Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Основы алгоритмизации и программирования»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Старший преподаватель ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Бутов |
|  | \_\_\_.\_\_\_\_.2025 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Автоматизация работы рекрутинговой фирмы»**

БГУИР КР 1-40 05 01-02 007 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 272303  Каштанов Матвей Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2025  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2025

**РЕФЕРАТ**

БГУИР КР 1-40 05 01-02 007 ПЗ

**Каштанов, М.С.** Автоматизация работы рекрутинговой фирмы/ М.С. Каштанов. – Минск: БГУИР, 2025. – 54 с.

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

ПРОСМОТР СОИСКАТЕЛЕЙ, сведения О СОИСКАТЕЛЯХ, пользователях, модели *IDEF*0, схемы алгоритмов, программное средство

*Цель* *курсовой работы*: Автоматизация работы рекрутинговой фирмы, которая позволяет облегчить процесс поиска работников (соискателей) различным предприятиям, дает возможность нахождения нужного работника по различным параметрам для рекрутеров (работодателей), а также позволяет администраторам контролировать и управлять процессом поиска и найма.

*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 разработаны модели бизнес-процессов предметной области на основе нотаций IDEF0; описаны основные алгоритмы работы программного средства; 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 2022*.

*Область применения результатов*: разработанное программное средство может применяться в сфере рекрутинга, поиска и найма персонала. В частности, использование программы рекрутинговыми агентствами и фирмами.

Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым при найме и поиске работников.

СОДЕРЖАНИЕ

[Введение 4](#_Toc135782961)

[1 Анализ и моделирование предметной области программного средства 6](#_Toc135782962)

[1.1 Описание предметной области 6](#_Toc135782963)

[1.2 Разработка функциональной модели предметной области в нотации IDEF0 7](#_Toc135782964)

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

[2 Проектирование и конструирование программного средства 13](#_Toc135782968)

[2.1 Постановка задачи 13](#_Toc135782969)

[2.2 Разработка модульной структуры 15](#_Toc135782970)

[2.3 Выбор способа организации данных 16](#_Toc135782971)

[2.4 Разработка перечня пользовательских функций программы 18](#_Toc135782972)

[2.5 Разработка схем алгоритмов программы 20](#_Toc135782973)

[3 Тестирование и проверка работоспособности программного средства 25](#_Toc135782974)

[4 Инструкция по развертыванию приложения и сквозной пример 28](#_Toc135782975)

[4.1 Авторизация 28](#_Toc135782976)

[4.2 Модуль администратора 29](#_Toc135782977)

[4.3 Модуль пользователя 34](#_Toc135782978)

[4.4 Регистрация 36](#_Toc135782979)

[Заключение 37](#_Toc135782980)

[Список использованных источников 38](#_Toc135782982)

[Приложение А 39](#_Toc135782984)

[Приложение Б 47](#_Toc135782985)

[Приложение В 48](#_Toc135782986)

# ВВЕДЕНИЕ

В нынешней ситуации жесткой конкуренции на рынке труда, подбор квалифицированных кадров становится определяющим условием процветания бизнеса. Это обуславливает высокую потребность в услугах рекрутинговых компаний, что, в свою очередь, заставляет их непрерывно совершенствовать свои рабочие подходы, увеличивать оперативность и качество поиска потенциальных сотрудников.

Рекрутинг представляет собой комплекс мер по поиску, отбору и найму компетентных специалистов для работы в компании-клиенте. Обычно он включает в себя анализ требований к вакансии, поиск претендентов с использованием баз данных, социальных сетей и специализированных сайтов, проведение интервью и налаживание коммуникации между работодателем и кандидатом. Внедрение автоматизированных систем в этот процесс позволяет быстрее обрабатывать резюме, уменьшить время, затрачиваемое на однообразные задачи, и повысить точность отбора за счет применения интеллектуальных алгоритмов.

Основная задача этого курсового проекта – создать программное обеспечение, которое автоматизирует ключевые процессы кадрового агентства. Это даст возможность:

* Облегчить поиск и сортировку кандидатов;
* Ускорить обработку запросов и взаимодействие с заказчиками;
* Увеличить продуктивность работы HR-менеджеров за счет уменьшения объема ручных операций.

Для реализации поставленной цели необходимо решить следующие задачи:

* Изучить специфику деятельности рекрутинговых агентств;
* Разработать алгоритмы функционирования программного обеспечения;
* Спроектировать структуру и ключевые компоненты системы;
* Проверить работоспособность разработанного решения.

Объектом исследования в этой работе является процесс рекрутинга, а предметом исследования – его автоматизация с помощью специализированного программного обеспечения.

Использование такой системы позволит рекрутинговым фирмам оптимизировать свою работу, сократить издержки и повысить лояльность клиентов благодаря более оперативному и точному подбору персонала.

# АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

## Описание предметной области

Современный рынок труда характеризуется высокой динамикой и острой конкуренцией за квалифицированных специалистов. Компании сталкиваются с проблемой поиска подходящих кандидатов: традиционные методы, такие как размещение вакансий на job-сайтах или ручной подбор через базы данных, требуют значительных временных затрат и не всегда обеспечивают нужное качество найма.

Одним из решений этой проблемы является автоматизация рекрутинговых процессов. Специализированное программное обеспечение позволяет оптимизировать поиск, отбор и взаимодействие с кандидатами, сокращая время закрытия вакансий и повышая точность подбора.

Современные рекрутинговые платформы представляют собой комплексные решения, которые:

* Автоматизируют поиск кандидатов.
* Сортируют и анализируют анкеты – применяют алгоритмы машинного обучения для ранжирования кандидатов по релевантности вакансии.
* Упрощают коммуникацию работодателей и соискателей.

Для работы с системой рекрутеру или HR-менеджеру достаточно ввести параметры вакансии (например, требуемые навыки, опыт работы, должность), после чего программа сформирует список наиболее подходящих кандидатов. Это значительно ускоряет процесс подбора и снижает нагрузку на сотрудников.

Преимущества автоматизации для рекрутинговых фирм:

* Скорость обработки заявок – система за минуты анализирует сотни резюме, тогда как вручную на это ушли бы часы.
* Снижение человеческого фактора – минимизация ошибок при отборе за счет четких алгоритмов.
* Удобство для клиентов – компании-заказчики получают подобранных кандидатов быстрее, что повышает их лояльность.
* Экономия ресурсов – сокращение затрат на рутинные операции позволяет рекрутерам сосредоточиться на стратегических задачах.

Несмотря на преимущества, многие рекрутинговые системы сталкиваются с проблемами:

* Низкая точность отбора – алгоритмы могут пропускать хороших кандидатов из-за жестких фильтров.
* Сложность интерфейса – не все системы удобны для пользователей, что замедляет работу.

Разработка эффективной системы подбора персонала позволяет:

* Увеличить количество успешных наймов за счет более точного соответствия кандидатов требованиям вакансий.
* Снизить операционные издержки – автоматизация рутинных процессов уменьшает затраты на HR-персонал.
* Повысить конкурентоспособность агентства

Автоматизация рекрутинга – это не просто тренд, а необходимость для современных HR-агентств. Внедрение умных систем подбора персонала помогает ускорить процессы, улучшить качество найма и повысить удовлетворенность как клиентов, так и кандидатов. Компании, которые игнорируют цифровизацию, рискуют потерять свою долю рынка, в то время как автоматизированные рекрутинговые платформы становятся ключевым инструментом для успешного роста бизнеса.

## Разработка функциональной модели предметной области в нотации IDEF0

IDEF0  — [методология](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) функционального моделирования ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *function modeling*) и графическая нотация, предназначенная для формализации и описания [бизнес-процессов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81) [1]. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность ([поток работ](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D1%80%D0%B0%D0%B1%D0%BE%D1%82)).

Использование IDEF0 позволяет более эффективно проектировать бизнес-процессы, повышать качество их выполнения и уменьшать затраты на их выполнение. Она является надежным инструментом для анализа и проектирования систем, включая систему поиска подходящих претендентов.

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”.

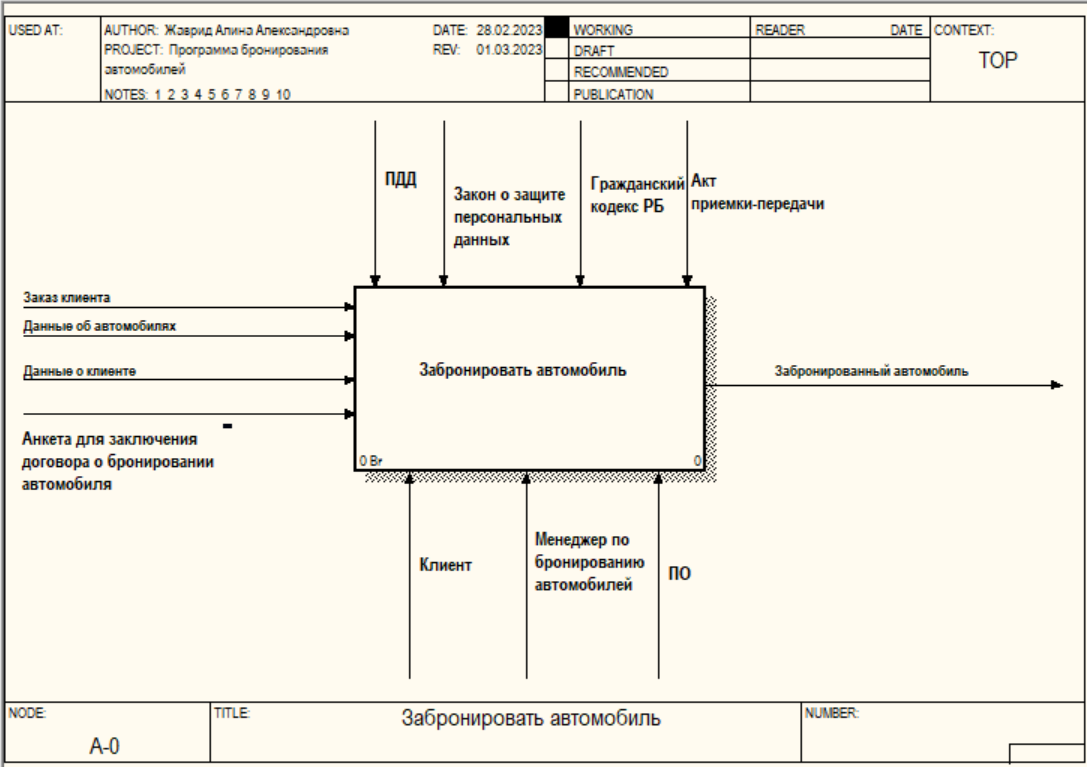


Рисунок 1.1 – Контекстная диаграмма модели А-0

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.

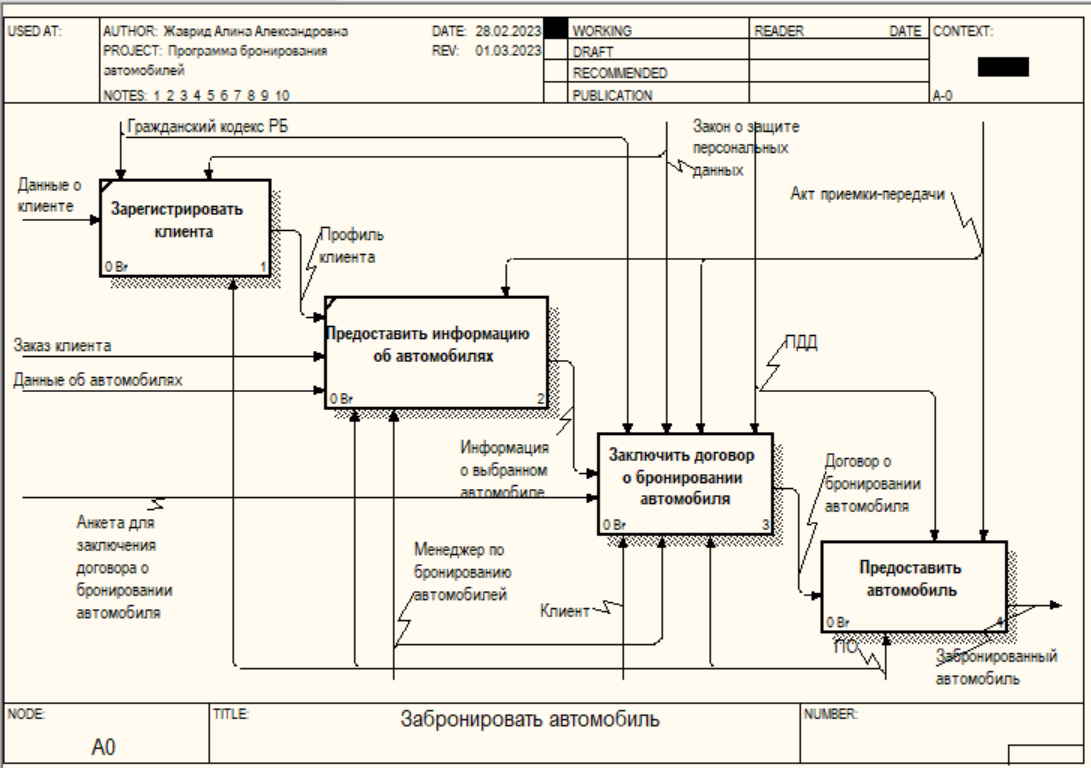


Рисунок 1.2 – Декомпозиция процесса “Забронировать автомобиль”

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.

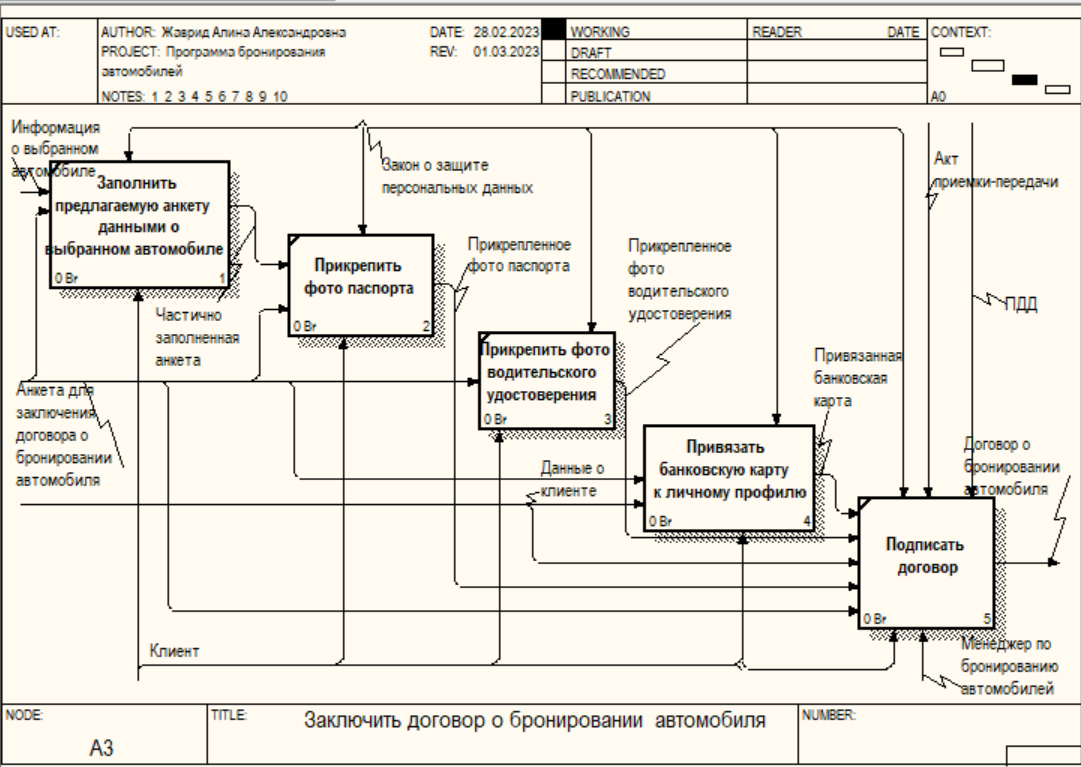


Рисунок 1.3 – Декомпозиция процесса “Заключить договор о бронировании автомобиля”

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.

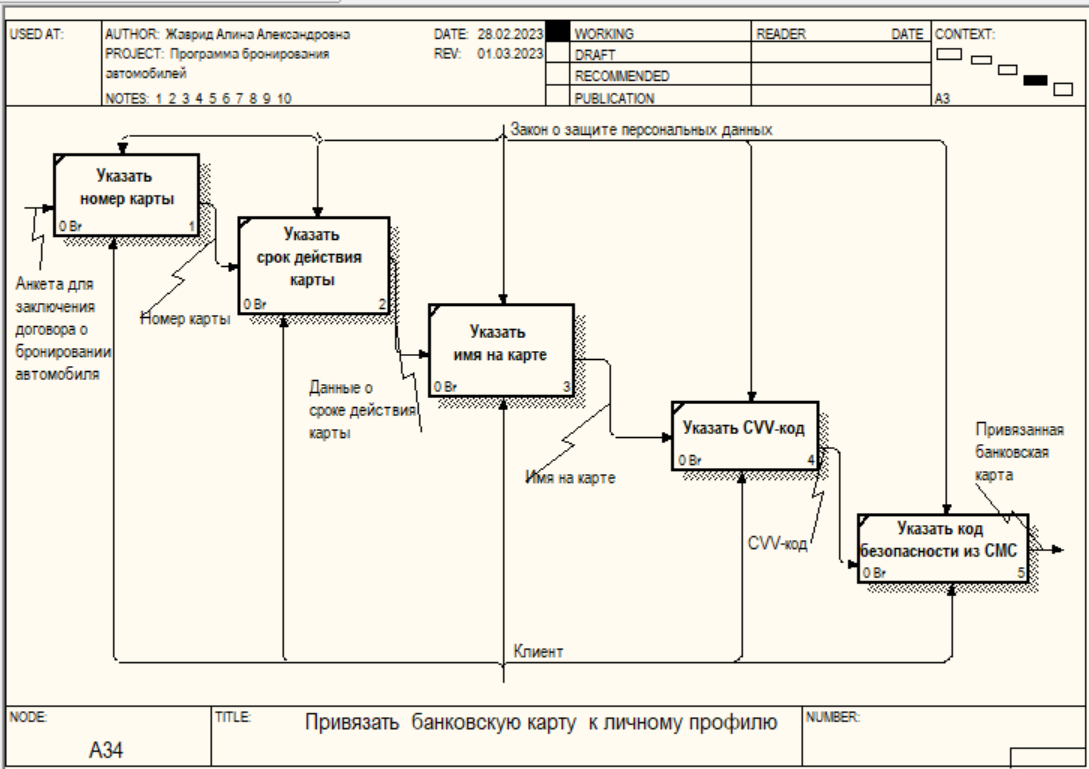


Рисунок 1.4 – Декомпозиция процесса “Привязать банковскую карту к личному профилю”

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.

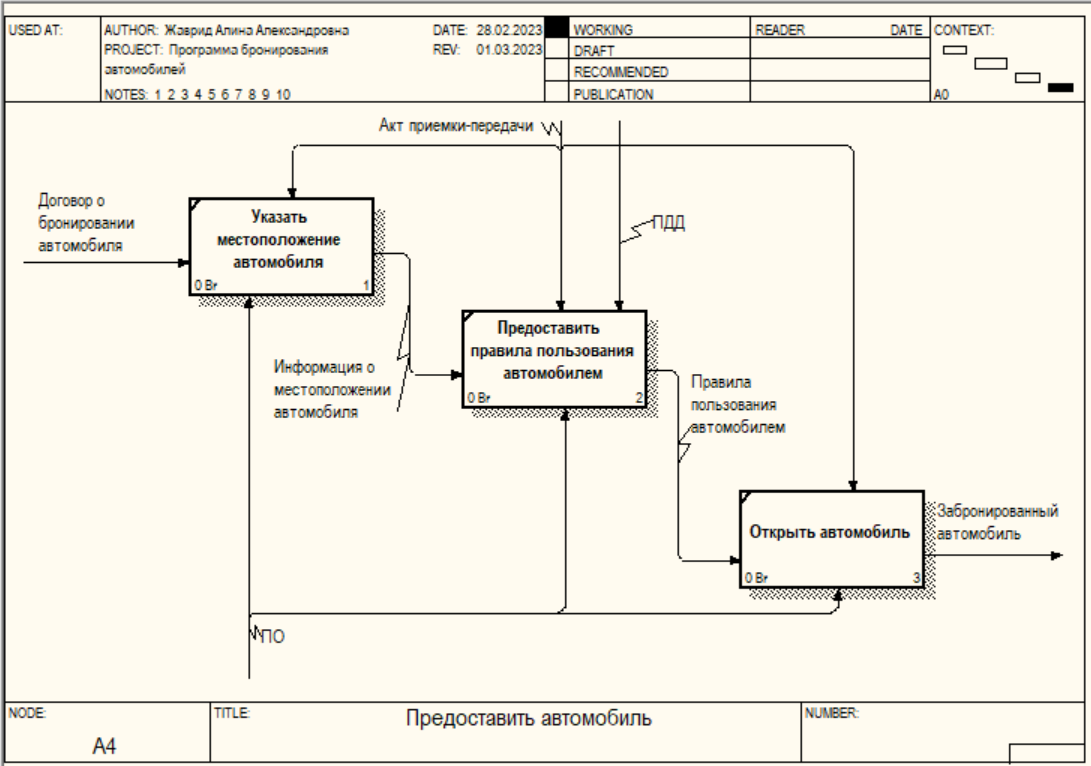


Рисунок 1.5 – Декомпозиция процесса “Предоставить автомобиль”

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

Разрабатываемое в рамках курсовой работы программное обеспечение должно соответствовать определенным условиям. Одним из основных требований к программе является наличие разделения на пользователя и администратора, а еще возможность разделения их полномочий. Пользователь должен иметь возможность просмотра всех претендентов. Администратор должен иметь возможность управлять перечнем соискателей, их резюмированными сведениями. Кроме того, необходимо предусмотреть возможность добавления новых претендентов, пользователей и корректировки информации об уже существующих, а также возможность поиска соискателей по определенным критериям, таким как опыт работы, наличие или отсутствие высшего образования или должность.

# ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## Постановка задачи

По условию задания требуется автоматизировать работу рекрутинговой фирмы. Система должна предусматривать режимы ведения данных о соискателях, отражающее все сведения о них, хранящееся в базе данных. О каждом претенденте будет известно следующее:

* Имя;
* Возраст;
* Наличие или отсутствие высшего образования;
* Опыт работы;
* Должность;
* E-mail.

Помимо данных об автомобилях, в ИС рекрутинга хранятся данные о соискателях, такие как:

* фамилия и имя;
* адрес почтового ящика;
* резюме.

С данной информационной системой должны работать следующие группы пользователей:

* пользователи;
* администраторы.

Модуль администраторавключает следующие подмодули:

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

* просмотр всех соискателей;
* добавление новых соискателей;
* удаление резюме соискателя;
* редактирование уже существующих профилей;

1. режим обработки информации:

* поиск соискателей (по 3 различным параметрам);
* сортировка данных (встроена в базу данных);

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

* просмотр всех соискателей;
* поиск соискателей по различным параметрам;
* сортировка данных (встроена в базу данных).

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);
* вывод сообщения об успешности создания /удаления записи.

Требования к программной реализации проекта:

* Все переменные и константы должны иметь осмысленные имена в рамках тематики варианта к курсовой работе.
* Имена функций должны быть осмысленными.
* 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).
* Код не должен дублироваться – для этого существуют функции!
* 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).
* Следует по возможности 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.

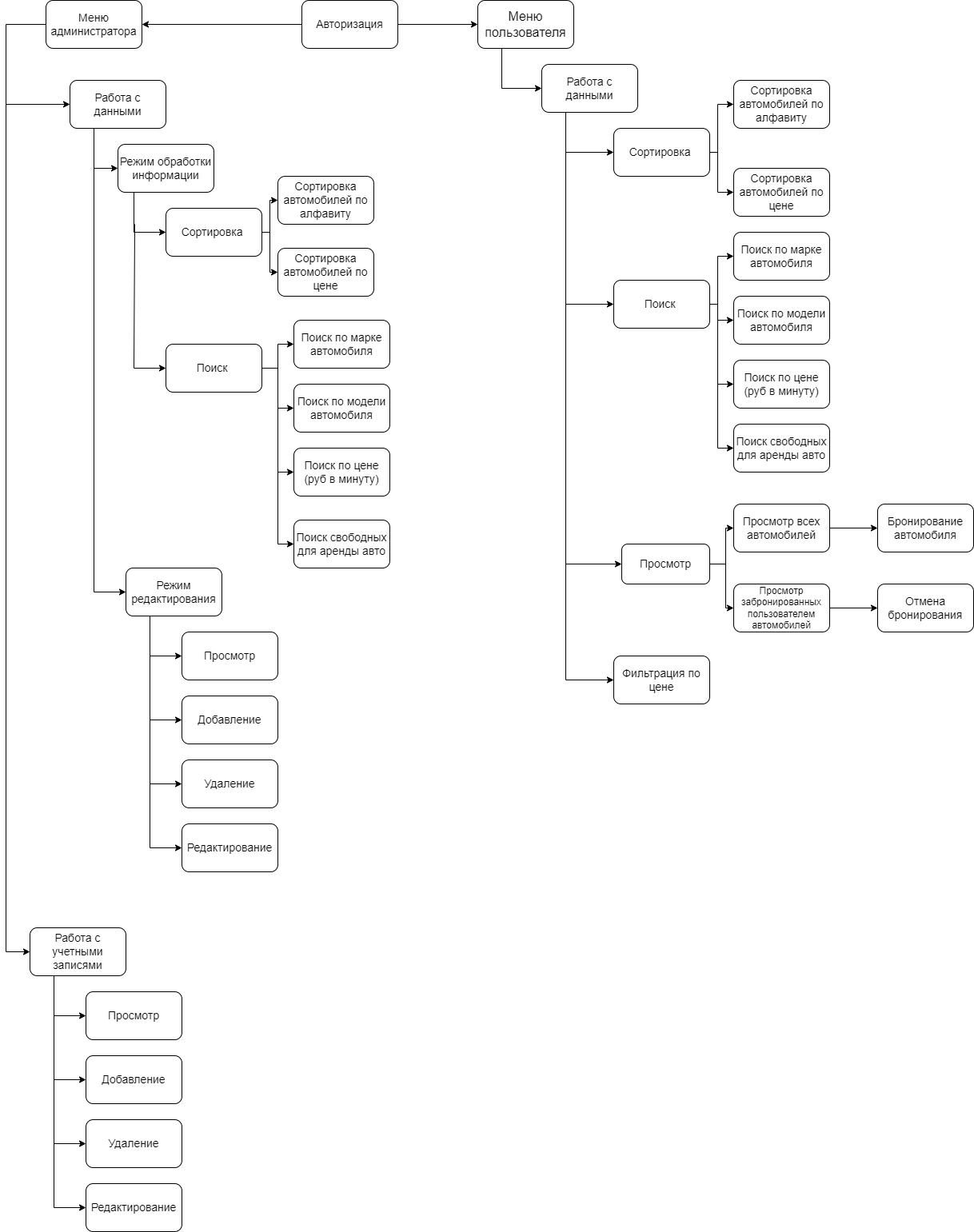


Рисунок 2.1 – Модульная структура программы

## 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 |

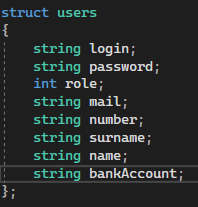


Рисунок 2.1 – Структура users

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.

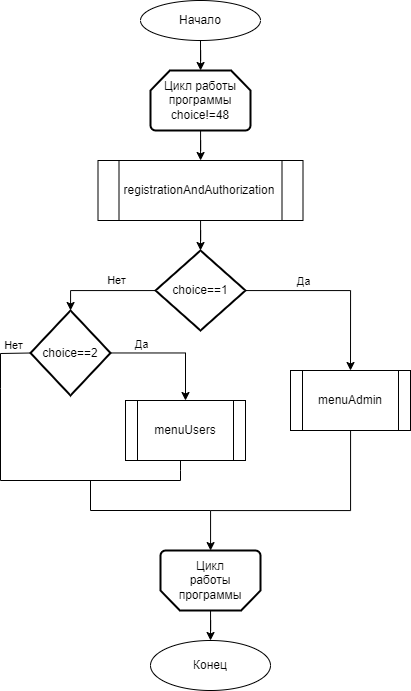


Рисунок 2.3 – Алгоритм функции main

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.

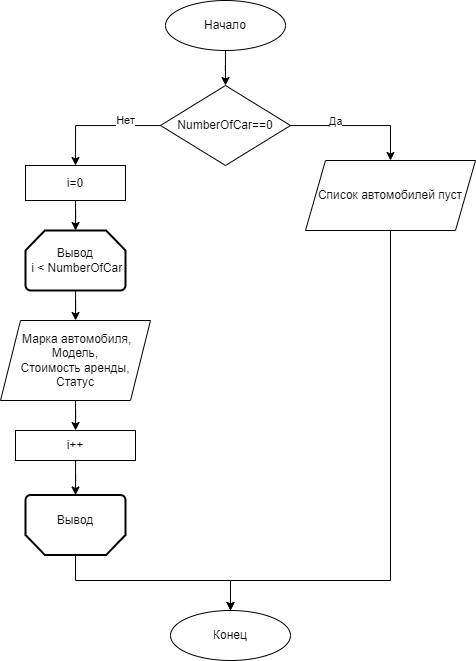


Рисунок 2.4 – Алгоритм функции ViewCar()

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.

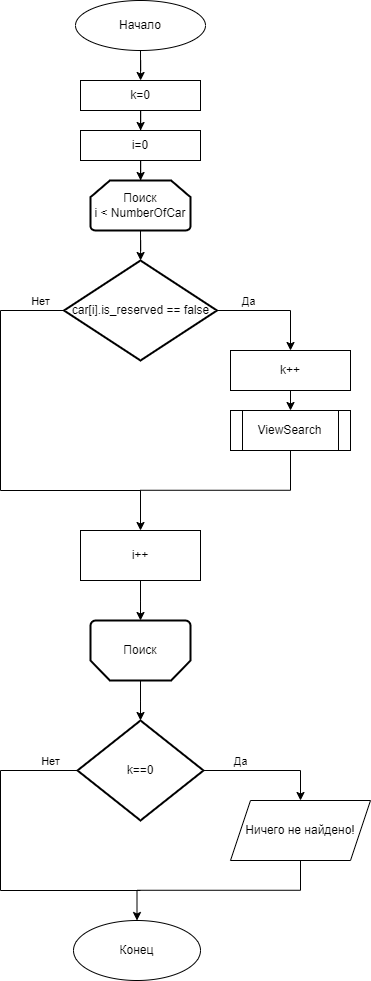


Рисунок 2.5 – Алгоритм функции searchNotReserved()

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).

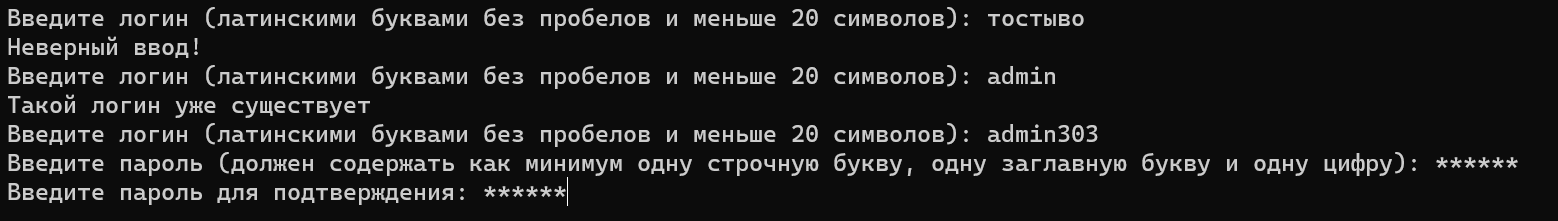


Рисунок 3.1 – Ввод логина и пароля

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

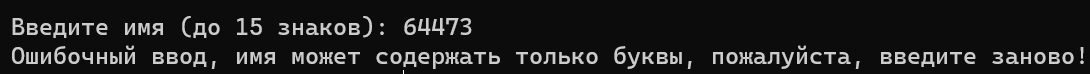


Рисунок 3.2 – Ввод не букв

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

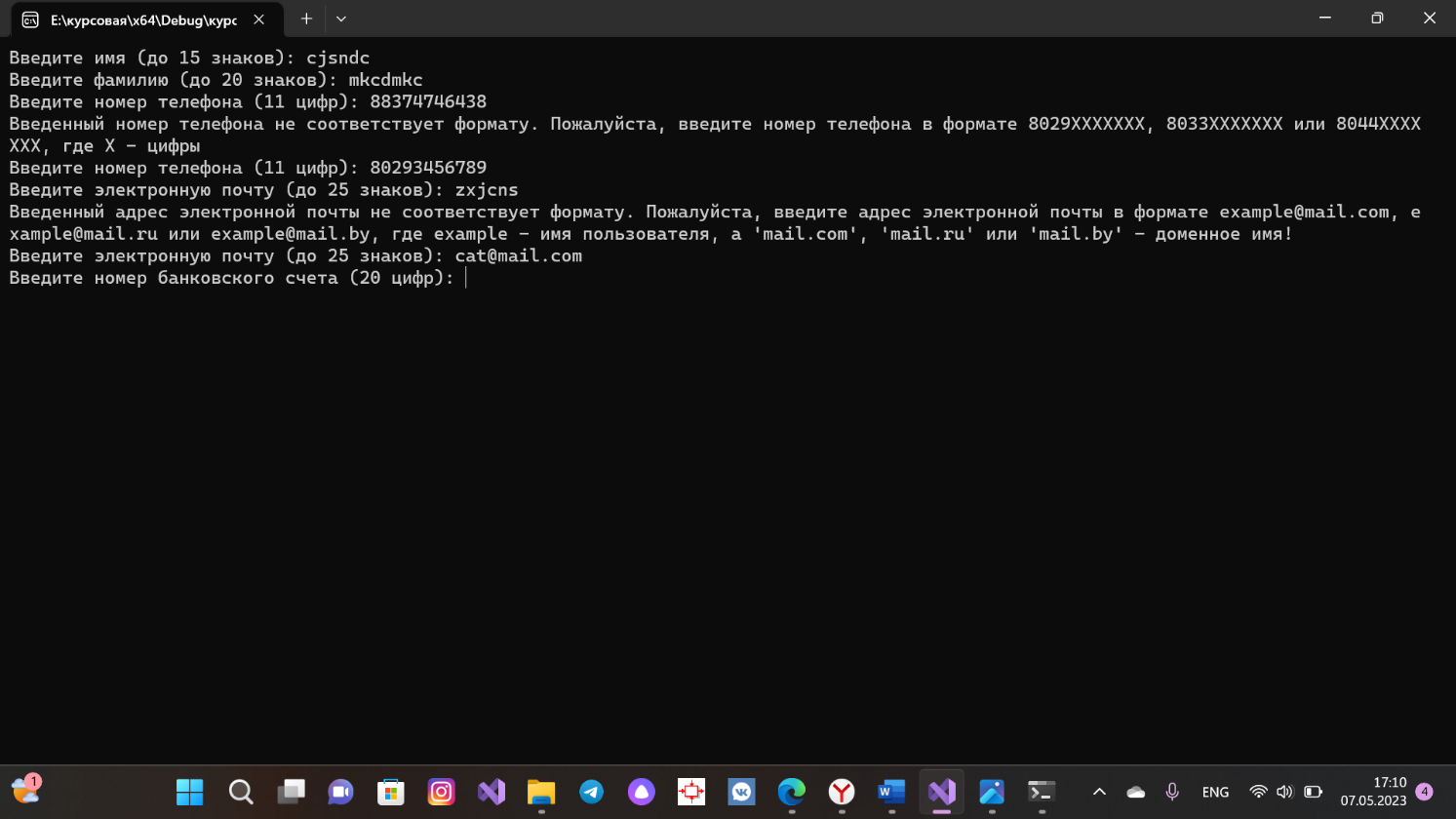


Рисунок 3.4 – Ввод некорректного формата номера телефона

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

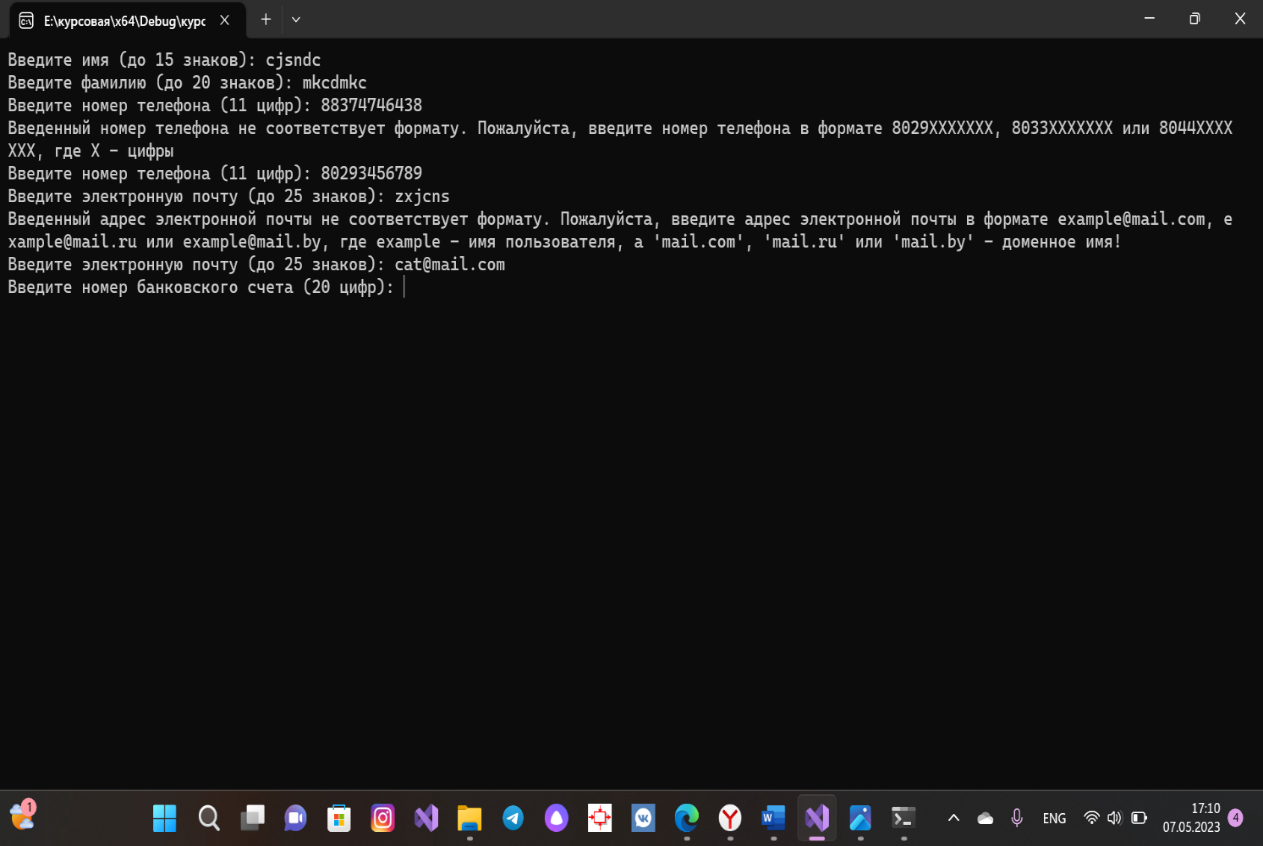


Рисунок 3.5 – Ввод некорректного формата электронной почты

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.

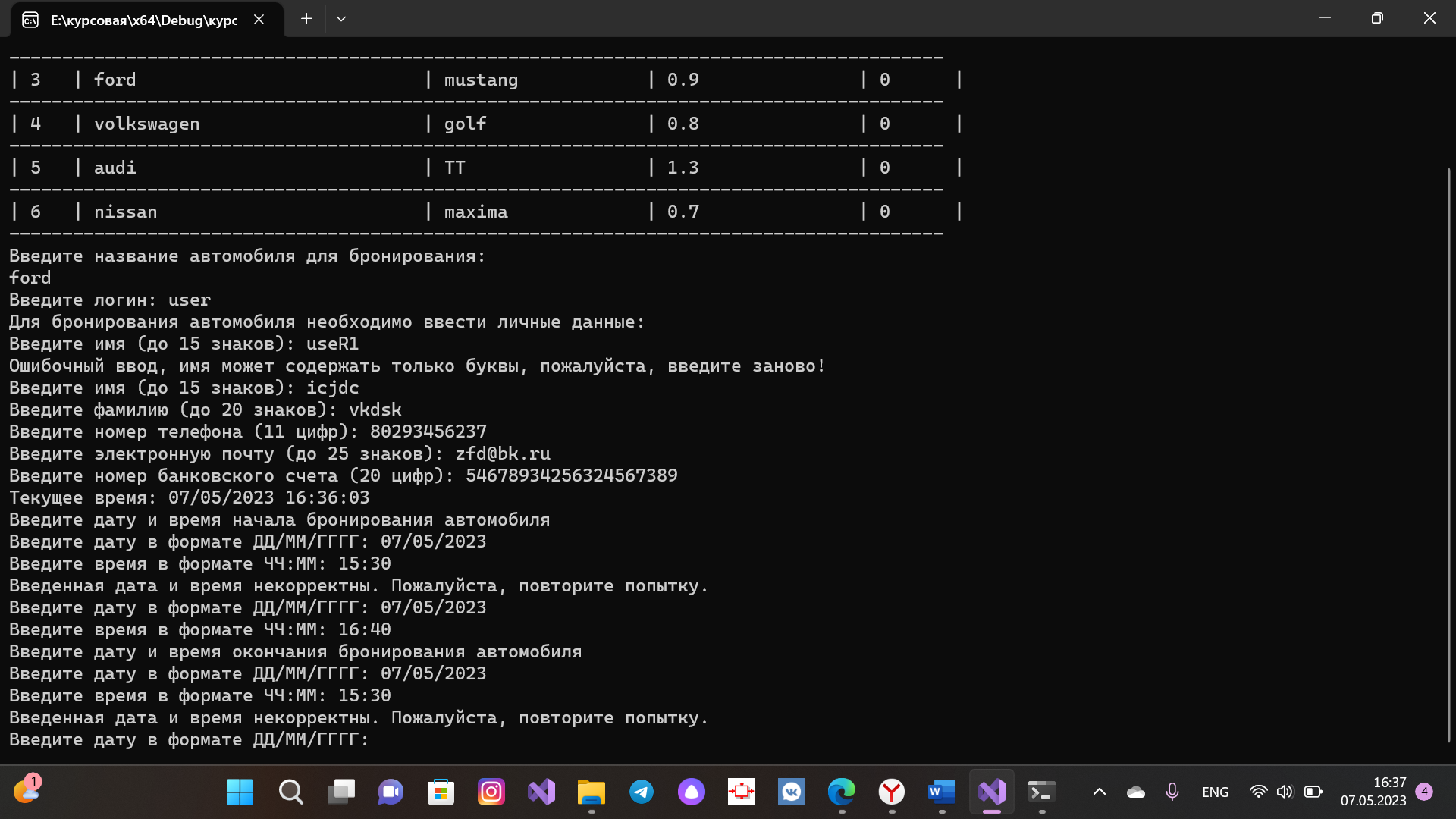


Рисунок 3.6 – Ввод некорректной даты

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

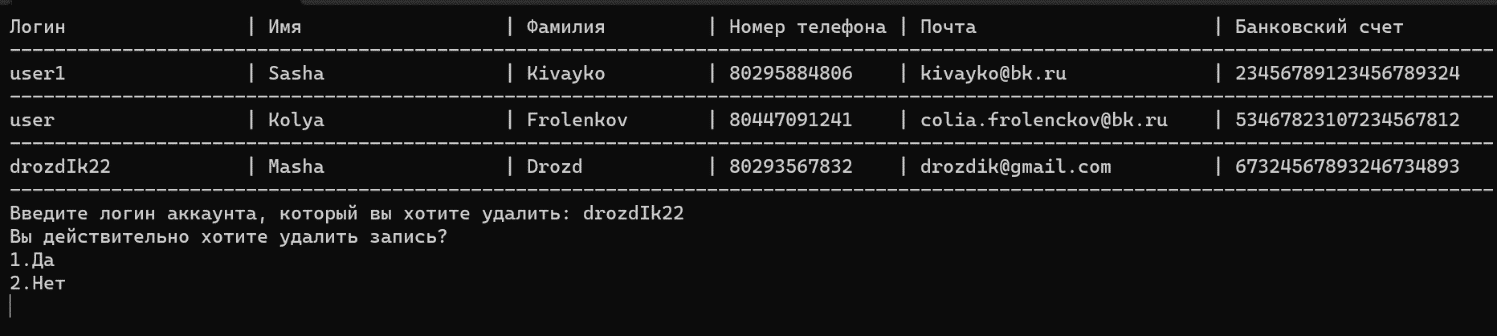


Рисунок 3.8 – Уведомление об уверенности удаления данных

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).

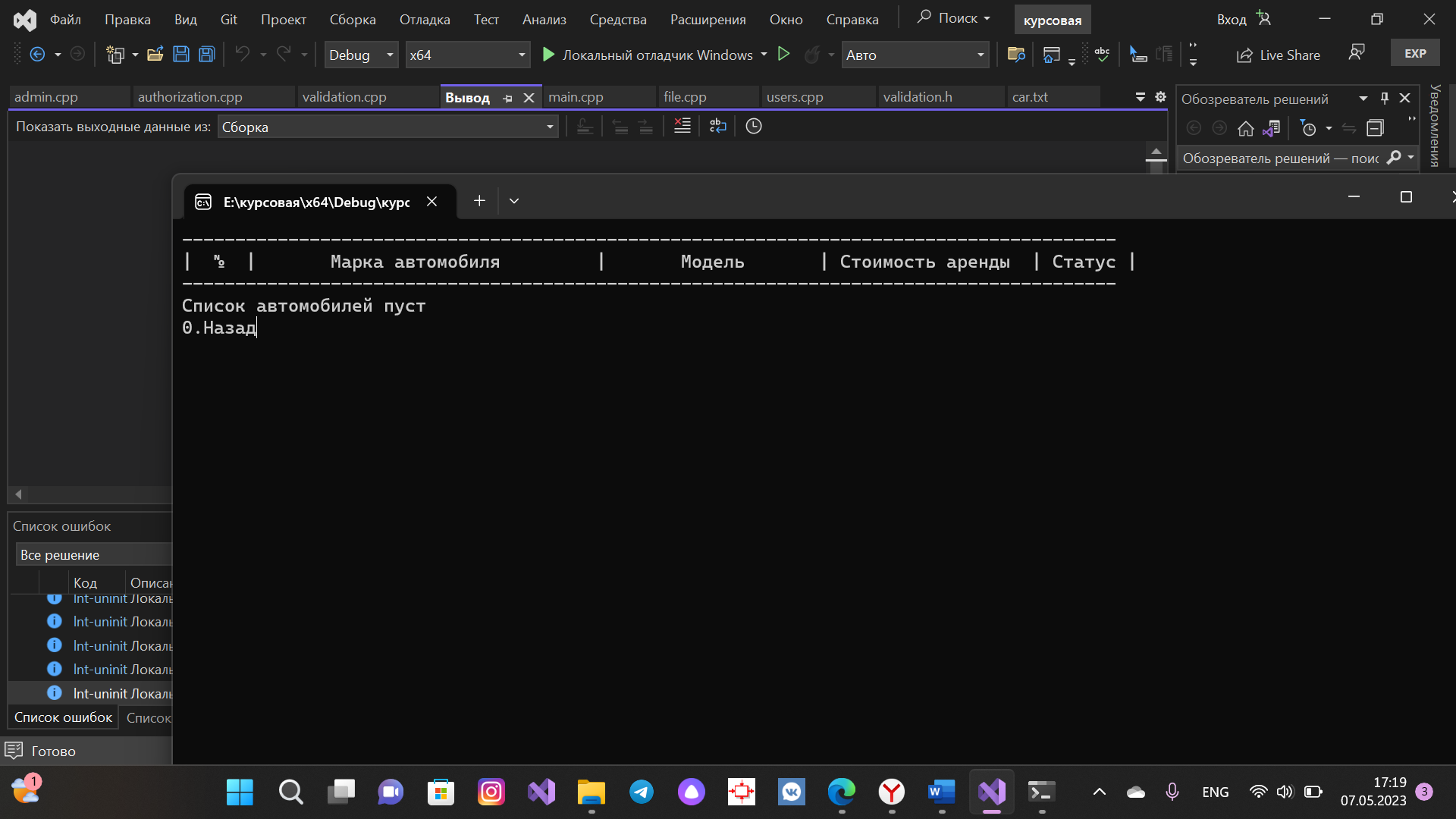


Рисунок 3.9 – Уведомление об отсутствии информации

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

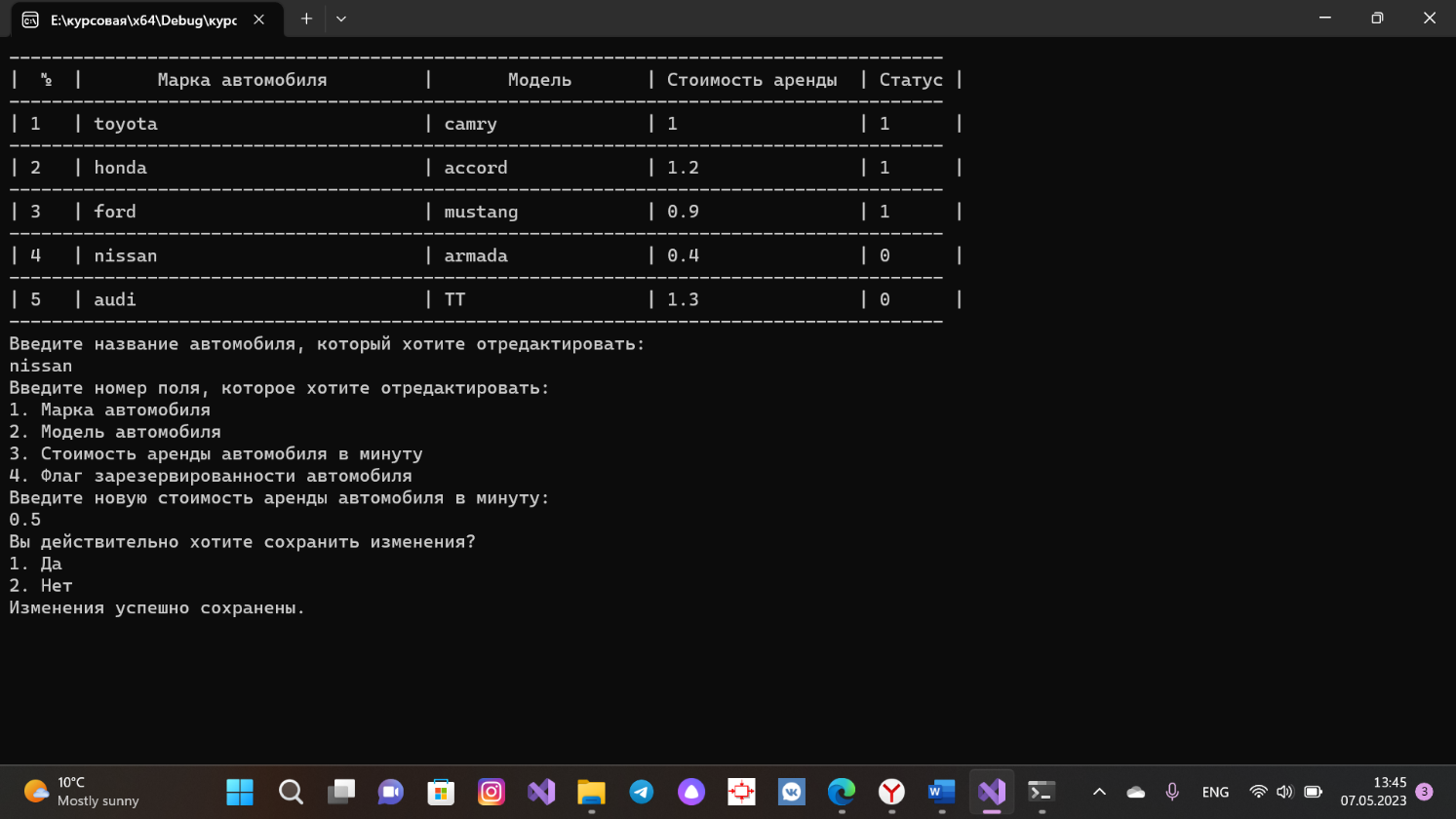


Рисунок 3.10 – Уведомление об уверенности редактирования

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.

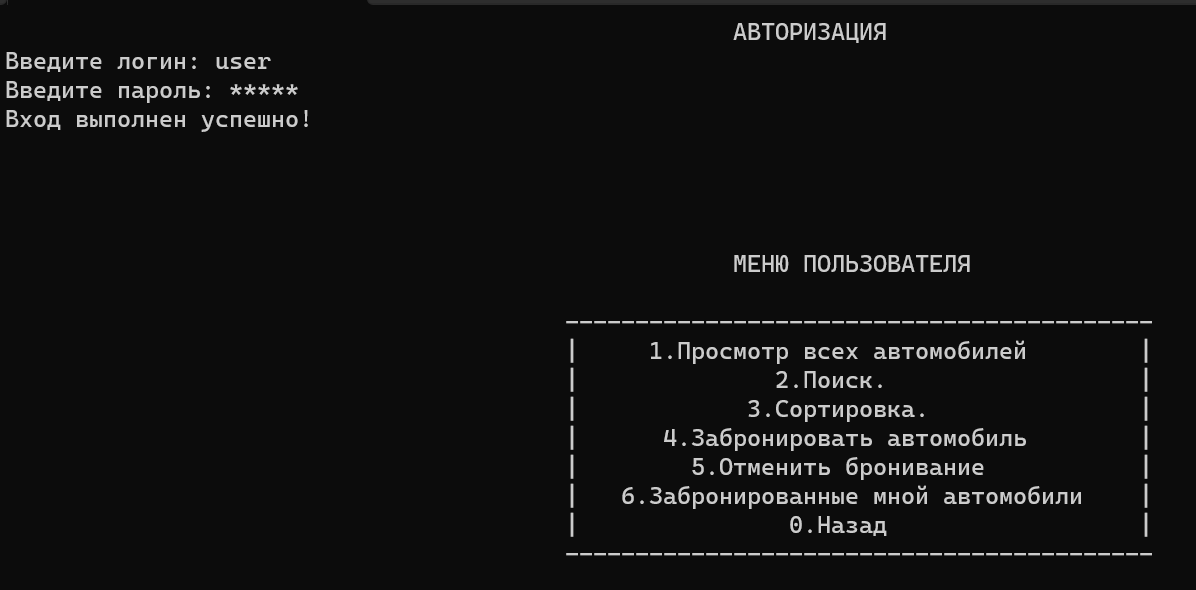
****

Рисунок 4.2 – Вход в учетную запись как пользователь

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

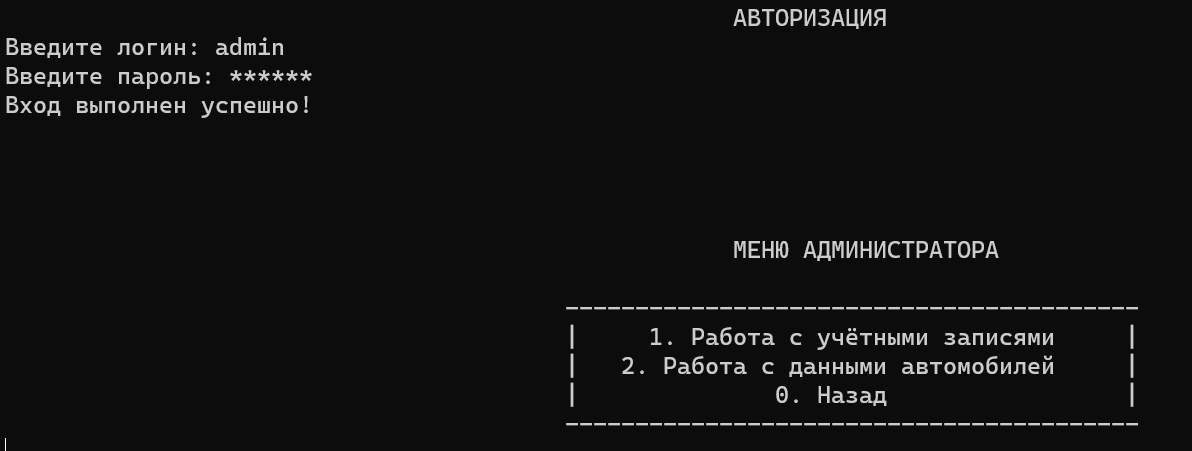


Рисунок 4.3 – Вход в учетную запись как администратор

## Модуль администратора

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

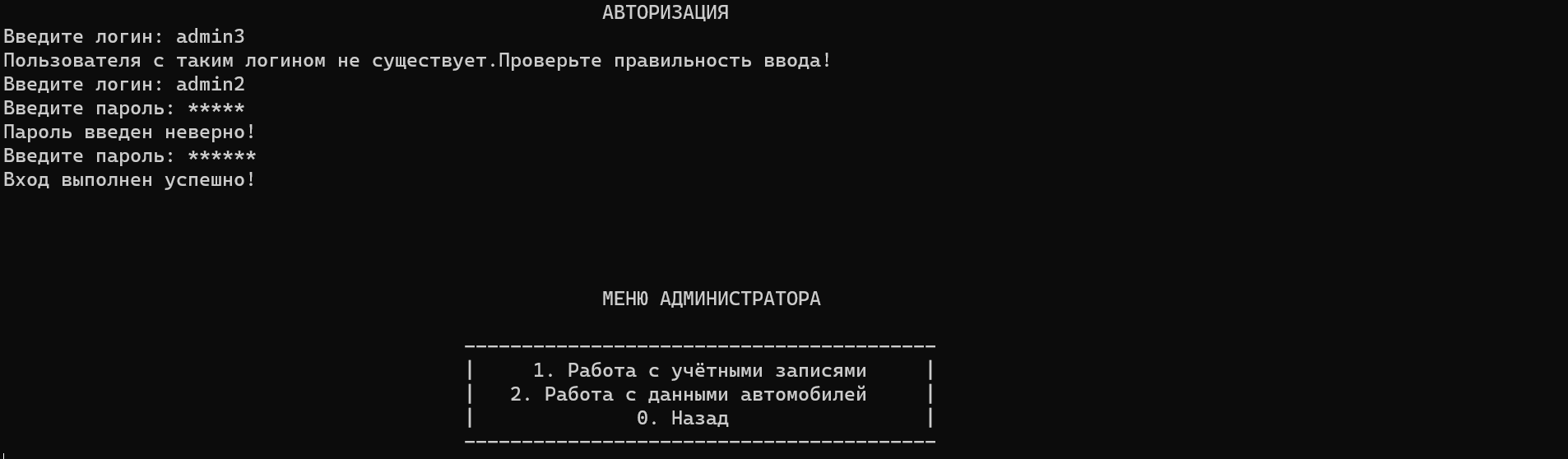


Рисунок 4.4 – Меню администратора

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

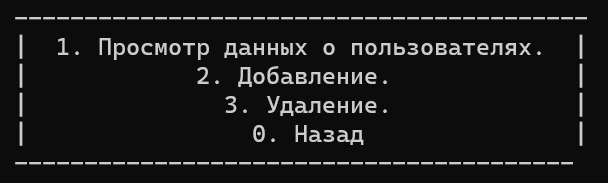


Рисунок 4.5 – Меню работы с учетными записями

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

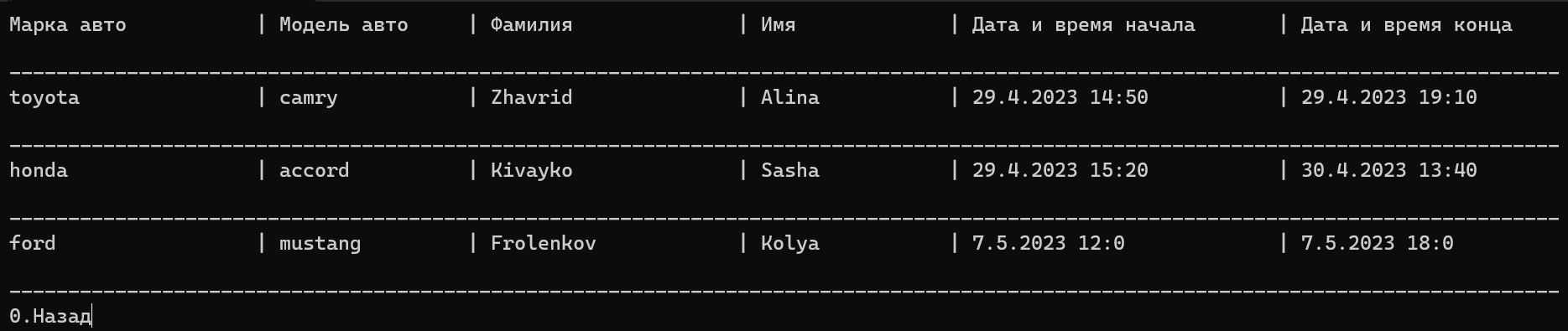


Рисунок 4.6 – Пример вывода информации о пользователях

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).

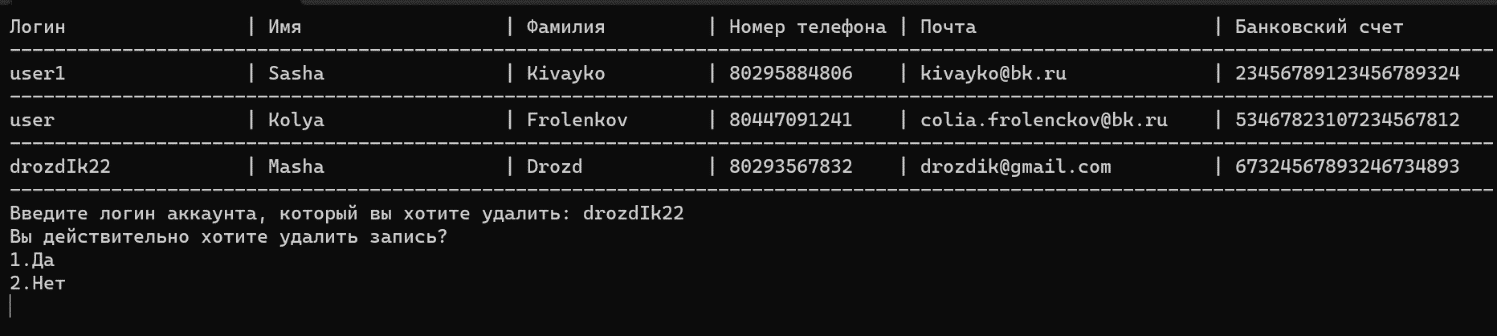


Рисунок 4.8 – Удаление аккаунта

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

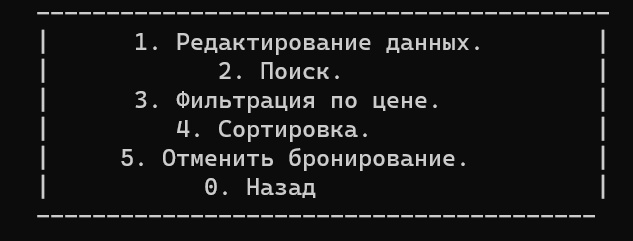


Рисунок 4.9 – Меню работы с данными автомобилей

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

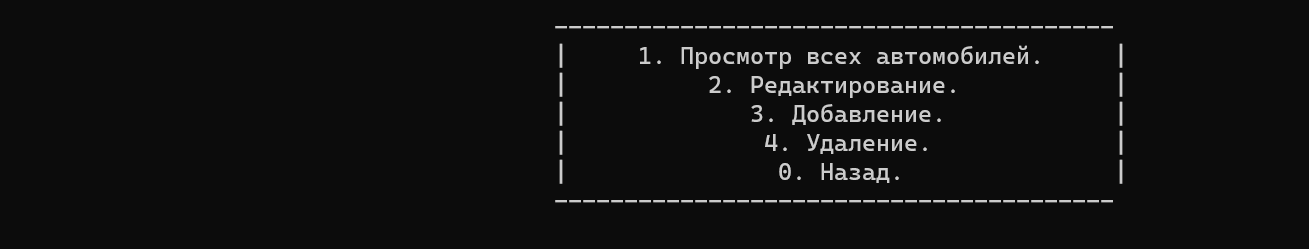


Рисунок 4.10– Меню редактирования данных

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

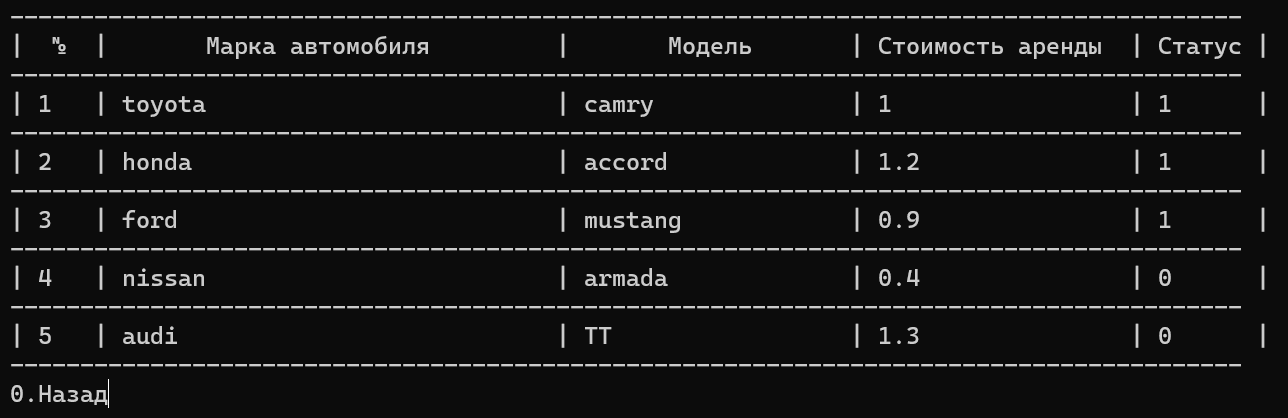


Рисунок 4.11 – Просмотр всех автомобилей

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

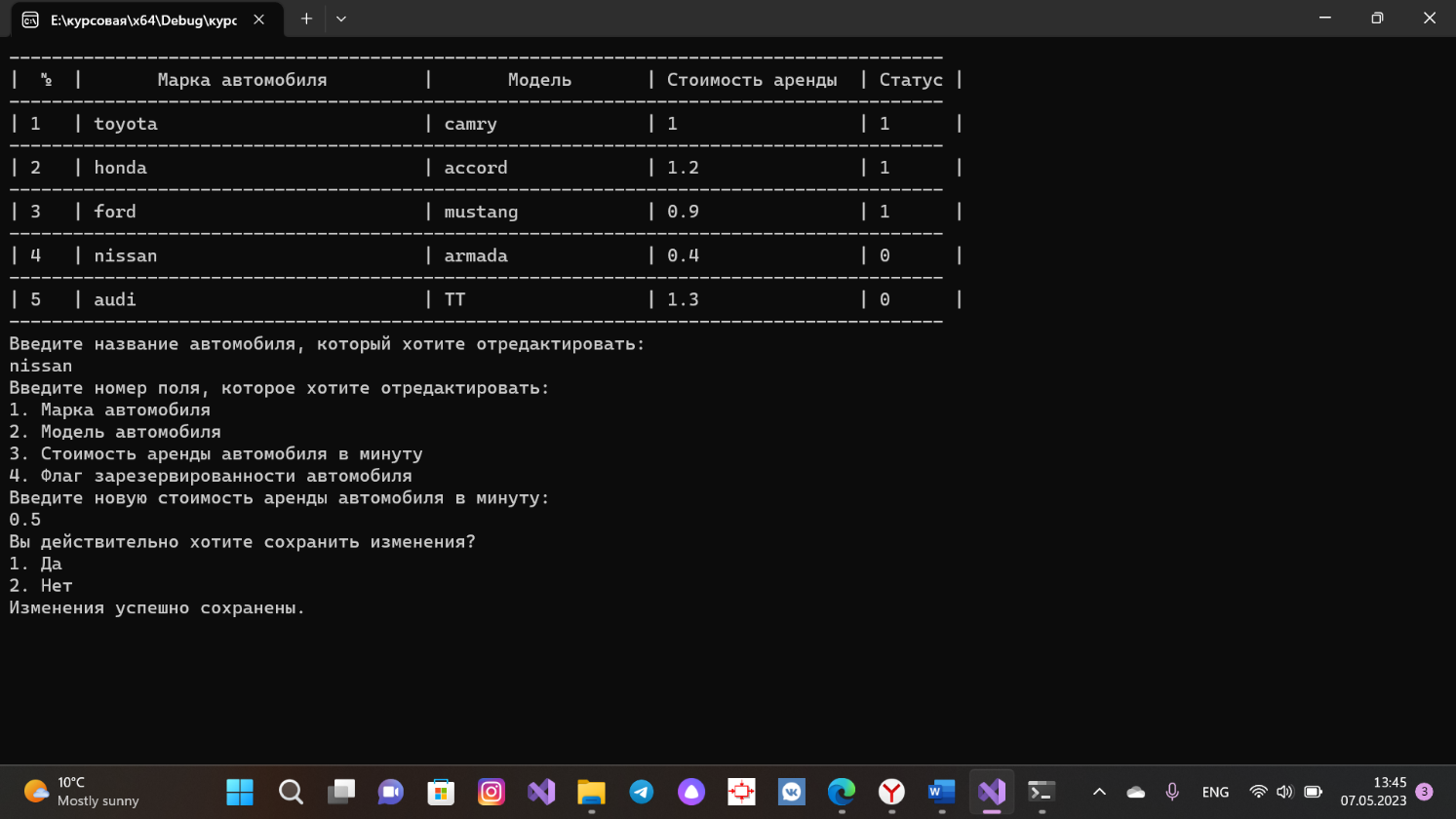


Рисунок 4.12 – Пример редактирования данных об автомобиле

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

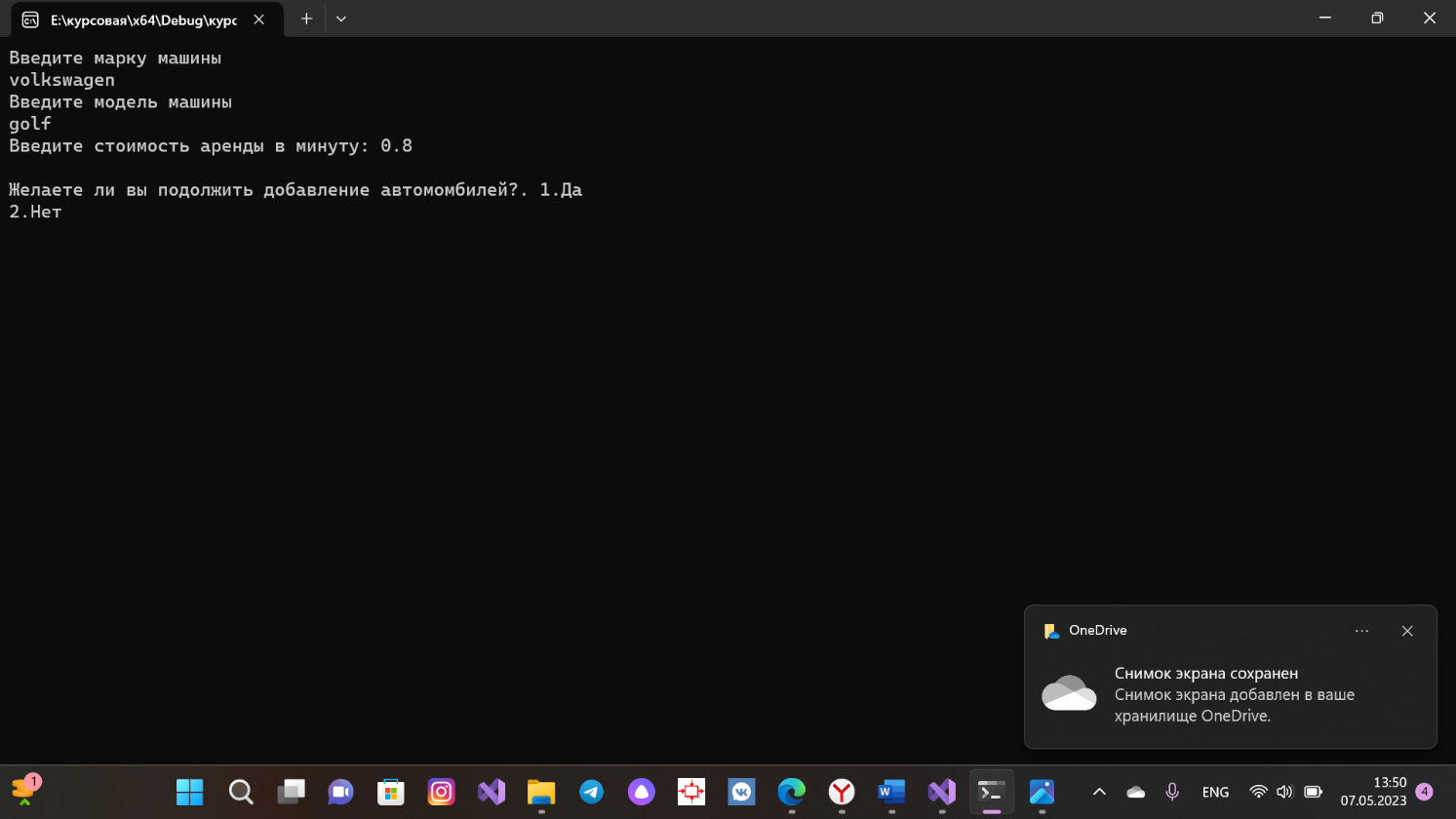


Рисунок 4.13 – Добавление записи об автомобиле

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

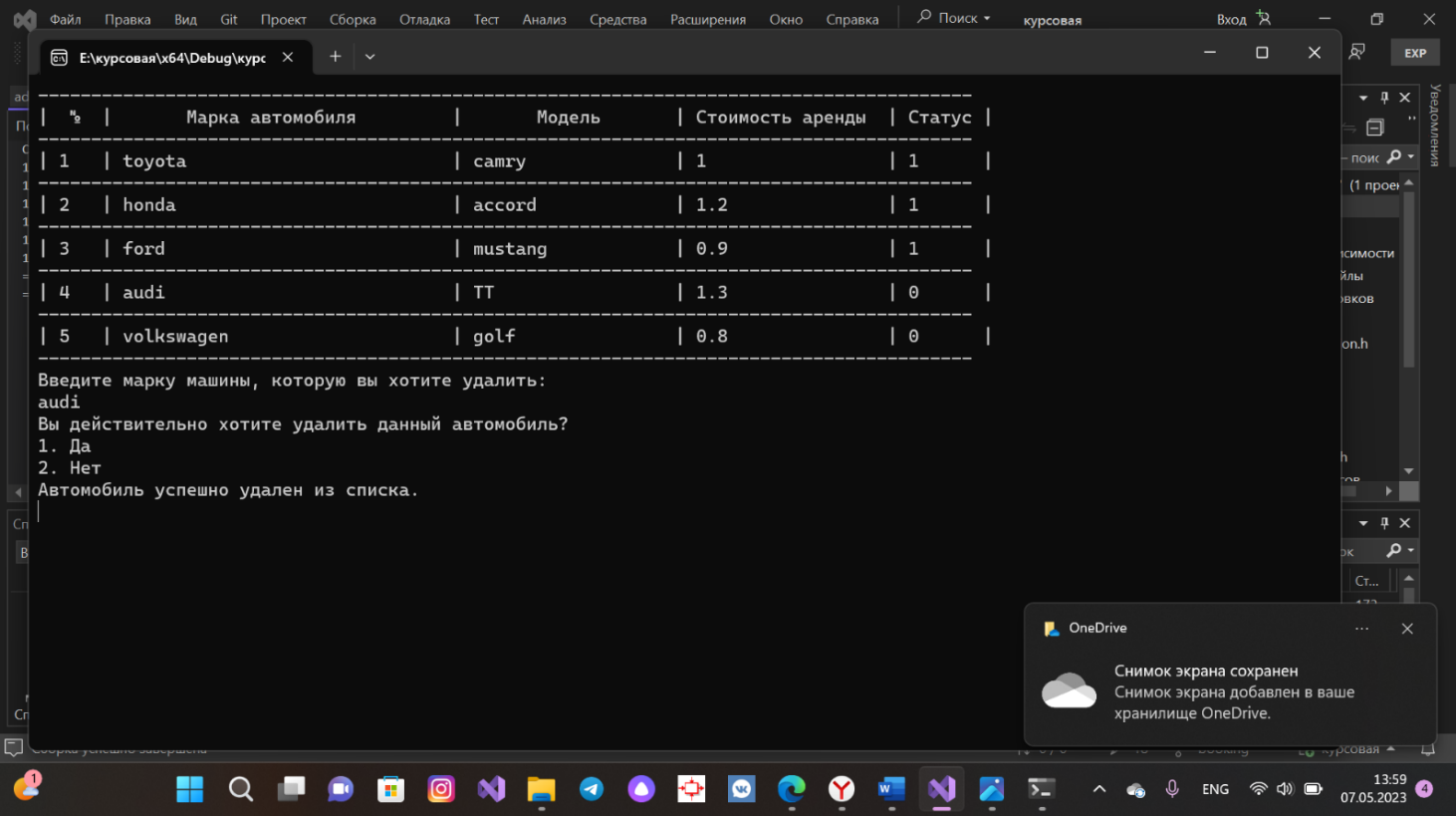


Рисунок 4.14– Удаление автомобиля

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

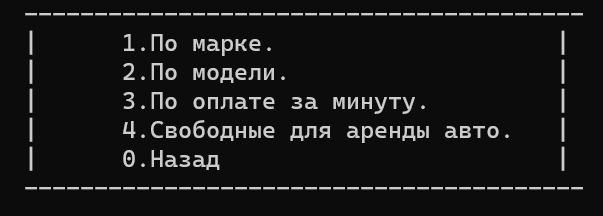


Рисунок 4.15 – Критерии поиска автомобиля

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).

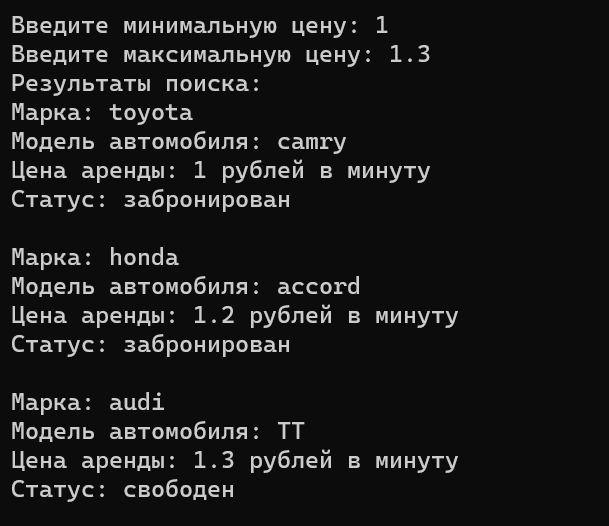


Рисунок 4.17 – Пример фильтрации данных

Сортировка автомобилей (рисунок 4.18) осуществляется по следующим критериям.

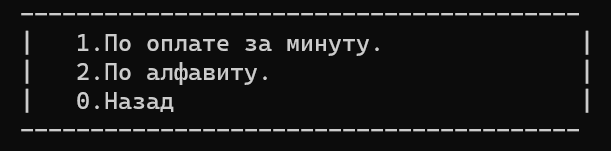


Рисунок 4.18 – Критерии сортировки автомобилей

Пример сортировки автомобилей отображен на рисунке 4.19 и 4.20.

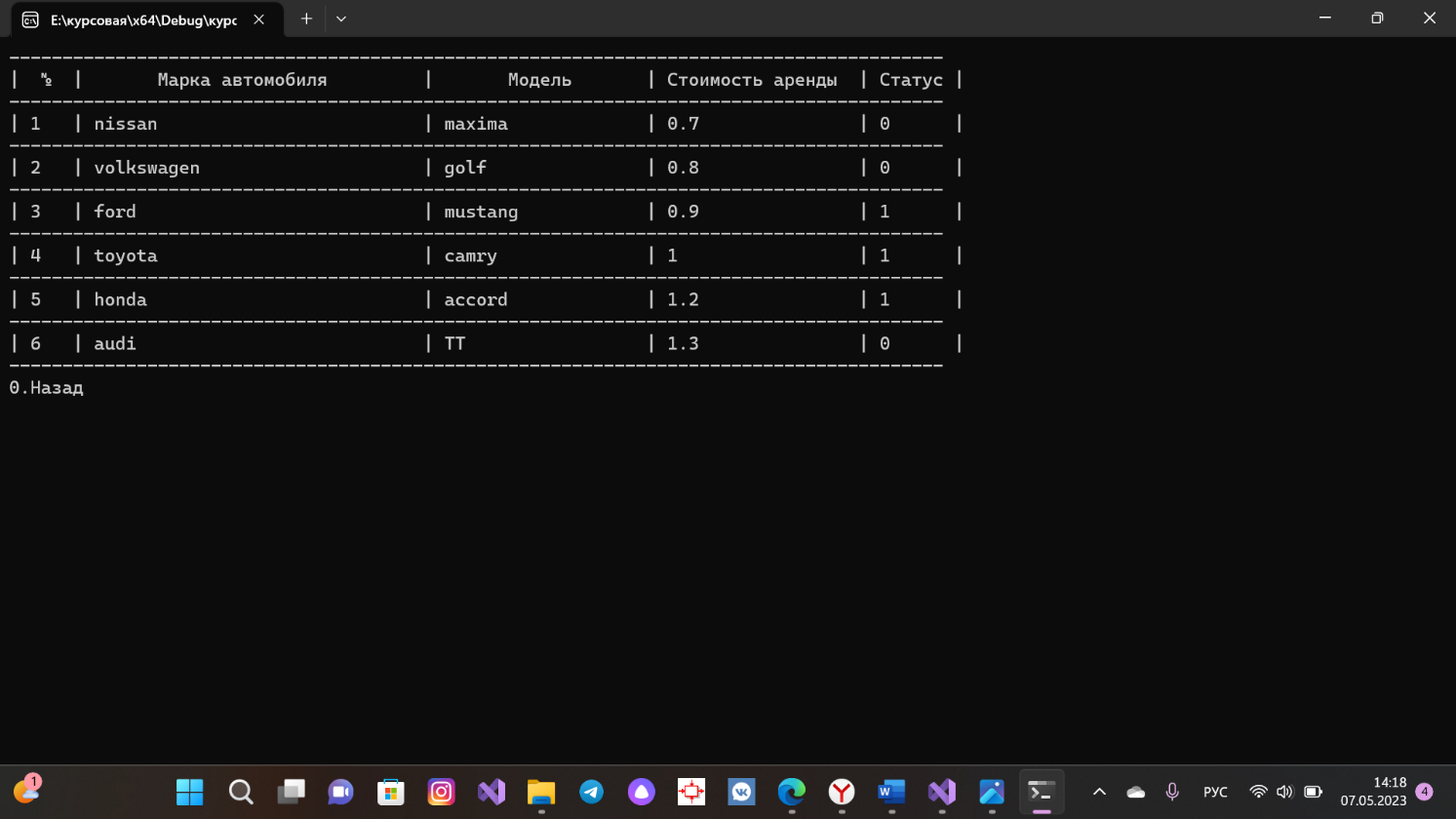


Рисунок 4.19 – Пример сортировки автомобилей по стоимости за минуту

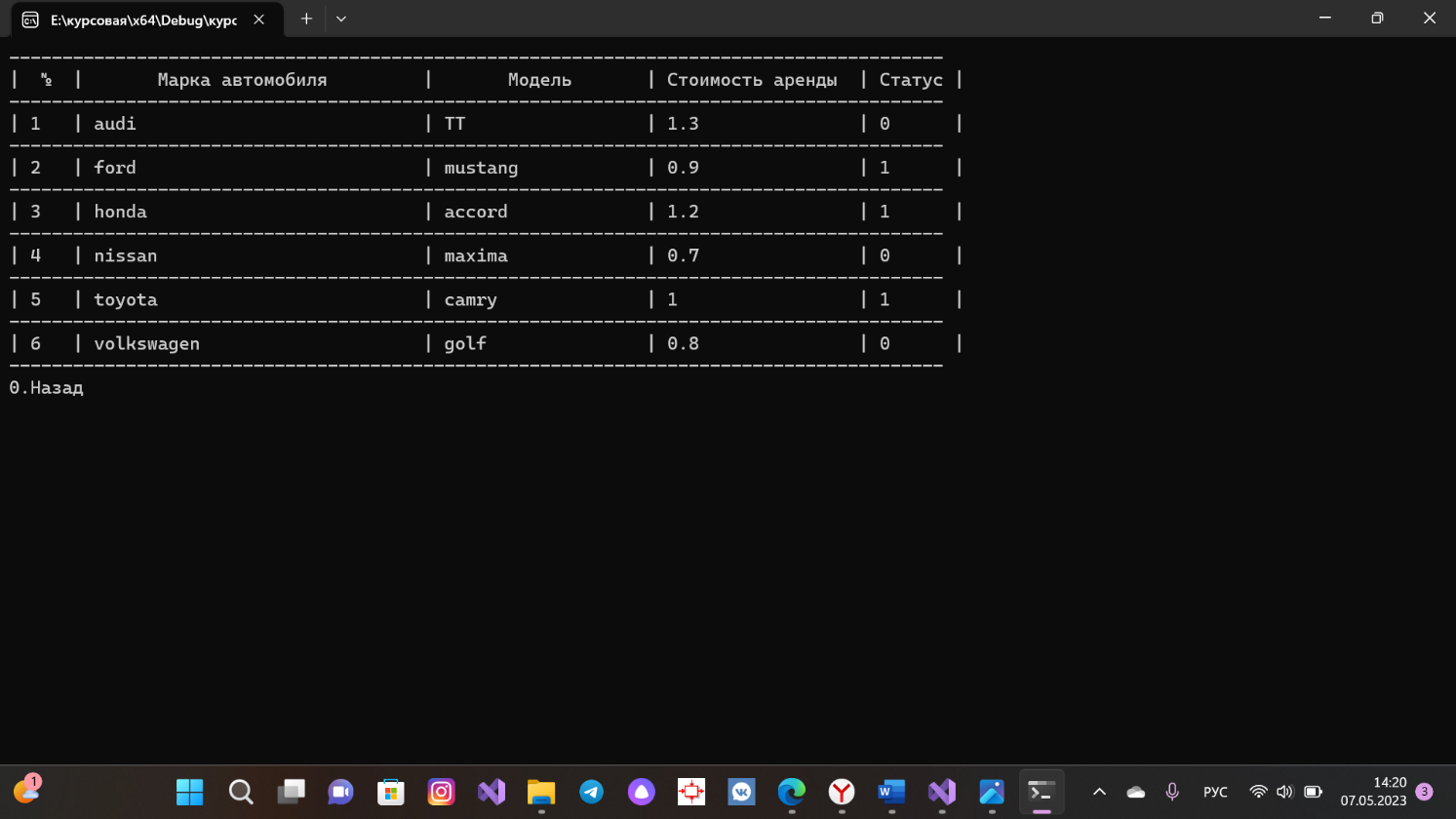


Рисунок 4.20 – Пример сортировки автомобилей по алфавиту

## 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).

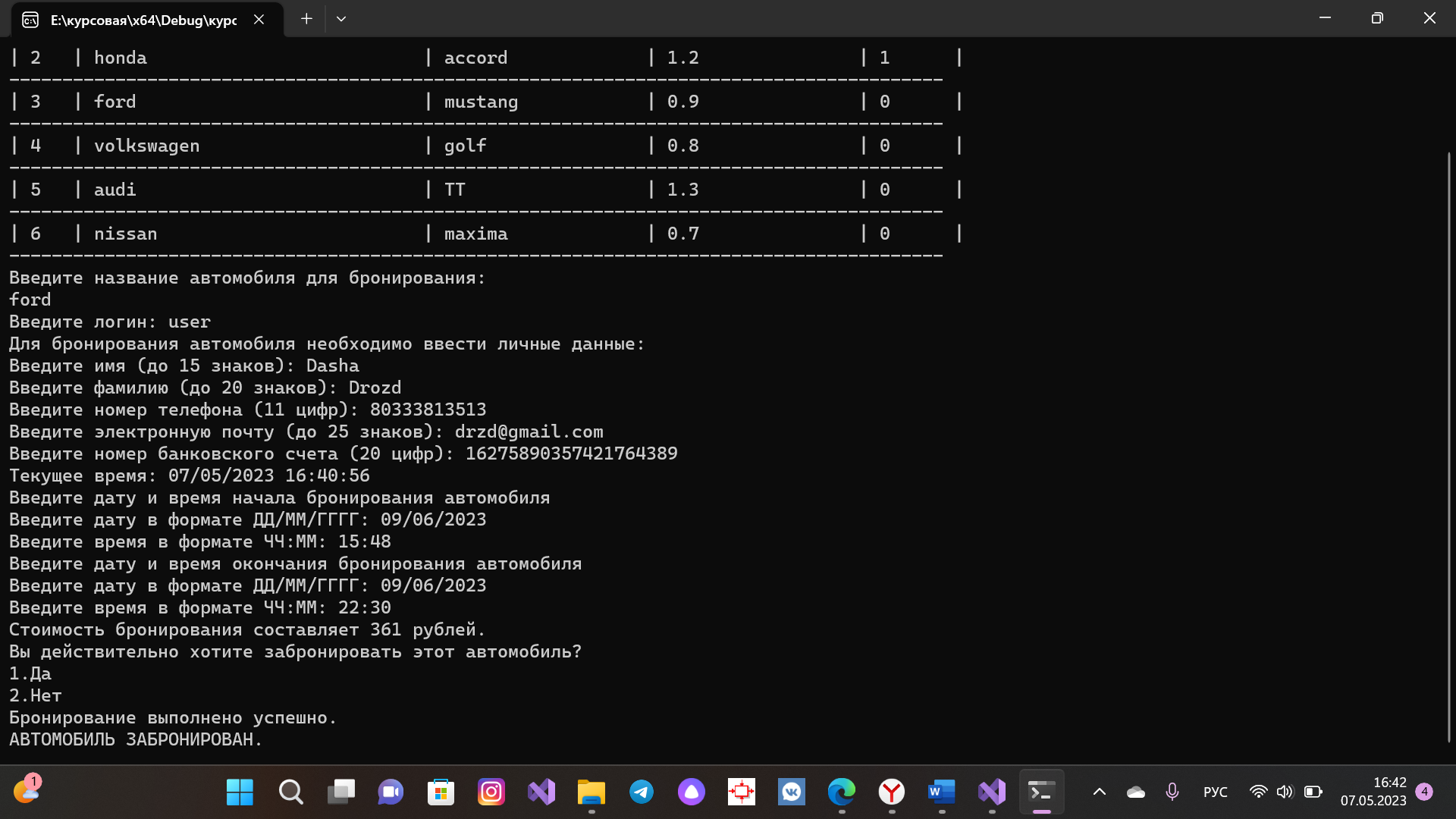


Рисунок 4.22 – Выполнение индивидуального задания

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).

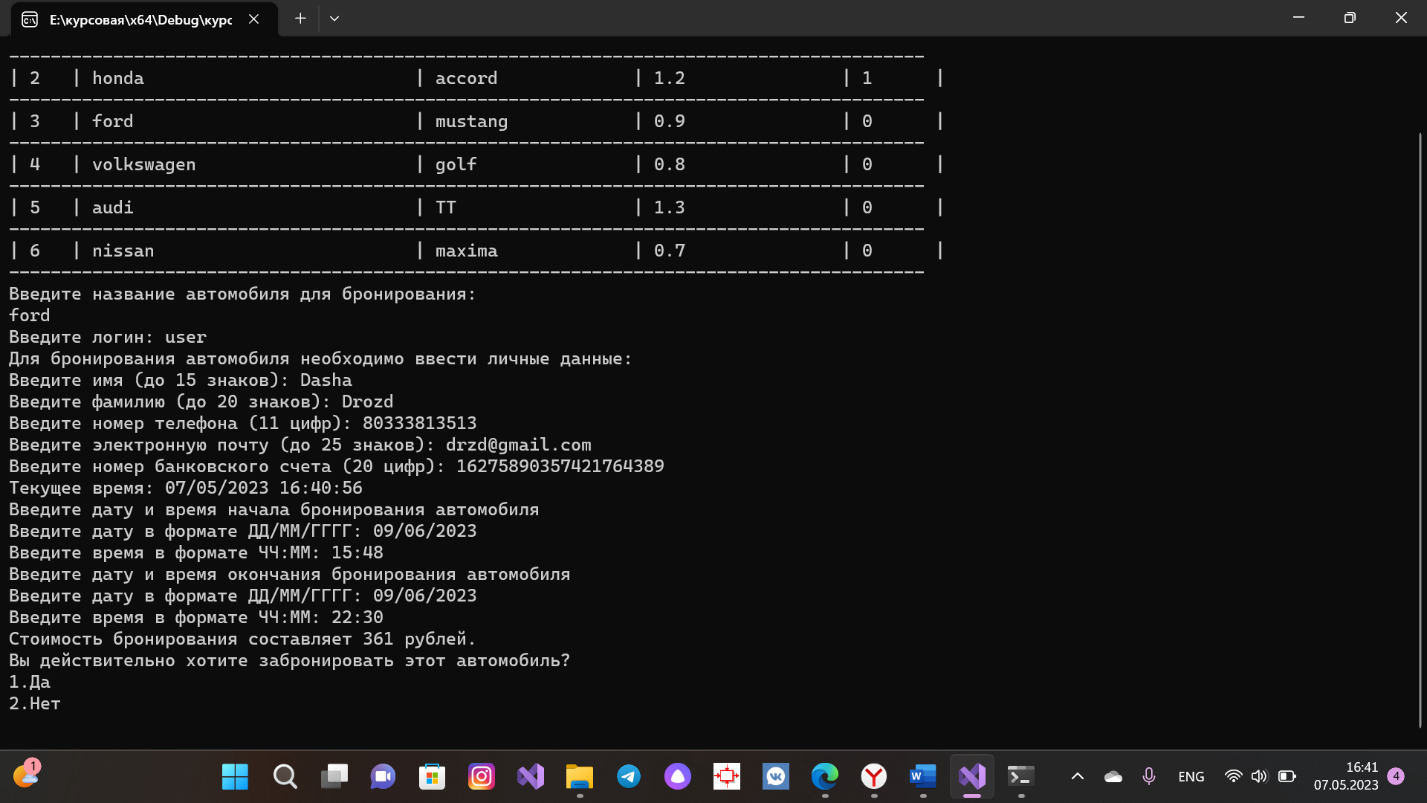


Рисунок 4.23 – Пример бронирования автомобиля

## 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).

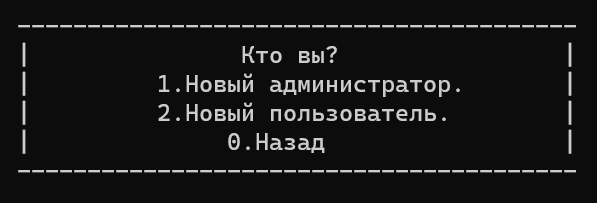


Рисунок 4.24 – Меню регистрации

Сам процесс регистрации приведен на рисунке 4.25.

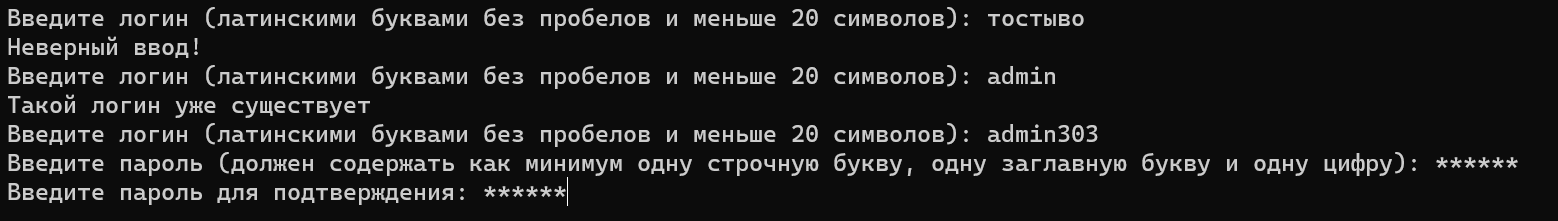


Рисунок 4.25 – Регистрация

**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]. –https://ru.wikipedia.org/wiki/IDEF0
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**

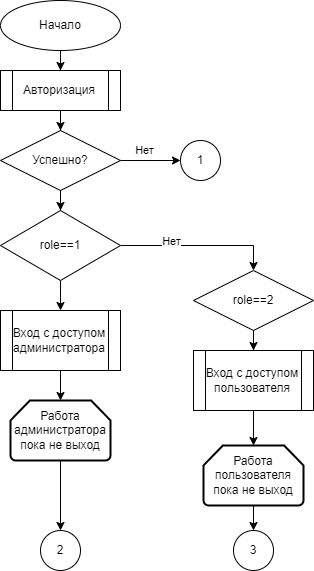
****

Рисунок Д.1 – Схема общего алгоритма работы программы

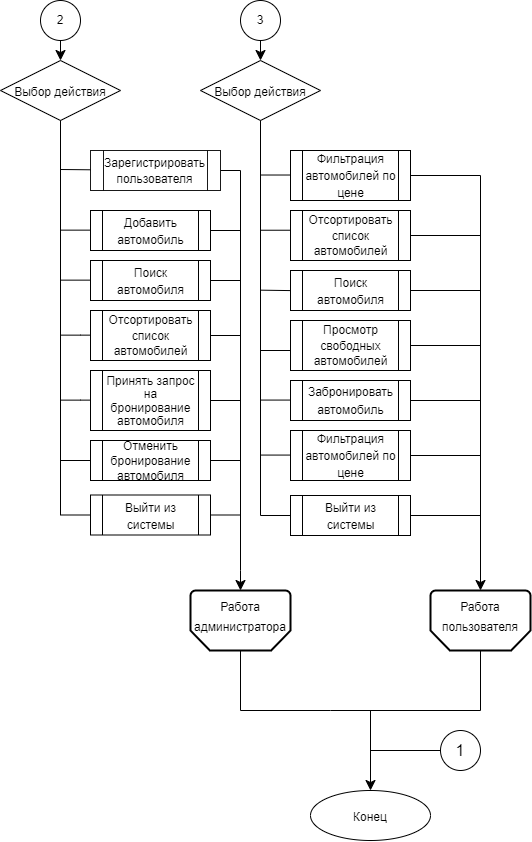
****

Рисунок Д.2 – Продолжение схемы общего алгоритма работы программы