Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Технологии разработки программного обеспечения»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Старший преподаватель ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Карпейчик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Разработка программы Бронирования автомобилей»**

БГУИР КР 1-40 05 01-02 007 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 272303  Жаврид Алина Александровна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

БГУИР КР 1-40 05 01-02 007 ПЗ

**Жаврид, А.А.** Разработка программы бронирования автомобилей/ А.А. Жаврид. – Минск: БГУИР, 2023. – 54 с.

Пояснительная записка 54 с., 54 рис., 2 табл., 6 источников, 3 приложения

бронирование автомобилей, сведения об автомобилях, пользователях, модели *UML*, *IDEF*0, *BPMN*, схемы алгоритмов, программное средство

*Цель* *курсовой работы*: разработка программы бронирования автомобилей, которая автоматизирует процесс бронирования автомобилей, облегчает поиск доступных автомобилей для пользователей, а также позволяет администраторам контролировать и управлять процессом бронирования.

*Методология проведения работы*: в процессе решения поставленных задач использованы принципы системного подхода, аналитические методы, методы компьютерной обработки экспериментальных данных и компьютерного моделирования.

*Результаты работы*: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработаны модели бизнес-процессов предметной области на основе нотаций IDEF0 и BPMN; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование программного средства, показавшее его соответствие функциональным требованиям, поставленным в задании на разработку.

Программный продукт разработан на языке *C++* с применением *MS Visual Studio 2019*.

*Область применения результатов*: разработанное программное средство может применяться в сфере каршеринга или аренды автомобилей.

Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым при учете и предоставлении данных об автомобилях.

СОДЕРЖАНИЕ

[Введение 4](#_Toc135782961)

[1 Анализ и моделирование предметной области программного средства 5](#_Toc135782962)

[1.1 Описание предметной области 5](#_Toc135782963)

[1.2 Разработка функциональной модели предметной области в нотации IDEF0 6](#_Toc135782964)

[1.3 Разработка BPMN-модели основного процесса предметной области 11](#_Toc135782965)

[1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований. 12](#_Toc135782966)

[1.5 UML-модели представления программного средства и их описание 12](#_Toc135782967)

[2 Проектирование и конструирование программного средства 18](#_Toc135782968)

[2.1 Постановка задачи 18](#_Toc135782969)

[2.2 Разработка модульной структуры 20](#_Toc135782970)

[2.3 Выбор способа организации данных 21](#_Toc135782971)

[2.4 Разработка перечня пользовательских функций программы 23](#_Toc135782972)

[2.5 Разработка схем алгоритмов программы 25](#_Toc135782973)

[3 Тестирование и проверка работоспособности программного средства 30](#_Toc135782974)

[4 Инструкция по развертыванию приложения и сквозной пример 33](#_Toc135782975)

[4.1 Авторизация 33](#_Toc135782976)

[4.2 Модуль администратора 34](#_Toc135782977)

[4.3 Модуль пользователя 39](#_Toc135782978)

[4.4 Регистрация 41](#_Toc135782979)

[Заключение 42](#_Toc135782980)

[Список использованных источников 43](#_Toc135782982)

[Приложение А 44](#_Toc135782984)

[Приложение Б 52](#_Toc135782985)

[Приложение В 53](#_Toc135782986)

# ВВЕДЕНИЕ

В настоящее время автомобиль является одним из самых востребованных видов транспорта. В связи с этим, услуги автопроката становятся все более популярными, что требует от компаний, оказывающих эту услугу, повышенного внимания к качеству сервиса и улучшения условий бронирования автомобилей.

Бронирование автомобилей – это процесс предварительного заказа автомобиля на определенное время для его использования. Обычно это используется в услугах каршеринга, где пользователь может заранее забронировать автомобиль на нужное ему время, чтобы гарантировать его наличие и удобство использования. Бронирование может осуществляться как онлайн, через сайт или мобильное приложение, так и офлайн, при посещении офиса каршеринга. Бронирование автомобилей позволяет пользователям избежать неудобств, связанных с отсутствием свободных автомобилей в нужное время и место, а также экономить время на поиске подходящей машины.

Цель данного курсового проекта – разработать программу, которая автоматизирует процесс бронирования автомобилей. Это обеспечит пользователям быстрый и удобный доступ к информации об доступных автомобилях, а также позволит им осуществлять бронирование. Кроме того, программа позволит администраторам контролировать процесс бронирования.

Для достижения поставленной цели необходимо решить следующие задачи:

* ознакомиться с предметной областью;
* проанализировать особенности процесса бронирования автомобилей;
* разработать алгоритмы работы программного средства;
* разработать модули структуры программного средства;
* протестировать работу программного средства.

Объектом исследования данной курсовой работы является процесс бронирования автомобилей.

# АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

## Описание предметной области

Современный городской образ жизни диктует свои условия и требования к мобильности. Люди сталкиваются с проблемой выбора транспорта для перевозки по городу. Один из вариантов – автомобильное такси, которое не всегда удобно и доступно. Другой вариант ­­– покупка или аренда собственного автомобиля. Однако покупка автомобиля может быть слишком дорогостоящей, а аренда может вызывать неудобства, если необходимо использовать автомобиль на длительный период времени.

Система бронирования автомобилей решает эту проблему, позволяя пользователям арендовать автомобиль на нужный период времени без необходимости покупать его или рассчитывать на такси. Она представляет собой программное решение, которое позволяет клиентам автопрокатной компании производить бронирование автомобилей через интернет.

В настоящее время все больше людей предпочитают бронировать автомобили онлайн, в удобное для них время, без необходимости обращаться в офис автопроката. Это значительно экономит время и сокращает количество ошибок, связанных с обработкой заказов вручную.

Для использования системы клиенту необходимо зарегистрироваться на сайте или в мобильном приложении автопрокатной компании и выбрать дату и время, на которое нужно арендовать автомобиль. Затем клиенту предоставляется список доступных автомобилей с указанием их характеристик и стоимости. Клиент может выбрать автомобиль, который соответствует его требованиям. После выбора автомобиля клиент вводит личные данные, такие как фамилия, имя, контактный телефон, адрес почтового ящика и номер банковского счета.

После подтверждения о соответствии личных данных требованиям компании автомобиль предоставляется клиенту. По истечению времени бронирования клиент должен произвести оплату. Оплата может производиться через сайт, мобильное приложение или путем перевода денег на банковский счет компании.

Система бронирования автомобилей позволяет упростить процесс аренды автомобилей и сделать его более удобным для клиентов и персонала компании.

Однако такие системы имеют некоторые недостатки. Одним из главных недостатков систем бронирования автомобилей является их ограниченная доступность. Некоторые компании не предоставляют достаточно информации о своих автомобилях и услугах, что может создавать трудности у клиентов. Также некоторые автопрокатные компании не имеют удобного онлайн-интерфейса, что затрудняет процесс бронирования автомобиля.

Все эти недостатки свидетельствуют о необходимости улучшения систем бронирования автомобилей. Автоматизация процесса бронирования может помочь компаниям устранить эти недостатки и улучшить качество обслуживания клиентов.

Актуальность разработки системы бронирования автомобилей заключается в том, что система может предложить клиенту больше вариантов автомобилей и дополнительных услуг, а также обеспечить точность и быстроту обработки заказов. Это значительно повышает удовлетворенность клиентов и может привести к увеличению количества бронирований. Компании, которые не могут предложить своим клиентам удобную и быструю систему бронирования, рискуют потерять свою клиентскую базу и снизить свою прибыль.

Кроме того, автоматизация системы бронирования позволяет снизить операционные затраты и сократить количество ошибок, связанных с обработкой заказов вручную. Это улучшает эффективность работы компании и позволяет сократить время на выполнение задач, которые могут быть автоматизированы.

Таким образом, разработка системы бронирования автомобилей является важным фактором для повышения эффективности работы автопрокатных компаний, улучшения качества обслуживания клиентов и контроля бронирования автомобилей.

## Разработка функциональной модели предметной области в нотации IDEF0

IDEF0 (Integration Definition for Function Modeling) – это методология и язык моделирования, используемые для анализа и проектирования бизнес-процессов. Она позволяет описывать функции, потоки данных и контроль в бизнес-процессах и представляет их в виде блок-схем.

Использование IDEF0 позволяет более эффективно проектировать бизнес-процессы, повышать качество их выполнения и уменьшать затраты на их выполнение [2]. Она является надежным инструментом для анализа и проектирования систем, включая систему бронирования автомобилей.

Рассмотрим информационную систему, выполненную с помощью средств моделирования функций IDEF0. Для начала необходимо сделать контекстную модель информационной системы. Контекстная диаграмма – самая верхняя диаграмма, на которой объект моделирования представлен единственным блоком с граничными стрелками. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой.

На рисунке 1.1 описан основной блок “Забронировать автомобиль”. Входящие стрелки – “Данные о клиенте”, ”Заказ клиента”, “Данные об автомобилях”, ”Анкета для заключения договора о бронировании автомобиля”. Это то, что необходимо иметь для начала работы. Управляющие для системы бронирования – “ПДД”, “Закон о защите персональных данных”, “Гражданский кодекс РБ”, ”Акт приемки-передачи”. В роли механизмов выступают клиент, менеджер по бронированию автомобилей, который регистрирует клиента, и компьютер, с помощью которого эта регистрация непосредственно осуществляется. После завершения процесса клиент получает “Забронированный автомобиль”.

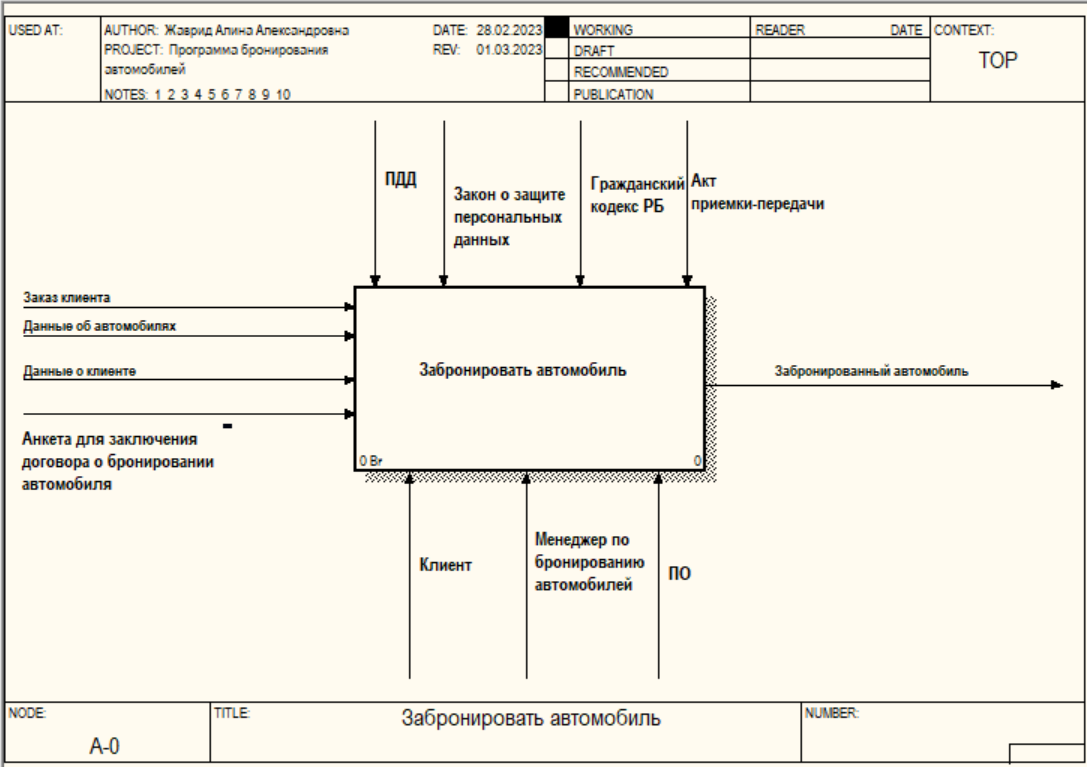


Рисунок 1.1 – Контекстная диаграмма модели А-0

На рисунке 1.2 представлена декомпозиция процесса “Забронировать автомобиль”. В данном случае мы получили диаграмму, состоящую из четырех процессов:

1. Зарегистрировать клиента – процесс регистрации новых пользователей в системе бронирования автомобилей, который включает ввод информации о персональных данных и контактных данных.
2. Предоставить информацию об автомобилях – процесс, который позволяет пользователям системы просматривать информацию о доступных для аренды автомобилях. В рамках данного процесса включается поиск по параметрам автомобиля, отображение описания, а также информация о цене аренды.
3. Заключить договор о бронировании автомобиля – процесс оформления заявки на аренду автомобиля, который включает выбор автомобиля, указание даты и времени аренды, информацию о стоимости и условиях аренды, а также подписание договора на аренду автомобиля.
4. Предоставить автомобиль – процесс передачи арендованного автомобиля пользователю, который может включать в себя проверку документов на вождение, инструктаж по эксплуатации автомобиля и подписание акта приема-передачи автомобиля.

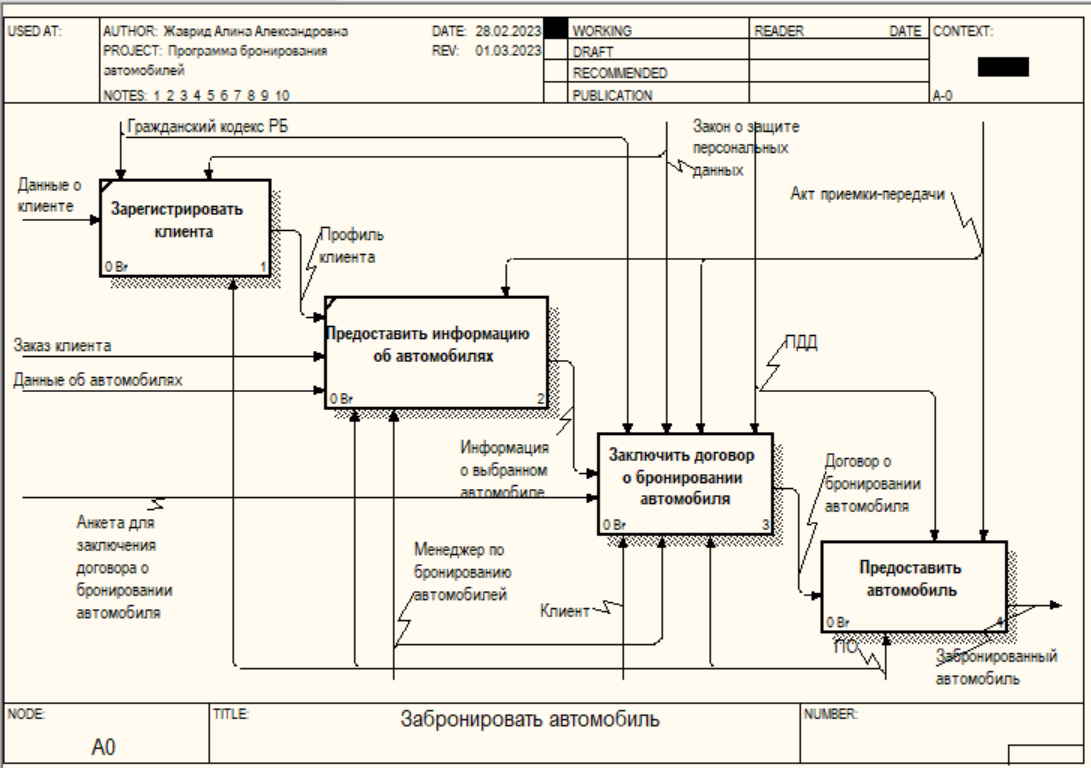


Рисунок 1.2 – Декомпозиция процесса “Забронировать автомобиль”

На рисунке 1.3 предоставлена декомпозиция процесса “Заключить договор о бронировании автомобиля”. Заключить договор довольно просто: заполняете анкету, прикладываете фото паспорта, прав. Также необходимо привязать банковскую карту, так как с нее будут списывать средства за оказание услуг. Далее все документы тщательно проверяются и заключается договор.

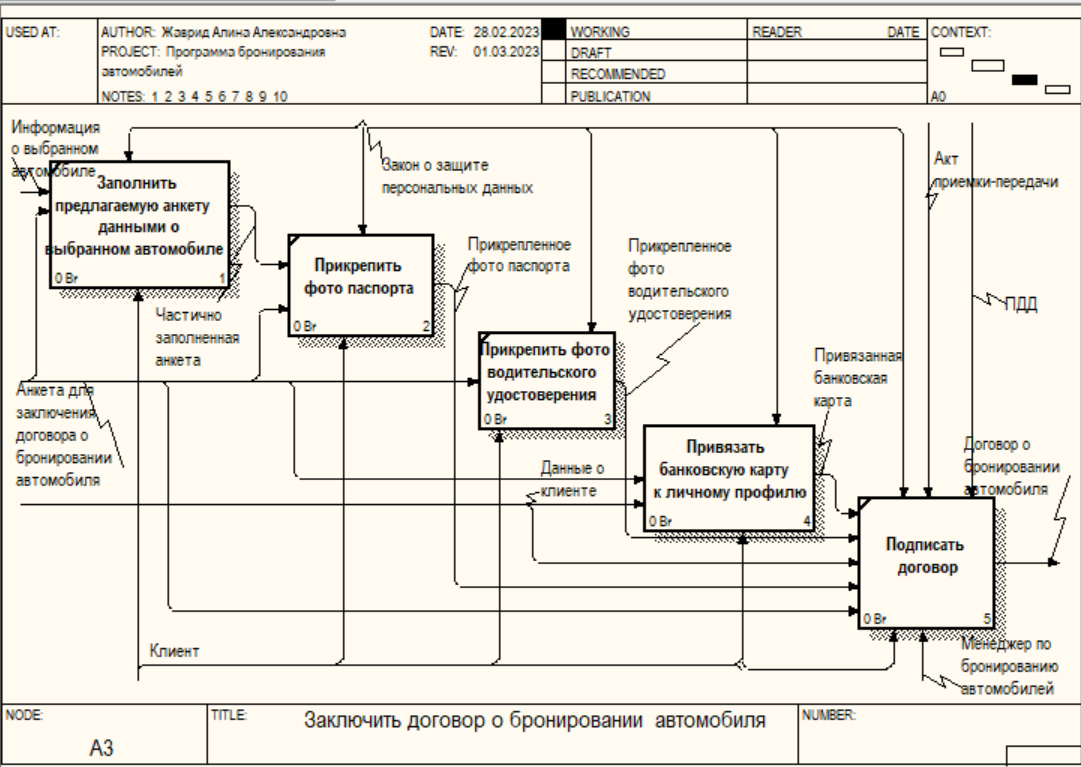


Рисунок 1.3 – Декомпозиция процесса “Заключить договор о бронировании автомобиля”

На рисунке 1.4 предоставлена декомпозиция процесса “Привязать банковскую карту к личному профилю”. Пользователь приложения может перейти на страницу платежей из соответствующей опции в главном меню и проделать следующие действия: “Указать номер карты”, “Указать срок действия карты”, “Указать имя на карте”, “Указать CVV-код”, “Указать код безопасности из СМС”. Результатом этого процесса является привязанная банковская карта к личному профилю.

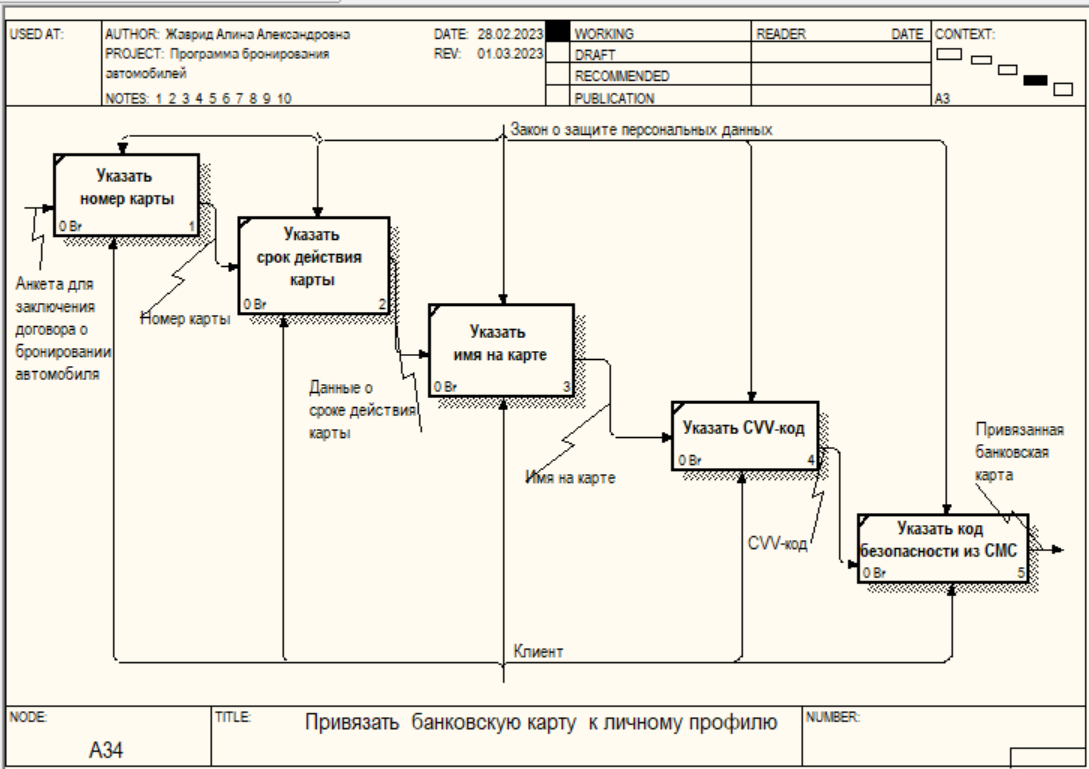


Рисунок 1.4 – Декомпозиция процесса “Привязать банковскую карту к личному профилю”

На рисунке 1.5 предоставлена декомпозиция процесса “Предоставить автомобиль”. Как только клиент заключит договор, приложение выдаст ближайшее к нему авто, после чего его нужно зарезервировать. Расстояние от клиента до машины определяется автоматически.

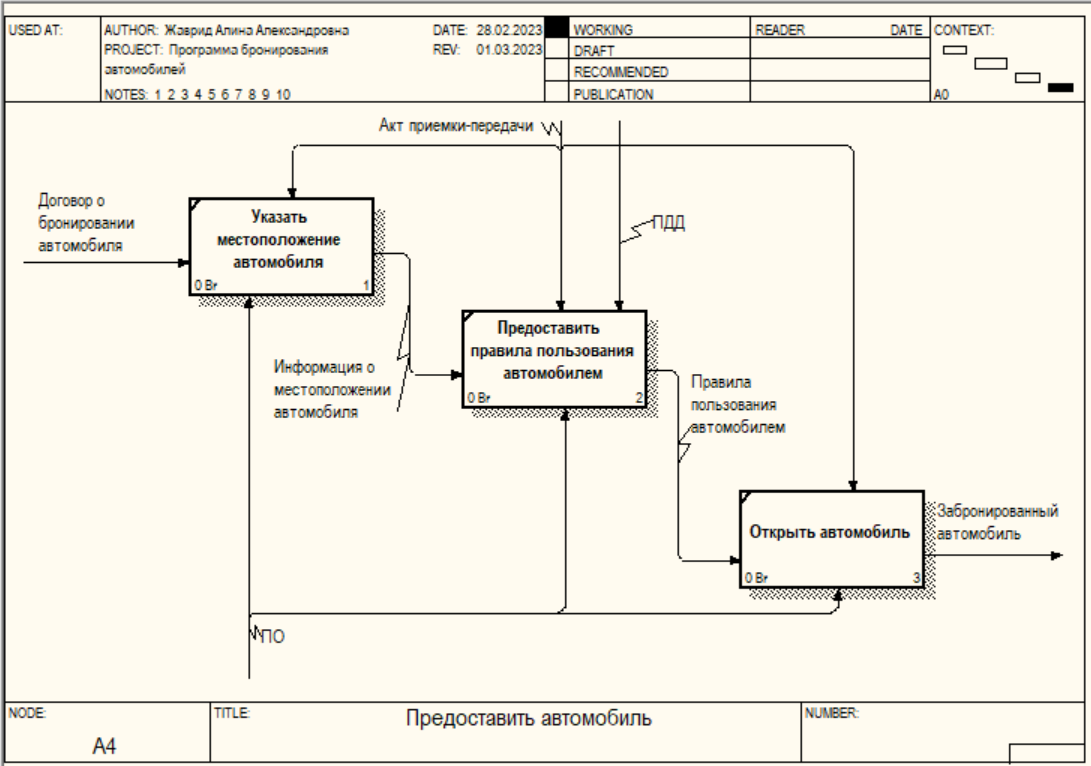


Рисунок 1.5 – Декомпозиция процесса “Предоставить автомобиль”

## Разработка BPMN-модели основного процесса предметной области

Business Process Model and Notation (BPMN) – это нотация моделирования бизнес-процессов, которая представляет бизнес-процессы в виде блок-схем [3].

В контексте разработки программы бронирования автомобилей, первым этапом является построение контекстной диаграммы. Для этого на диаграмме рисуем дорожки, а также события "начало" и "конец". Затем на диаграмму добавляем действия, используя логические операторы для создания разветвлений.

Для разработки программы бронирования автомобилей, использование BPMN поможет наглядно представить последовательность действий и информационные потоки, что облегчит процесс разработки и внедрения системы (см. рисунок 1.6).

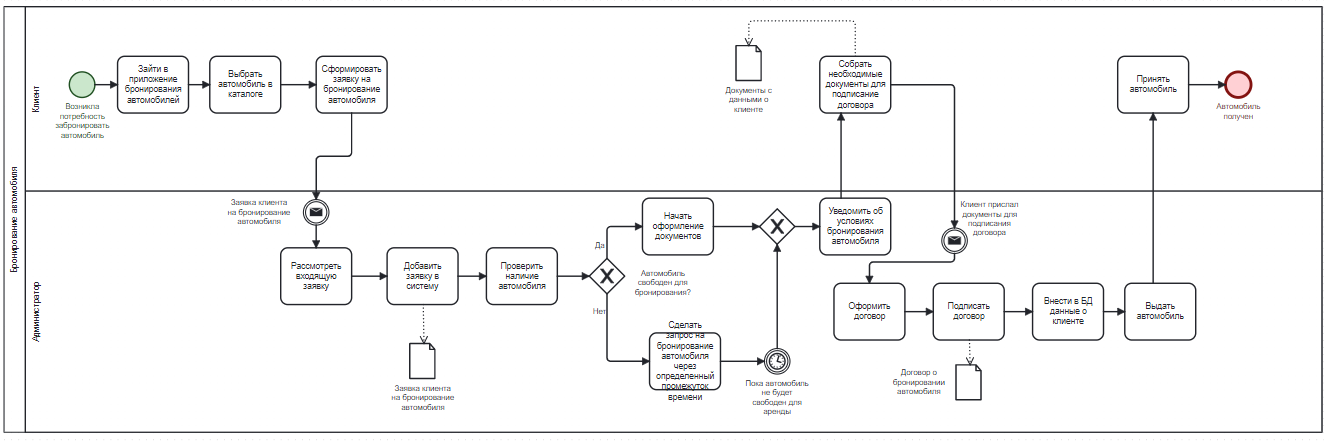
****

Рисунок 1.6 – BPMN-модель основного процесса предметной области

## Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований.

Разрабатываемое в ходе курсовой работы программное средство должно соответствовать определенным требованиям. Одним из главных требований к программе является наличие разделения на пользователя и администратора, а также возможность разделения их прав доступа. Пользователь должен иметь возможность зарегистрироваться в системе и выбрать свой автомобиль для бронирования, а также отслеживать статус своих заказов. Администратор должен иметь возможность управлять списком автомобилей, их стоимостью, сроками аренды и статусом заказов. Другим важным требованием является наличие функционала модуля действий, таких как запрос на подтверждение удаления записей или внесения изменений. Это поможет избежать случайных удалений или изменений информации, которые могут негативно повлиять на работу всей системы. Кроме того, необходимо предусмотреть возможность добавления новых автомобилей и изменения информации об уже существующих, а также возможность поиска автомобилей по определенным критериям, таким как модель, цена или дата аренды. Каждое из этих требований является важным и необходимым для обеспечения качественной работы программы.

## UML-модели представления программного средства и их описание

UML (Unified Modeling Language) – это язык визуального моделирования, используемый для разработки программного обеспечения. UML предоставляет стандартизированные нотации и синтаксис для описания различных аспектов программной системы, таких как ее структура, поведение и взаимодействие между ее компонентами. Использование UML-моделей позволяет разработчикам лучше понимать и описывать архитектуру программного обеспечения, представлять ее в виде графических диаграмм и обеспечивать более наглядное взаимодействие между членами команды [2].

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. На рисунке 1.7 представлены два актёра: Администратор и Клиент, а также сценарии их поведения.

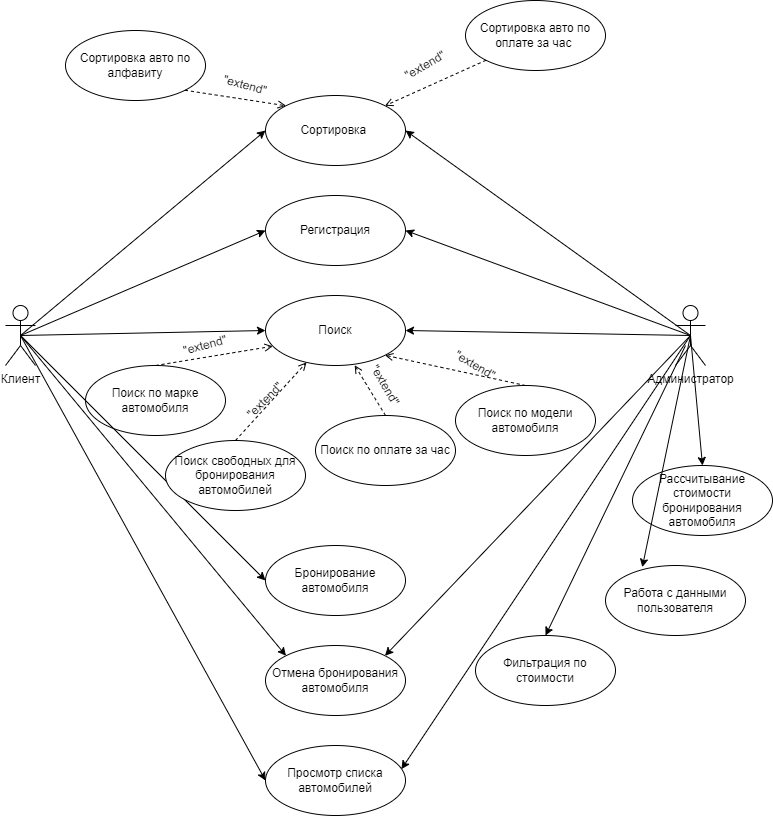


Рисунок 1.7 – UML-диаграмма вариантов использования

Диаграмма компонентов – это структурная диаграмма, которая используется для моделирования и отображения архитектуры программного обеспечения. Эта диаграмма позволяет показать, как компоненты программного обеспечения взаимодействуют друг с другом, какие зависимости между ними существуют и какие интерфейсы предоставляются каждым компонентом. Диаграмма компонентов обычно используется на начальном этапе проектирования системы для создания общего представления об архитектуре. Она может быть использована в качестве документации для программистов, тестировщиков и других участников проекта, а также в качестве средства для обмена информацией между различными участниками проекта [2].

На рисунке 1.8 представлена диаграмма компонентов, на которой показано взаимодействие различных функций программного обеспечения между собой и результаты их выполнения.

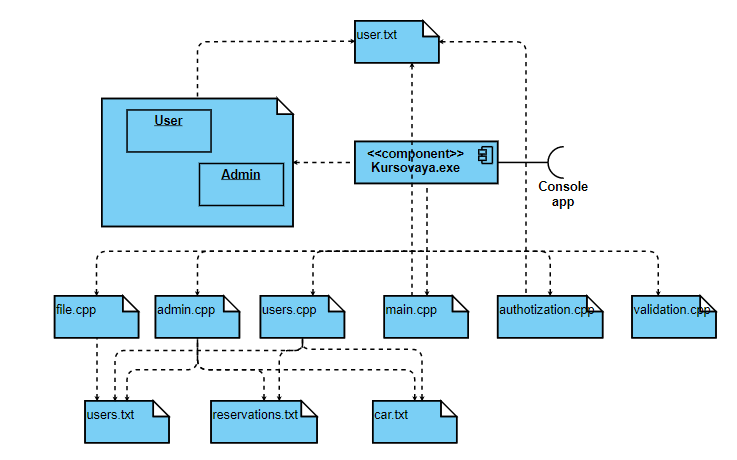


Рисунок 1.8 – UML-диаграмма компонентов

Диаграмма состояний является графическим инструментом, который позволяет моделировать поведение объектов и систем в различных состояниях и переходах между этими состояниями. Это позволяет разработчикам и аналитикам лучше понимать функциональность системы и ее возможные сценарии использования [2]. На рисунке 1.9 представлена UML-диаграмма состояний. Данная диаграмма описывает основной процесс: бронирование автомобиля.

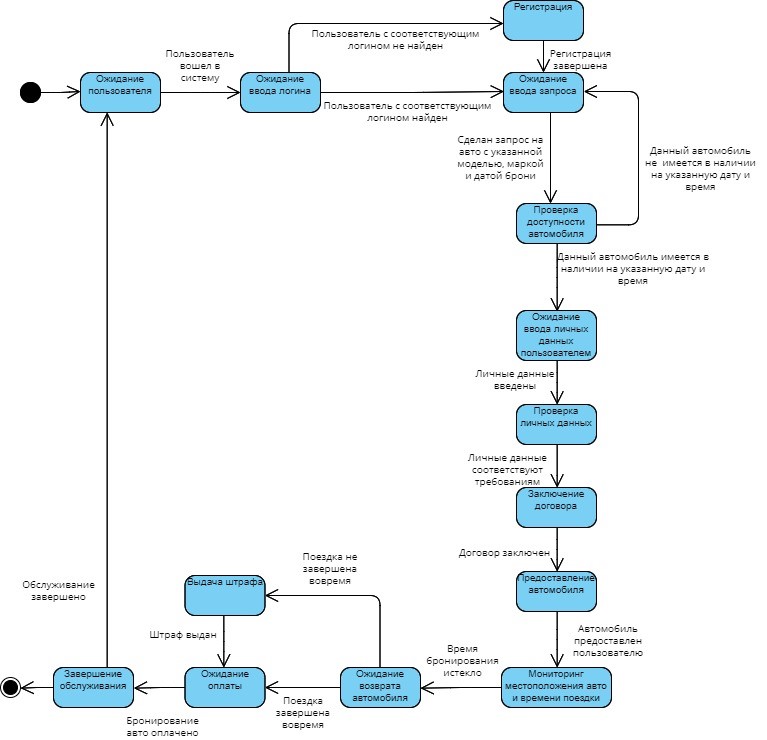


Рисунок 1.9 – UML-диаграмма состояния

Диаграмма развертывания – это диаграмма UML, которая используется для визуализации физического аппаратного и программного обеспечения системы [2].

В ходе выполнения программы данные о клиентах и автомобилях будут сохраняться в файлах:

* Kursovaya.exe – сама программа;
* user.txt – файл, в котором хранится информация о логинах и паролях пользователей;
* car.txt – файл, в котором хранится информация о всех автомобилях;
* users.txt – файл, в котором хранится личная информация всех пользователей;
* reservations.txt – файл, в котором хранится информация о пользователях и забронированных ими автомобилях.

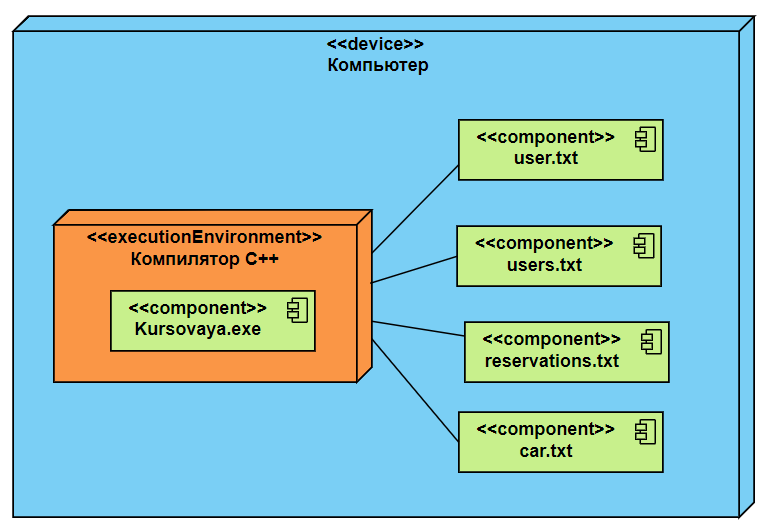


Рисунок 1.10 – UML-диаграмма развертывания

Диаграмма последовательности – это тип диаграммы UML, который используется для описания взаимодействия между объектами в рамках некоторой системы. Она показывает, как объекты обмениваются сообщениями и как эти сообщения связаны во времени. На диаграмме последовательности объекты представлены в виде вертикальных линий (столбцов), а сообщения между объектами – в виде стрелок, которые соединяют соответствующие столбцы. Диаграммы последовательности позволяют увидеть порядок вызова методов и передачу данных между объектами, что помогает понять работу системы и выявить возможные проблемы [2].

На UML-диаграмме последовательностей (см. рисунок 1.11) в общих чертах изображена основная суть разрабатываемого программного средства. Пользователь вводит данные (пароль и логин). В случае, если данные введены верно, пользователь получает доступ к определенному набору действий, определенных разработчиком. В противном же случае, пользователь будет уведомлен о том, что данные были введены неверно.

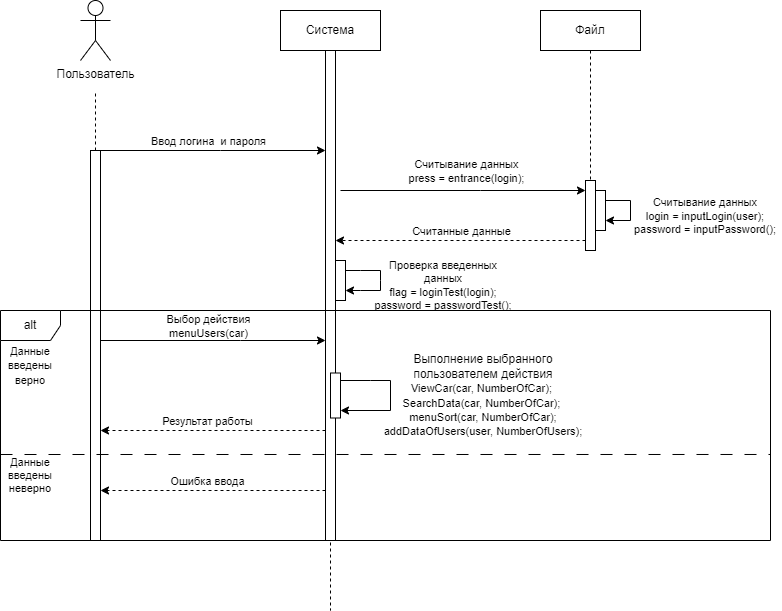


Рисунок 1.11 – UML-диаграмма последовательности

Анализ и моделирование бизнес-процессов, связанных с бронированием автомобилей, с использованием диаграмм IDEF0, UML и BPMN, позволяет лучше понять структуру системы, визуализировать модели объектов и процессов, определить потоки информации и выявить возможные узкие места и проблемы в процессе. Использование такого подхода помогает разработчикам создать эффективную и удобную систему бронирования автомобилей, которая удовлетворит потребности пользователей и бизнес-процессов.

# ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## Постановка задачи

По условию задания требуется разработать программу для бронирования автомобилей. Система должна предусматривать режимы ведения данных об автомобилях, отражающее все сведения о них, хранящееся в базе данных. О каждом автомобиле будет известно следующее:

* марка;
* модель;
* стоимость аренды;
* статус бронирования.

Помимо данных об автомобилях, в ИС бронирования автомобилей хранятся данные о клиентах, такие как:

* фамилия и имя;
* контактный телефон;
* адрес почтового ящика;
* номер банковского счета.

С данной информационной системой должны работать следующие группы пользователей:

* пользователи;
* администраторы.

Модуль администраторавключает следующие подмодули:

1. Управление учетными записями пользователей:
   * просмотр всех учетных записей;
   * добавление новой учетной записи;
   * редактирование учетной записи;
   * удаление учетной записи.
2. Работа с данными:
3. режим редактирования:

* просмотр всех данных;
* добавление новой записи;
* удаление записи;
* редактирование записи;

1. режим обработки информации:

* поиск данных (по четырем различным параметрам);
* сортировка данных (по двум различным параметрам);
* фильтрация данных по цене;
* отмена бронирования автомобиля.

Модуль пользователя включает в себя подмодуль работы с данными со следующими функциональными возможностями:

* просмотр всех данных автомобилей;
* бронирование автомобиля;
* отмена бронирования автомобиля;
* вывод забронированных пользователем автомобилей;
* фильтрация данных по цене;
* поиск данных (по четырем параметрам);
* сортировка данных (по двум параметрам).

Для реализации перечисленных модулей/подмодулей создано меню с соответствующими пунктами.

Необходимо предусмотреть:

* обработку исключительных ситуаций (ничего не найдено по результатам поиска; введенные пользователем данные не соответствуют формату поля; файл с записями для чтения не существует; логин новой учетной записи уже существует; неправильно введен логин/пароль, повторное введение пароля при регистрации);
* возможность возврата назад (навигация);
* запрос на подтверждение удаления вида «Вы действительно хотите удалить запись?»;
* запрос на подтверждение редактирование вида «Вы действительно хотите редактировать запись?»;
* вывод сообщения об успешности создания /удаления записи.

Требования к программной реализации проекта:

1. Все переменные и константы должны иметь осмысленные имена в рамках тематики варианта к курсовой работе.
2. Имена функций должны быть осмысленными.
3. Код не должен содержать неименованных числовых констант (так называемых «магических» чисел), неименованных строковых констант (например, имен файлов и др.). Подобного рода информацию следует выносить в глобальные переменные с атрибутом const. Код необходимо комментировать (как минимум в части объявления структур, массивов/векторов, заголовков функций, нетривиальной логики).
4. Код не должен дублироваться – для этого существуют функции!
5. Выполнение операций чтения/записи в файл должно быть сведено к минимуму (т.е. после однократной выгрузки данных из файла в массив/вектор дальнейшая работа ведется с этим массивом/вектором, а не происходит многократное считывание данных из файла в каждой функции).
6. Следует избегать длинных функций и глубокой вложенности: текст функции должен умещаться на один экран, а вложенность блоков и операторов должна быть не более трёх.

## Разработка модульной структуры

Модульное программирование – это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам. Использование модульной структуры позволяет разбить большую сложную систему на небольшие независимые блоки, называемые модулями. Для разработки системы был выбран метод восходящей разработки. Модульная структура предоставлена на рисунке 2.1.

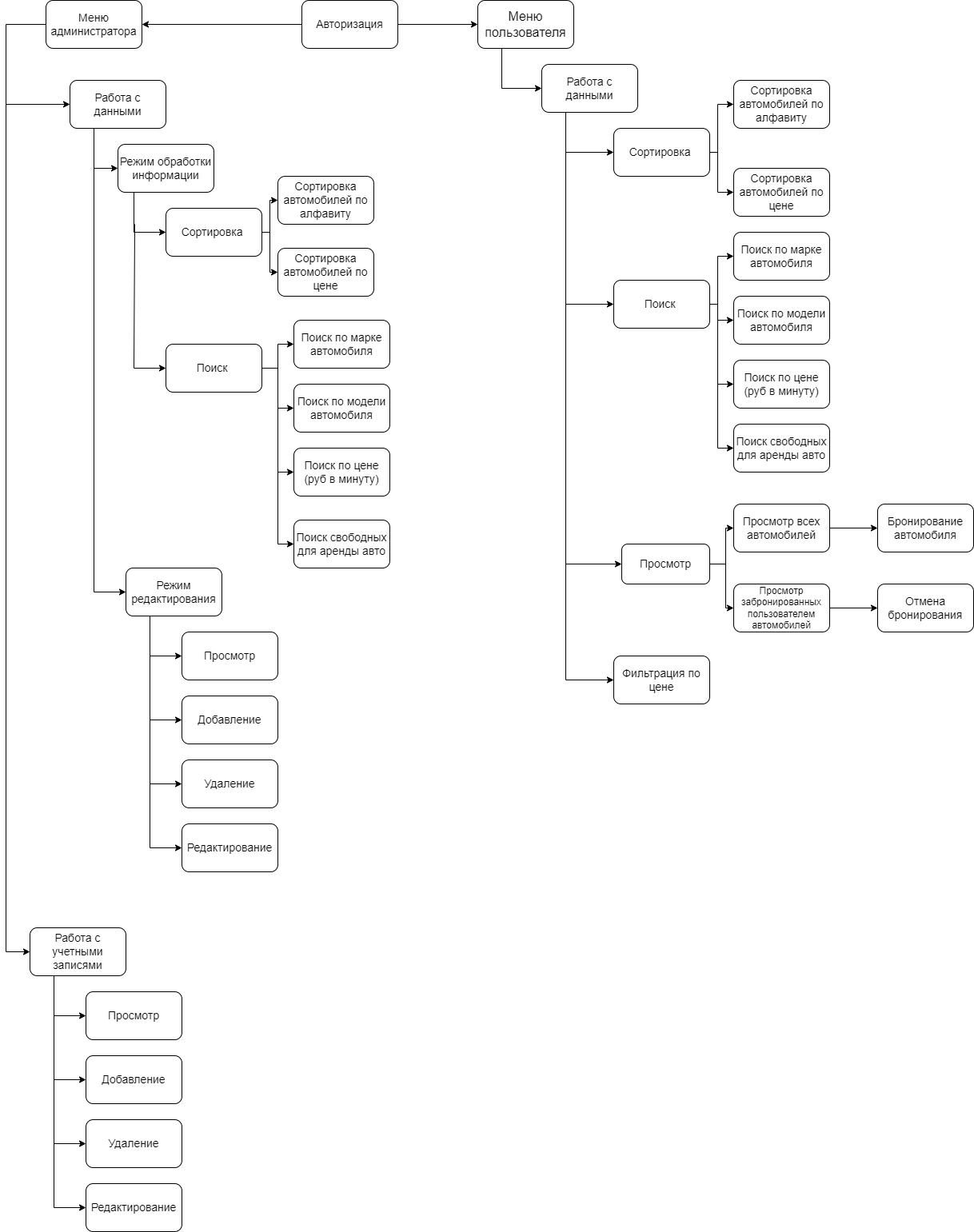


Рисунок 2.1 – Модульная структура программы

## 2.3 Выбор способа организации данных

В курсовом проекте используется множество различных переменных. Для более удобной работы с ними организованы структуры. Они имеют одно имя для обозначения группы переменных. Но эта группа должна иметь родственные, схожие признаки. Таким образом, мы группируем их на модули, что позволяет быстрее изменять код и делать его компактным и структурированным.

Описание используемых структур представлено ниже в таблице 2.1.

Таблица 2.1 – Описание структуры users

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| login | string | Логин пользователя |
| password | string | Пароль пользователя |
| role | int | Роль пользователя |
| mail | string | Почтовый ящик клиента |
| number | string | Номер телефона клиента |
| surname | string | Фамилия клиента |
| name | string | Имя клиента |
| bankAccount | string | Номер банковского счета клиента |

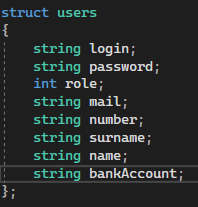


Рисунок 2.1 – Структура users

Таблица 2.2 – Описание структуры cars

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| model | string | Марка авто |
| carType | string | Модель авто |
| rentPrice | float | Цена брони авто (руб/мин) |
| is\_reserved | bool | Статус брони авто |
| reservedTimes | vector<time\_t> | Время брони авто |



Рисунок 2.2 – Структура cars

## 2.4 Разработка перечня пользовательских функций программы

Для разработки программы были использованы следующие функции:

int registrationAndAuthorization(users\*& login, int& press);//Меню регистрации и авторизации.

void menuAdmin(users\*& user, cars\*& car);//Меню администратора.

void menuWorkWithDatausers(users\*& user, int& NumberOfUsers, cars\*& car, int& NumberOfCar);//Меню для управления уч. записями(администратор).

void menuWorkWithDataCar(cars\*& car, int& NumberOfCar);//Меню для работы с данными автомобилей(администратор).

void menuUsers(cars\*& car, int& NumberOfCar,string login, users\*& user, int& NumberOfUsers);//Меню пользователя.

void menuSort(cars\* car, int NumberOfCar);//Меню сортировки.

void menuSearch(cars\* car, int NumberOfCar);//Меню поиска.

void editDataAboutCar(cars \*&car, int& NumberOfCar);//Меню режима редактирования данных об автомобилях.

void editUserData(users\*& user, int& NumberOfUsers);//Меню редактирования полей данных о пользователях.

void editCarData(cars\*& car, int& NumberOfCar);//Меню редактирования полей данных об автомобиле.

void ViewUserData(users\* user, int NumberOfUsers, cars\* car, int NumberOfCars);//Просмотр данных о пользователях и забронированных ими автомобилях.

void addDataOfUsersA(users\*& user, int& NumberOfUsers);//Добавление данных о пользователе.

void ViewUserPersonalData(users\* user, int NumberOfUsers);//Просмотр личных данных о пользователях.

void deleteUserData(users\*& user, int& NumberOfUsers, cars\*& car, int& NumberOfCar);//Удаление аккаунта пользователя.

void deleteCarData(cars\*& car, int& NumberOfCar);//Удаление информации об автомобиле.

void filterByPriceRange(cars\* car, int numberOfCars);//Фильтрация автомобилей по цене.

void addDataOfCar(cars\*& car, int& NumberOfCar);//Добавление данных об автомобилях.

void viewReservations(string phoneNumber);//Просмотр информации о забронированном пользователем автомобиле.

void ViewCar(cars\*& car, int NumberOfCar);//Просмотр списка автомобилей.

void cancel\_reservation(cars\*& car, int& NumberOfCar, bool is\_user);//Отмена бронирования автомобиля.

void ViewSearch(cars\* car, int k, int i);//Просмотр при поиске.

void searchModel(cars\* car, int NumberOfCar); void searchCarType(cars\* car, int NumberOfCar);void searchRentPrice(cars\* car, int NumberOfCar); void searchNotReserved(cars\* car, int NumberOfCar);//Поиск по определенному параметру.

void sortingPay(cars \*&car, int NumberOfCar); void sortingCar(cars \*&car, int NumberOfCar); //Сортировка по определенному параметру.

string getCurrentDateTime();//Функция для определения текущей даты.

void getStartDateTime(int& startDay, int& startMonth, int& startYear, int& startHour, int& startMinute);void getEndDateTime(int& startDay, int& startMonth, int& startYear, int& startHour, int& startMinute, int& endDay, int& endMonth, int& endYear, int& endHour, int& endMinute);//Ввод начала/окончания даты и времени бронирования автомобиля.

void logReservation(cars&car, users&user, int startDay, int startMonth, int startYear, int startHour, int startMinute, int endDay, int endMonth, int endYear, int endHour, int endMinute); //Запись данных о пользователе и забронированном автомобиле.

void reserveCar(cars\* car, int NumberOfCar, users\*& user, int& NumberOfUsers, string login); //Бронирование автомобиля.

string enterName();string enterSurname();string enterEmail();string enterPhoneNumber();string enterBankAccount();//Ввод данных о пользователе

bool isPasswordValid(string& password);bool isTimeValid(int hour, int minute);bool isDateValid(int day, int month, int year);bool isEmailValid(string& email);bool isPhoneValid(string& phone); float enterRentPrice();string passwordTest();bool loginTest(string login);//Проверки на корректность ввода данных.

int listOfPersonalData(users\*& user); int listDataOfCar(cars\*& car); string\* listUsersLogin(int& NumberOfUsers);int listUsers(users\*& user); users\* listUsers(int NumberOfUsers); void loadDataOfCar(cars\*& car, int& NumberOfCar); void loadDataOfUsers(users\*& user, int& NumberOfUsers); //Чтение данных из файла.

void addCar(string model, string carType, float rentPrice,bool is\_reserved); void addRegistration(string login, string password,int role); void addUser(string login, string password, string surname,string name,string mail,int number, int bankAccount, int role); void saveUsers(users\* user, int NumberOfUsers);//Запись информации в файл.

string inputPasswordStars();//Ввод пароля звёздачками.

string inputLogin(users\*& user);//Ввод логина.

string inputPassword();//Ввод пароля.

void registration(users\*& user);//Регистрация нового пользователя.

int stringComparison(string s, string a);//Сравнение двух паролей.

int searchLoginPassword(users\*& user, int numberOfUsers, int& count, string login, string password);//Поиск логина и пароля.Определение пользователя или администратора.

int entrance(users\*& user);// авторизация.

int registrationAndAuthorization(users\*& user, int& press);//Выбор авторизация или регистрация.

## 2.5 Разработка схем алгоритмов программы

Алгоритм в программировании – это последовательность шагов, которые описывают порядок выполнения задачи. Он представляет собой набор инструкций, которые нужно выполнить, чтобы получить желаемый результат. Хорошо разработанные алгоритмы помогают программистам создавать эффективный и оптимизированный код, который выполняет задачу быстро и точно.

На рисунке 2.3 представлен алгоритм функции main.

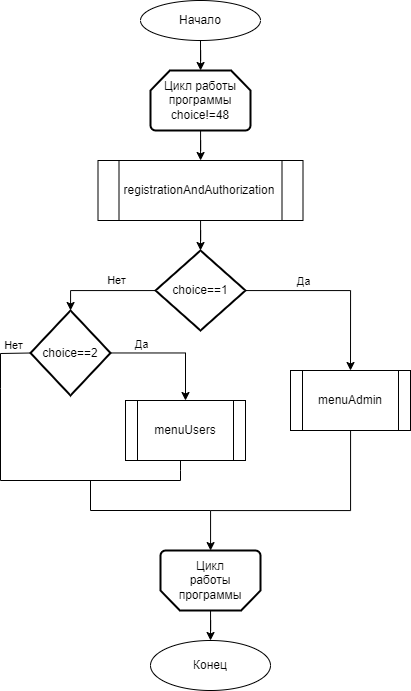


Рисунок 2.3 – Алгоритм функции main

Реализация функции int main():

int main()

{

users\* user = nullptr;

cars\* car = nullptr;

int NumberOfUsers = 0;

int NumberOfCar = 0;

int choice;

string login;

do {

registrationAndAuthorization(user, choice);

if (choice == 1)

menuAdmin(user, car);

else if (choice == 2)

menuUsers(car, NumberOfCar, login, user, NumberOfUsers);

} while (choice != 48);

delete[] user;

}

Функция ViewCar() выводит на экран информацию об автомобилях. Алгоритм функции представлен на рисунке 2.4.

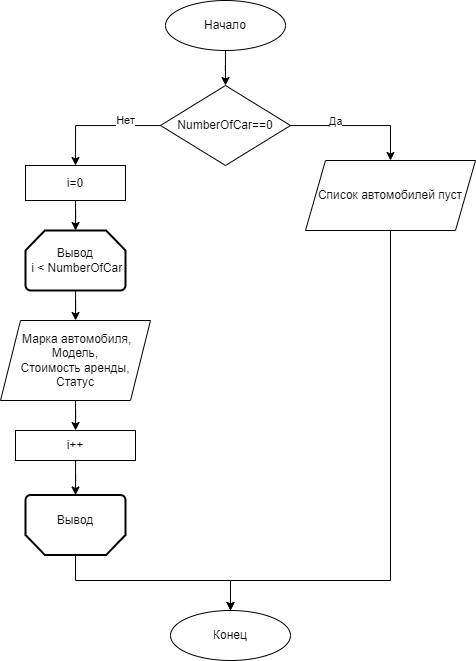


Рисунок 2.4 – Алгоритм функции ViewCar()

Реализация функции ViewCar():

cout << "------------------------------------------------------------\n"

<< "| № | Марка автомобиля | Модель | Стоимость аренды | Статус |\n"

<< "----------------------------------------------------\n";

if (NumberOfCar == 0) cout << "Список автомобилей пуст" << endl;

else {

for (int i = 0; i < NumberOfCar; i++) {

cout << "| " << setw(3) << left << i + 1 << " | "

<< setw(30) << left << car[i].model << " | "

<< setw(18) << left << car[i].carType << " | "

<< setw(17) << left << car[i].rentPrice << " | "

<< setw(6) << left << car[i].is\_reserved << " |\n";

cout << "-----------------------------------------\n";

}

}

Функция searchNotReserved() позволяет найти незабронированные автомобили и вывести список на экран. Предоставление пользователю списка незабронированных автомобилей помогает для дальнейшего выбора и бронирования. Алгоритм функции представлен на рисунке 2.5.

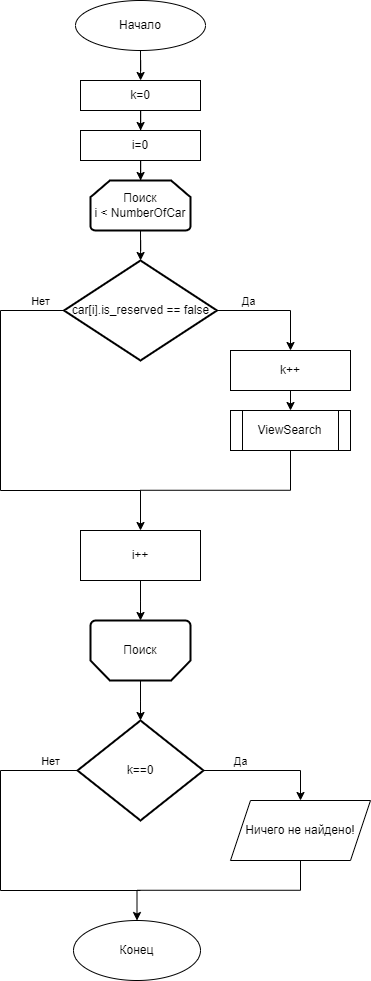


Рисунок 2.5 – Алгоритм функции searchNotReserved()

Реализация функции searchNotReserved():

{

int k = 0;

cout << "-----------------------------------------\n"

<< "| № | Марка автомобиля | Модель | Стоимость аренды | Статус |\n"

<< "----------------------------------------------------\n";

for (int i = 0; i < NumberOfCar; i++) {

if (car[i].is\_reserved == false)

{

k++;

ViewSearch(car, k, i);

continue;

}

}

if (k == 0) {

cout << "Ничего не найдено!\n";

}

system("CLS");

}

Данные схемы алгоритмов работы программы помогают разработчикам лучше понимать логику работы программы и выявлять возможные проблемы и узкие места в ее функционировании. Они могут также быть использованы для обучения новых разработчиков, которые могут быстрее овладеть функциональностью программы, используя такие схемы.

Таким образом, разработка схем алгоритмов работы программы является важным этапом в процессе разработки программного обеспечения, который помогает повысить качество и эффективность работы программы.

В этом разделе была выполнена постановка задачи, выбран способ организации данных, а именно с помощью структуры на языке программирования C++. Далее был разработан перечень пользовательских функций программы.

Кроме того, были разработаны три схемы алгоритмов работы программы. В целом, процесс проектирования и конструирования программного средства был успешно завершен, и были разработаны все необходимые элементы для реализации программы.

1. **ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

При написании кода программы необходимо предусмотреть обработку исключительных ситуаций, таких как:

* введенные пользователем данные не соответствуют формату поля (например, символы в числовом поле);
* неверно введен логин или пароль;
* логин уже существует;
* введенные пользователем данные нелогичны (например, год бронирования больше 2023);
* подтверждение удаления и редактирования;
* ничего не найдено по результатам поиска.

При добавлении или редактировании учетной записи, программа определяет корректный ввод логина и пароля. Пароль должен содержать как минимум одну букву верхнего регистра, одну букву нижнего регистра и одну цифру. Логин должен быть введен латинскими буквами не длиннее 20 символов. При вводе неправильного логина и/или пароля, программа будет просить ввод правильных данных, пока не будут введены правильные данные, а также запросит ввести пароль для подтверждения (рисунок 3.1).

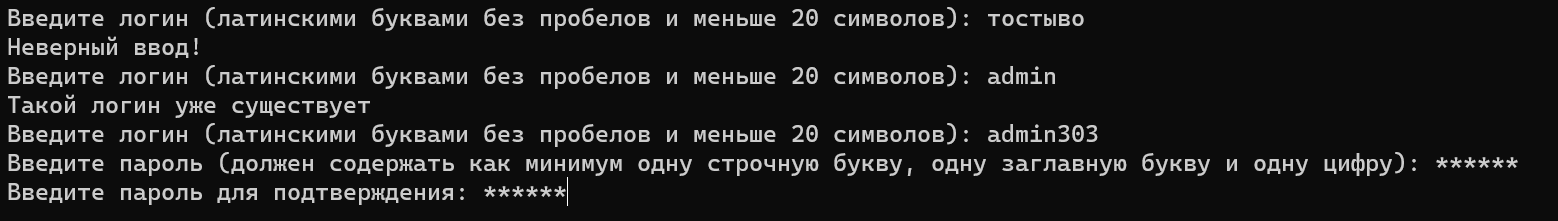


Рисунок 3.1 – Ввод логина и пароля

Предусмотрено ввод только букв в случае, если необходимо ввести имя, фамилию (рисунок 3.2).

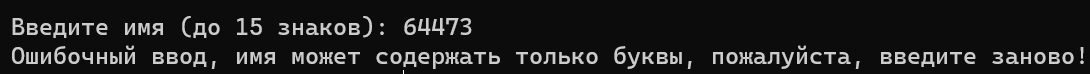


Рисунок 3.2 – Ввод не букв

При записи номера телефона предусмотрен ввод только цифр и определенный формат ввода. Пример ввода приведен на рисунке 3.3 и 3.4.



Рисунок 3.3 – Ввод не цифры

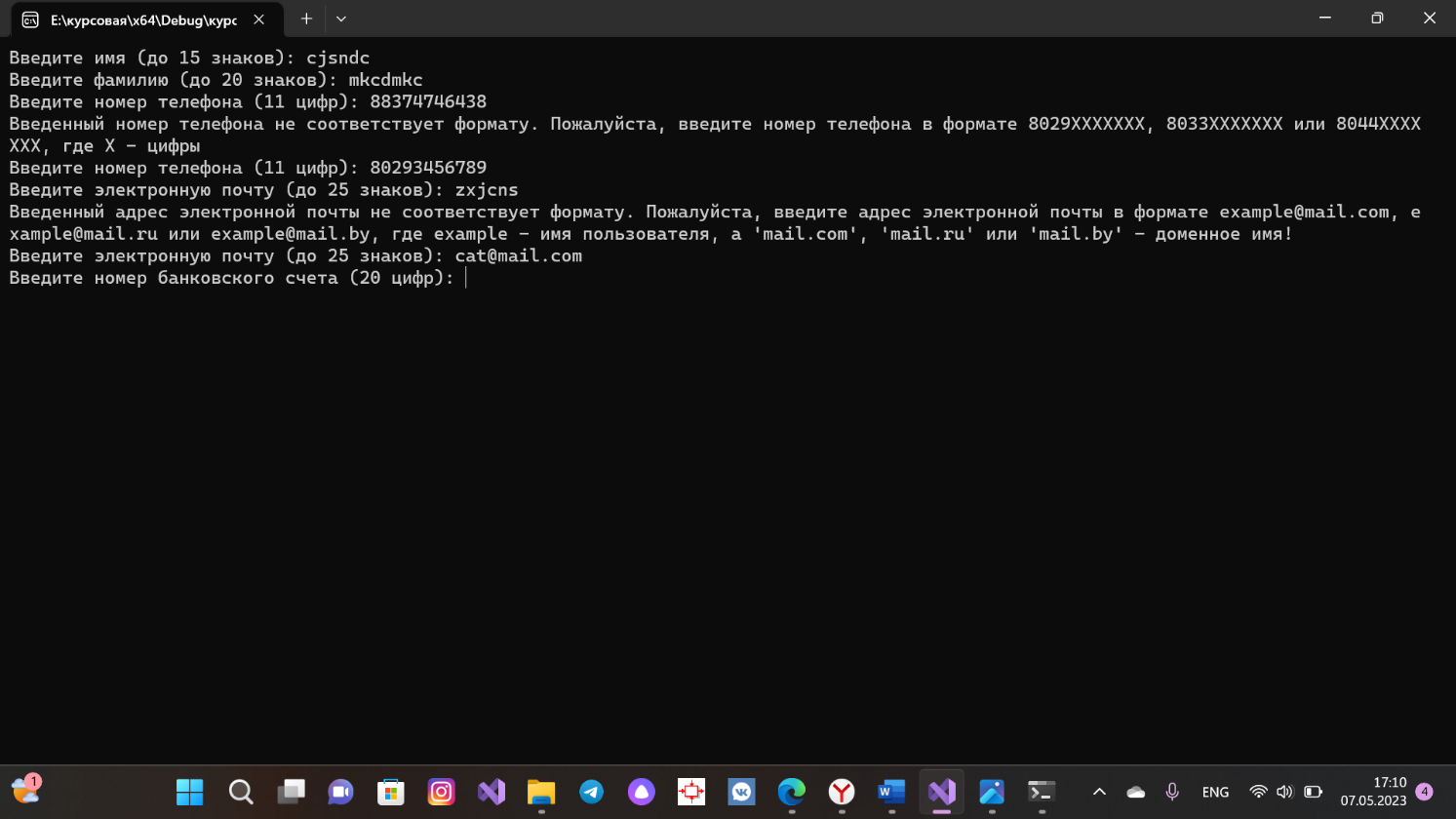


Рисунок 3.4 – Ввод некорректного формата номера телефона

При записи электронной почты предусмотрен определенный формат ввода (рисунок 3.5).

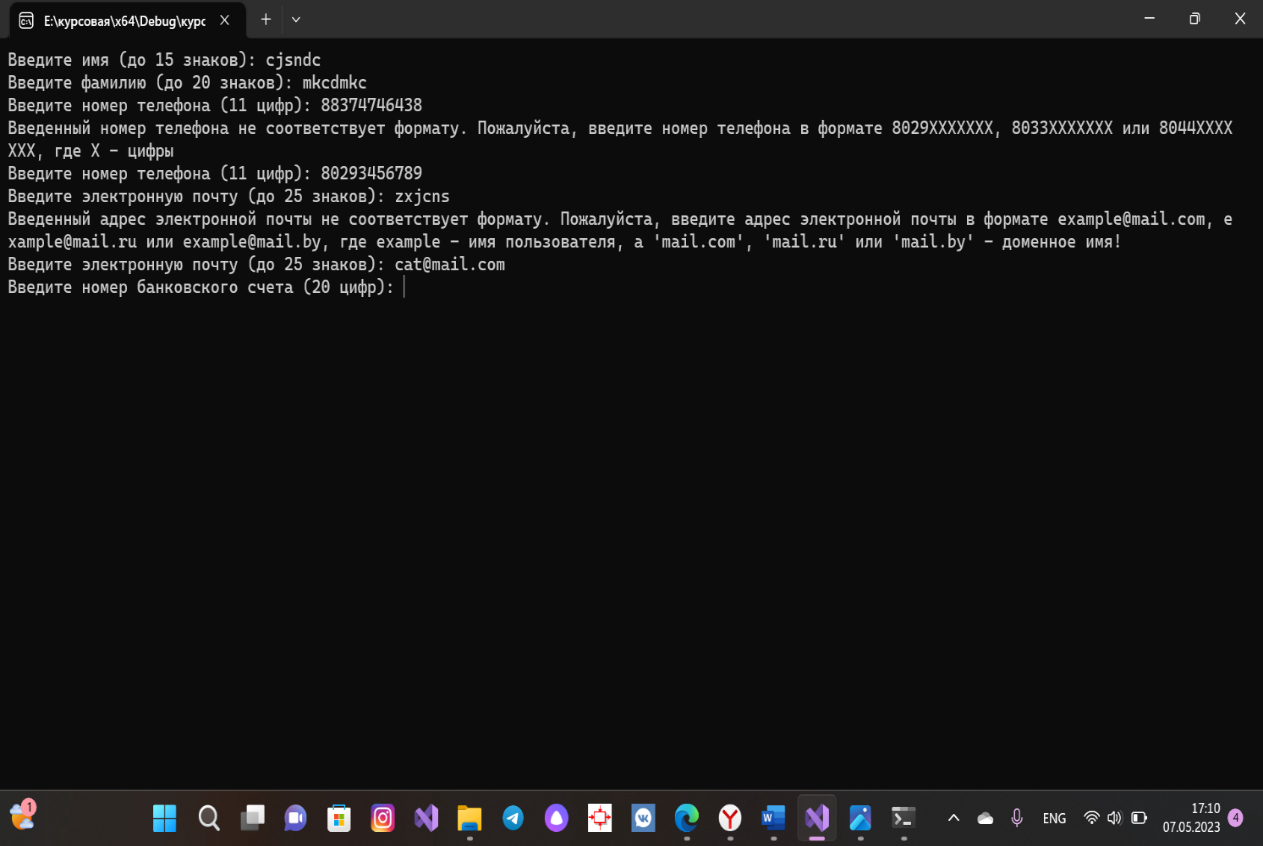


Рисунок 3.5 – Ввод некорректного формата электронной почты

Для ввода даты и времени бронирования предусмотрены следующие ограничения: дата и время бронирования должны быть позже текущей даты и времени (рисунок 3.6), недопустим ввод букв, а также выход за часовые рамки.

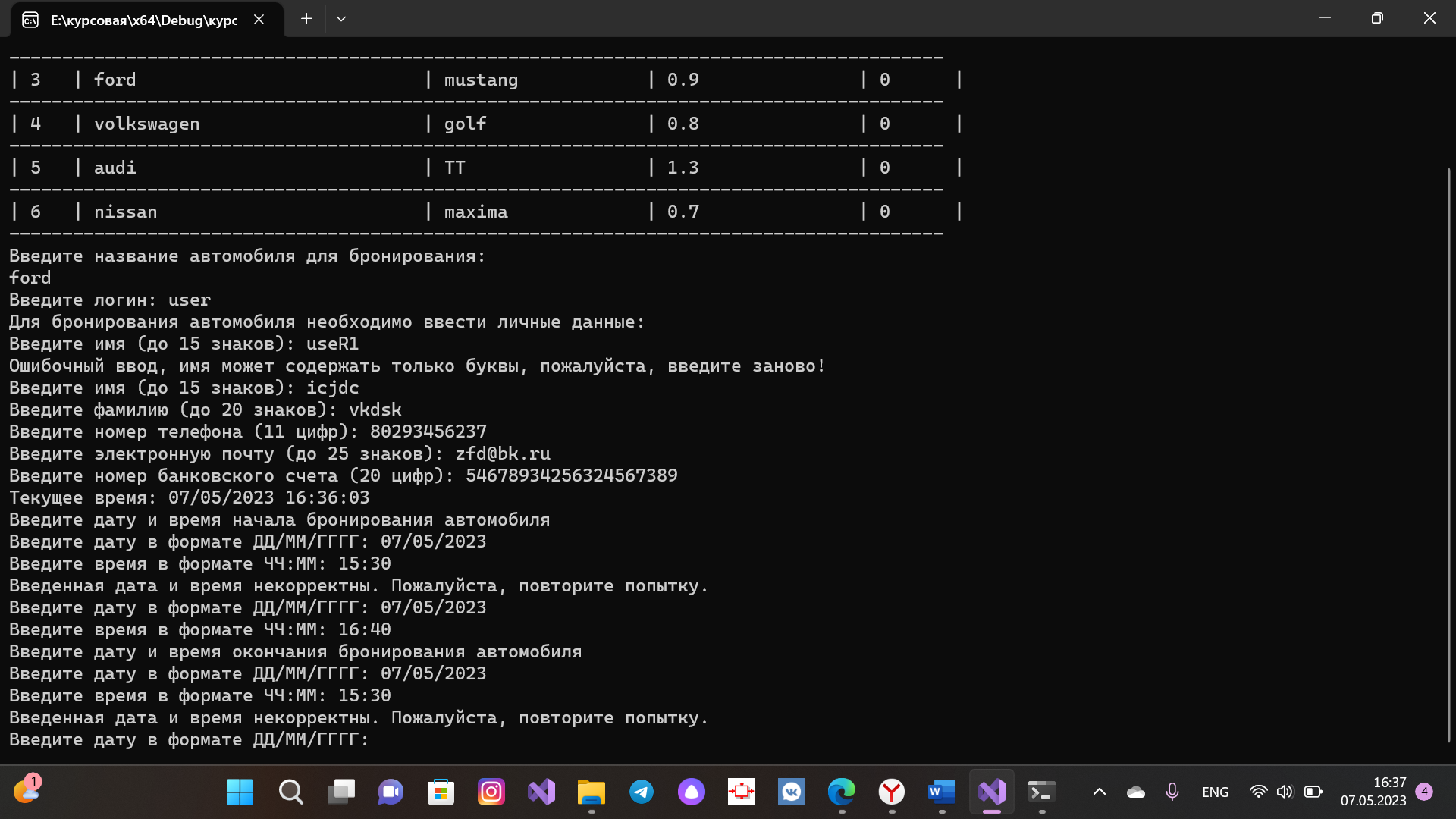


Рисунок 3.6 – Ввод некорректной даты

При попытке удаления записи пользователь получается уведомление, об уверенности в том, что данную запись нужно удалить (рисунок 3.8).

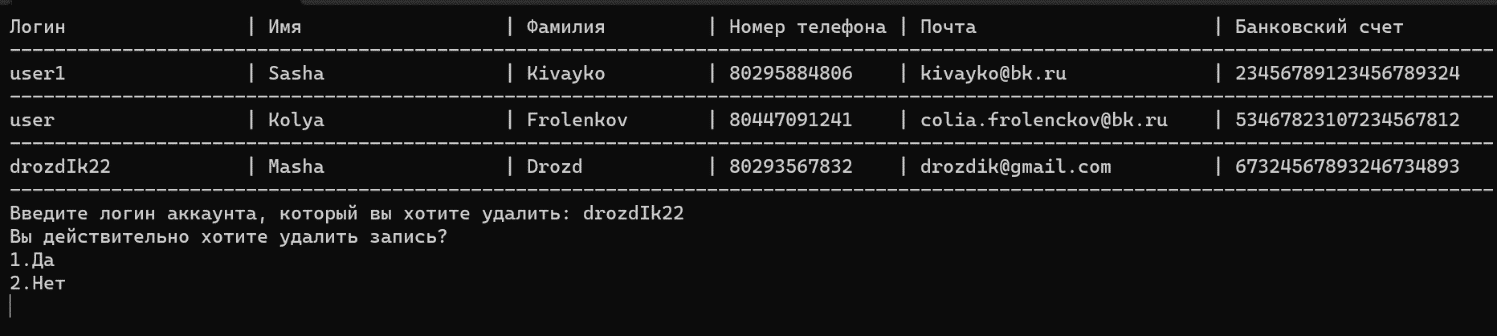


Рисунок 3.8 – Уведомление об уверенности удаления данных

В случае если пользователь пытается начать с работу с данными, которые еще не были записаны в файл, он получает сообщение об ошибке (рисунок 3.9).

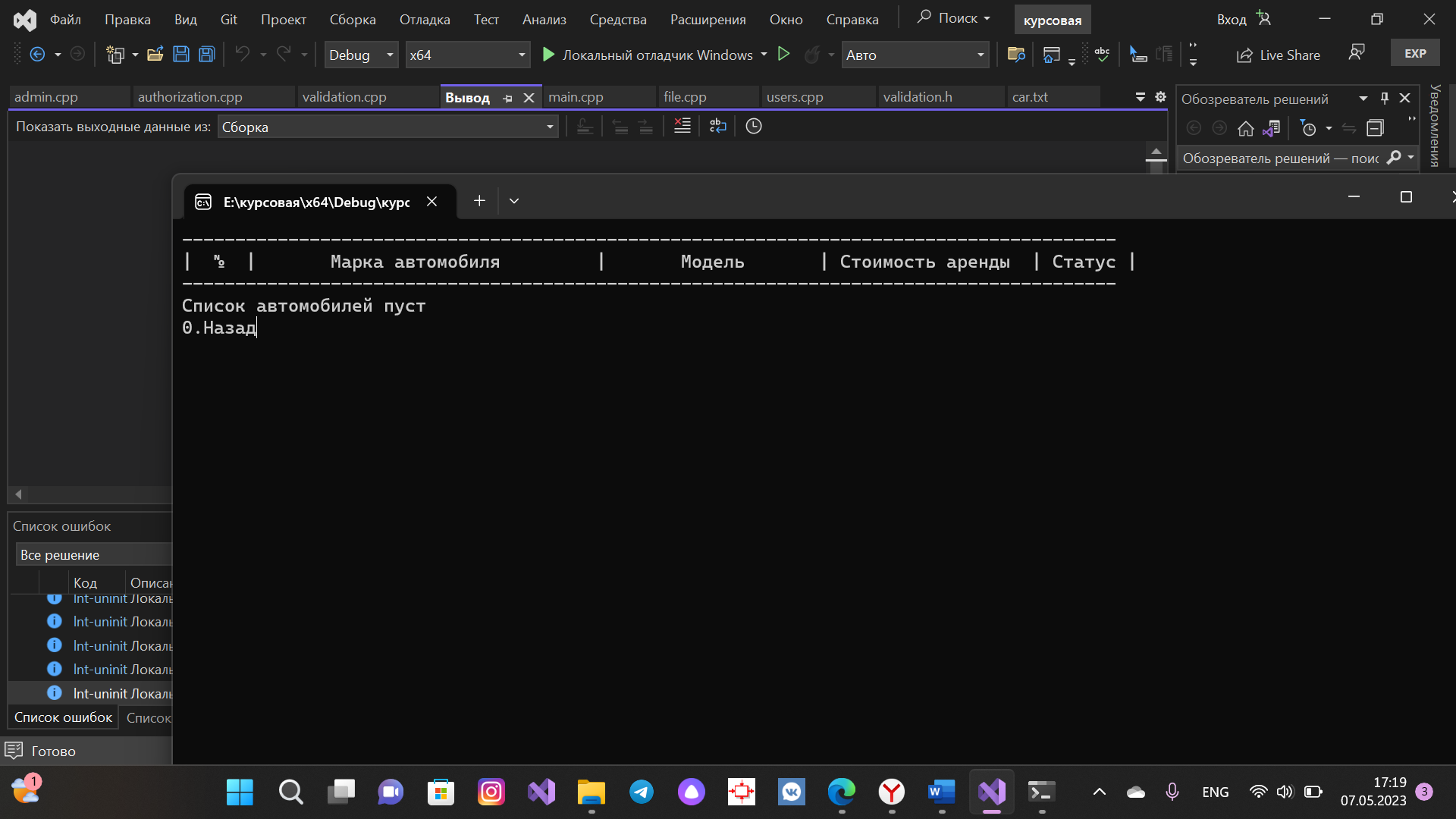


Рисунок 3.9 – Уведомление об отсутствии информации

При редактировании информации пользователь получает уведомление об уверенности редактирования данной информации (рисунок 3.10).

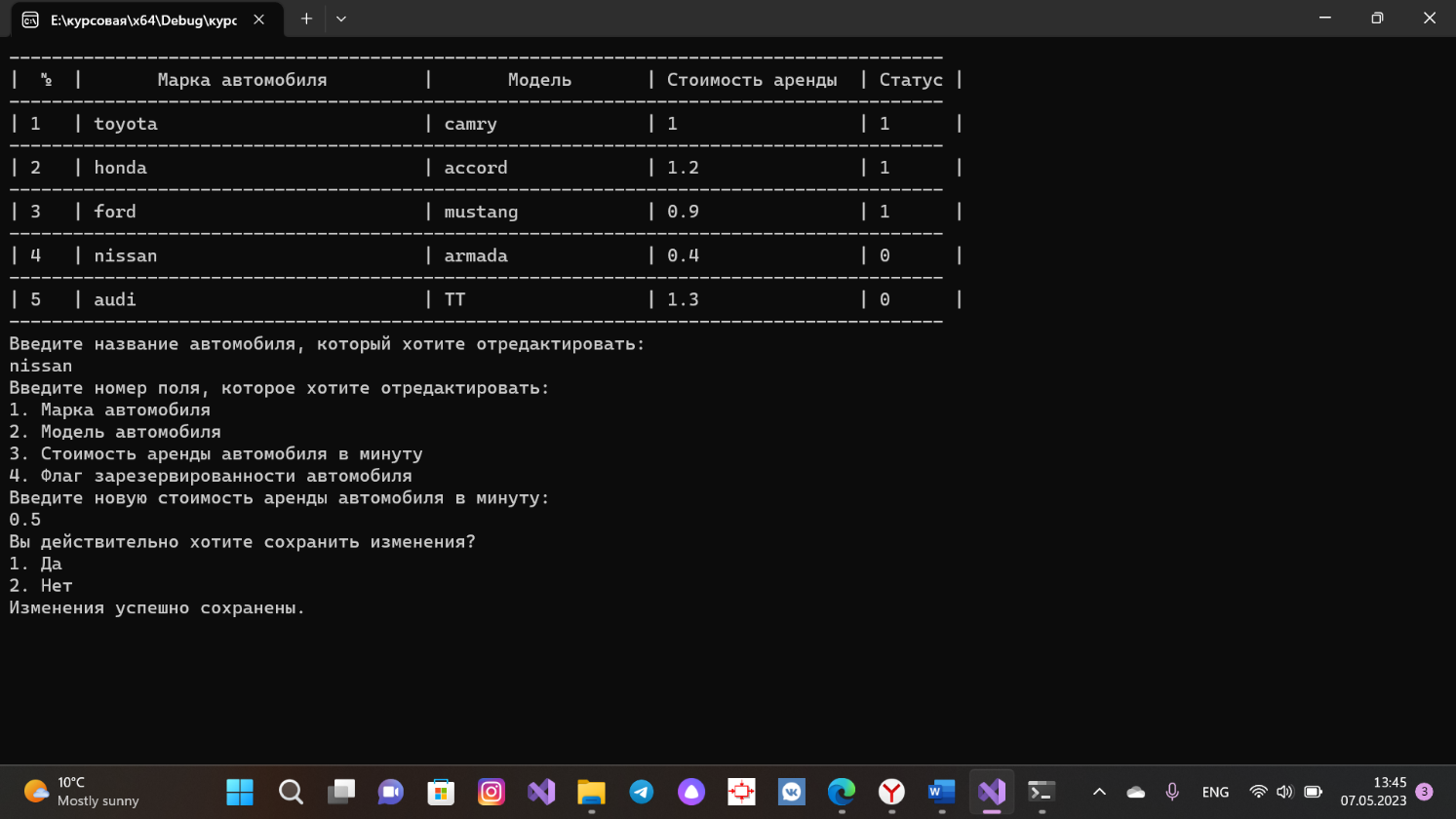


Рисунок 3.10 – Уведомление об уверенности редактирования

В случае, если пользователь попытается забронировать автомобиль, которого нет в наличии или уже забронирован, то получит сообщение об ошибке (рисунок 3.11).



Рисунок 3.11 – Уведомление об отсутствии информации

# ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ПРИМЕР

В начале работы с программой появляется главное меню, которое предоставляет выбор входа в программу (рисунок 4.1).



Рисунок 4.1 – Главное меню

## 4.1 Авторизация

Для работы программы предусмотрены два модуля: модуль администратора и модуль пользователя. Чтобы получить доступ к нужному модулю, необходимо авторизоваться.

При выборе авторизации в главном меню пользователю необходимо ввести свой логин и пароль. Если введенные данные не совпадают с зарегистрированными в системе, пользователь может попробовать ввести их еще раз.

В зависимости от “Роли”, назначенной пользователю в соответствующей структуре, он получает доступ либо к модулю пользователя, либо к модулю администратора. На рисунке 4.2 приведен пример входа в учетную запись под ролью пользователя.

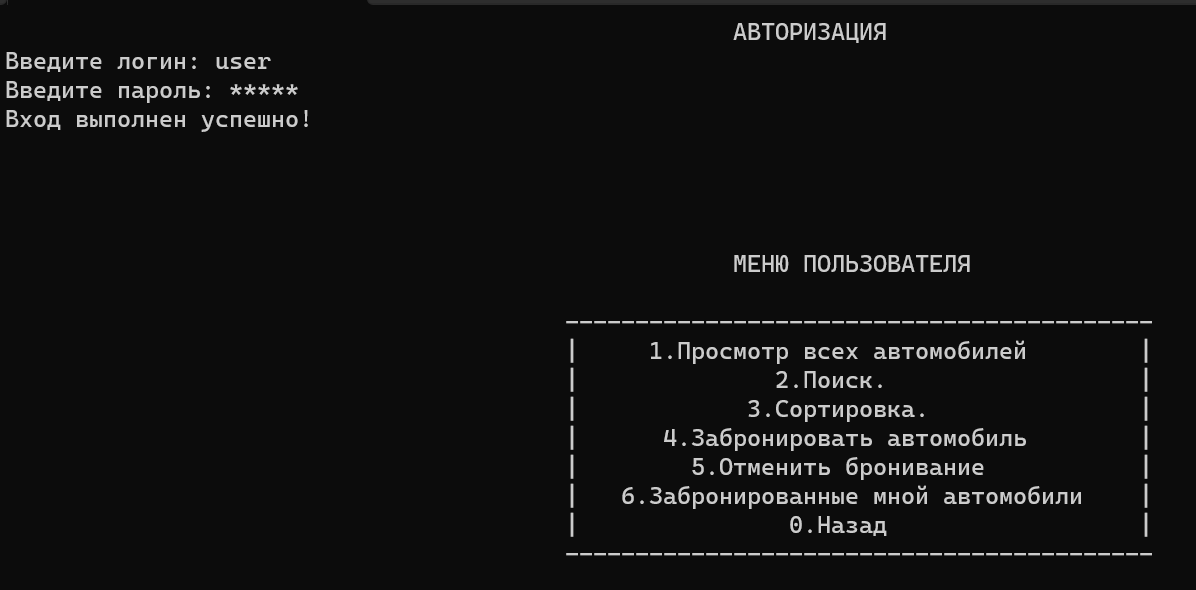
****

Рисунок 4.2 – Вход в учетную запись как пользователь

На рисунке 4.3 приводится результат входа в учетную запись как администратор.

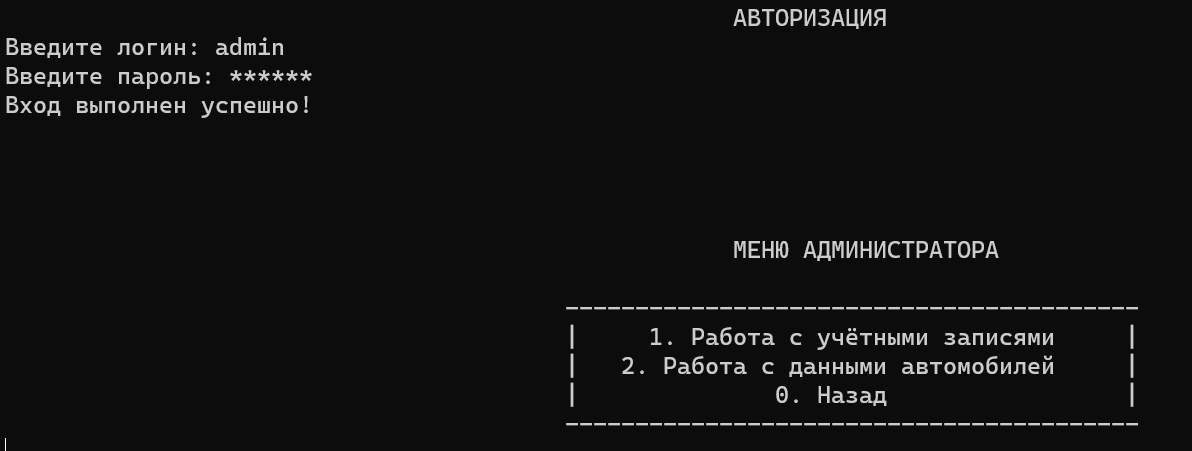


Рисунок 4.3 – Вход в учетную запись как администратор

## Модуль администратора

После успешного входа под именем администратора пользователю предоставляется меню администратора (рисунок 4.4).

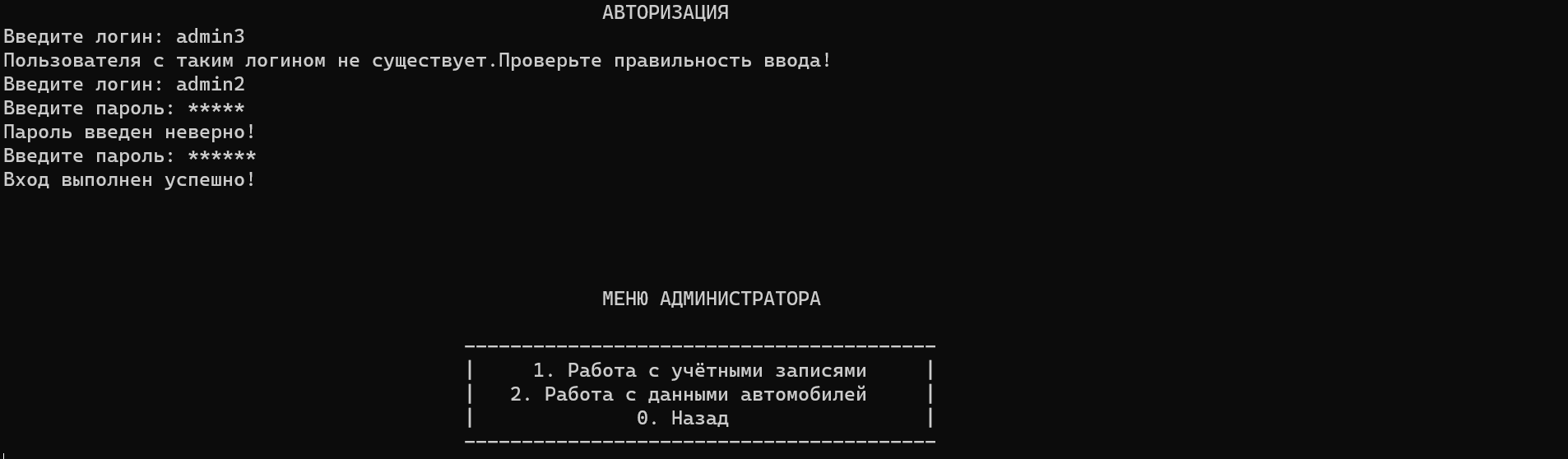


Рисунок 4.4 – Меню администратора

При выборе работы с учетными записями администратору предоставляются следующие действия (рисунок 4.5).

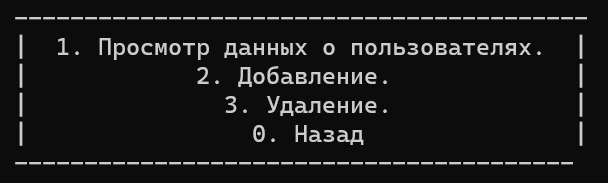


Рисунок 4.5 – Меню работы с учетными записями

При выборе просмотра данных о пользователях на экране появится список пользователей и забронированные ими автомобили (рисунок 4.6).

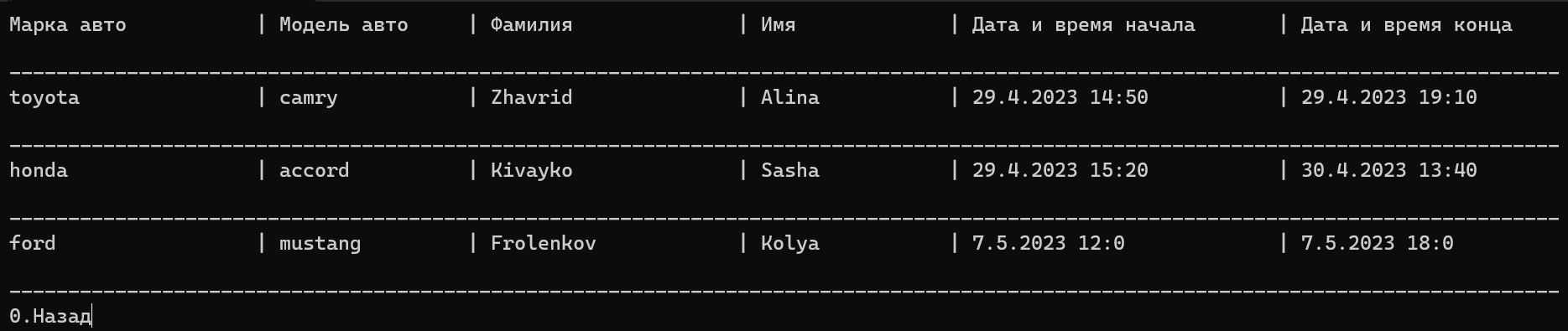


Рисунок 4.6 – Пример вывода информации о пользователях

При добавлении нового пользователя необходимо ввести личные данные пользователя и придумать логин и пароль (рисунок 4.7).



Рисунок 4.7 – Добавление нового пользователя в систему

При попытке удаления аккаунта у администратора запрашивается подтверждения для совершения данного действия (рисунок 4.8).

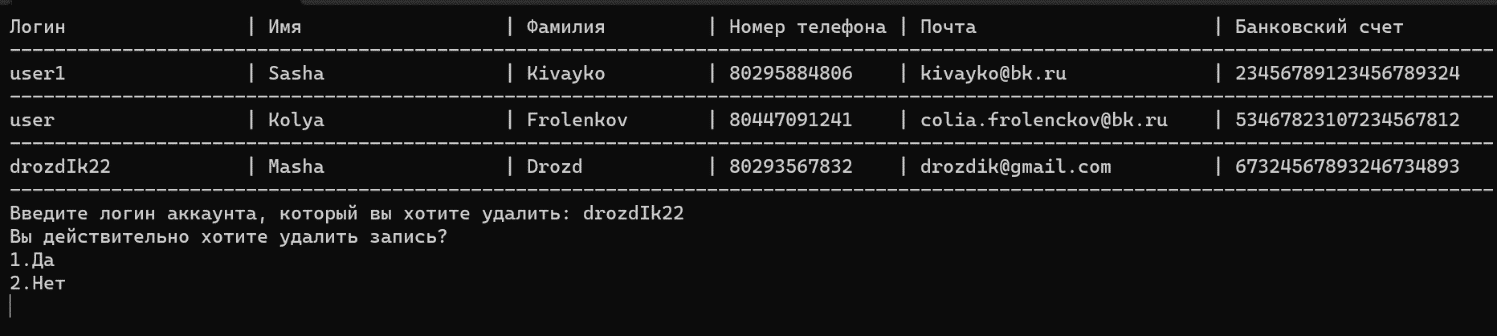


Рисунок 4.8 – Удаление аккаунта

При выборе работы с данными автомобилей администратору предоставляются следующие действия (рисунок 4.9).

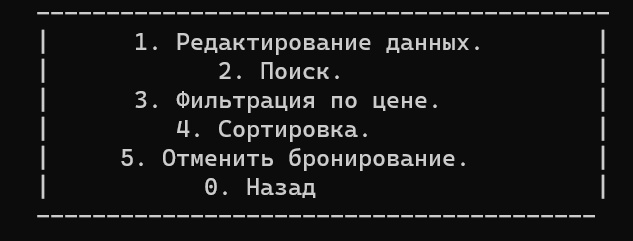


Рисунок 4.9 – Меню работы с данными автомобилей

При выборе варианта “Редактирование данных” будет выведено следующее меню (рисунок 4.10).

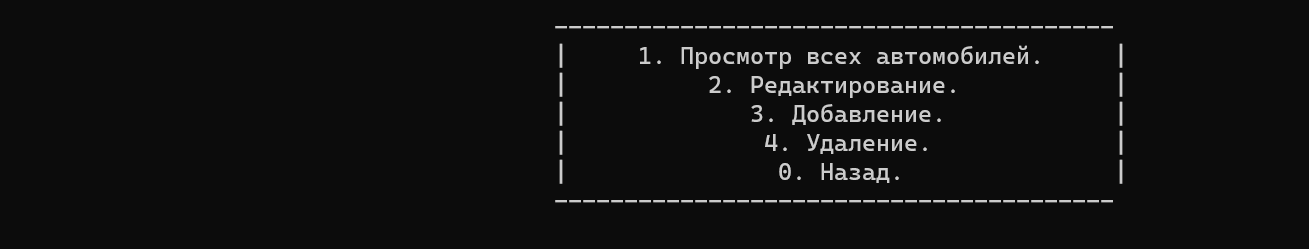


Рисунок 4.10– Меню редактирования данных

При выборе варианта “Просмотр всех автомобилей” на экран будут выведены все автомобили, имеющиеся в автопарке системы (рисунок 4.11).

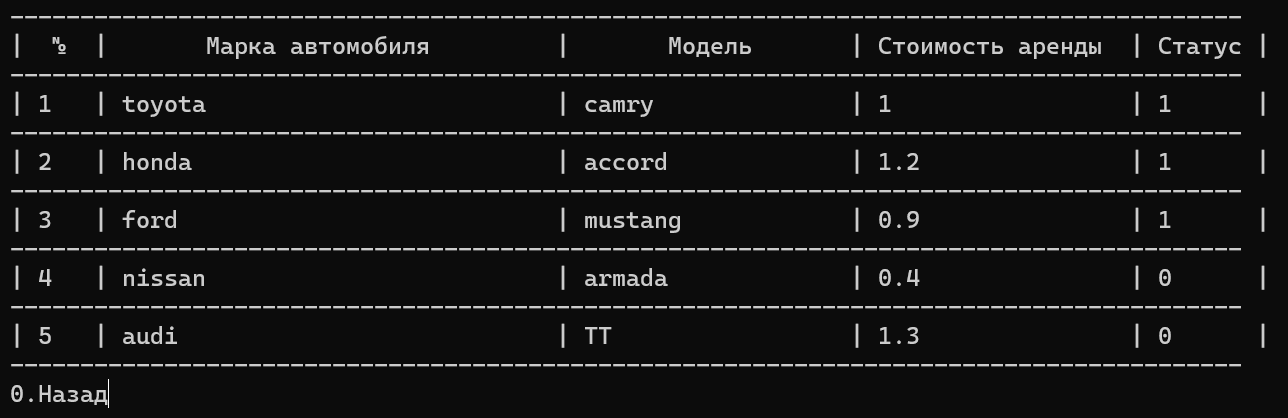


Рисунок 4.11 – Просмотр всех автомобилей

При редактирования данных об автомобиле предоставлены следующие поля (рисунок 4.12):

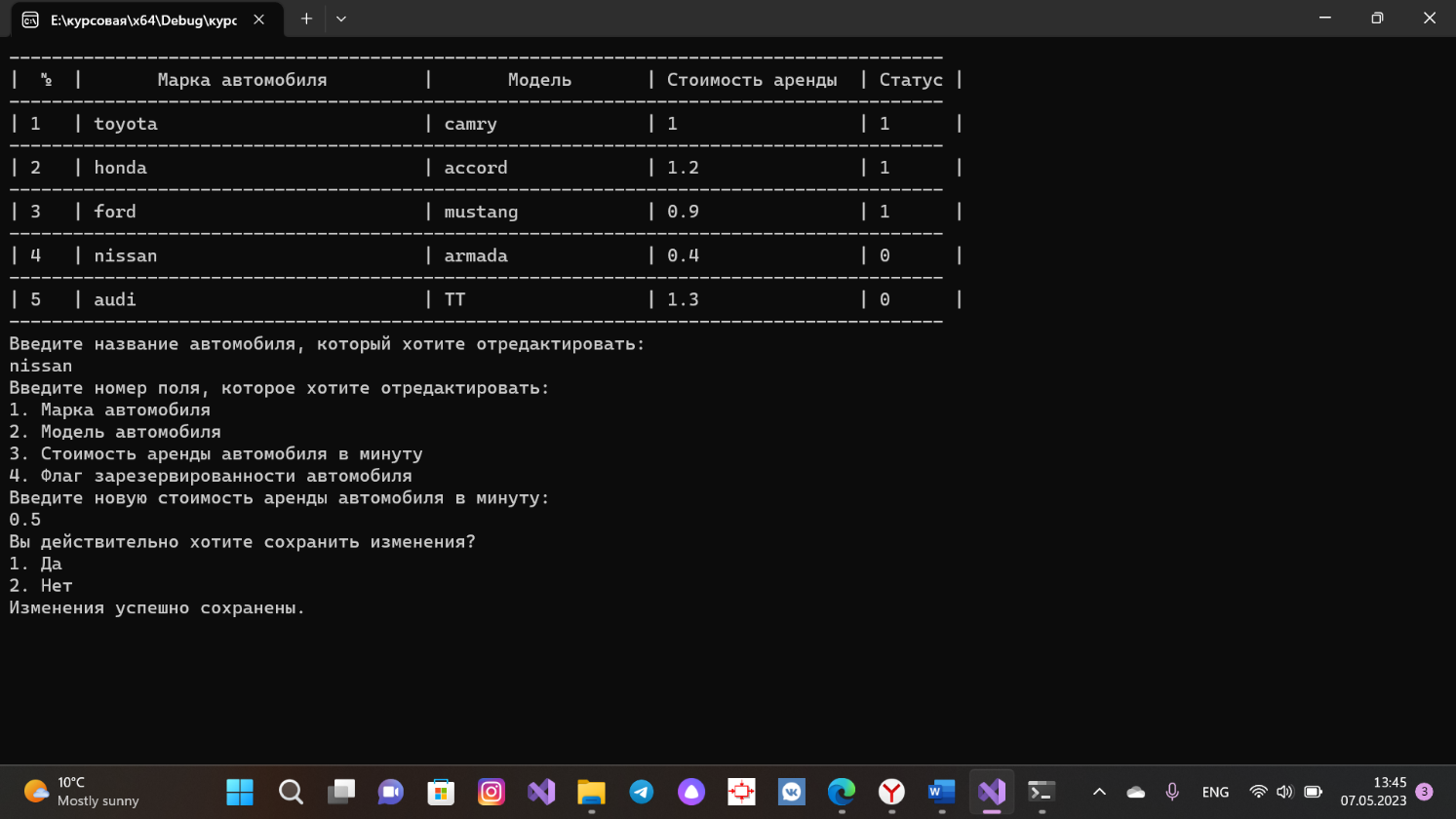


Рисунок 4.12 – Пример редактирования данных об автомобиле

Добавление новой записи об автомобиле предлагает ввести данные для записи нового автомобиля (рисунок 4.13).

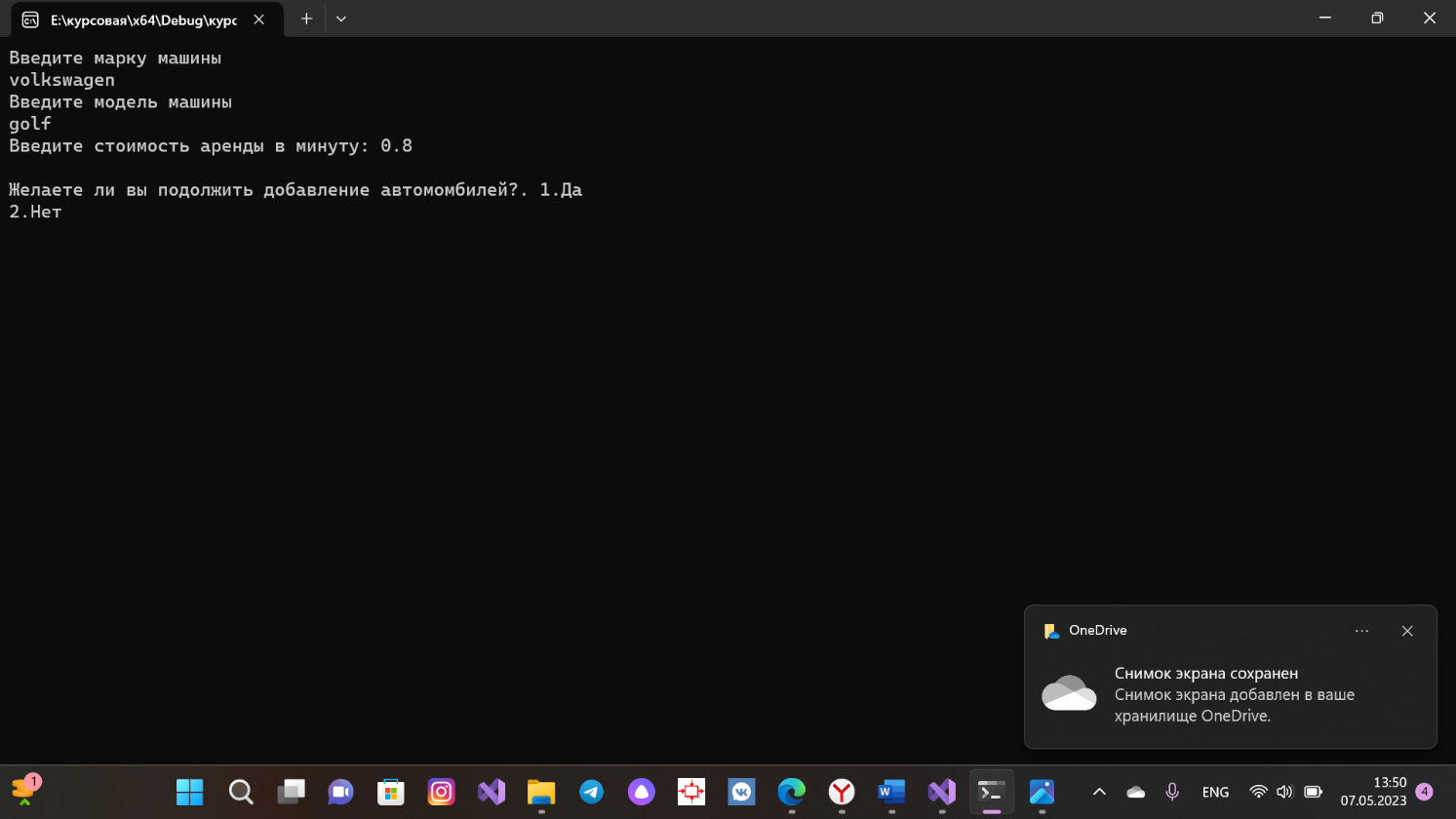


Рисунок 4.13 – Добавление записи об автомобиле

При попытке удаления записи об автомобиле у администратора запрашивается подтверждения для совершения данного действия (рисунок 4.14).

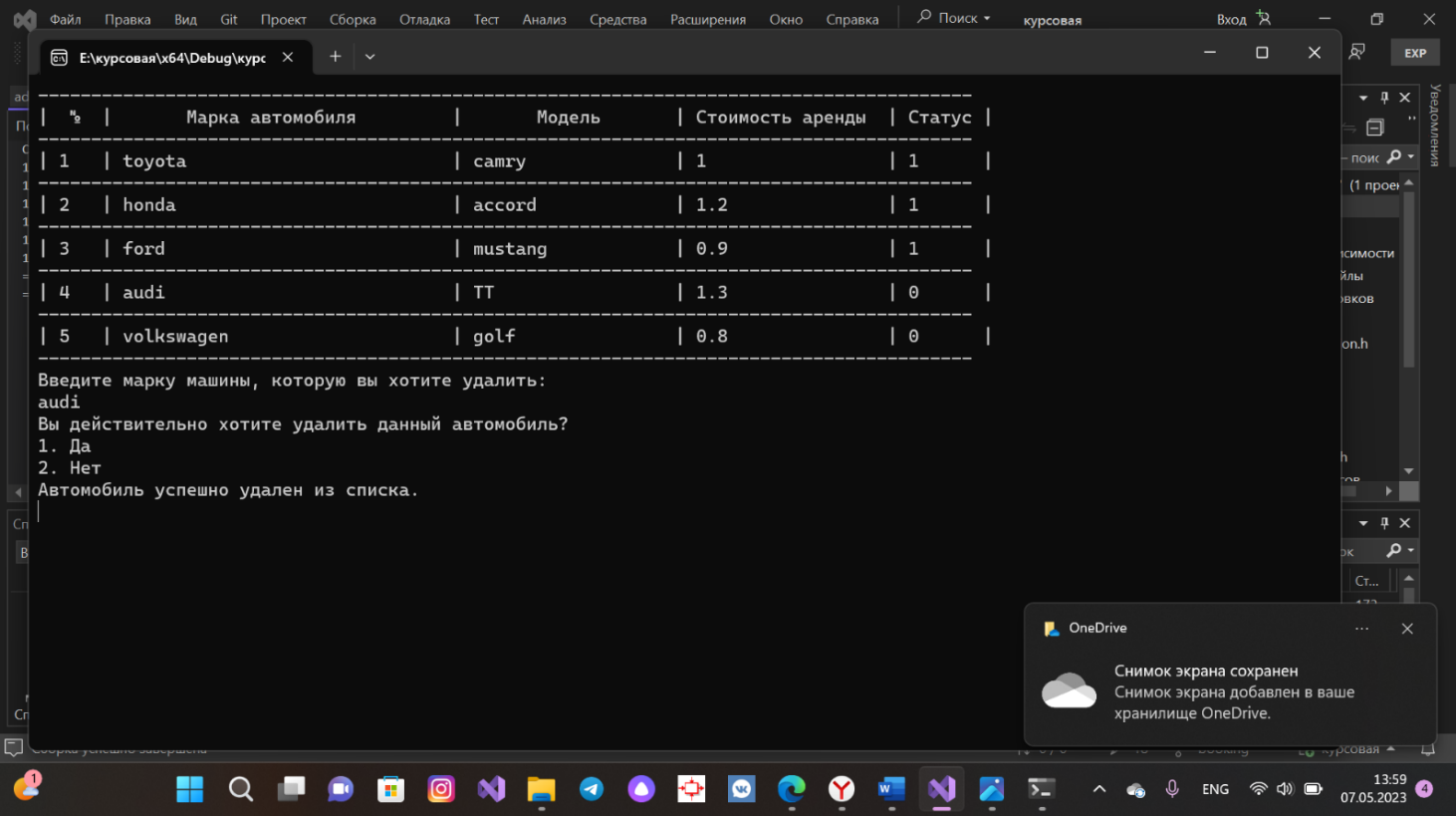


Рисунок 4.14– Удаление автомобиля

Поиск автомобилей (рисунок 4.15) может осуществляться по следующим критериям.

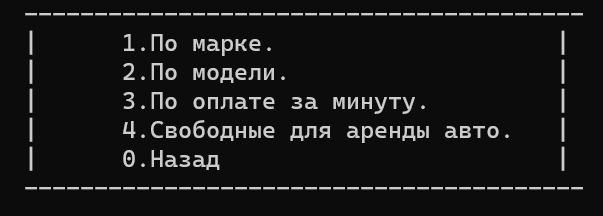


Рисунок 4.15 – Критерии поиска автомобиля

Пример результата поиска информации об автомобиле представлен на рисунке 4.16.



Рисунок 4.16 – Пример результата поиска автомобиля

При выборе варианта “Фильтрация по цене” необходимо ввести ценовой диапазон, в результате чего будет представлена информация о доступных автомобилях в данном ценовом диапазоне (рисунок 4.17).

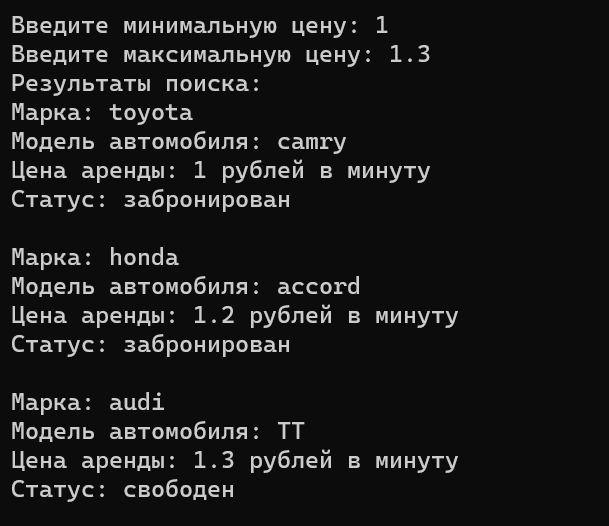


Рисунок 4.17 – Пример фильтрации данных

Сортировка автомобилей (рисунок 4.18) осуществляется по следующим критериям.

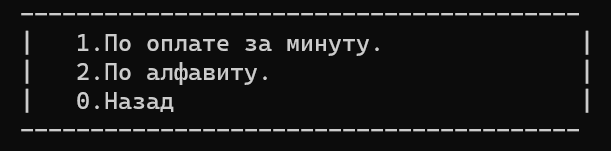


Рисунок 4.18 – Критерии сортировки автомобилей

Пример сортировки автомобилей отображен на рисунке 4.19 и 4.20.

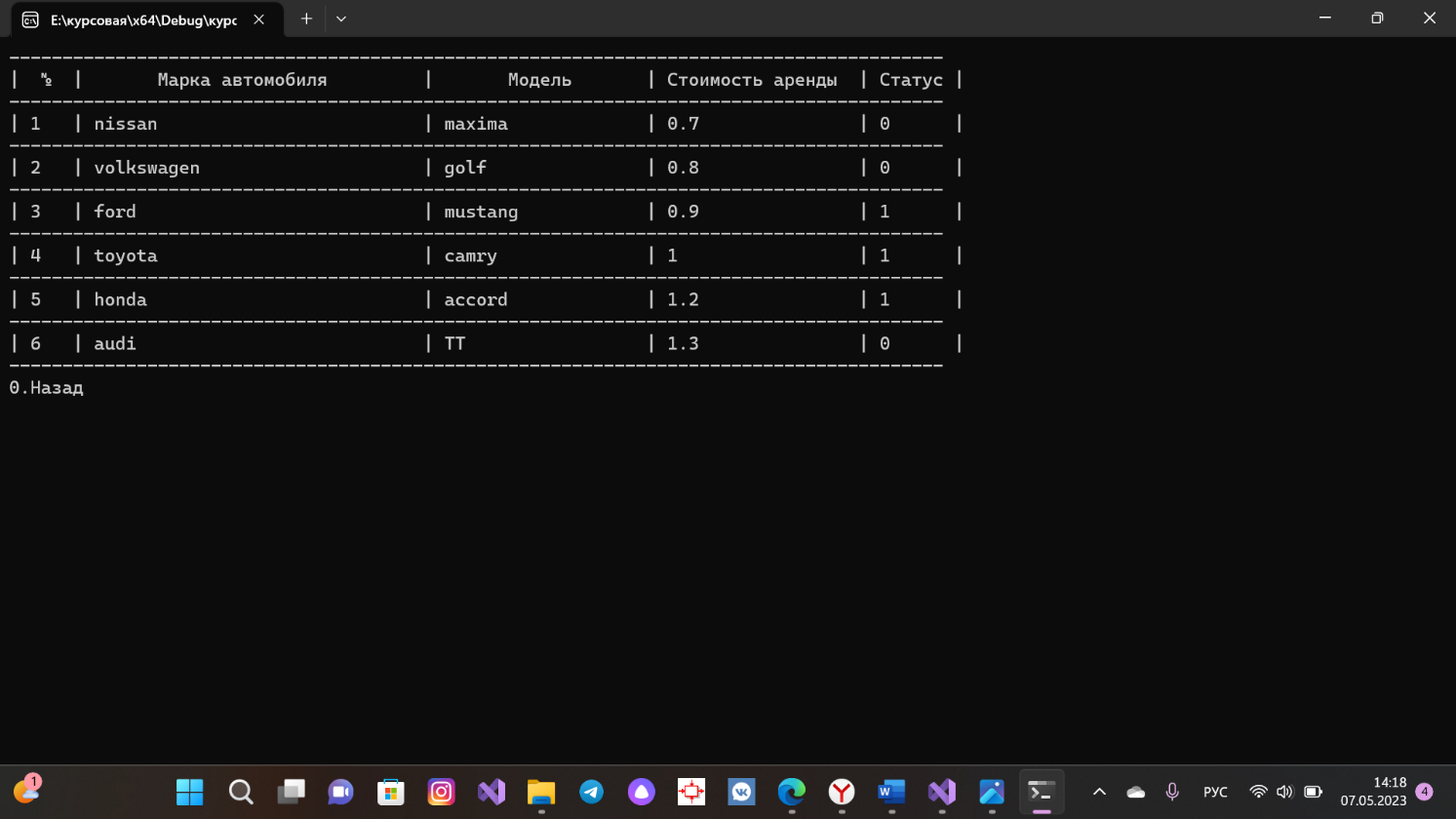


Рисунок 4.19 – Пример сортировки автомобилей по стоимости за минуту

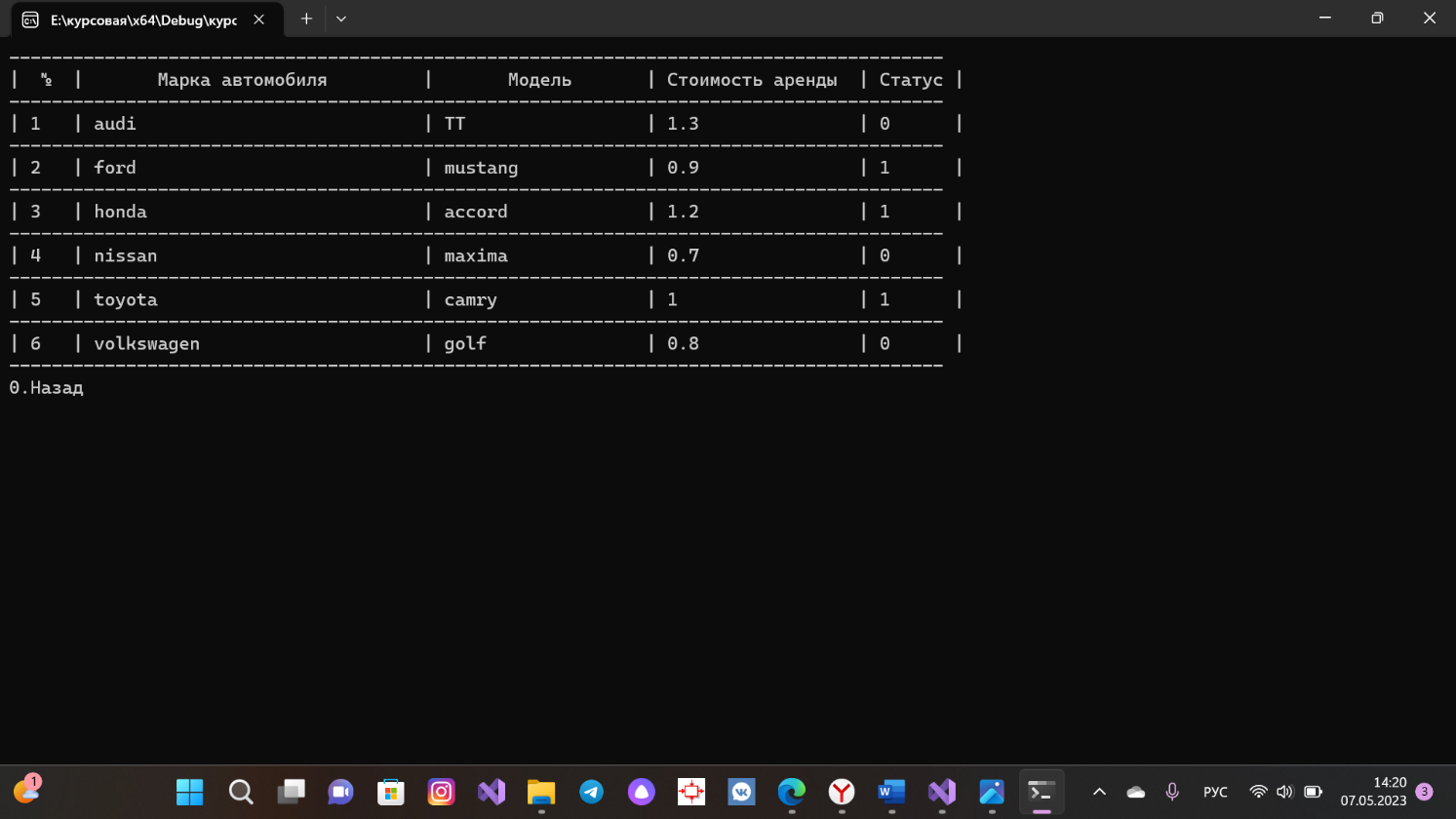


Рисунок 4.20 – Пример сортировки автомобилей по алфавиту

## 4.3 Модуль пользователя

После успешного входа в учетную запись под ролью пользователя ему предоставляется меню, в котором перечислены действия, доступные для него (рисунок 4.21).



Рисунок 4.21 – Меню пользователя

Вывод данных об автомобилях, критерии поиска и сортировки аналогичны с идентичными действиями администратора.

Выполнение индивидуального задания представляет собой определение текущего времени, ввод начала и конца бронирования автомобиля, а также расчет стоимости в соответствии с введенными данными (рисунок 4.22).

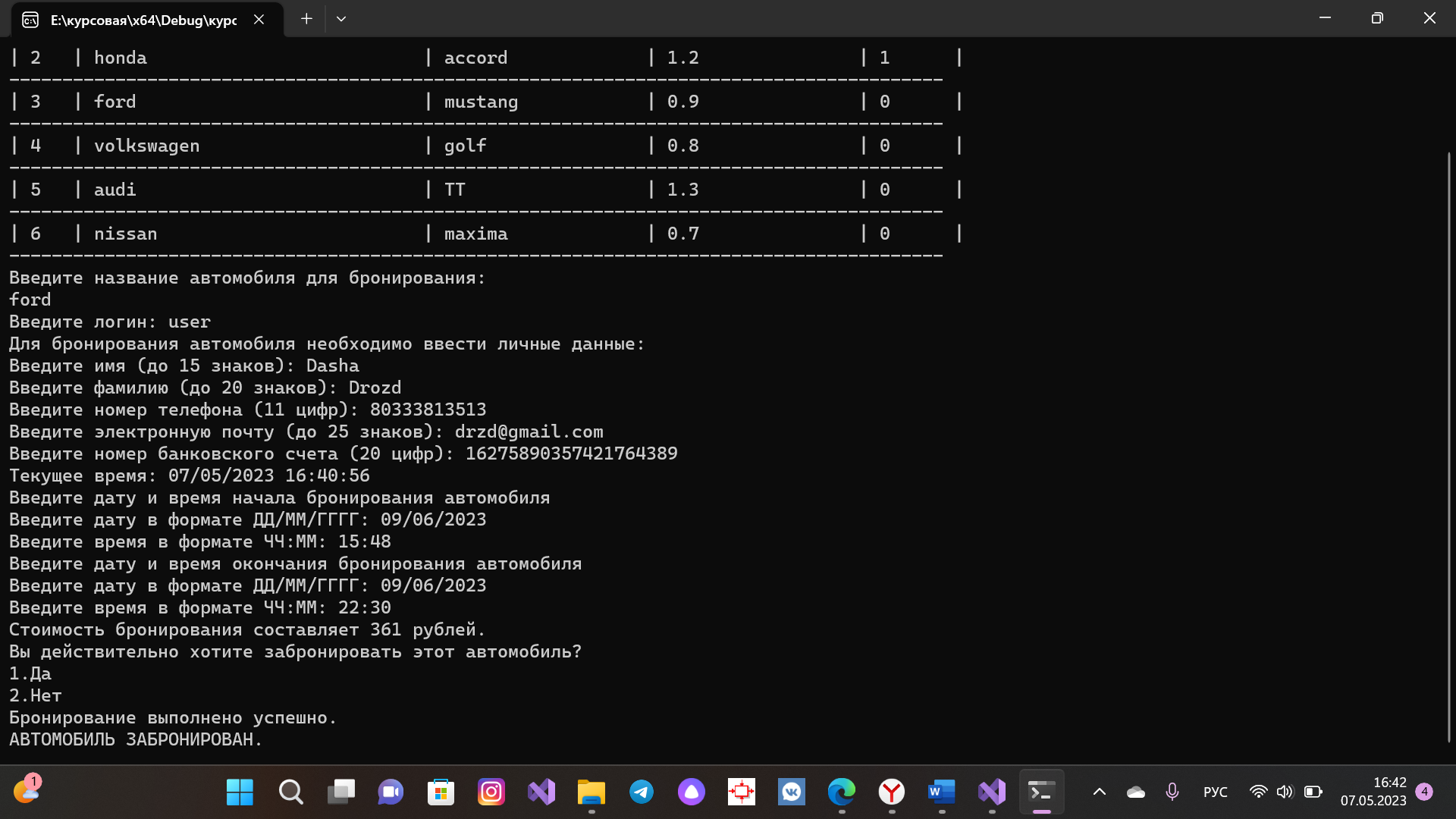


Рисунок 4.22 – Выполнение индивидуального задания

При выборе варианта “Забронировать автомобиль” необходимо выбрать подходящий автомобиль, ввести персональные данные и время бронирования, в результате чего будет рассчитана стоимость аренды автомобиля за указанный промежуток времени (рисунок 4.23).

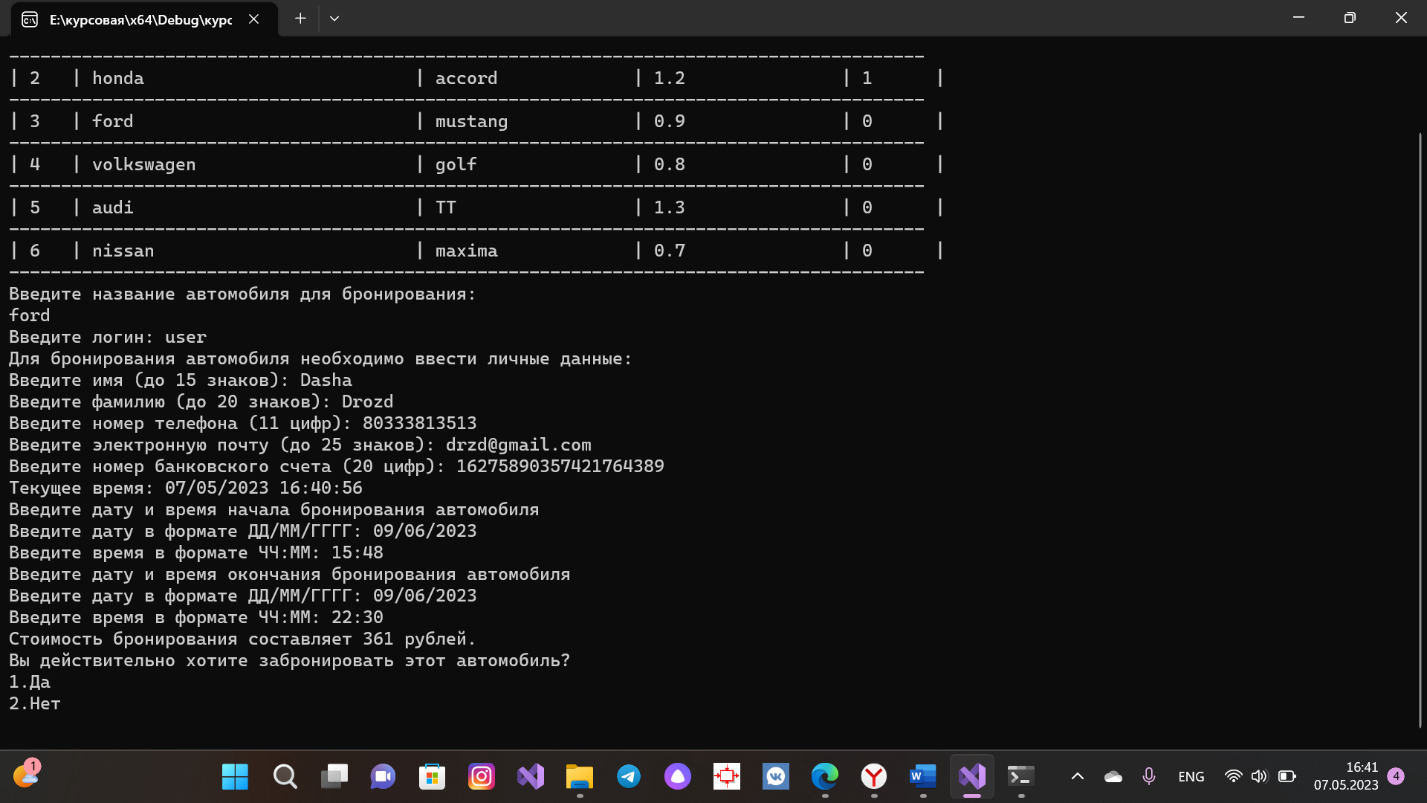


Рисунок 4.23 – Пример бронирования автомобиля

## 4.4 Регистрация

Модуль регистрации дает возможность самому создать свой аккаунт из главного меню для дальнейшего пользования программой и модуля соответствующей роли включительно (рисунок 4.24).

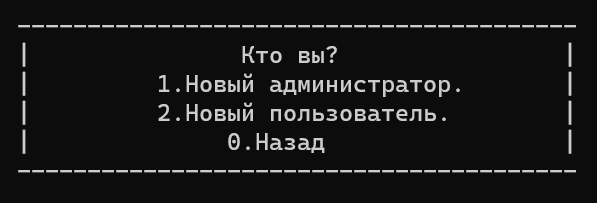


Рисунок 4.24 – Меню регистрации

Сам процесс регистрации приведен на рисунке 4.25.

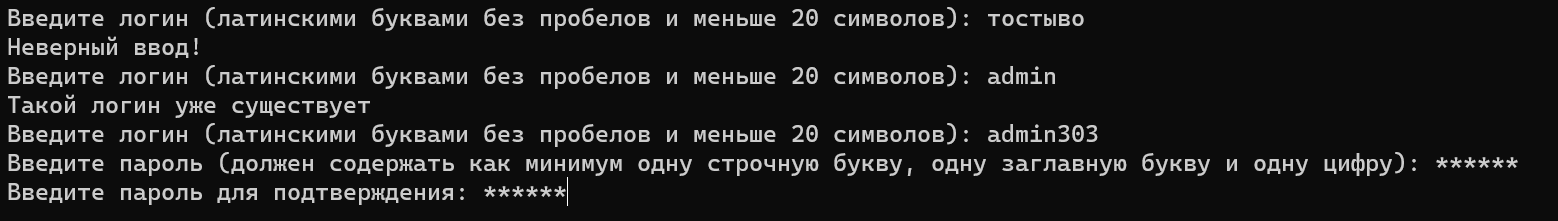


Рисунок 4.25 – Регистрация

**ЗАКЛЮЧЕНИЕ**

В ходе разработки программного средства для бронирования автомобилей были изучены основные принципы организации процесса бронирования, а также были рассмотрены различные варианты реализации данной задачи. Был создан удобный и быстрый доступ к информации об автомобилях, их редактированию и просмотру.

Созданные пользовательские функции позволяют эффективно работать с имеющимися данными посредством ввода, просмотра, редактирования, удаления, поиска, сортировки и других операций.

Были рассмотрены исключительные ситуации и их обработка с помощью функций проверок (на основе языка С++).

Функциональная схема работы программы представлена в виде алгоритмов некоторых функций.

Программное средство имеет понятный графический интерфейс, позволяющий с минимальными навыками использования персонального компьютера использовать преимущества бронирования автомобилей в цифровом виде. Таким образом, система готова к использованию и может обеспечить пользователя поступлением необходимой информации о доступных автомобилях и ценах на них. Благодаря разработанным функциям, пользователь может быстро и удобно найти подходящий автомобиль для аренды, а также произвести бронирование и оплату онлайн.

Для понимания логики программного средства были разработаны модели бизнес-процессов предметной области в нотациях IDEF0 и BPMN, проектные решения программного средства на основе языка UML.

Как итог данной работы можно назвать функциональное приложение для бронирования автомобилей, которое позволяет быстро и удобно находить доступные автомобили и осуществлять их бронирование, а также контролировать процесс бронирования.

В дальнейшем система может быть доработана с добавлением новых функций, например, уведомлений об изменении цен на автомобили или наличии новых моделей в парке аренды. Также можно реализовать возможность оценки и комментирования автомобилей.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методология IDEF0 [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО Лабораторная работа №2.docx](file:///C:\Users\Анюта\трпо\ТРПО%20Лабораторная%20работа%20№2.docx)
2. Методическое пособие для выполнения лабораторной роботы «Язык Uml. Диаграммы UML» [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО - Лабораторные работы №7-8.doc](file:///C:\Users\Анюта\трпо\ТРПО%20-%20Лабораторные%20работы%20№7-8.doc)x
3. Методическое пособие для выполнение лабораторной работы «Моделирование бизнес-процессов на основе нотации BPMN в BizagiProcess Modeler» [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО Лабораторная работа №4 (2).docx](file:///C:\Users\Анюта\трпо\ТРПО%20Лабораторная%20работа%20№4%20(2).docx)
4. IDEF0 [Электронный ресурс]. –Электронные данные. –Режим

доступа : <https://ru.wikipedia.org/wiki/IDEF0>

1. ТРПО – Курсовое проектирование (работа) - Пособие [Электронный ресурс]. – Режим доступа: <https://cloud.mail.ru/public/oaoe/asWuomL8y>.
2. https://habr.com/ru/post/504008/ «Алгоритмы и структуры данных»
3. Куклина И.Г, Информационные технологии и платформы разработки информационных систем: Куклина И.Г. – 2018. – 283с.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Листинг кода**

void viewReservations(string phoneNumber) { //Просмотр информации о забронированном пользователем автомобиле

int press;

ifstream logfile;

logfile.open("reservations.log");

if (logfile.is\_open()) {

string line;

bool found = false;

while (getline(logfile, line)) {

stringstream ss(line);

string carModel, carType, surname, name, mail, number, startDate, startTime, endDate, endTime;

ss >> carModel >> carType >> surname >> name >> mail >> number >> startDate >> startTime >> endDate >> endTime;

if (number == phoneNumber) {

found = true;

cout << "Марка автомобиля: " << carModel << endl;

cout << "Модель автомобиля: " << carType << endl;

cout << "Дата начала аренды: " << startDate << endl;

cout << "Время начала аренды: " << startTime << endl;

cout << "Дата окончания аренды: " << endDate << endl;

cout << "Время окончания аренды: " << endTime << endl << endl;

}

}

if (!found) {

cout << "Номер телефона не найден в логе." << endl;

}

logfile.close();

}

else {

cout << "Ошибка открытия файла для чтения!" << endl;

}

cout << "0.Назад";

do {

press = \_getch();

} while (!(press >= 48 && press <= 54));

system("CLS");

}

void logReservation(cars&car, users&user, int startDay, int startMonth, int startYear, int startHour, int startMinute, int endDay, int endMonth, int endYear, int endHour, int endMinute) { //Запись данных о пользователе и забронированном автомобиле

ofstream logfile;

logfile.open("reservations.log", ios\_base::app);

if (logfile.is\_open()) {

logfile << car.model << " " << car.carType << " " << user.surname << " " << user.name << " " << user.mail << " " << user.number << " "

<< startDay << "." << startMonth << "." << startYear << " " << startHour << ":" << startMinute << " "

<< endDay << "." << endMonth << "." << endYear << " " << endHour << ":" << endMinute << endl;

logfile.close();

}

else {

cout << "Ошибка открытия файла для записи!" << endl;

}

}

void reserveCar(cars\* car, int NumberOfCar, users\*& user, int& NumberOfUsers, string login) { //Бронирование автомобиля

cout << "----------------------------------------------------------------------------------------\n"

<< "| № | Марка автомобиля | Модель | Стоимость аренды | Статус |\n"

<< "----------------------------------------------------------------------------------------\n";

if (NumberOfCar == 0) cout << "Список автомобилей пуст" << endl;

else {

for (int i = 0; i < NumberOfCar; i++) {

cout << "| " << setw(3) << left << i + 1 << " | "

<< setw(30) << left << car[i].model << " | "

<< setw(18) << left << car[i].carType << " | "

<< setw(17) << left << car[i].rentPrice << " | "

<< setw(6) << left << car[i].is\_reserved << " |\n";

cout << "----------------------------------------------------------------------------------------\n";

}

}

string model;

int press;

cout << "Введите название автомобиля для бронирования:\n";

cin >> model;

int carIndex = findCarByModel(car, NumberOfCar, model);

if (carIndex == -1) {

cout << "Данной модели автомобиля нет в наличии или он уже забронирован.\n";

return;

}

bool userExists = false;

do {

cout << "Введите логин: ";

cin >> login;

string existingLogin = getLoginFromFile(login);

if (existingLogin != "") {

userExists = true;

login = existingLogin; }

else {

cout << "Пользователь с таким логином не найден. Попробуйте еще раз.\n"; }

} while (!userExists);

cout << "Для бронирования автомобиля необходимо ввести личные данные:\n";

addDataOfUsers(login, user, NumberOfUsers);

int startDay, startMonth, startYear, startHour, startMinute;

getStartDateTime(startDay, startMonth, startYear, startHour, startMinute);

int endDay, endMonth, endYear, endHour, endMinute;

getEndDateTime(startDay, startMonth, startYear, startHour, startMinute, endDay, endMonth, endYear, endHour, endMinute);

int totalMinutes = ((endYear - startYear) \* 525600) + ((endMonth - startMonth) \* 43800) + ((endDay - startDay) \* 1440) + ((endHour - startHour) \* 60) + (endMinute - startMinute);

float rentPrice;

rentPrice = car[carIndex].rentPrice;

int totalCost = totalMinutes \* rentPrice;

cout << "Стоимость бронирования составляет " << totalCost << " рублей.\n";

cout << "Вы действительно хотите забронировать этот автомобиль?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49) {

car[carIndex].is\_reserved = true;

cout << "Бронирование выполнено успешно.\n";

if (car[carIndex].is\_reserved == true) {

cout << "АВТОМОБИЛЬ ЗАБРОНИРОВАН.";

Sleep(3000);

system("cls");

ofstream f(FILE\_LIST, ios::out | ios::trunc);

if (f.is\_open()) {

for (int i = 0; i < NumberOfCar; i++) {

f << setw(25) << left << car[i].model

<< setw(20) << left << car[i].carType

<< setw(7) << left << car[i].rentPrice

<< setw(7) << left << car[i].is\_reserved << endl;

}

f.close();

}

else {

cout << "Ошибка записи в файл\n";

}

logReservation(car[carIndex], user[NumberOfUsers - 1], startDay, startMonth, startYear, startHour, startMinute, endDay, endMonth, endYear, endHour, endMinute);

}

else {

cout << "Бронирование отменено пользователем.\n";

}

}

}

void menuUsers(cars\*& car, int& NumberOfCar,string login, users\*& user, int& NumberOfUsers)//Меню пользователя

{

int press;

bool is\_user = true;

string phoneNumber;

cout << "\n\n\n\n\t\t\t\t\t\t МЕНЮ ПОЛЬЗОВАТЕЛЯ\n\n";

do {

cout << "\t\t\t\t\t------------------------------------------\n"

<< "\t\t\t\t\t| 1.Просмотр всех автомобилей |\n"

<< "\t\t\t\t\t| 2.Поиск. |\n"

<< "\t\t\t\t\t| 3.Сортировка. |\n"

<< "\t\t\t\t\t| 4. Фильтрация по цене. |\n"

<< "\t\t\t\t\t| 5.Забронировать автомобиль |\n"

<< "\t\t\t\t\t| 6.Отменить бронивание |\n"

<< "\t\t\t\t\t| 7.Забронированные мной автомобили |\n"

<< "\t\t\t\t\t| 0.Назад |\n"

<< "\t\t\t\t\t------------------------------------------\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 55));

NumberOfCar = listDataOfCar(car);

system("CLS");

switch (press)

{

case 49:ViewCar(car, NumberOfCar); break;

case 50:menuSearch(car, NumberOfCar); break;

case 51:menuSort(car, NumberOfCar); break;

case 52:filterByPriceRange(car, NumberOfCar); break;

case 53:reserveCar(car,NumberOfCar, user, NumberOfUsers,login); break;

case 54:cancel\_reservation(car, NumberOfCar, is\_user); break;

case 55:{cout << "Введите ваш номер телефона для того, чтобы просмотреть забронированные автомобили "; cin >> phoneNumber; viewReservations(phoneNumber); } break;

case 48:system("CLS"); return;

}

} while (press != 56);

}

string getCurrentDateTime() {

time\_t currentTime = time(nullptr);

struct tm localTime;

localtime\_s(&localTime, &currentTime);

char dateTime[20];

strftime(dateTime, 20, "%d/%m/%Y %H:%M:%S", &localTime);

return isDateTimeValid(localTime.tm\_mday, localTime.tm\_mon + 1, localTime.tm\_year + 1900, localTime.tm\_hour, localTime.tm\_min) ? dateTime : "Invalid DateTime";

}

void getStartDateTime(int& startDay, int& startMonth, int& startYear, int& startHour, int& startMinute) {

time\_t now = time(nullptr);

tm ltm{};

localtime\_s(&ltm, &now);

int currentDay = ltm.tm\_mday;

int currentMonth = ltm.tm\_mon + 1;

int currentYear = ltm.tm\_year + 1900;

int currentHour = ltm.tm\_hour;

int currentMinute = ltm.tm\_min;

cout << "Текущее время: " << getCurrentDateTime() << endl;

cout << "Введите дату и время начала бронирования автомобиля\n";

bool isInputValid = false;

while (!isInputValid) {

getDateTime(startDay, startMonth, startYear, startHour, startMinute);

if (!isDateTimeValid(startDay, startMonth, startYear, startHour, startMinute)) {

cout << "Дата и время должны быть не раньше текущей даты и времени.\n";

isInputValid = false;

}

else {

isInputValid = true;

}

}

}

void getEndDateTime(int& startDay, int& startMonth, int& startYear, int& startHour, int& startMinute, int& endDay, int& endMonth, int& endYear, int& endHour, int& endMinute) { //Ввод окончания даты и времени бронирования

cout << "Введите дату и время окончания бронирования автомобиля\n";

getDateTime(endDay, endMonth, endYear, endHour, endMinute);

bool isInputValid = false;

while (!isInputValid) {

if (endYear < startYear || (endYear == startYear && endMonth < startMonth) || (endYear == startYear && endMonth == startMonth && endDay < startDay) || (endYear == startYear && endMonth == startMonth && endDay == startDay && endHour <= startHour && endMinute <= startMinute)) {

cout << "Время окончания должно быть позже времени начала.\n";

getDateTime(endDay, endMonth, endYear, endHour, endMinute);

}

else {

isInputValid = true;

}

}

}

void editCarData(cars\*& car, int& NumberOfCar) { //Меню редактирования полей данных об автомобиле

cout << "------------------------------------------------------\n"

<< "| № | Марка автомобиля | Модель | Стоимость аренды | Статус |\n"

<< "----------------------------------------------------------------------------------------\n";

if (NumberOfCar == 0) cout << "Список автомобилей пуст" << endl;

else {

for (int i = 0; i < NumberOfCar; i++) {

cout << "| " << setw(3) << left << i + 1 << " | "

<< setw(30) << left << car[i].model << " | "

<< setw(18) << left << car[i].carType << " | "

<< setw(17) << left << car[i].rentPrice << " | "

<< setw(6) << left << car[i].is\_reserved << " |\n";

cout << "-----------------------------------------\n";

}

}

string model;

int press, fieldIndex;

bool isReserved;

float rentPrice;

cout << "Введите название автомобиля, который хотите отредактировать:\n";

cin >> model;

for (int i = 0; i < NumberOfCar; i++) {

if (car[i].model == model) {

cout << "Введите номер поля, которое хотите отредактировать:\n"

<< "1. Марка автомобиля\n"

<< "2. Модель автомобиля\n"

<< "3. Стоимость аренды автомобиля в минуту\n"

<< "4. Флаг зарезервированности автомобиля\n";

do {

press = \_getch();

fieldIndex = press - 48;

} while (fieldIndex < 1 || fieldIndex > 4);

switch (fieldIndex) {

case 1:

cout << "Введите новую марку автомобиля:\n";

car[i].model = containsOnlyLetters(car[i].model);

break;

case 2:

cout << "Введите новую модель автомобиля:\n";

cin >> car[i].carType;

break;

case 3:

cout << "Введите новую стоимость аренды автомобиля в минуту:\n";

cin >> car[i].rentPrice;

break;

case 4:

cout << "Введите новое значение флага зарезервированности автомобиля (0 или 1):\n";

cin >> isReserved;

car[i].is\_reserved = isReserved;

break;

default:

break;

}

cout <<"Вы действительно хотите сохранить изменения?\n"

<< "1. Да\n"

<< "2. Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49) {

ofstream fout(FILE\_LIST);

for (int j = 0; j < NumberOfCar; j++) {

fout << car[j].model << " " << car[j].carType << " " << car[j].rentPrice << " " << car[j].is\_reserved << endl;

}

fout.close();

cout << "Изменения успешно сохранены.\n";

Sleep(2000);

system("CLS");

return;

}

else {

loadDataOfCar(car, NumberOfCar);

cout << "Изменения отменены.\n";

Sleep(2000);

system("CLS");

return;

}

}

}

cout << "Извините! Данного автомобиля нет в этом списке.\n";

return;

}

void filterByPriceRange(cars\* car, int numberOfCars) { //Фильтрация автомобилей по цене

float minPrice, maxPrice;

int press;

cout << "Введите минимальную цену: ";

cin >> minPrice;

cout << "Введите максимальную цену: ";

cin >> maxPrice;

bool found = false;

for (int i = 0; i < numberOfCars; i++) {

if (car[i].rentPrice >= minPrice && car[i].rentPrice <= maxPrice) {

if (!found) {

cout << "Результаты поиска:\n";

found = true;

}

cout << "Марка: " << car[i].model << endl;

cout << "Модель автомобиля: " << car[i].carType << endl;

cout << "Цена аренды: " << car[i].rentPrice << " рублей в минуту" << endl;

if (car[i].is\_reserved) {

cout << "Статус: забронирован\n";

}

else {

cout << "Статус: свободен\n";

}

cout << endl;

}

}

if (!found) {

cout << "Нет доступных автомобилей в указанном ценовом диапазоне.\n";

}

cout << "0.Назад";

do {

press = \_getch();

} while (!(press >= 48 && press <= 54));

system("CLS");

}

# ПРИЛОЖЕНИЕ Б

**(обязательное)**

**Проверка на заимствование**

****

# ПРИЛОЖЕНИЕ В

**(справочное)**

**Схема общего алгоритма работы программы**

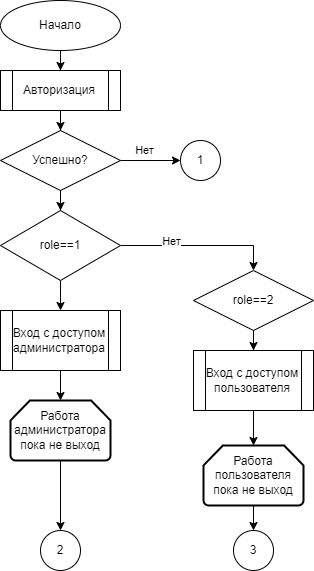
****

Рисунок Д.1 – Схема общего алгоритма работы программы

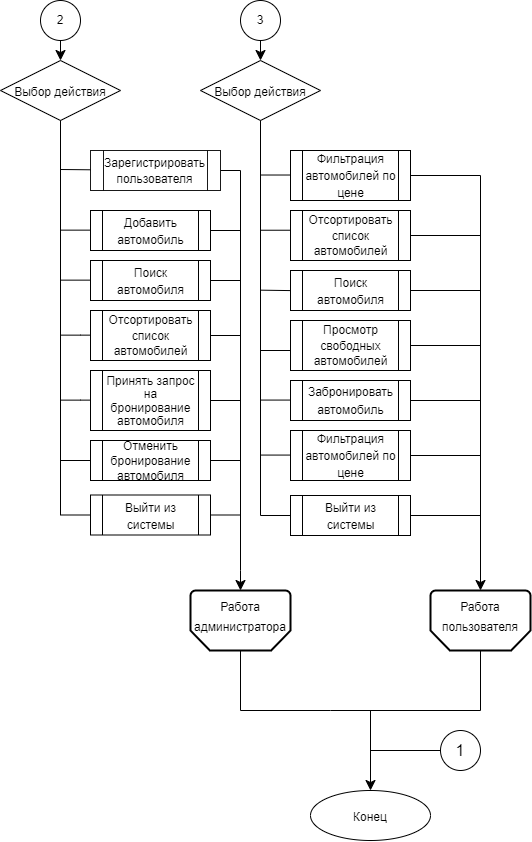
****

Рисунок Д.2 – Продолжение схемы общего алгоритма работы программы