Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_\_Б. В. Никульшин

(подпись)

––"–––––––––––––––––2022г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Тюшкевичу Максиму Андреевичу.*

1. Тема проекта *Сервис аренды автомобилей.*

2. Срок сдачи студентом законченного проекта *10 декабря 2022 г.*

3. Исходные данные к проекту *Файлы: PetrolCars – содержит информацию о бензиновых автомобилях, DieselCars – бензиновые автомобили, ElectricCars –электрические автомобили, HybridCars – гибридные автомобили. Users – информация о пользователях. Admin – информация об администраторах. CarInUsage – содержит недоступные автомобили. Session – содержит историю заказов. UserAuthentication и AdminAuthentication – содержимт информацию о паролях и логинов для аккаунтов. Контейнер: STL – vector.*

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

*Введение. Содержание. 1. Постановка задачи 2. Структурное проектирование. 3. Функциональное проектирование. 4. Описание классов. 5. Разработка алгоритмов. 5.1 Разработка алгоритмов по шагам. 5.1 Разработка схем алгоритмов. 6. Заключение. Список использованной литературы.*

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

*1. Диаграмма классов*

*2. Схема алгоритма метода* *void ShowSpecifiCarSessions()*

*3. Схема алгоритма метода**bool UserAuthentication(User&)*

6. Консультант по проекту *Марзалюк А. В*.

7. Дата выдачи задания *16 сентября 2022 г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*разделы 1, 2 к 10 октября 2022 г. – 20 %;*

*разделы 3, 4 к 10 ноября 2022 г. – 40 %;*

*раздел 5 к 10 декабря 2022 г. – 20 %;*

*оформление пояснительной записки и графического материала к 10 декабря 2022 г. – 20 %*

*Защита курсового проекта с 19 декабря 2022 г. по 30 декабря 2022 г.*

РУКОВОДИТЕЛЬ– –––––––––*Марзалюк А. В.*

Задание принял к исполнению –Х *Тюшкевич М. А.*

СОДЕРЖАНИЕ

[ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ 5](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303619)

[ВВЕДЕНИЕ 6](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303620)

[1 ОБЗОР ИСТОЧНИКОВ](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303621) 8

1.1 [Анализ аналогов программного средства 8](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303622)

1.2 [Постановка задачи](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303624) 8

2 [СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303625) 9

[2.1 Структура программы](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303626) 9

[2.2 Структура входных данных](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303627) 9

3 [ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303628) 11

3.1 [Диаграмма классов](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303628) 11

[3.2 Обзор используемых классов](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303628) 11

4 РАЗРАБОТКА АЛГОРИТМОВ 19

4.1 [Алгоритмы по шагам](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303628) 19

4.2 Схемы алгоритмов20

5 [РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303628) 21

[ЗАКЛЮЧЕНИЕ](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303628) 23

[Список использованной литературы](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303629) 24

[Приложение А Диаграмма классов](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303630) 25

[Приложение Б Схема алгоритма метода ShowSpecificCarSessions()](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303631) 26

[Приложение В Схема алгоритма метода UserAuthentication(User&)](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303632) 27

[Приложение Г Листинг кода программы](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303633) 28

[Приложение Д Ведомость документов](file:///D:\1ФЛЕШКА\Курсовая\Варианты%20курсачей\Сахарук\Мое(О%20Господи(Чё))детище(да)(Содержание).docx#_Toc450303634) 29

**ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ**

ID – идентификационный номер.

ОС – операционная система.

ООП – Объектно-ориентированное программирование.

STL – (*Standard Template Library*) –Библиотека стандартных шаблонов.

**ВВЕДЕНИЕ**

В наше время стали очень востребованны различные сервисы аренды. Многие из нас пользовались сервисами краткосрочной аренды самокатов и велосипедов, например «Whoosh» или «Jet». Существуют и подобные сервисы для краткосрочной аренды автомобилей, например «Hello» или «Anytime». Особенность таких сервисов в том, что они предоставляют бюджетные автомили для поездок в пределах города, что формирует стоимость в зависимости от пройденного расстояния. Из этого вытекает потребность людей в другой форме аренды автомобилей – традиционных компаниях, предлагающих подобную услугу. Их отличие от вышеописанных сервисов в том, что они могут предлагать автомобили более высокого класса и на более выгодных условиях, чем и создают спрос у тех людей, которые желают взять автомобиль на больший срок и по более выгодной цене.

Современные сервисы аренды автомобилей могут предоставлять огромное множество услуг. Это и аренда автомобилей абсолютно любого класса, от бюджетных малолитражек до представительских автомобилей самого высокого уровня, возможность наема водителя, круглосуточная техническая поддержка, консультация по всем вопросам.

Объектно-ориентированное программирование представляет собой парадигму, в которой программа предствляется в виде связанных между собой объектов и методов, описывающих их поведение. Данная парадигма вытекла из необходимости писать более расширяемый и поддерживаемый код.

С++ — компилируемый, статически типизируемый язык общего назначения. Преимущество языка C++ в том, что он жестко не привязан к объектно-ориентированному программированию и позволяет писать программы в различных стилях, например в процедурном. Данная особенность обусловлена тем, что язык является развитием языка С и, следовательно, полностью поддерживает его и позволяет работать его командами и конструкциями. Язык предоставляет и возможность работы на низком уровне (ассемблерные вставки, управление памятью).

Язык С++ имеет огромное количество стандартых функций и возможностей. Это и стандартная библиотека шаблонов (STL), потоки ввода-вывода, поддержка многопоточности, регулярные выражения.

Основными концепциями ООП являются инкапсуляция, полиморфизм и наследование.

Инкапсуляция – механизм сокрытия и защиты данных, исключающий плияние на них извне класса. Помогает скрывать данные, чтобы предотвратить нежелательные последствия и предотвратить будущие проблемы.

Наследование – механизм, позволяющий расширять и дополнять созданные классы и объекты. Позволяет не писать или переписывать старые классы, а просто дополнять уже существующие. Мощный инструмент для расширения программ.

Полиморфизм – механизм, позволяющий сущностям изменять свое поведение в замисимости от класса. Как и наследование, позволяет не переписывать существующий код, а дополнять его и расширять возможности.

Язык С++ предоставляет широкие возможности по реализации описанных концепций. Грамотное использование данных принципов позволяет создавать надежный, защищенный и легко расширяемый код, что является неоспоримым плюсом языка.

Универсальность и гибкость языка С++ позволяют использовать его в самых разных сферах. Инструменты языка необходимы в написании высокопроизводительных систем и программ. Возможность работы на низком уровне востребована в написании драйверов и частей операционных систем. Язык широко использвуется в игровой индустрии, являясь основным языком для многих современных игр.

Хотя язык C++ и является достаточно сложным для изучения, он является мощнейшим инструментом, позволяющим решить как вышеописанные задачи, так и еще огромный круг проблем, которые могут возникнуть в нынешнем мире высоких технологий.

Используя все то, что было приведено выше, считаю, что язык С++ более чем подходит для написания курсового проекта, так как в нем широко используются ООП-концепции и возможности, предоставляемые языком.

**1 ОБЗОР ИСТОЧНИКОВ**

**1.1 Анализ аналогов программного средства**

Сервисы аренды возникли практически одновременно с широким распространением автомобилей, и в течении следующего времени они только расширялись и предлагали все больше услуг и возможностей.

На сегодняшиний день одними из ведущих сервисов являются Europcar.com (доступен в более чем 50 странах), Alamo.com (84 страны), DiscoverCars.com (более 145 стран). Эти сервисы работают в основном в Европе, Северной Америке и Азии.

Среди белорусских сервисов можно выделить Westgroup.by и Car-rent.by, предоставляющие широкий выбор автомобилей любого класса.

На всех этих платформах доступен выбор даты поездки, расчет стоимости, выбор места получения автомобиля, существует множество акций и скидок, позволяющих экономить и получать выгодные предложения.

* 1. **Постановка задачи**

Реализация проекта будет использована интегрированныя среда разработки CLion и система сборки проектов CMake.

В программе будут реализованы функции для пользователя:

– Просмотр доступных автомобилей.

– Выбор автомобиля для аренды.

– Оформление заказа.

– Возможность редактирования информации об аккаунте.

Для администратора будут реализованы функции:

– Возможность просмотра всех данных автомобилей.

– Возможность просмотра всех пользователей.

– Возможность просматривать все заказы, заказы определенного автомобиля или пользователя.

– Добавление и удаление автомобилей.

Для реализации данного программного обеспечения будет использоватся объектно-ориентированный язык программирования C++ (стандарт C++14) и операционная система macOS Ventura 13.0.1.

**2 CИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

**2.1 Структура программы**

В структуре программы можно выделить несколько основных модулей: модуль интерфейса, модуль обработки ошибок и модуль функционирования программы, блок работы с базой данных автомобилей.

Блок интерфейса предназначен для взаимодействия пользователя с программой. Пользователь может быть как рядовым потребителем, так и администратором.

Модуль функционирования программы включает в себя все те механизмы, позволяющие комфортно взаимодействать с программой и выполнять требуемые задачи.

Модуль обработки ошибок предназначен для борьбы с возможными исключениями и обеспечивает надежную работу всей программы.

Модуль работы с базой данных автомобилей предназначен для сериализации объектов различных типов автомобилей, поскольку в языке C++ прямая запись объектов в файл может в определенных ситуациях привести к ошибкам чтения и потери доступа к части информации.

**2.2 Структура файлов программы**

Вся информация будет хранится в бинарных файлах.

Файлы, содержащие информацию о различных типах автомобилей имеют крайне схожую структуру, запись и чтение из них происходит через класс Database, служащий для корректного считывания и записи в файлы.

Данные файлы имеют названия DieselCars.bin (содержит информацию об автомобилях с дизельным двигателем), ElectricCars.bin (об электромобилях), HybridCars.bin (о гибридных автомобилях), PetrolCars.bin (об автомобилях с бензиновым двигателем).

Далее рассмотрим их структуру и поля. Некоторые поля могут содержатся только в некоторых файлах, если об этом ничего не сказано, то следует считать, что поле содержится во всех файлах.

– int id. ID автомобиля.

– bool deleteMark. Удален ли автомобиль.

– wchar\_t brand[30], mode[30], country[30], bodyType[30]. Марка, модель, страна и тип кузова.

­– int manufacturedYear. Год выпуска автомобиля.

– int numberOfSeats. Количество мест.

– int maxSpeed. Максимальная скорость автомобиля.

– float consumption, volume. Расход топлива и объем двигателя. Содержится во всех файлах, кроме ElectricCars.bin.

– wchar\_t interior.color[30], interior.material[30]. Цвет и материал интерьера автомобиля.

– wchar\_t color[30]. Цвет автомобиля.

– int costPerDay. Цена аренды автомобиля на 1 день.

– int petrolType. Минимально допустимая марка бензина для данного автомобиля. Содержится в файле PetrolCars.bin.

– int batteryCapacity. Объем батареи. Содержится в файлах ElectricCars.bin и HybridCars.bin.

– wchar\_t fuelType[8]. Тип топлива. Содержится в файле HybridCars.bin.

Для реализации функций регистрации и входа в аккаунт были созданны файлы User.bin, UserAuthentication.bin, Admin.bin, AdminAuthentication.bin.

Файлы User.bin и Admin.bin содержат информацию о пользователях и администраторах соответственно. Файлы UserAuthentication.bin и AdminAuthentication.bin хранят логины и пароли от аккаунтов.

Файлы User.bin и Admin.bin соответствующие объекты классов, содержащие следующие общие поля:

– int id. ID автомобиля.

– wchar\_t name[30] и surname[30]. Имя и фамилия пользователя.

– int age. Возраст пользователя.

Файл User.bin содержит так же свои собственные поля:

– wchar\_t passportNum[15]. Содержит номер паспорта пользователя.

– wchar\_t status[10]. Содержит статус пользователя в скидочной программе.

– float count. Содержит текущий баланс аккаунта.

– float spendMoney. Содержит количество потраченных средств.

– wchar\_t phoneNumber[15]. Телефонный номер пользователя.

– wchar\_t adress.country[30]. Страна проживания пользователя.

– wchar\_t adress.city[30]. Город проживания пользователя.

– wchar\_t adress.street[30]. Улица проживания пользователя.

– int adress.house[30]. Номер дома проживания пользователя.

– int adress.flat[30]. Номер квартиры проживания пользователя.

Файлы UserAuthentication.bin и AdminAuthentication.bin содержат одинаковые данные:

– wchar\_t login[30]. Содержит логин.

– wchar\_t password[30]. Содержит пароль.

– int id. ID пользователя, которому принадлежат логин и пароль.

Файл Session.bin содержит информации о заказах в виде объектов класса Session и имеет поля:

– int userID. ID заказчика.

– int carID. ID автомобиля.

– int DateFrom.day, DateFrom.month, DateFrom.year. Дата начала заказа.

– int DateTo.day, DateTo.month, DateTo.year. Дата окончания заказа.

Файлы CarId.bin и PersonID.bin содержат число типа int, соответствующее последнему идентификационному номеру, выданному классам Car и его наследникам и классам Person и его наследникам соответственно.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

**3.1 Диаграмма классов**

Диаграмма классов является важнейшим элементом любого проекта. Она позволяет графически представить связь классов между собой, избегая просмотра кода всей программы, который может быть очень объемным.

Диаграмма классов программы представлена в приложении А.

**3.2 Обзор используемых классов**

**3.2.1 Класс Car**

Поскольку главной услугой сервиса аренды автомобилей, очевидно, является аренда автомобилей, то в программе реализовано 6 классов. Базовым классом для всех остальных является класс Car, от него наследуются классы OilCar и ElectricCar. В свою очередь от класса OilCar наследуются классы PetrolCar и DieselCar. От классов OilCar и ElectricCar одновременно наследуется класс HybridCar, представляющий гибридные автомобили.

Классы Car и OilCar не предназначены для создания объектов соответствующих классов.

Поля класса Car:

– int id. ID автомобиля.

– bool deleteMark. Метка о том, удален ли автомобиль.

– wchar\_t brand[30], model[30], country[30], bodyType[30]. Марка, модель, страна и тип кузова.

– int manufacturedYear. Год выпуска автомобиля.

– int numberOfSeats. Количество мест.

– float maxSpeed. Максимальная скорость автомобиля.

– wchar\_t interior.color, interior.material. Цвет и материал интерьера автомобиля.

– wchar\_t color. Цвет автомобиля.

– int costPerDay. Цена аренды автомобиля на 1 день.

Методы класса Car:

– Car(). Конструктор.

– ~Car(). Деструктор.

– virtual void SetInfo(). Заполнение полей класса.

– bool IsDelete(). Проверяет, удален ли автомомбиль.

– void DeleteCar(). Удаляет автомомбиль.

– int GetID(). Возвращает ID.

– int GetManufacturedYear(). Возвращает год выпуска.

– wchar\_t\* GetBrand(). Возвращает марку.

– wchar\_t\* GetModel(). Возвращает модель.

– wchar\_t\* GetCountry(). Возвращает страну производства.

– wchar\_t\* GetBodyType(). Возвращает тип кузова.

– int GetNumberOfSeats(). Возвращает количество мест

– float GetMaxSpeed(). Возвращает максимальную скорость.

– wchar\_t\* GetInteriorColor(). Возвращает цвет интерьера.

– wchar\_t\* GetInteriorMaterial(). Возвращает материал интерьера.

– wchar\_t\* GetColor(). Возвращает цвет автомобиля.

– int GetCostPerDay(). Возвращает стоимость аренды на один день.

**3.2.2 Класс ElectricCar**

Описывает электромобили.

Поля класса ElectricCar:

– int batteryCapacity. Емкость батареи электромобиля.

Методы класса ElectricCar:

­– ElectricCar(). Конструктор.

– ~ElectricCar(). Деструктор.

– void SetInfo(). Заполнение полей класса.

– int GetBatteryCapacity(). Возвращает емкость батареи.

**3.2.3 Класс HybridCar**

Описывает гибридные автомобили.

Поля класса HybridCar:

– wchar\_t fuelType[8].

Методы класса HybridCar:

– HybridCar(). Конструктор.

– ~HybridCar(). Деструктор.

– void SetInfo(). Заполнение полей класса.

– wchar\_t\* GetFuelType(). Возвращает тип топлива.

**3.2.4 Класс OilCar**

Является родительским для DieselCar и PetrolCar. Описывает автомобили с двигателем внутреннего сгорания.

Поля класса OilCar:

– float consumption. Расход топлива.

– float volume. Объем двигателя.

Методы класса OilCar:

– OilCar()

– ~OilCar()

– void SetInfo(). Заполнение полей класса.

– float GetConsumtion(). Возвращает расход топлива.

– float GetVolume(). Возвращает объем двигателя.

**3.2.5 Класс DiseselCar**

Описывает автомобили с дизельным двигателем.

Методы класса DieselCar:

– DieselCar(). Конструктор.

– ~ DieselCar(). Деструктор.

– void SetInfo(). Заполнение полей класса.

**3.2.6 Класс PetrolCar**

Описывает автомобили с бензиновым двигателем.

Поля класса PetrolCar:

– int petrolType. Минимальная марка бензина.

Методы класса PetrolCar:

– PetrolCar ()

– ~ PetrolCar ()

– void SetInfo(). Заполнение полей класса.

– float GetPetrolType(). Возвращает минимальную марку бензина.

**3.2.7 Класс Person**

Класс Person и его наследники Admin и User предназначены для описания аккаунтов людей, которые взаимодействуют с программой.

Поля класса Person:

– int id. ID пользователя.

– wchar\_t name[30]. Имя пользователя.

– wchar\_t surname[30]. Фамилия пользователя.

– int age. Возраст пользователя.

Методы класса Person:

– Person(). Конструктор.

– ~Person(). Деструктор.

– int GetID(). Возвращает ID пользователя.

– wchar\_t\* GetName(). Возвращает имя пользователя.

– wchar\_t\* GetSurname(). Возвращает фамилию пользователя.

– int GetAge(). Возвращает возраст пользователя.

– virtual void SetInfo(). Заполнение полей класса.

**3.2.8 Класс Admin**

Описывает администратора.

Методы класса Admin:

– Admin(). Конструктор.

– ~Admin(). Деструктор.

– void SetInfo(). Заполнение полей класса.

Поля класса User:

– wchar\_t passportNum[15]. Номер паспорта.

– wchar\_t status[10]. Статус пользователя.

– float count. Баланс.

– float spendMoney. Количество потраченных средств.

– wchar\_t phoneNumber[15]. Номер телефона пользователя.

– adress.country[30]. Страна проживания пользователя.

– adress.city[30]. Город проживания пользователя.

– adress.street[30]. Улица проживания пользователя.

– adress.house[30]. Номер дома проживания пользователя.

– adress.flat[30]. Номер квартиры проживания пользователя.

**3.2.9 Класс User**

Описывает пользователя.

Методы класса User:

– User(). Конструктор.

– ~User(). Деструктор.

– void SetInfo(). Заполнение полей класса.

– void SetNewNumber(). Установка нового номера телефона.

– void SetNewAdress().Установка нового адреса.

– void SetNewPassport(). Установка нового номера паспорта.

– void SetCount(float plus). Изменение баланса аккаунта.

– void SetSpendMoney(float). Возвращает количество потраченных средств.

– void SetStatus(wchar\_t ). Устанавливает статус.

– wchar\_t\* GetPassportNum(). Возвращает номер паспорта.

– wchar\_t\* GetStatus(). Возвращает статус.

– float GetCount(). Возвращает баланс.

– float GetSpendMoney(). Возвращает количество потраченных средств.

– wchar\_t\* GetPhoneNumber(). Возвращает номер телефона.

– wchar\_t\* GetCountry(). Возвращает страну проживания.

– wchar\_t\* GetCity(). Возвращает город проживания.

– wchar\_t\* GetStreet(). Возвращает улицу проживания.

– int GetHouse(). Возвращает номер дома проживания.

– int GetFlat(). Возвращает номер квартиры проживания.

**3.2.10 Класс Interface**

Является самым масштабным классом проекта и реализовывает взаимодействие пользователя с программой. Условно разделен на потребительскую, администраторскую и общую части.

Поля класса:

– vector<PetrolCar> petrolCars. Все автомоибили типа PetrolCar.

– vector<DieselCar> dieselCars. Все автомоибили типа DieselCar.

– vector<ElectricCar> electricCars. Все автомоибили типа ElectricCar.

– vector<HybridCar> hybridCars. Все автомоибили типа HybridCar.

– vector<int> carIdInUsage. ID недоступных автомобилей.

– Session newSession. Новый заказ.

– vector<Session> allSessions. Все заказы.

– User user. Пользователь.

– Admin admin. Администратор.

– Методы общей части класса:

– Interface(). Конструктор.

– ~Interface(). Деструктор.

– void FirstMenu(). Выводит на экран меню входа и считывает выбор.

– void SeeUserHistory(int). Выводит на экран все заказы определенного пользователя.

Методы пользовательской части класса:

– void UserMainMenu(). Выводит на экран меню пользователя и считывает выбор.

– void Registration(). Осуществляет регистрацию нового пользователя.

– void UserAccountInfo(). Выводит информацию об аккауте.

– void SeeCars(). Показывает все автомобили, доступные для заказа

– void CarOrder(int). Осуществляет заказ автомобилей.

– void ShowDetailedInfo(int, int). Показывает подробную информацию об автомобиле.

– bool ShowCarsForChoosing(int). Служит для выбора автомобиля для заказа.

– void AddCount(). Пополняет баланс аккаунта.

– int FindCost(int, int). Служит для поиска стоимости автомобиля при создании заказа.

– void RefreshUserInFile(). При завершении программы, обновляет файл с пользователями, чтобы сохранить изменения.

– void SetAccountSettings(). Предназначен для изменения информации об аккаунте.

Методы администраторской части класса:

– void AdminMainMenu(). Выводит на экран меню администратора и считывает выбор.

– void ShowAllSessions(). Показывает абсолютно все заказы.

– void ShowAllUsers(). Показывает всех пользователей, зарегистрированных в программе.

– void ShowAllCars(). Показывает все автомобили.

– void ShowSpecificUserSessions(). Показывает все заказы конкретного автомобиля.

– void DeleteSpecificCar(). Удаление автомобиля из базы данных.

– void RefreshAllCars(). Обновляет файлы с автомобилями, что бы сохранить изменения.

– void AddNewCar(). Добавление нового автомобиля в базу данных.

**3.2.11 Класс CarInUsage**

Предназначен для контроля автомобилей, которые недоступны для заказа.

Поля класса CarInUsage:

– int lastDay. Хранит последний день аренды в виде юлианской даты.

– int carID. Содержит ID автомобиля, взятого в аренду.

Мотоды класса CarInUsage:

– CarInUsage(). Конструктор.

– CarInUsage(int, int). Конструктор с параметрами

– ~CarInUsage(). Деструктор.

– void WriteInFile(). Записывает недоступные автомобили в файл CarInUsage.bin.

– void RefreshFile(). Проверяет даты окончания и удаляет из файла автомобили, у которых закончилась аренда.

– void GetCarsInUsage(vector<int>&). Считывает в вектор идентификационные номера автомобилей, недоступных для заказа.

**3.2.12 Класс Authentication**

Предназначен для реализации функции входа в аккаунт.

Поле класса Authentication:

– Userdata user. Объект входа.

Методы класса Authentication:

­– Authentication(). Конструктор.

– ~Authentication(). Деструктор.

– bool AdminAuthentication(Admin&). Предназначен для аутентификации администратора.

– bool UserAuthentication(User&). Предназначен для аутентификации пользователя.

**3.2.13 Класс Userdata**

Предназначен для работы с логином и паролем от аккаунта.

Поля класса Userdata:

– wchar\_t login[30]. Содержит логин аккаунта.

– wchar\_t password[30]. Содержит пароль аккаунта.

– int id. Содержит ID аккаунта, которому принадлежат логин и пароль.

Методы класса Userdata:

– Userdata(). Конструктор.

– ~Userdata(). Деструктор.

– void SetInfo(int). Осуществляет чтение логина и пароля.

– void SetData(wchar\_t\*, wchar\_t\*). Служит для установки логина и пароля.

– bool CheckAccess(int). Проверяет, существуют введенные логин и пароль.

­– wchar\_t\* GetLogin(). Возвращает логин.

– wchar\_t\* GetPassword(). Возвращает пароль.

– static bool CheckLogin(wchar\_t\*). Проверяет логин, что бы не был введен уже существующий логин.

– static void ChangeLogin(int). Осуществляет смену логина.

– static void ChangePassword(int). Осуществляет смену пароля.

**3.2.14 Класс CarBrand**

Предназначен для выбора марки автомобиля, при добавлении нового автомобиля.

Поля класса CarBrand:

– wchar\_t brand[30]. Содержит марку автомобиля.

– wchar\_t country[30]. Содержит страну автомобиля.

Методы:

– CarBrand(). Конструктор.

– CarBrand(). Деструктор.

– void GetCarBrands(). Выводит список марок автомобилей.

– wstring GetCountry(int). Возращает страну, в зависимости от выбранного автомобиля.

– wstring GetBrand(int). Возращает марку, в зависимости от выбранного автомобиля.

**3.2.15 Класс Database**

Предназначет для чтения и записи в файлы объектов классов PetrolCar, DieselCar, ElectricCar и HybridCar. Не содержит полей.

Методы класса Database:

– static void AddInDatabase(DieselCar&). Запись в файл.

– static void AddInDatabase(PetrolCar&). Запись в файл.

– static void AddInDatabase(ElectricCar&). Запись в файл.

– static void AddInDatabase(HybridCar&). Запись в файл.

– static void GetFromDatabase(vector<DieselCar>&). Чтение из файла в ветктор.

– static void GetFromDatabase(vector<PetrolCar>&). Чтение из файла в ветктор.

– static void GetFromDatabase(vector<ElectricCar>&). Чтение из файла в ветктор.

– static void GetFromDatabase(vector<HybridCar>&). Чтение из файла в ветктор.

– static void DeleteFromDatabase(DieselCar&). Помечает объект как удаленный.

– static void DeleteFromDatabase(PetrolCar). Помечает объект как удаленный.

– static void DeleteFromDatabase(ElectricCar&). Помечает объект как удаленный.

– static void DeleteFromDatabase(HybridCar&). Помечает объект как удаленный.

**3.2.16 Класс Session**

Предназначен для создания заказов.

Поля класса Session:

– int userID. Содержит идентификатор заказчика.

– int carID. Содержит идентификатор взятого автомобиля.

– int DateFrom.day, DateFrom.month, DateFrom.year. Содержат дату начала заказа.

– int DateTo.day, DateTo.month, DateTo.year. Содержат дату окончания заказа.

Методы класса Session:

– Session(). Конструктор.

­– ~Session(). Деструктор.

– bool CreateSession(int, int, int, User&). Предназначет для создания заказа.

– static void GetAllSessions(vector<Session>&). Записывает в переданный вектор все заказы из файла.

– bool CheckDate(int). Проверяет дату, что бы она была верной.

– bool CheckDate(int, int). Переопределение предыдущей функции

– int JDDate(int, int, int). Переводит дату в юлианску дату.

– int NowDays(). Возвращает текущую дату в виде юлианской.

– bool InputDateCheck(int, int, int). Проверяет корректность ввода даты.

**3.2.17 Класс IdGenerator**

Предназначен для установки идентификационных номеров для классов Car и его наследников и классов Person и его наследников. Не содержит полей.

Методы класса IdGenerator:

– static int GetIdForCar(). Возращает ID для автомобиля.

– static int GetIdForPerson(). Возращает ID для человека

**4 РАЗРАБОТКА АЛГОРИТМОВ**

**4.1 Алгоритмы по шагам**

**4.1.1 Алгоритм по шагам метода bool CreateSession(int, int, int, User&) – создание заказа автомобиля.**

1. Начало.
2. Входные данные:

userID – ID заказчика;

carID – ID автомобиля;

costPerDay – стоимость аренды автомобиля на день;

user – объект класса User, содержит информацию о заказчике.

3. Проверятся, не равна ли costPerDay нулю, поскольку стоимость 0 означает, что автомобиль удален. Если не равна, то переход в пункт 4, если равна то переход в пункт 20.

4. Полям this->userID и this->carID присваиваются значения carID и userID.

5. Пользователь вводит дату начала заказа.

6. Введенная дата преобразуется в юлианскую дату.

7. Если дата введена неверно, то переходим в пункт 5, иначе в пункт 8.

8. Если количество дней между нынешней датой и введенной больше 3, то переход в пункт 20, иначе в пункт 9.

9. Пользователь вводит дату окончания заказа.

10. Если дата введена неверно, то нужно переход в пункт 9, иначе в пункт 11.

11. Расчет количества дней, на которое арендуется автомобиль. Если разница между началом и окончанием заказа 0, то стоимость будет браться за 1 день.

13. Расчет стоимости заказа.

14. Проверяется, достаточно ли у пользователя средств, если нет, то переход в пункт 20, иначе переход в пункт 15.

15. К расчитанной стоимости заказа применяются скидки, в зависимости от статуса пользователя.

16. Запрашивается подтверждение заказа.

17. Если пользователь отказался, то переход в пункт 20, иначе вычитается стоимость заказа с баланса пользователя и арендованный автомобиль заносится в список недоступных для заказа и переход в пункт 19.

18. Информация о заказе записывается в файл.

19. Возращаем true и переходим в пункт 21.

20. Возвращаем false и переходим в пункт 21.

21. Конец.

**4.1.2 Алгоритм по шагам метода void ChangeLogin(int) – изменение логина аккунта.**

1. Начало.

2. Входные данные:

id – ID пользователя.

3. Создается новый объект newUserdata типа Userdata.

4. Открывается для чтения файл UserAuthentication.bin

5. Считывается информацию из файла в объект newUserdata.

6. Проверяется ID считанных данных, если считанный ID совпал с id, переданным в качестве параметра, то переход в пункт 6, иначе переход в пункт 4.

7. Закрытие файла.

8. Пользователь вводит новый логин. Обработка ошибок, если пользователь ввел пустую строку, либо строку с пробелами, то ввод повторяется.

9. Проверятся доступность логина методом CheckLogin(login).

10. Если логин недоступен, то переход в пункт 8.

11. Методом SetData() устанавливатся объекту newUserdata новый логин.

12. Открывается файл UserAuthentication.bin для чтения и создается временный файл tmp.bin для записи.

13. Создается временный объект tmp типа Userdata, служащий в качестве буфера.

14. Из файла UserAuthentication.bin считываются объекты типа Userdata используя объект tmp, пока не достигнут конец файла.

15. Проверяется, равны ли ID объекта tmp и ID объекта newUserdata. Если совпадают, то в файл tmp.bin записывается объект newUserdata, иначе tmp.

16. Закрывается файл UserAuthentication.bin он удаляется.

17. Закрывается файл tmp.bin и он переименовывается в UserAuthentication.bin.

18. Конец.

**4.2 Схемы алгоритмов**

Схема алгоритма метода void ShowSpecificCarSessiom() приведена в приложении Б. – показывает все заказы определенного автомомбиля.

Схема алгоритма метода bool UserAuthentication(User&) приведена в приложении В. – аутентификация пользователя в программе.

**5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Для использования программы в качестве пользователя необходимо сначала зарегистрироваться. В процессе регистрации задаются логин и пароль. Они не должны содержать пробелы и не могут являться пустыми строками. Логин и пароль можно сменить в любой момент. На рисунке 5.1 главное меню программы.

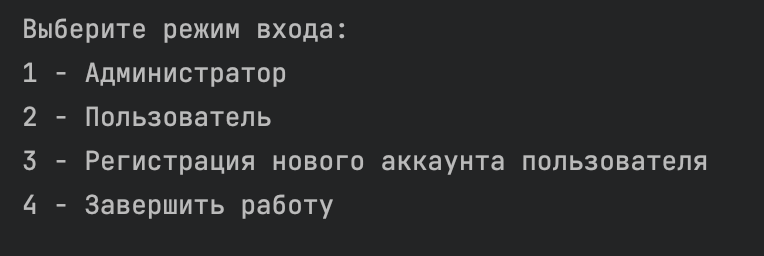


Рисунок 5.1 – Главное меню программы

В случае, если аккаунт уже был создан, достаточно просто войти.

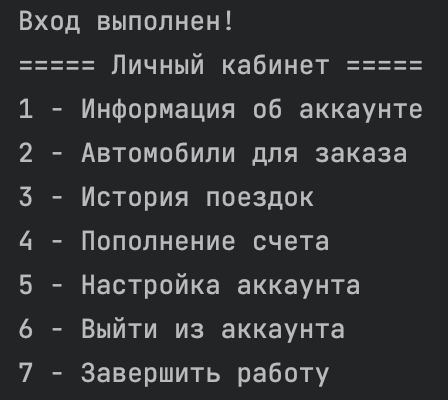


Рисунок 5.2 – Главное меню пользователя

Все пункты в меню программы выбираются с помощью цифр и нажатия клавиши Enter. Неверный ввод обрабатывается.

Пользователь выбирает тип желаемого автомобиля и выбирает сам автомобиль из соответствующего списка. В конце у пользователя запрашивается подтверждение заказа. На рисунке 5.3 представлен пример заказа автомобиля.

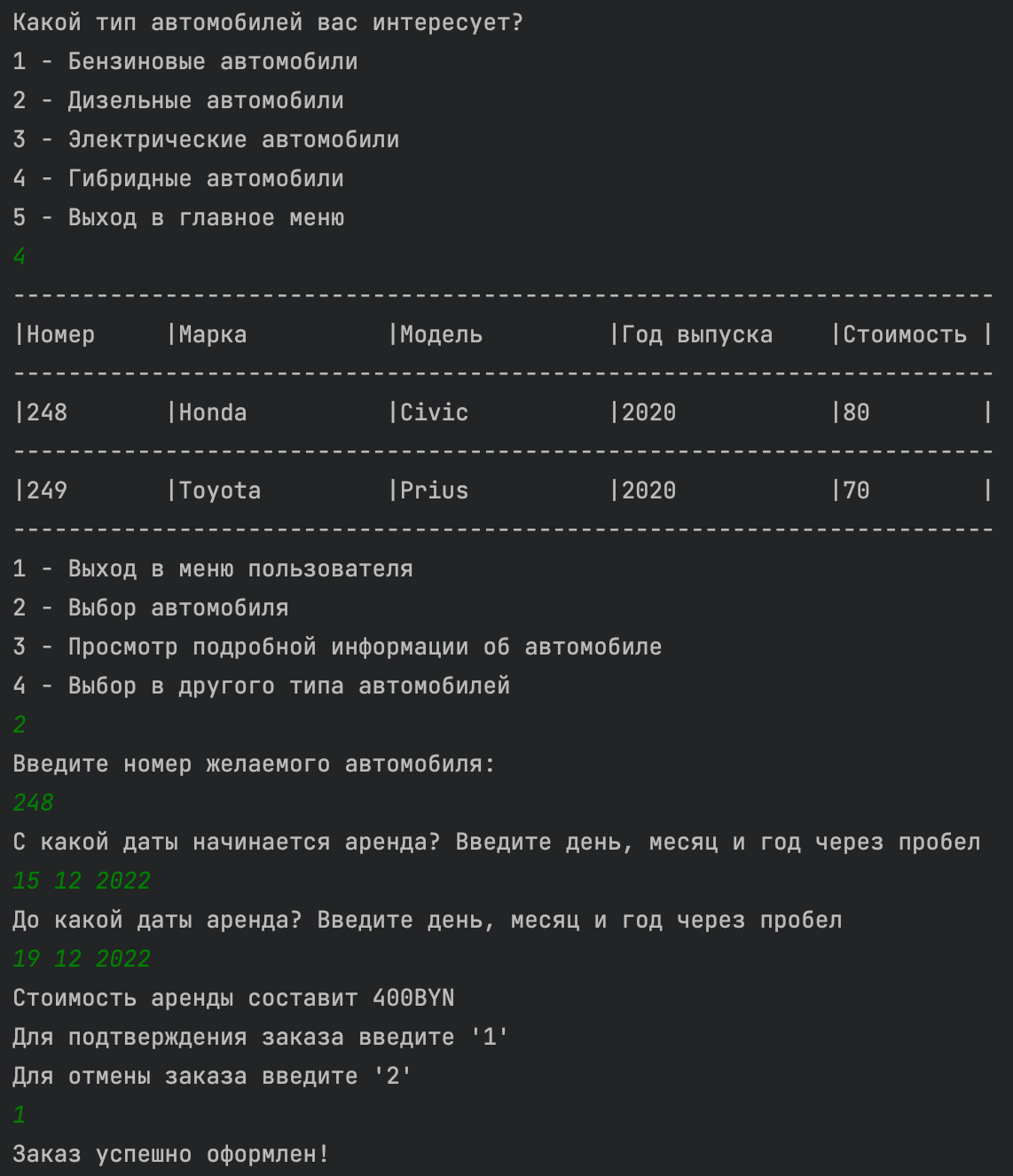
**

Рисунок 5.3 – Оформление заказа

На рисунке 5.4 показан пример просмотра прошлых заказов.

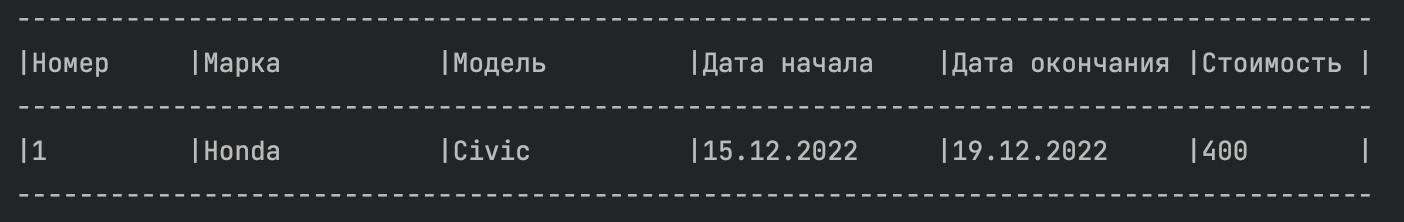


Рисунок 5.4 – Просмотр прошлых заказов

**ЗАКЛЮЧЕНИЕ**

В ходе работы над курсовым проектом был разработан сервис аренды автомобилей, который имеет понятный текстовый пользовательский интерфейс и отображает основные возможности такого рода сервисов.

Было изучено и успешно использовано большое количество текстовой и графической информации, в результате чего стало возможным проектирование программного продукта, его тестирование и устранение ошибок.

Были получены практические знания о технологиях написания программ, создании их архитектуры и поддержки. Были дополненны знания об основных конструкциях языка и способах их использования.

Все эти знания без сомнения пригодятся в будущем обучении и помогут дальше развиваться в выбранном направлении.

Данный проект может быть усовершенствован и расширен в следующих местах:

– Добавление поддержки систем управденияя баз данных для хранения и долнения информации.

– Разработка графического интерфейса для любой платформы.

­– Разработка мобильного приложения и web-версии приложения.

– Добавление локализации интерфейса для различных стран.

– Добавление возможности аренды других транспортных средств, например мотоциклов или коммерческого транспорта, например автобусов.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

[1] Герберт, Ш. Самоучитель C++/Ш. Герберт. Санкт-Петербург 2003. – 845 с.

[2] Дейтел, Х.М. Как программировать на С++ / Х.М. Дейтел, П.Д. Дейтел; пер. с англ. – М. : Бином, 2007. – 1152 с.

[3] Страуструп, Б. Язык программирования С++ / Б. Страуструп; специальное издание. Пер. с англ. – СПб. : BHV, 2008. – 1098 с.

[4] Объектно-ориентированное программирование в С++/ Роберт Лафоре г. пер. с англ. – Санкт-Петербург, 2019. – 1152 с.

**ПРИЛОЖЕНИЕ А**

(*обязательное*)

Диаграмма классов

**ПРИЛОЖЕНИЕ Б**

(*обязательное*)

Схема алгоритма метода void ShowSpecificCarSessions()

**ПРИЛОЖЕНИЕ В**

(*обязательное*)

Схема алгоритма метода bool UserAuthentication(User&)

**ПРИЛОЖЕНИЕ Г**

(*обязательное*)

Листинг кода программы

**ПРИЛОЖЕНИЕ Д**

(*обязательное*)

Ведомость документов