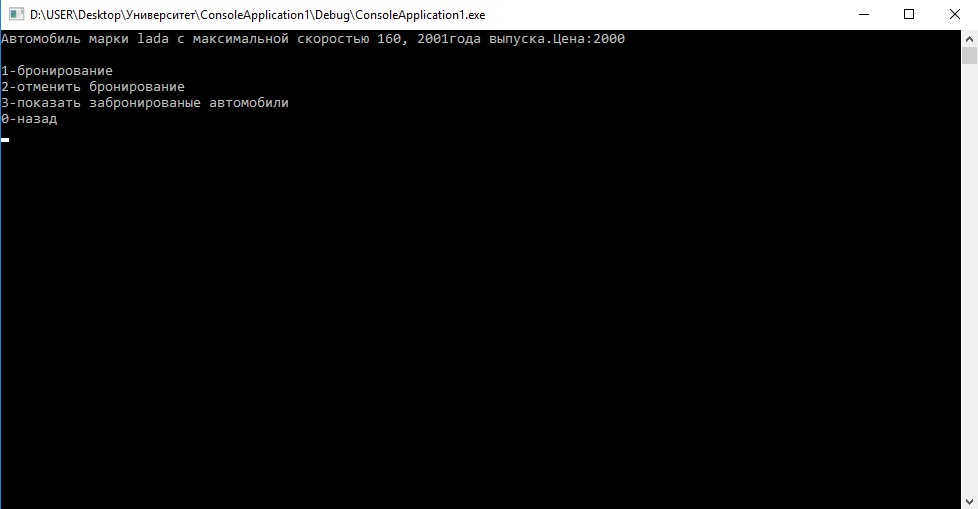


Рис.2.5 Скриншот окна “Список забронированных Автомобилей”.

Вернувшись в главное меню, вы сможете Найти автомобиль нажав на клавиатуре клавишу “3” – “Поиск" и выбрав различные виды поиска, найти интересующий вас Автомобиль.

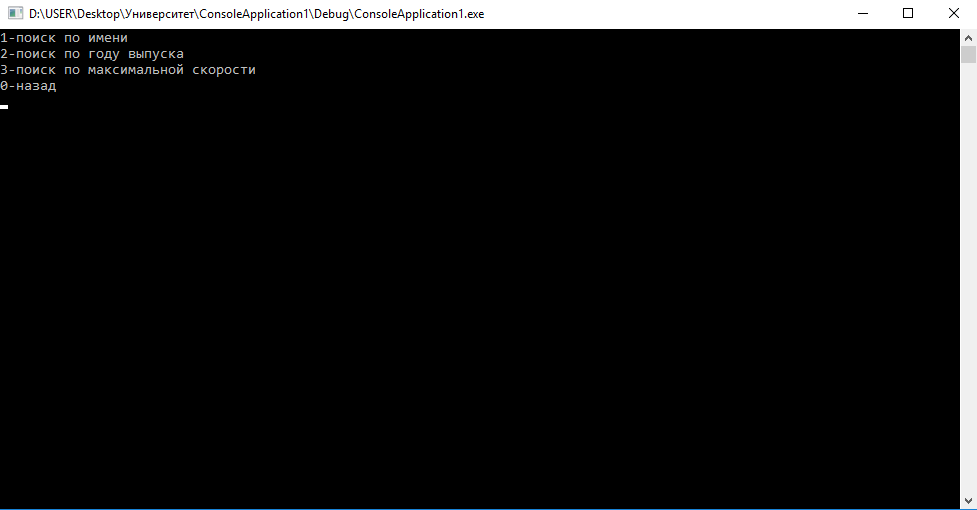


Рис.2.6 Скриншот окна “Виды поиска интересующего Автомобиля”.

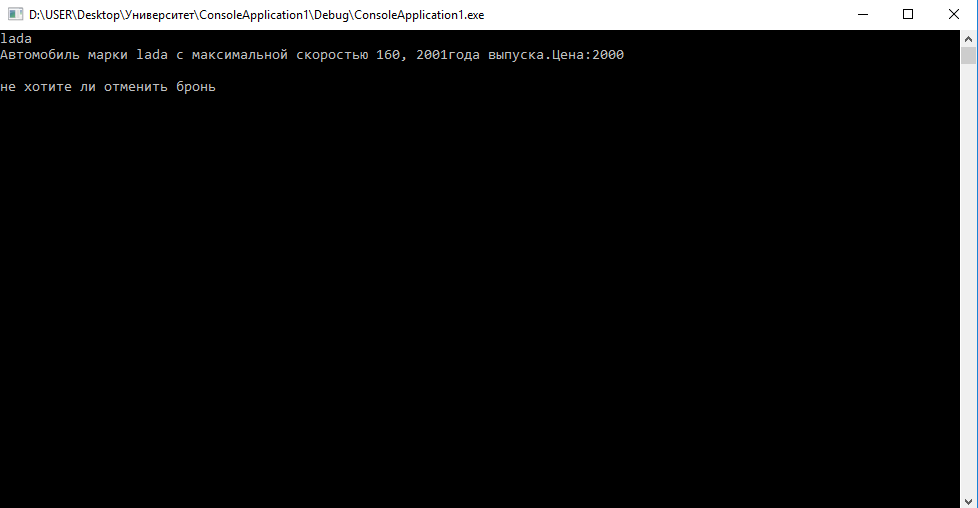


Рис.2.7 Скриншот окна “Поиск Автомобиля по названию”.

Покинуть программу вы сможете закрыв Консольное приложение или нажав клавишу “0” на клавиатуре.



# Заключение

# В результате выполнения курсовой работы была полностью реализована программа «Программа для управления продажей автомобилей», используя объектно-ориентированное программирование. Написание программы способствовало закреплению теоретического материала на практических занятиях.

# В ходе выполнения курсовой работы были более подробно изучены базовые элементы управления, их свойства и методы.

# 

# Список литературы

1. А. Александреску. Современное программирование на C++. Обобщенное программирование и прикладные шаблоны проектирования. Книга для опытных программистов не С++. 2002 год, 330 стр.
2. Т. А. Павловская. C/C++. Программирование на языке высокого уровня. Из серии "300 лучших учебников".2003 год. 461 стр.
3. Прата Стивен. Язык программирования С++. Лекции и упражнения. Учебник. 2005 год. 1100 стр.
4. Бланшет, Саммерфилд - Qt4 Программирование GUI на С++. 2ed. 2008
5. Шлее Макс - Профессиональное программирование на C++. +CD. Qt 4.8. 2012
6. Марк Саммерфилд - Qt Профессиональное программирование (High tech). 2011

# Приложение №1. Исходный код программы

Avto.cpp

|  |
| --- |
| #include "Avto.h" |
|  | bool skidka = false; |
|  | int schet = 0; |
|  | using namespace std; |
|  |  |
|  | fstream file("file.txt"); |
|  | Avto::Avto() |
|  | { |
|  |  |
|  | file >> name; |
|  | file >> maxspeed; |
|  | file >> year; |
|  | file >> price; |
|  | br = false; |
|  | } |
|  |  |
|  |  |
|  | Avto::~Avto() |
|  | { |
|  | } |
|  |  |
|  | bool Avto::find\_name(string& str) |
|  | { |
|  | if (name == str)return true; |
|  | else |
|  | return false; |
|  | } |
|  |  |
|  | bool Avto::find\_year(int ye) |
|  | { |
|  | if (year == ye)return true; |
|  | else |
|  | return false; |
|  | } |
|  |  |
|  | bool Avto::find\_maxspeed(int sp) |
|  | { |
|  | if (maxspeed == sp)return true; |
|  | else |
|  | return false; |
|  | } |
|  |  |
|  | void Avto::bron() |
|  | { |
|  | if (br == false) { |
|  |  |
|  |  |
|  | if (balance >= price) { |
|  | schet++; |
|  | balance -= price; |
|  |  |
|  |  |
|  | br = true; |
|  |  |
|  | cout << "àâòîìîáèëü çàáðîíèðîâàí" << endl; |
|  | cout << "âàø áàëàíñ:"<<get\_balance() << endl; |
|  |  |
|  | } |
|  | else cout << "íåäîñòàòî÷íî ñðåäñòâ" << endl; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | } |
|  | else cout << "ýòîò àâòîìîáèëü óæå çàáðîíèðîâàí" << endl; |
|  | } |
|  |  |
|  | void Avto::del\_bron() |
|  | { |
|  | if (br == true) { |
|  | br = false; |
|  | balance += (price / 2); |
|  | cout << "áðîíü îòìåíåíà" << endl; |
|  | cout << "âàø áàëàíñ:" << get\_balance() << endl; |
|  | } |
|  | else cout << "âû íå áðîíèðîâàëè ýòîò àâòîìîáèëü" << endl; |
|  | } |
|  |  |
|  | bool Avto::get\_br() |
|  | { |
|  |  |
|  | return br; |
|  | } |
|  |  |
|  | int Avto::get\_balance() |
|  | { |
|  | return balance; |
|  | } |
|  |  |
|  | void Avto::bron\_skidka() |
|  | { |
|  | if (br == false) { |
|  |  |
|  |  |
|  | if (balance >= (price/2)) { |
|  | schet++; |
|  | balance -= (price/2); |
|  |  |
|  |  |
|  | br = true; |
|  |  |
|  | cout << "àâòîìîáèëü çàáðîíèðîâàí" << endl; |
|  | cout << "âàø áàëàíñ:" << get\_balance() << endl; |
|  |  |
|  | } |
|  | else cout << "íåäîñòàòî÷íî ñðåäñòâ" << endl; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | } |
|  | else cout << "ýòîò àâòîìîáèëü óæå çàáðîíèðîâàí" << endl; |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  | std::ostream& operator<<(std::ostream& out, const Avto& avto) |
|  | { |
|  | out <<"Àâòîìîáèëü ìàðêè " << avto.name << " c ìàêñèìàëüíîé ñêîðîñòüþ " << avto.maxspeed << ", " << avto.year << "ãîäà âûïóñêà." << "Öåíà:" << avto.price << endl; |
|  |  |
|  | return out; |
|  | } |

Avto.h

|  |
| --- |
| #pragma once |
|  |  |
|  | #include <iostream> |
|  | #include <fstream> |
|  | #include <string> |
|  | using namespace std; |
|  | static int balance=10000; |
|  | extern bool skidka; |
|  | extern int schet; |
|  | class Avto |
|  | { |
|  | string name; |
|  | int price, maxspeed, year, skidka; |
|  | bool br; |
|  |  |
|  | public: |
|  | Avto(); |
|  |  |
|  | ~Avto(); |
|  | friend std::ostream& operator<< (std::ostream& out, const Avto& avto); |
|  | bool find\_name(string &str); |
|  | bool find\_year(int ye); |
|  | bool find\_maxspeed(int sp); |
|  | void bron(); |
|  | void del\_bron(); |
|  | bool get\_br(); |
|  | static int get\_balance(); |
|  | void bron\_skidka(); |
|  |  |
|  |  |
|  | }; |

ConsoleApplication1.cpp

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include "Avto.h" |
|  | using namespace std; |
|  | void broni(Avto\* p, int n); |
|  | void find(Avto\* p, int n); |
|  | int main() |
|  | { |
|  | setlocale(LC\_ALL, "Russian"); |
|  | int n; |
|  | n = 5; |
|  | Avto\* p = new Avto[n]; |
|  | //cout << skidka; |
|  | cout << "Добро пожаловать в автосалон" << endl; |
|  | int k; |
|  | do{ |
|  | cout << "1-посмотреть автомобили" << endl; |
|  | cout << "2-перейти к бронированию" << endl; |
|  | cout << "3-поиск" << endl; |
|  | cout << "4-активировать скидку" << endl; |
|  | cout << "0-выход" << endl; |
|  | cin >> k; |
|  | system("cls"); |
|  | switch (k) |
|  | { |
|  | case 1: { |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << p[i] << endl; |
|  | } |
|  | break; |
|  | } |
|  | case 2: { |
|  | broni(p, n); |
|  | } |
|  | break; |
|  | case 3: { |
|  | find(p, n); |
|  | } |
|  | case 4: { |
|  | //schet++; |
|  | //skidka = true; |
|  | //cout << schet <<" "<< skidka << endl; |
|  |  |
|  | if (schet > 5) { |
|  | cout << "вы активировали скидку 50%" << endl; |
|  | skidka = true; |
|  | } |
|  | else cout << "недостаточно бронирований" << endl; |
|  | }break; |
|  | default: |
|  | break; |
|  | } |
|  |  |
|  | } while (k != 0); |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  | void broni(Avto\* p, int n) |
|  | { |
|  | int k; |
|  | do { |
|  | cout << "1-бронирование" << endl; |
|  | cout << "2-отменить бронирование" << endl; |
|  | cout << "3-показать забронированые автомобили" << endl; |
|  | cout << "0-назад" << endl; |
|  | cin >> k; |
|  | system("cls"); |
|  | switch (k) |
|  | { |
|  | case 1: { |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << p[i] << endl; |
|  | } |
|  | cout << "введите номер ввтомобиля" << endl; |
|  | int z; |
|  | cin >> z; |
|  | system("cls"); |
|  | if (!skidka) |
|  | p[z - 1].bron(); |
|  | else p[z - 1].bron\_skidka(); |
|  |  |
|  | }break; |
|  | case 2: { |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | cout << p[i] << endl; |
|  | } |
|  | cout << "введите номер ввтомобиля" << endl; |
|  | int z; |
|  | cin >> z; |
|  | system("cls"); |
|  | p[z-1].del\_bron(); |
|  | }break; |
|  | case 3: { |
|  | int k = 0; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  |  |
|  | if (p[i].get\_br()) { |
|  | cout << p[i] << endl; |
|  | k++; |
|  | } |
|  |  |
|  | } |
|  | if (k == 0)cout << "нет забронированных автомобилей" << endl; |
|  | }break; |
|  | default: |
|  | break; |
|  | } |
|  | } while (k != 0); |
|  | } |
|  |  |
|  | void find(Avto\* p, int n) |
|  | { |
|  | int k; |
|  | do { |
|  | cout << "1-поиск по имени" << endl; |
|  | cout << "2-поиск по году выпуска " << endl; |
|  | cout << "3-поиск по максимальной скорости" << endl; |
|  | cout << "0-назад" << endl; |
|  | cin >> k; |
|  | system("cls"); |
|  | switch (k) |
|  | { |
|  | case 1: { |
|  | string na; |
|  | cin >> na; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | if (p[i].find\_name(na)) { |
|  | cout << p[i] << endl; |
|  | if (!p[i].get\_br()) { |
|  | cout << "не хотите ли забронировать этот автомобиль" << endl; |
|  | int q; |
|  | cin >> q; |
|  | if (q == 1)if (!skidka) |
|  | p[i ].bron(); |
|  | else p[i ].bron\_skidka();; |
|  | } |
|  | else { |
|  | cout << "не хотите ли отменить бронь" << endl; |
|  | int q; |
|  | cin >> q; |
|  | if (q == 1)p[i].del\_bron(); |
|  | } |
|  |  |
|  | } |
|  |  |
|  | } |
|  | }break; |
|  | case 2: { |
|  | int na; |
|  | cin >> na; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | if (p[i].find\_year(na)) { |
|  | cout << p[i] << endl; |
|  | if (!p[i].get\_br()) { |
|  | cout << "не хотите ли забронировать этот автомобиль" << endl; |
|  | int q; |
|  | cin >> q; |
|  | if (q == 1)if (!skidka) |
|  | p[i ].bron(); |
|  | else p[i ].bron\_skidka();; |
|  | } |
|  | else { |
|  | cout << "не хотите ли отменить бронь" << endl; |
|  | int q; |
|  | cin >> q; |
|  | if (q == 1)p[i].del\_bron(); |
|  | } |
|  | } |
|  | } |
|  | }break; |
|  | case 3: { |
|  | int na; |
|  | cin >> na; |
|  | system("cls"); |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  | if (p[i].find\_maxspeed(na)) { |
|  | cout << p[i] << endl; |
|  | if (!p[i].get\_br()) { |
|  | cout << "не хотите ли забронировать этот автомобиль" << endl; |
|  | int q; |
|  | cin >> q; |
|  | if (q == 1)if (!skidka) |
|  | p[i ].bron(); |
|  | else p[i ].bron\_skidka();; |
|  | } |
|  | else { |
|  | cout << "не хотите ли отменить бронь" << endl; |
|  | int q; |
|  | cin >> q; |
|  | if (q == 1)p[i].del\_bron(); |
|  | } |
|  | } |
|  | } |
|  | }break; |
|  |  |
|  | default: |
|  | break; |
|  | } |
|  | } while (k != 0); |
|  |  |
|  | } |

# Приложение №2. UML-диаграмма

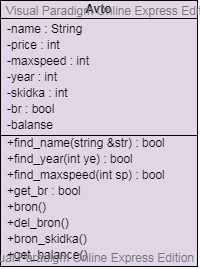


Рис. 3.1 Общая UML диаграмма