Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Объектно-ориентированное проектирование и программирование»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Старший преподаватель кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.Н. Салапура |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Разработка АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ Аренды автомобилей»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 272303  ПИВОВАРЧИК Егор Дмитриевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc153908046)

[1 Обзор автоматизируемой предметной области, программных аналогов, методов и алгоритмов решения поставленной задачи 6](#_Toc153908047)

[1.1 Описание предметной области 6](#_Toc153908048)

[1.2 Обзор программ-аналогов по данной предметной области 7](#_Toc153908049)

[1.2.1 Сайт mir-avto.by 7](#_Toc153908050)

[1.2.2 Сайт westgroup.by 8](#_Toc153908051)

[2 Функциональное моделирование на основе стандарта IDEF0 9](#_Toc153908052)

[3 Структура используемых данных 12](#_Toc153908053)

[4 Описание созданных программных конструкций 14](#_Toc153908054)

[5 Разработка и описание диаграммы классов приложения 19](#_Toc153908055)

[6 Разработка и описание диаграммы вариантов использования 20](#_Toc153908056)

[7 Блок-схема алгоритма работы всей программы и двух основных методов 21](#_Toc153908057)

[7.1 Блок-схема алгоритма функции main 21](#_Toc153908058)

[7.2 Блок-схема алгоритма метода Admin::coutCars() 24](#_Toc153908059)

[7.3 Блок-схема алгоритма метода User::login() 25](#_Toc153908060)

[8 Описание алгоритма запуска приложения, его использования, результаты работы программы, тестирования обработки ошибок 27](#_Toc153908061)

[Заключение 33](#_Toc153908062)

[Список использованных источников 34](#_Toc153908063)

[Приложение А (обязательное). Листинг программы с комментариями 35](#_Toc153908064)

[Приложение Б. Проверка на заимствование в системе «Антиплагиат» 40](#_Toc153908065)

[Приложение В (обязательное). Диаграмма классов. 41](#_Toc153908066)

[Приложение Г (обязательное). Диаграмма вариантов использования. 42](#_Toc153908067)

[Приложение Д (обязательное). Блок-схемы алгоритмов. 43](#_Toc153908068)

# Введение

В современном мире технологического прогресса и цифровой трансформации, системы автоматизированной аренды автомобилей становятся неотъемлемой частью индустрии транспорта. Эти системы предоставляют эффективные и удобные решения для арендаторов и арендодателей, улучшая процессы управления автопарком и повышая уровень обслуживания клиентов.

Автоматизированные системы аренды автомобилей помогают управлять всем процессом от начала до конца. Они включают в себя функции отслеживания состояния автомобилей, их местоположения, обслуживание и регулирование тарифов. Онлайн-сервисы предоставляют удобный и быстрый способ для клиентов арендовать автомобили. Они могут выбирать автомобили и бронировать их прямо онлайн, обеспечивая максимальное удобство и доступность. Необходимость создания приложения может благоприятно повлиять на деятельность отдельных арендных агентств, и помочь им стать более успешными на конкурентном рынке аренды авто.

Автоматизированные системы позволяют предоставлять более высокий уровень обслуживания клиентов. Это включает в себя уведомления, отзывы, программы лояльности и другие функции, которые способствуют улучшению взаимодействия с клиентами.

Приложение значительно сокращает время, затрачиваемое на выполнение различных операций вручную. Это снижает нагрузку на персонал и позволяет более эффективно использовать ресурсы.

Также системы автоматизации предоставляют данные и аналитику, которые могут быть использованы для принятия бизнес-решений, оптимизации процессов и предсказания трендов в индустрии.

Все перечисленные факторы обосновывают создание автоматизированной системы, что и является целью курсового проекта.

Задачи работы:

* ознакомиться с предметной областью;
* выбрать подходящие аналоги и исследовать их функционал;
* проанализировать процессы;
* спроектировать основной процесс;
* продумать систему работы основных функций программы;
* создать программу, реализующую функционал;
* протестировать её работу.

Объектом исследования данной курсовой работы является автоматизированная система аренды автомобилей

# 1 Обзор автоматизируемой предметной области, программных аналогов, методов и алгоритмов решения поставленной задачи

## 1.1 Описание предметной области

Аренда автомобилей - это популярный и важный сервис в наше время, предоставляющий клиентам доступ к автомобилям без необходимости их приобретения. Аренда автомобилей стала особенно востребованной в наше время так как она предлагает многочисленные возможности для путешествия людей без необходимости приобретения авто. Аренда автомобилей является очень гибким сервисом, так как клиент сам в праве выбирать транспорт в зависимости от его конкретных потребностей.

Мобильность становится ключевым элементом в современной жизни, и аренда автомобилей позволяет быстро и удобно перемещаться.

Также владение собственным автомобилем связано с значительными затратами на его покупку, страхование, обслуживание и топливо. Аренда автомобиля позволяет экономить средства, освобождая клиентов от финансовых обязательств по владению транспортом.

Аренда предоставляет путешественникам уникальную возможность исследовать новые места без необходимости пересадок и зависимости от общественного транспорта.

В дополнение к обычным путешествиям для бизнес-путешественников аренда автомобиля является удобным способом перемещения между встречами и деловыми мероприятиями.

Корпорации часто используют услуги аренды автомобилей для обеспечения мобильности своих сотрудников без необходимости содержания собственного автопарка.

Модели совместного использования, такие как каршеринг, становятся все более популярными, обеспечивая удобство использования автомобилей только по мере необходимости.

С появлением мобильных приложений и онлайн-платформ, аренда автомобилей стала более доступной и удобной для заказа и управления. Нужно реализовать приложение для какой-либо компании, которое предоставит возможность реализовать аренду мобильнее и быстрее, что даст ей преимущество на рынке конкурентов. Нужно разобраться как происходит аренда автомобиля офлайн, чтобы спроецировать онлайн систему:

* Выбор автомобиля (клиенту предоставляется выбор из имеющегося автопарка);
* Предъявление документов (аренда авто предоставляется клиентам, имеющим допуск к участию в дорожном движении, то есть права на управление автомобилем соответствующей категории);
* Заполнение договора (после выбора автомобиля и предъявления документов клиент заполняет договор аренды. В этом документе указываются условия аренды, сроки, тарифы, страховка и другие важные параметры);
* Проверка автомобиля (перед выдачей автомобиля клиенту производится проверка его состояния);
* Оплата и залог (клиент оплачивает аренду автомобиля и может внести залог в виде блокировки средств на кредитной карте. Этот залог используется в случае ущерба автомобилю или несоблюдения условий договора);
* Получение ключей (после завершения формальностей клиент получает ключи от автомобиля и может начать использовать его в соответствии с условиями договора);
* Возврат автомобиля (в конце срока аренды клиент возвращает автомобиль в пункт проката);

## 1.2 Обзор программ-аналогов по данной предметной области

В данной предметной области имеется огромное количество Web-аналогов, которые предлагают аренду машин с множеств компаний-сотрудников Web-приложения. Можно реализовать их интерфейс в своем приложении, и узнать побольше информации о требуемых данных.

## 1.2.1 Сайт mir-avto.by

Сайт предоставляет каршеринговые услуги онлайн. Позволяет выбрать автомобиль для аренды, написать менеджеру и отправить ему фотографии документов, допускающие участие в дорожном движении, и забрать автомобиль с указанного места в указанное время.

Рисунок 1.1 показывает интерфейс выбора автомобиля.

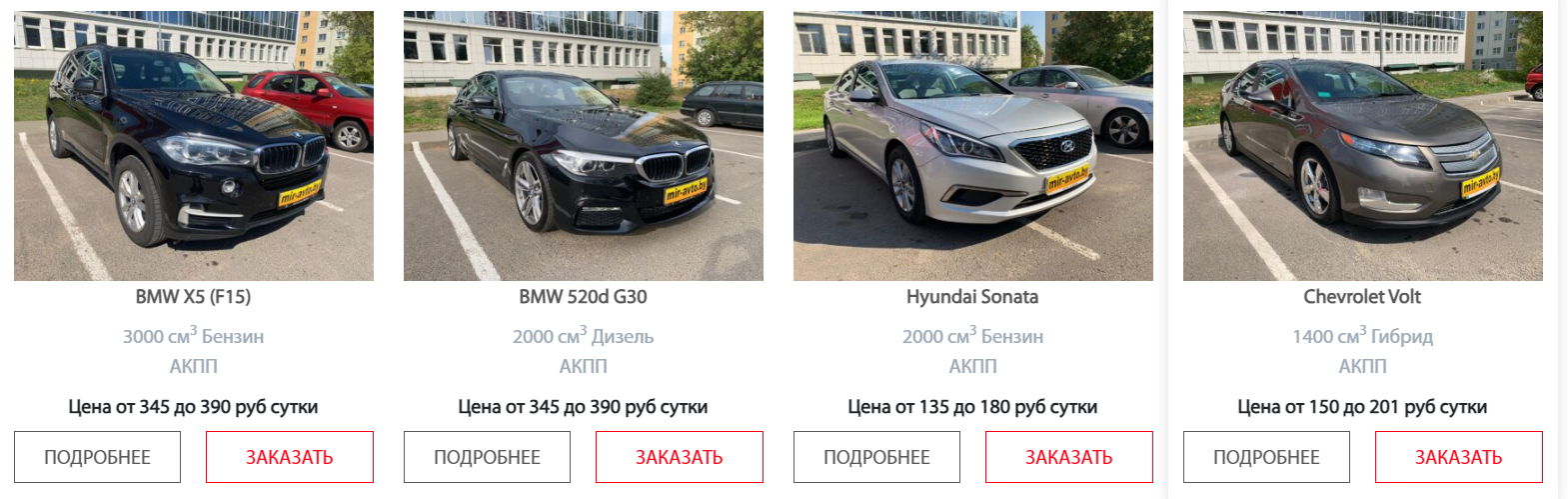


Рисунок 1.1 – Популярные авто на сайте mir-avto.by

## 1.2.2 Сайт westgroup.by

Сайт предоставляет возможность бронирования автомобилей от конкретной фирмы WestGroup и получения автомобиля в указанном месте.

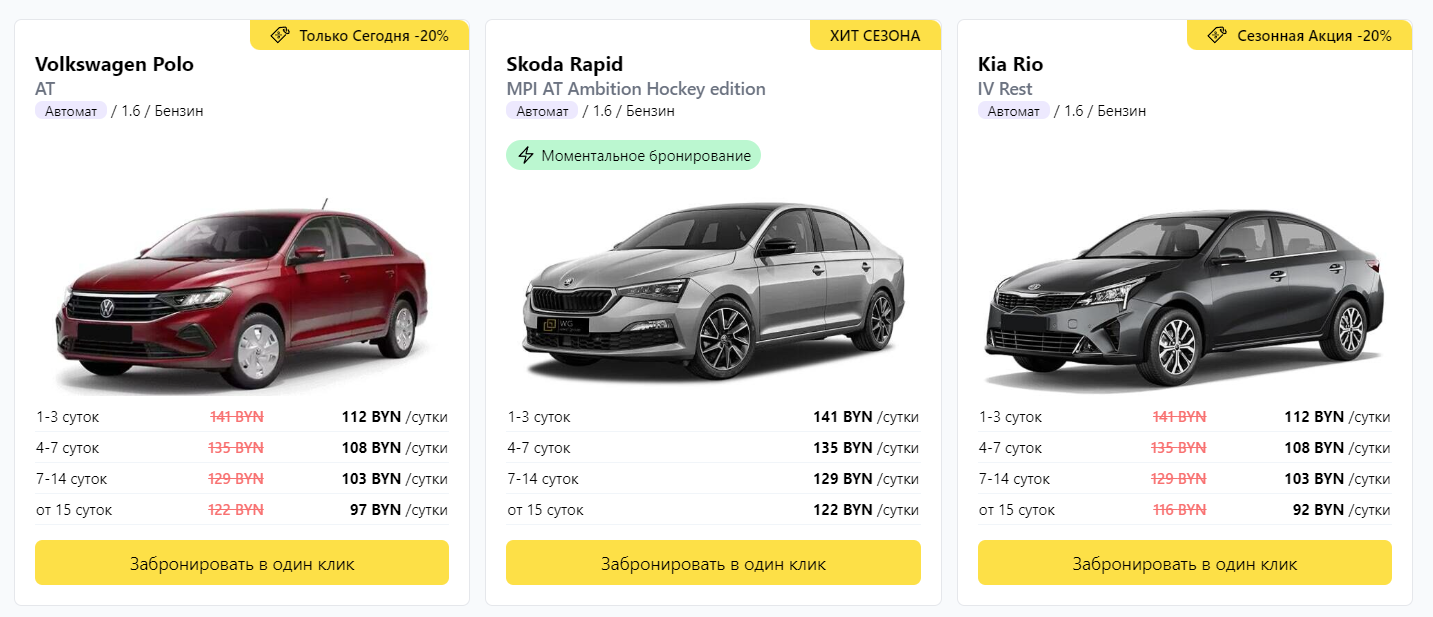


Рисунок 1.2 – Возможность выбора авто на сайте WestGroup

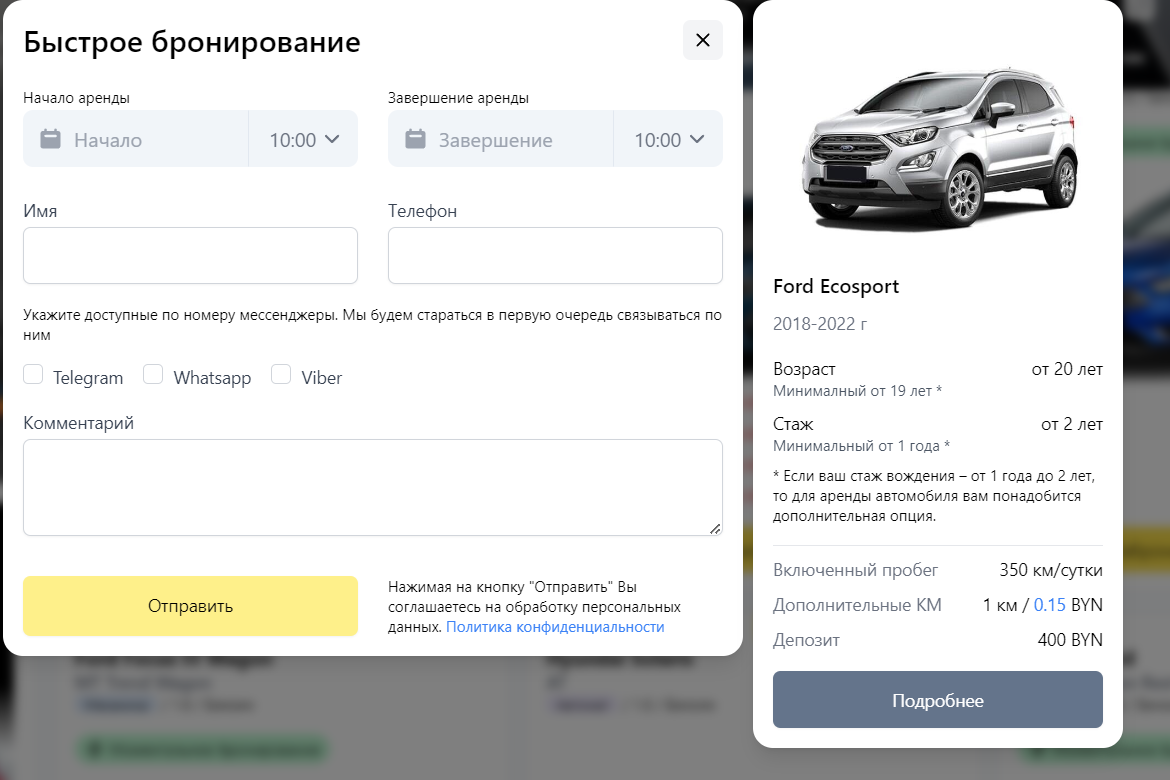


Рисунок 1.3 – Ввод данных для бронирования автомобиля

# 2 Функциональное моделирование на основе стандарта IDEF0

Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки. Для передачи информации о конкретной системе источником графического языка является сама методология IDEF0.

Методология IDEF0 предписывает построение иерархической системы диаграмм – единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция – система разбивается на подсистемы, и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

Нотация IDEF0 позволяет моделировать функционал описываемой области. Это дает возможность наглядно увидеть все достоинства и недостатки предмета исследования, что обеспечивает удобство анализа протекающих процессов.

Основные процессы разработки программы отражены на IDEF0-диаграмме.

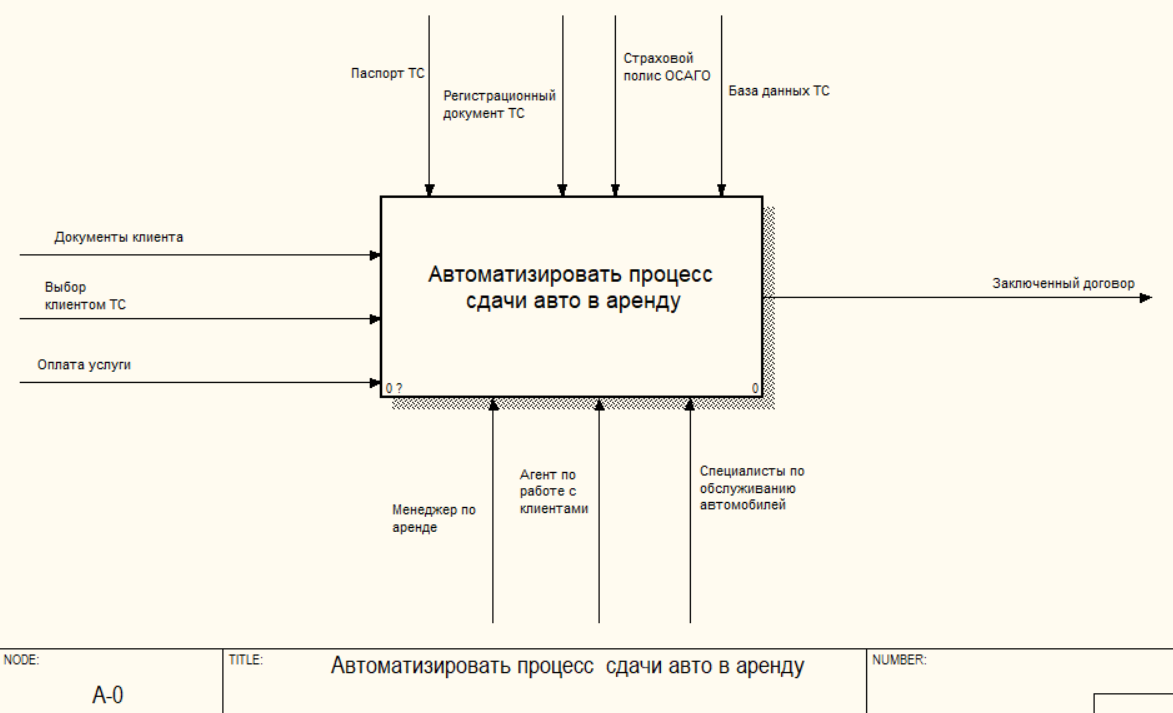


Рисунок 2.1 – Контекстная диаграмма процесса «Автоматизировать процесс сдачи авто в аренду»

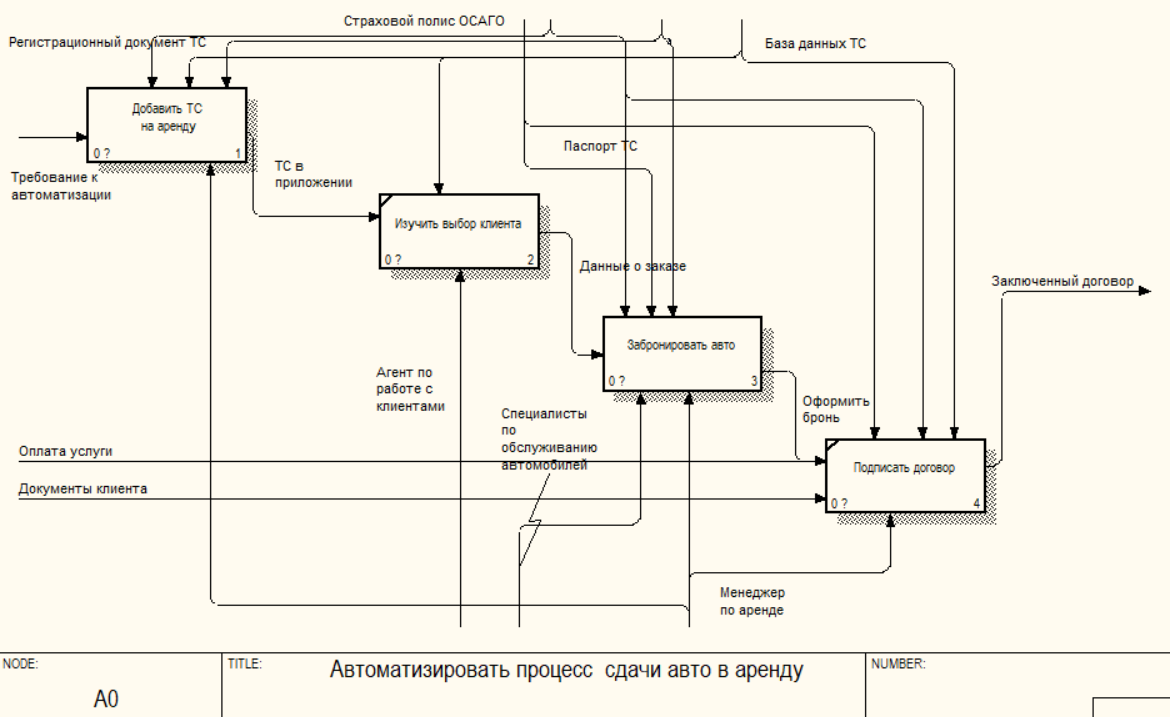


Рисунок 2.2 – Декомпозиция контекстной диаграммы

На Рисунке 2.2 представлены основные шаги для автоматизации сдачи авто в аренду, а именно:

* Добавление авто в программу;
* Изучение выбора клиента;
* Бронь автомобиля;
* Непосредственное подписание договора об аренде.

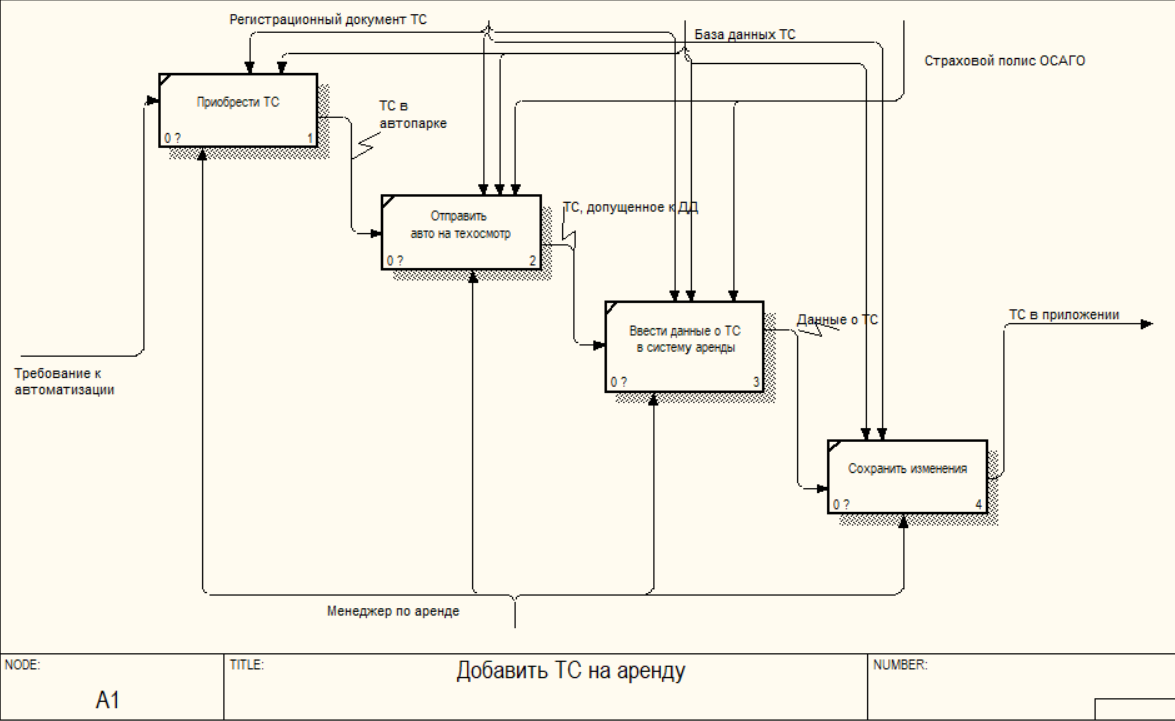


Рисунок 2.3 – Декомпозиция процесса «Добавить ТС на аренду»

Декомпозиция этого этапа показывает все шаги, которые стоит предпринять менеджеру по аренде авто, для его добавления в автопарк.

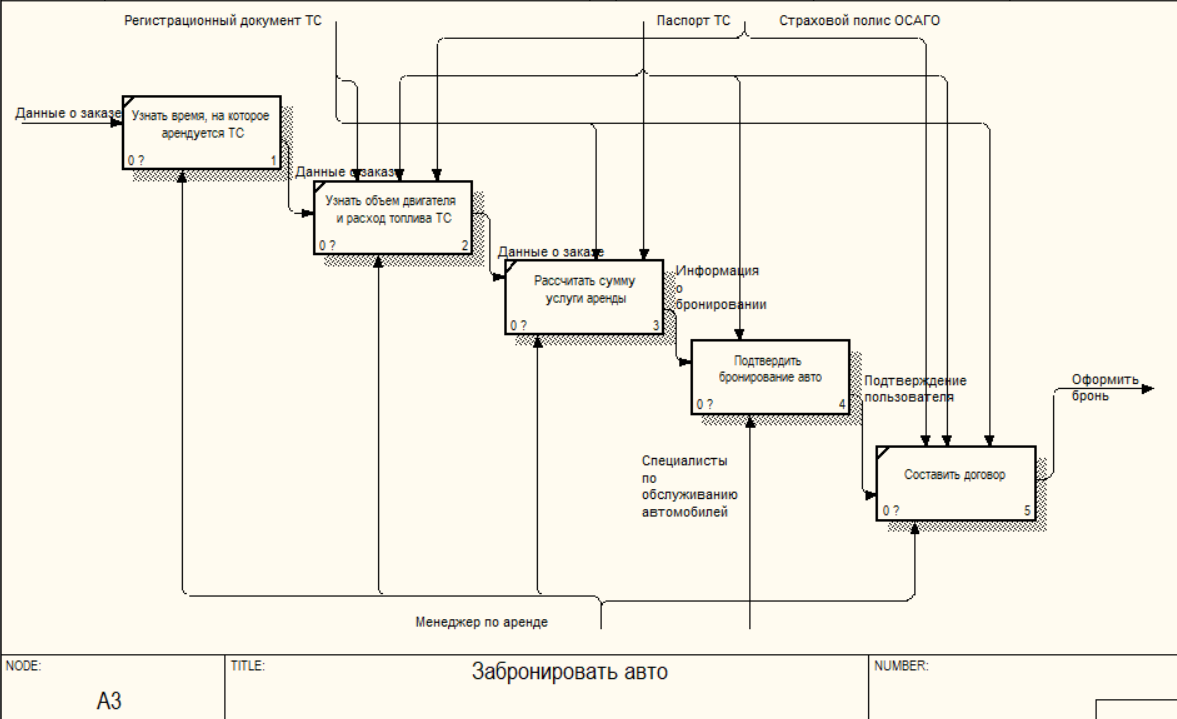


Рисунок 2.4 – Декомпозиция процесса «Забронировать авто»

При декомпозиции этого процесса мы можем видеть, как при желании клиента забронировать автомобиль действуют внутри приложения и какие потоки данных при этом образуются.

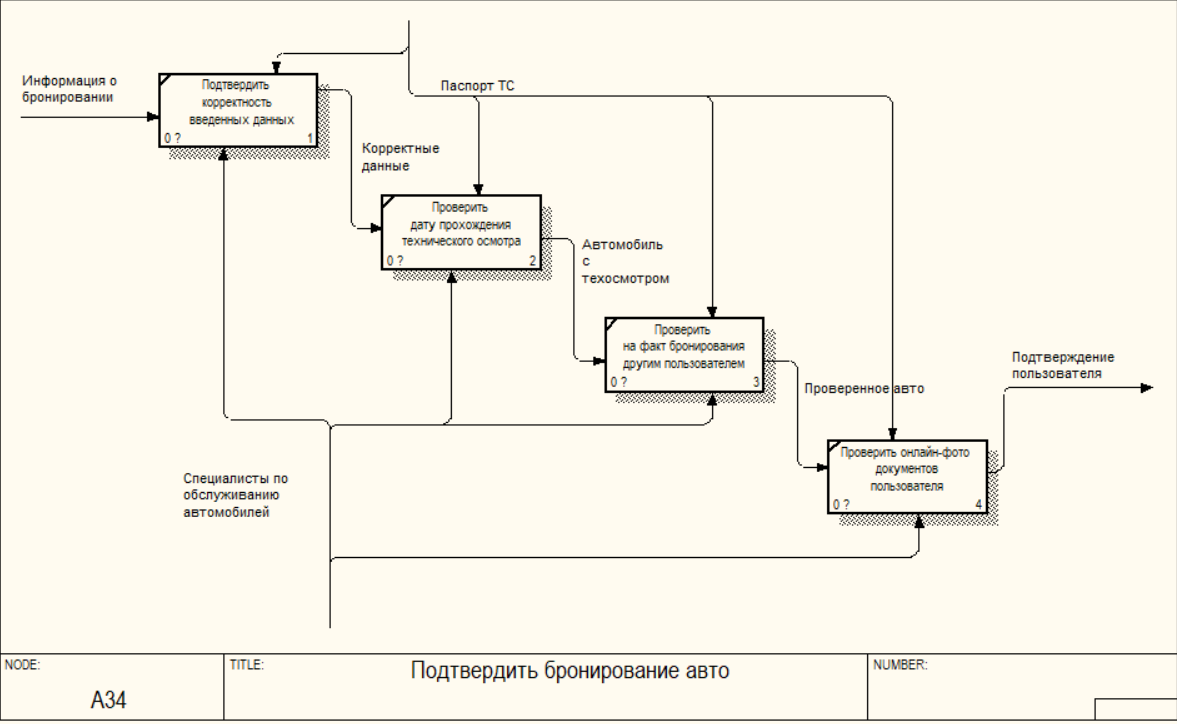


Рисунок 2.5 – Декомпозиция процесса «Подтвердить бронирование авто»

# 3 Структура используемых данных

Для организации данных были выбраны классы и структуры, список которых приведён ниже.

Таблица 3.1 – Переменные класса Car

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| body | string | Хранит кузов автомобиля |
| type | string | Хранит класс автомобиля |
| name | string | Хранит модель автомобиля |
| engine\_capacity | float | Хранит объем двигателя |
| fuel\_type | string | Хранит тип топлива |
| transmission | string | Хранит тип трансмиссии |
| fuel\_use | float | Хранит расход топлива |
| booked | int | Хранит количество бронирований |
| cost | float | Хранит стоимость аренды в день |

Таблица 3.2 – Переменные класса User

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| name | string | Хранит имя пользователя |
| surname | string | Хранит фамилию пользователя |
| login | string | Хранит логин пользователя |
| password | string | Хранит пароль пользователя |
| cars | static vector<car> | Хранит все автомобили в программе |
| all\_reservations | static vector<reservation> | Хранит все брони в программе |

Таблица 3.3 – Переменные класса Reservation

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| receiver\_login | string | Хранит логин пользователя, бронирующего авто |
| vehicle | Car | Хранит автомобиль, который бронируется |
| acquisition\_date | Date | Хранит дату получения автомобиля |
| return\_date | Date | Хранит дату возвращения автомобиля |

Таблица 3.4 – Переменные структуры Date

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| day | int | Хранит день в составе даты |
| month | int | Хранит месяц в составе даты |
| year | int | Хранит год в составе даты |

Классы Admin и Client наследуются от класса User и имеют уникальные методы для каждой роли.

Методы класса Admin:

* void addCar();
* void addAdmin();
* void coutCars();
* void changeAccountinfo();
* void cout\_reservations();

Методы класса Client:

* void choosecar();
* void cout\_reservations() {};
* void notifications() {};
* void remove\_reservation(){};

Данные классы помогают удобно организовать работу системы, храня в себе данные, и позволяют проводить различные операции, требуемые для работы.

# 4 **Описание созданных программных конструкций**

В коде программы реализованы следующие программные конструкции:

* реализация базовых принципов объектно-ориентированного программирования (инкапсуляции, полиморфизм, наследование) представлена на рисунке 4.1, рисунке 4.2, рисунке 4.3 :

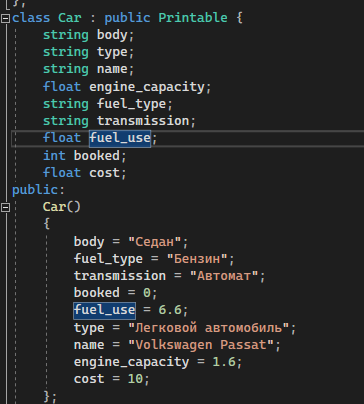
****

Рисунок 4.1 – Реализация public, private полей и конструкторов, как пример инкапсуляции

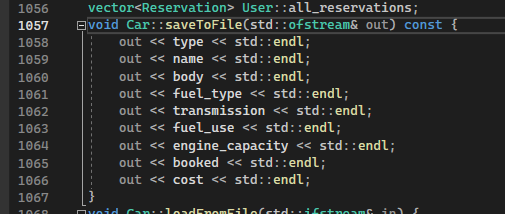


Рисунок 4.2 – Реализация перегрузки метода saveToFile, как пример полиморфизма

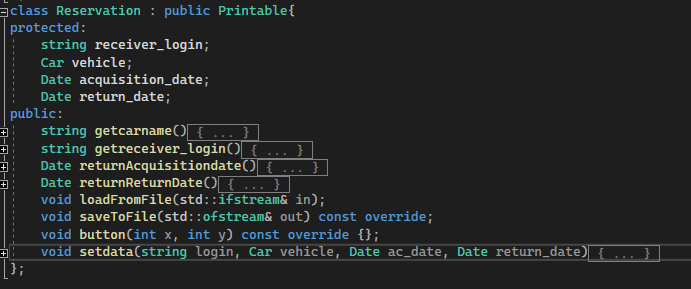


Рисунок 4.3 – Реализация наследования

– использование абстрактных классов (рисунок 4.4);

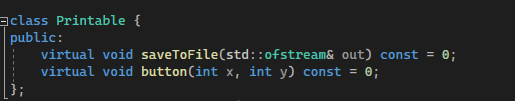


Рисунок 4.4 – Реализация абстрактного класса

– использование передачи параметров по ссылке и по значению (рисунок 4.5);



Рисунок 4.5 – Передача параметров в функции по ссылке и по значению

– использование встроенных (рисунок 4.6), пользовательских (рисунок 4.7), дружественных (рисунок 4.8), виртуальных функции (рисунок 4.9);

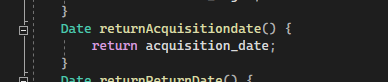


Рисунок 4.6 – Использование встроенной функции

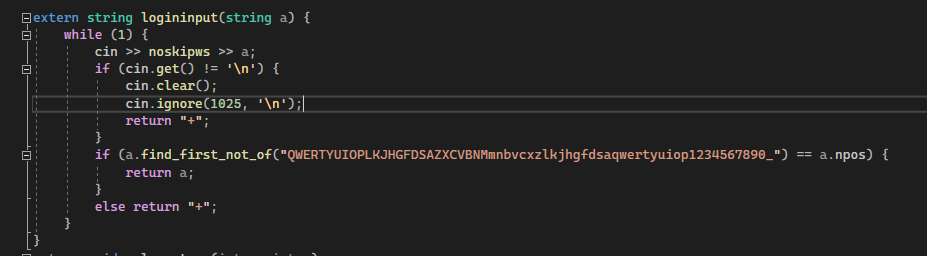


Рисунок 4.7 – Использование пользовательских функций

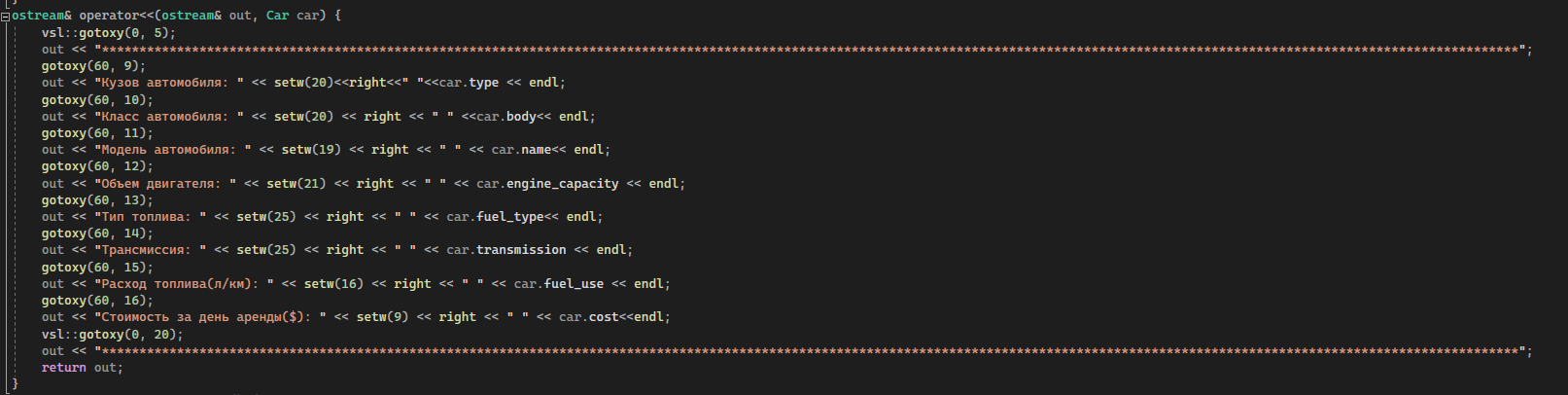


Рисунок 4.8 – Использование дружественных функций

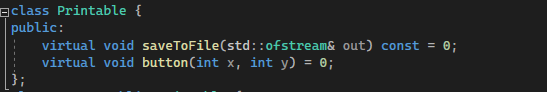


Рисунок 4.9 – Использование виртуальных функций

– использование пространств имён (встроенных и собственных (рисунок 4.10));

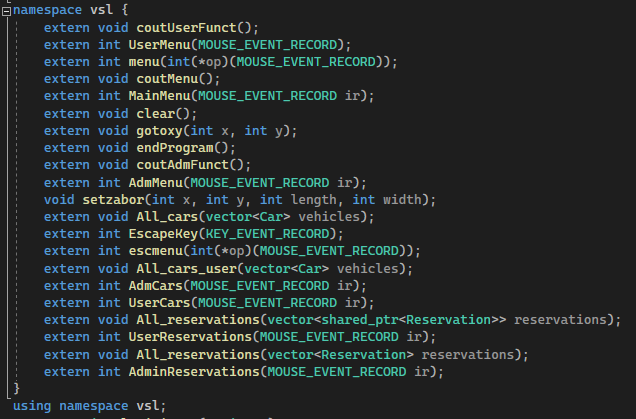


Рисунок 4.10 – Реализация собственного пространства имён

– реализация обработки ошибок программы (средствами языка С++) (рисунок 4.11);

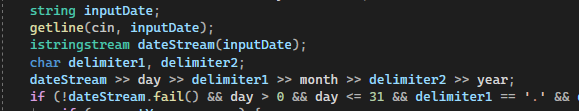


Рисунок 4.11 – Реализация обработки ошибок

– использование перегрузки методов (рисунок 4.12) и операторов (рисунок 4.13), переопределение методов;



Рисунок 4.12 – Использование перегруженного метода saveToFile

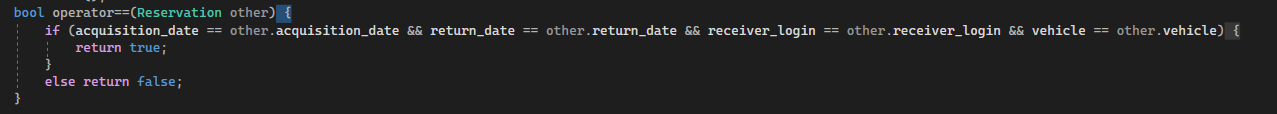


Рисунок 4.13 – Использование перегруженного оператора==

– использование шаблонов классов и методов (рисунок. 4.14);

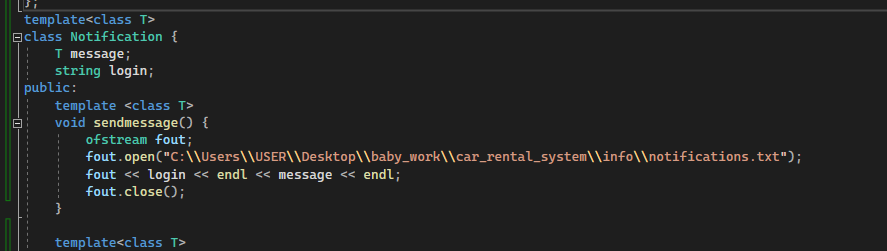


Рисунок 4.14 – Реализация шаблонного класса и шаблонного метода

– использование статических методов и полей (рисунок. 4.15, 4.16);

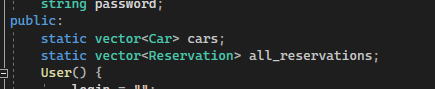


Рисунок 4.15 – Реализация статических полей

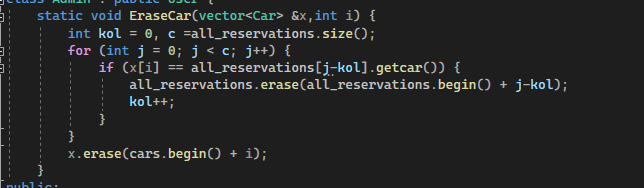


Рисунок 4.16 – Реализация статических методов

– использование динамического выделения памяти и умных указателей (smart pointers) (рисунок. 4.17);



Рисунок 4.17 – Использование умного указателя shared\_ptr

– использование потоков С++, перегрузки операторов ввода/вывода (рис. 4.18), контролирования работы с потоком.

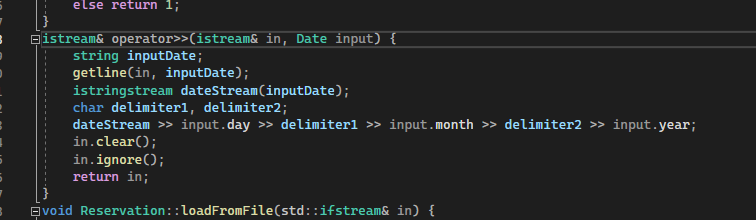


Рисунок 4.18 – Реализация перегрузки операторов ввода/вывода в поток и работы с потоком

# 5 Разработка и описание диаграммы классов приложения

Диаграмма классов используется для представления физического и логического строения разрабатываемого продукта. На UML-диаграмме классов показаны классы и их методы (рис. 5.1).

В первую очередь при погружении в предметную область необходимо было разработать основные классы: User, Admin, Client – которые бы хранили общие данные пользователей и обеспечили основные процессы автоматизации аренды автомобилей. Классы Car и Reservation предназначены для хранения данных о машинах и бронированиях соответственно, что упрощает написание программного кода для приложения. Класс Printable и структура Date являются вспомогательными средствами для реализации других возможностей программы.

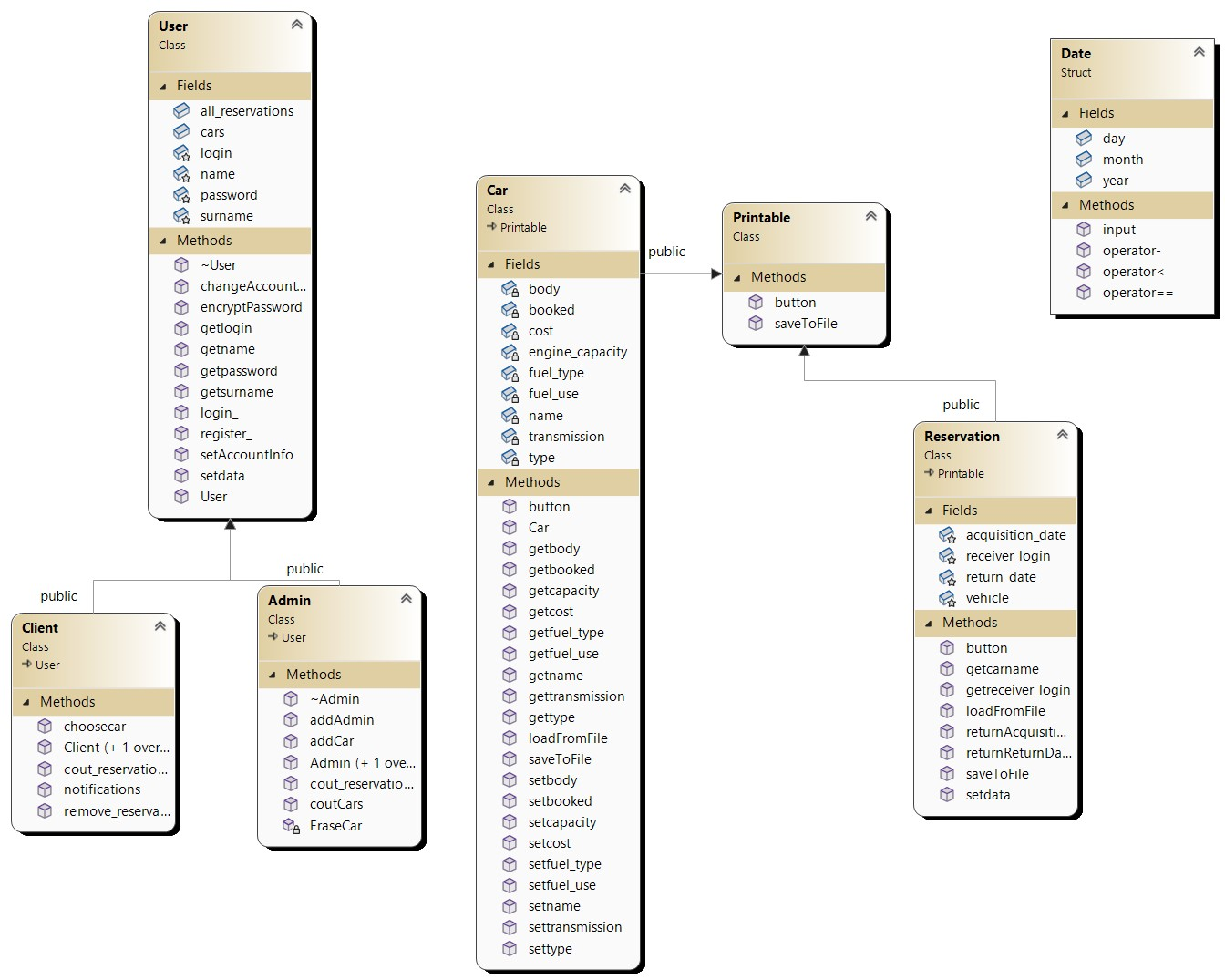


Рисунок 5.1 – Диаграмма используемых классов

# 6 Разработка и описание диаграммы вариантов использования

Диаграмма вариантов использования — это визуальное представление функциональности системы, описывающее, как различные типы пользователей взаимодействуют с системой.

В программе выделяются 2 основные роли (рисунок 6.1): клиент и администратор. Каждая из ролей представляет собой набор сценариев использования приложения, которые отражают уникальный функционал каждой роли.

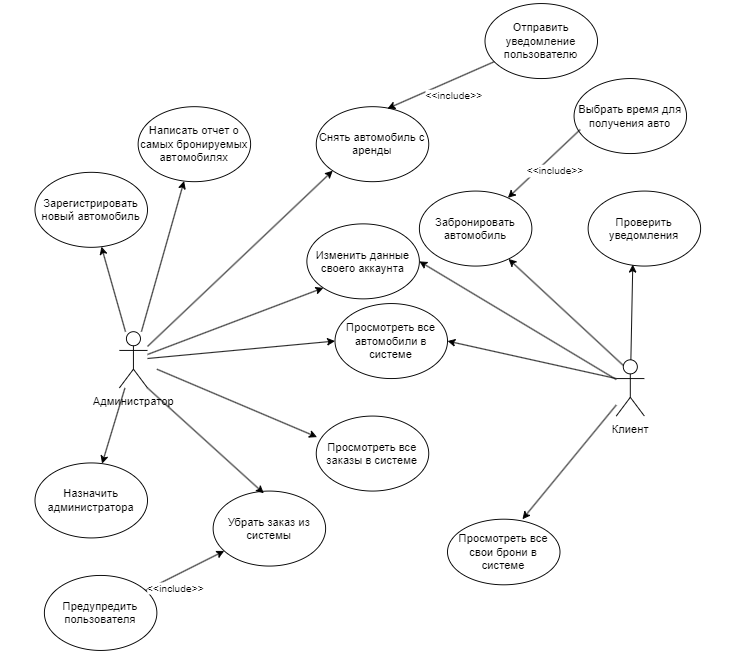


Рисунок 6.1 – Диаграмма вариантов использования

Диаграмма вариантов использования позволяет подробно изучить особенности и возможности каждой роли при пользовании приложением, а также показывает полный потенциал функционала каждого варианта использования с помощью расширений и включений.

# 7 Блок-схема алгоритма работы всей программы и двух основных методов

## 7.1 Блок-схема алгоритма функции main

Функция int main является объединяющей функцией всей программы. Ее алгоритм представлен на рис. 7.1.

Изображение выглядит как текст, диаграмма, зарисовка, План

Автоматически созданное описание

Рисунок 7.1 – Алгоритм функции int main()

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("chcp 1251");

keybd\_event(VK\_MENU, 0x38, 0, 0);

keybd\_event(VK\_RETURN, 0x1c, 0, 0);

keybd\_event(VK\_RETURN, 0x1c, KEYEVENTF\_KEYUP, 0);

keybd\_event(VK\_MENU, 0x38, KEYEVENTF\_KEYUP, 0);

User new\_user;

Admin new\_admin;

Client new\_client;

int state;

while (1)

{

clear();

switch (menu(MainMenu)) {

case 1:

new\_user.register\_();

gotoxy(80, 12);

cout << "Вы были успешно зарегистрированы" << endl;

gotoxy(80, 13);

system("pause");

clear();

coutMenu();

break;

case 2:

state = new\_user.login\_();

gotoxy(80, 12);

cout << "Успешная авторизация" << endl;

gotoxy(80, 13);

system("pause");

clear();

switch (state) {

case 1:

new\_client.setdata(new\_user);

while (state != 6) {

coutUserFunct();

gotoxy(80, 2);

new\_client.getreservations();

cout << "Добро пожаловать " << new\_client.getname();

gotoxy(75, 3);

cout << "Это система по аренде автомобилей!" << endl;

state = menu(UserMenu);

switch (state) {

case 1:new\_client.choosecar();

clear();

break;

case 2:new\_client.cout\_reservations();

clear();

break;

case 3:new\_client.changeAccountinfo();

clear();

break;

case 4:new\_client.notifications();

clear();

break;

case 5:new\_client.change\_reservation();

clear();

break;

case 6:

break;

}

}

break;

case 2:

new\_admin.setdata(new\_user);

while (state != 6) {

coutAdmFunct();

gotoxy(80, 2);

cout << "Добро пожаловать " << new\_admin.getname();

gotoxy(80, 3);

cout << "Это меню администратора!" << endl;

state = menu(AdmMenu);

switch (state) {

case 1:new\_admin.addCar();

clear();

break;

case 2:new\_admin.coutCars();

clear();

break;

case 3:new\_admin.changeAccountinfo();

clear();

break;

case 4:new\_admin.cout\_reservations();

clear();

break;

case 5:new\_admin.addAdmin();

break;

case 6:break;

}

}

}

break;

case 3:endProgram();

return 0;

break;

}

}

return 0;

}

## 7.2 Блок-схема алгоритма метода Admin::coutCars()

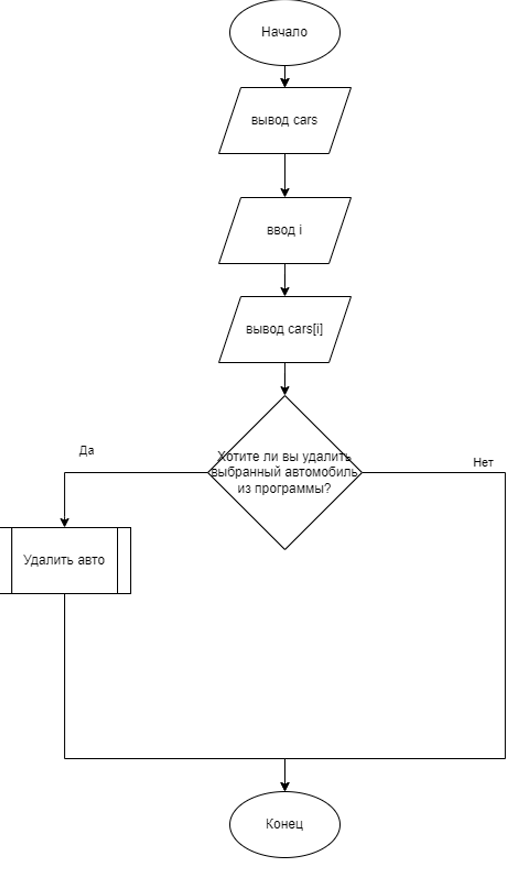


Рисунок 7.2 – Алгоритм метода Admin::coutCars()

Метод coutCars() предназначен для предоставления возможности администратору просмотреть все автомобили, имеющиеся в программе. Также этот метод после просмотра предоставляет возможность удалить из программы те автомобили, которые администратор считает нужным, например для проведения технического осмотра или при сильном повреждении автомобиля. Блок-схема алгоритма представлена на рисунке 7.2

void Admin::coutCars() {

while (1)

{

clear();

All\_cars(this->cars);

int i;

i = escmenu(AdmCars);

Sleep(100);

if (i == 123) break;

else if (i >= 0) {

system("cls");

cout << cars[i] << endl;

gotoxy(60, 3);

system("echo Введите любую клавишу чтобы продолжить или DELETE чтобы убрать автомобиль из аренды && pause > nul");

if (GetAsyncKeyState(VK\_DELETE) & 0x8000) {

EraseCar(cars, i);

}

}

}

}

## 7.3 Блок-схема алгоритма метода User::login()

Метод предоставляет возможность авторизации пользователя в программе. Также метод проверяет файлы “users.txt” и “admins.txt” на факт существования такого логина и пароля и возвращает соответствующую цифру: 1 – пользователь, 2 – администратор. Алгоритм этого метода можно посмотреть на рисунке 7.3

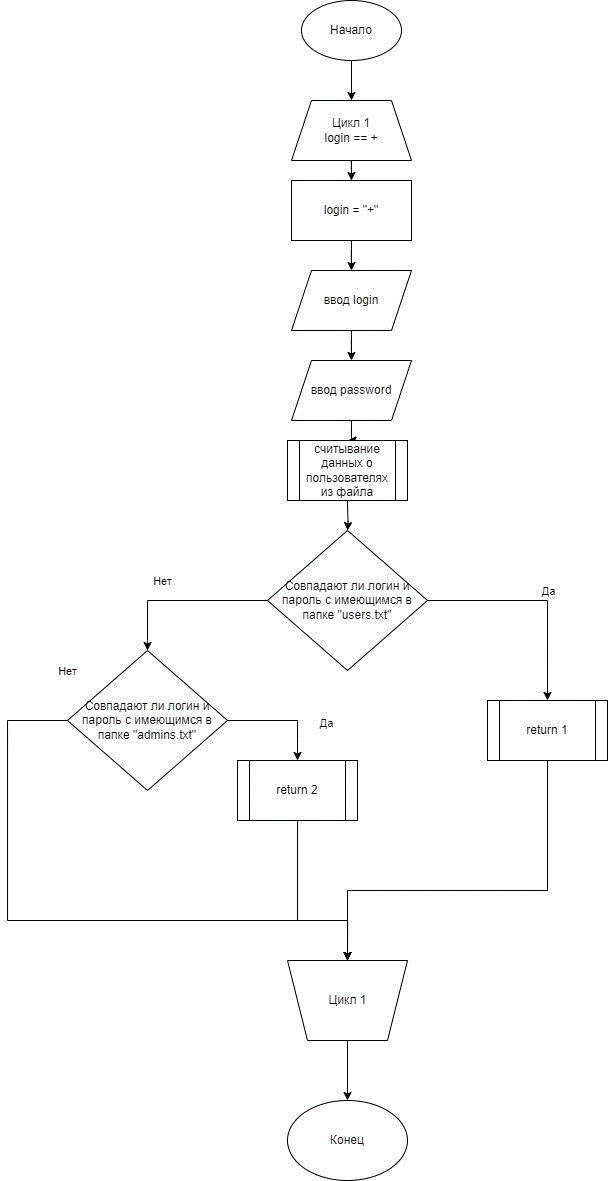


Рисунок 7.3 – Алгоритм метода User::login()

# 8 Описание алгоритма запуска приложения, его использования, результаты работы программы, тестирования обработки ошибок

После запуска программы пользователю предоставляется меню (рисунок 8.1), в котором он может зарегистрироваться или войти в приложение с уже имеющегося аккаунта.

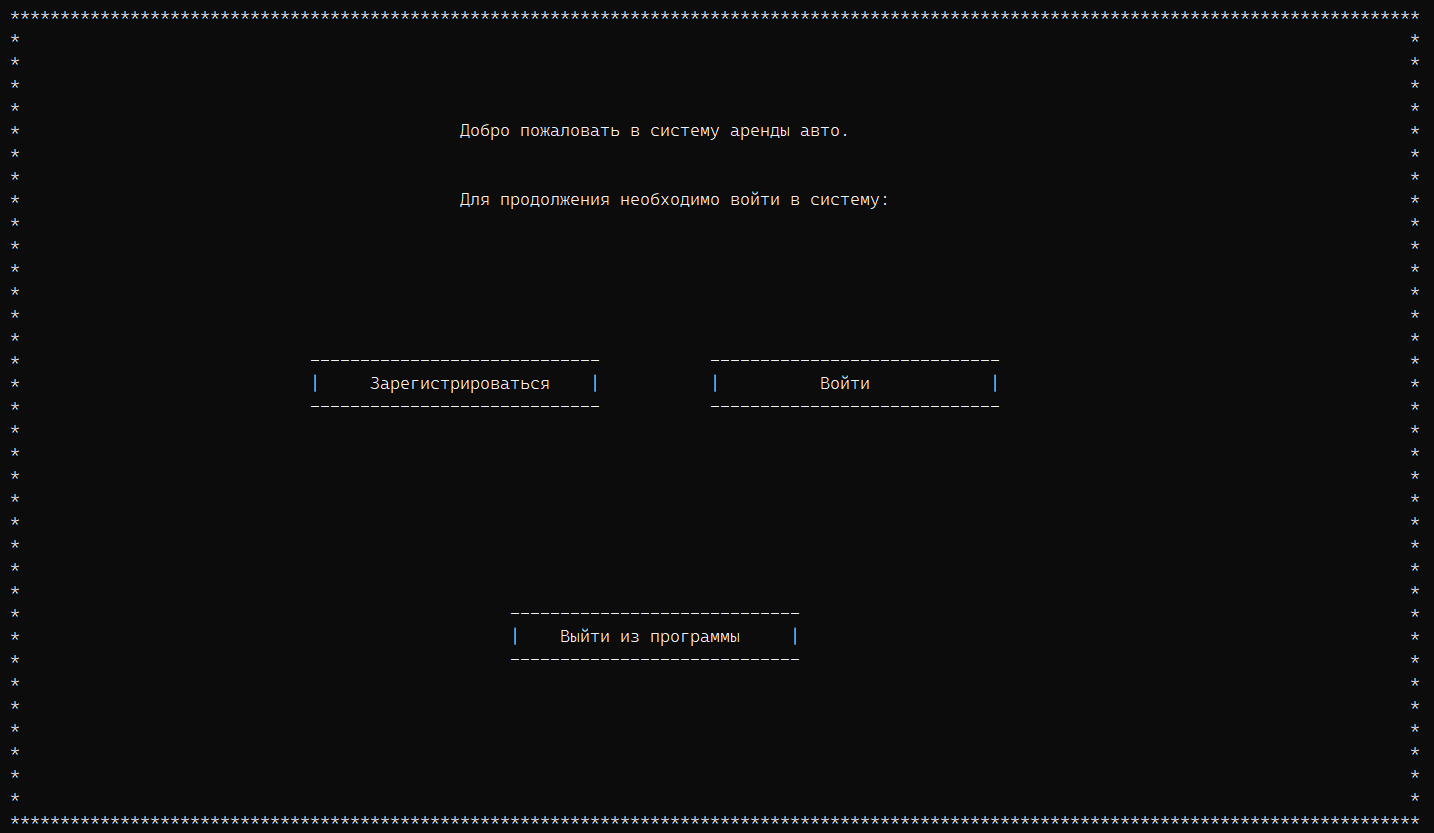


Рисунок 8.1 – Главное меню

После того как пользователь выберет регистрацию, откроется окно для считывания данных пользователя (рисунок 8.2).

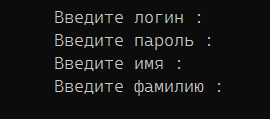


Рисунок 8.2 – Поле авторизации

Ввод логина и пароля разрешается только с использованием английских символов, цифр и знака нижнего подчеркивания. При некорректном вводе логина или пароля программа запросит пользователя попробовать ввести логин еще раз (рисунок 8.3, рисунок 8.4).

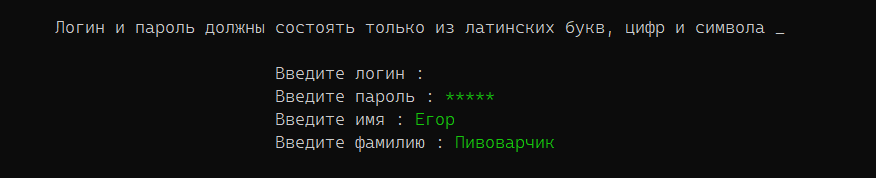


Рисунок 8.3 – Проверка на ввод логина

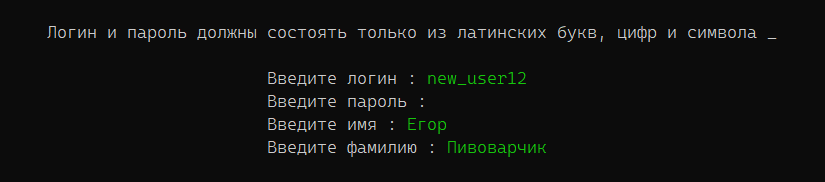


Рисунок 8.4 – Проверка на ввод пароля

После успешной авторизации пользователь вернется в главное меню где сможет авторизоваться в систему при помощи кнопки «Войти». Авторизация обеспечивает сохранность данных и существование такого аккаунта в системе либо в папке с пользователями, либо в папке с администраторами. Также меню предоставляет проверку на ввод данных существующего в системе аккаунта (рисунок 8.5).

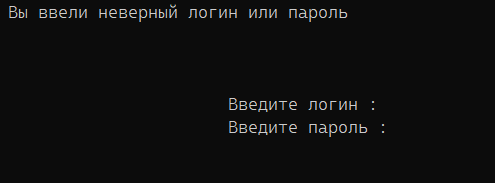


Рисунок 8.5 – Проверка авторизации пользователя приложения

После успешной авторизации пользователь попадет в меню приложения (рисунок 8.6). В меню он может выбрать желаемое действие.

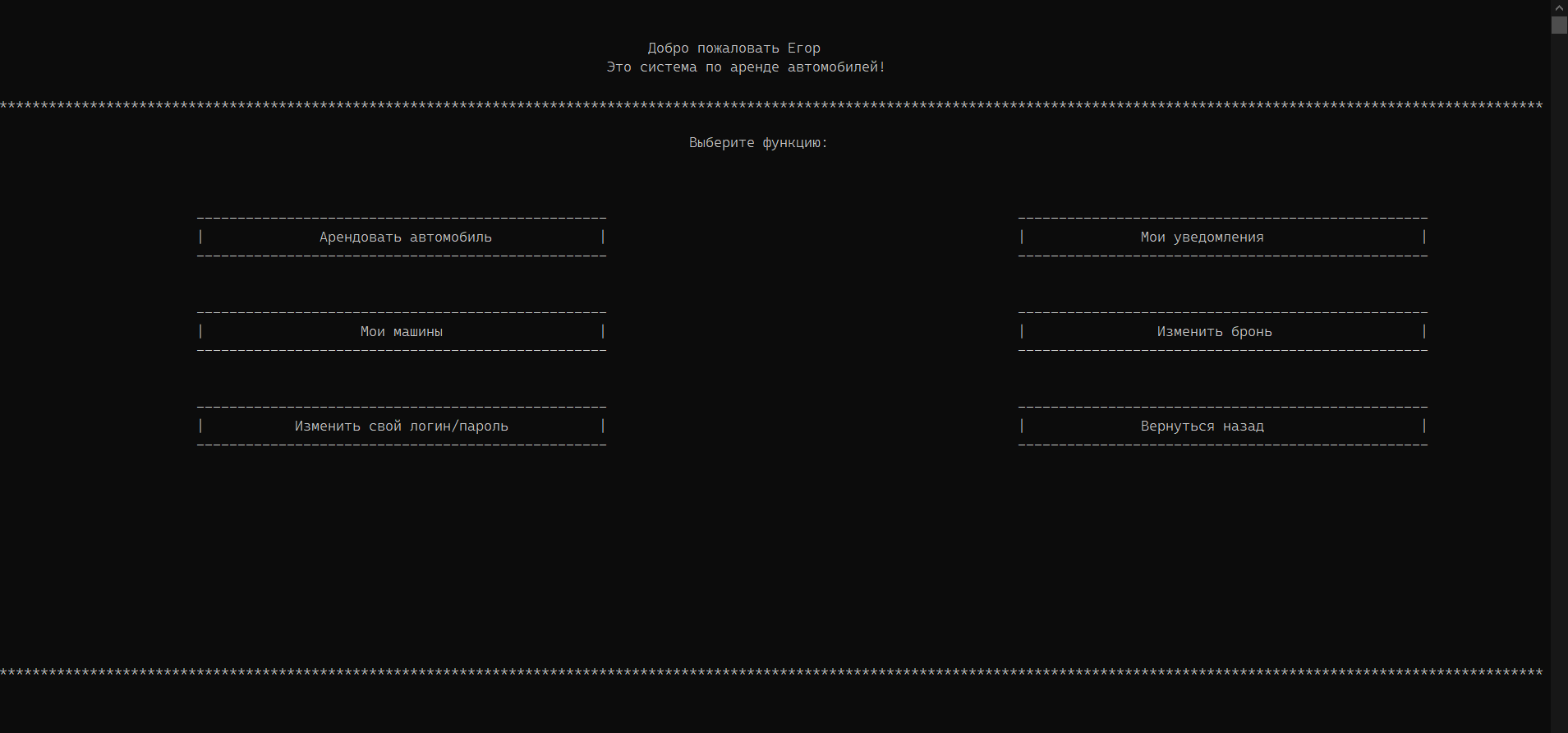


Рисунок 8.6. – Главное меню пользователя

На рисунке 8.7. можно увидеть главное меню аренды нового автомобиля.

В этом меню пользователь может увидеть название автомобиля и его стоимость в сутки, затем нажать на понравившийся вариант и рассмотреть его характеристики (рисунок 8.8).



Рисунок 8.7. – Главное меню аренды автомобиля

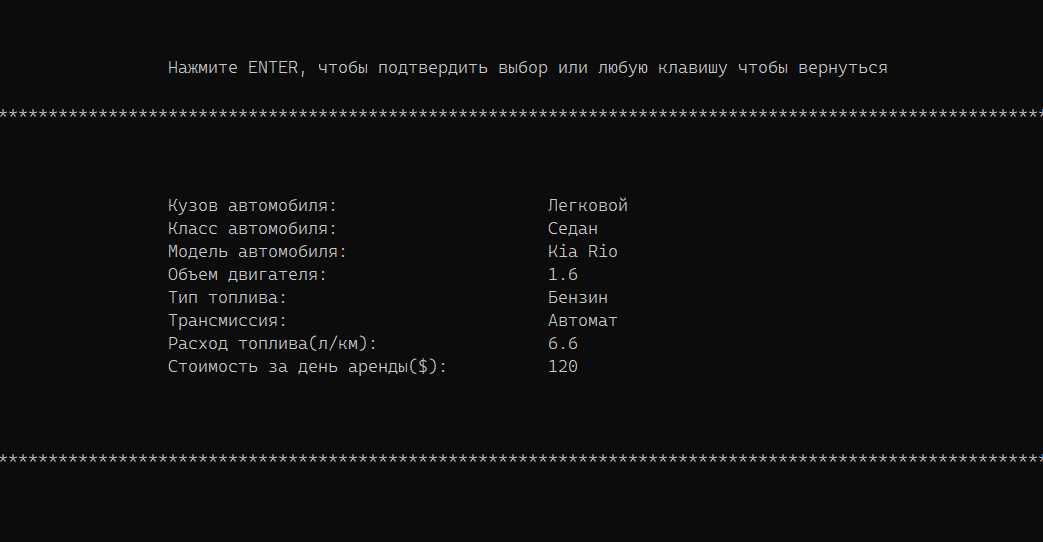


Рисунок 8.8 – Данные о транспортном средстве

Если пользователь решит забронировать выбранный автомобиль, то ему придется ввести время, на которое он хочет забронировать данное транспортное средства и выведется общая сумма, которую отдаст пользователь за всю аренду этого транспортного средства. Также в вводе даты присутствуют проверки на ввод пользователем корректной даты (рисунок 8.9, рисунок 8.10, рисунок 8.11)

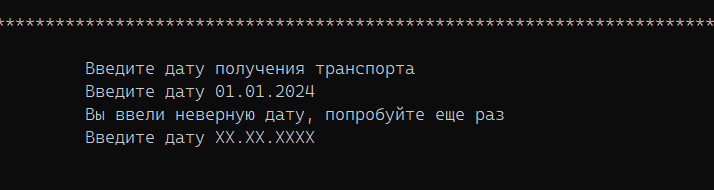


Рисунок 8.9 – Выбор некорректной даты бронирования ТС

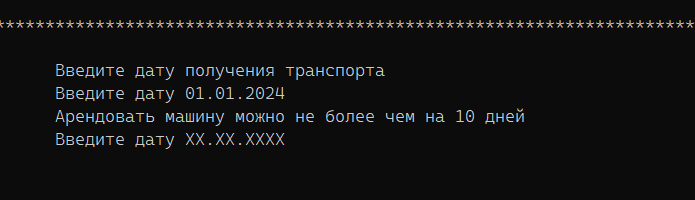


Рисунок 8.10 – Выбор аренды ТС более чем на 10 дней

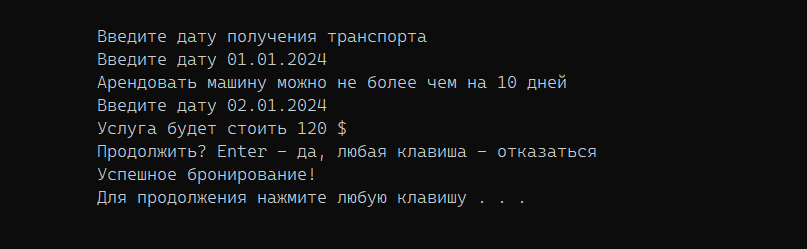


Рисунок 8.11 ­– Реализация бронирования транспортного средства

Забронированные транспортные средства будут выводится в отдельном меню для каждого пользователя с возможностью удаления брони (рисунок 8.12).

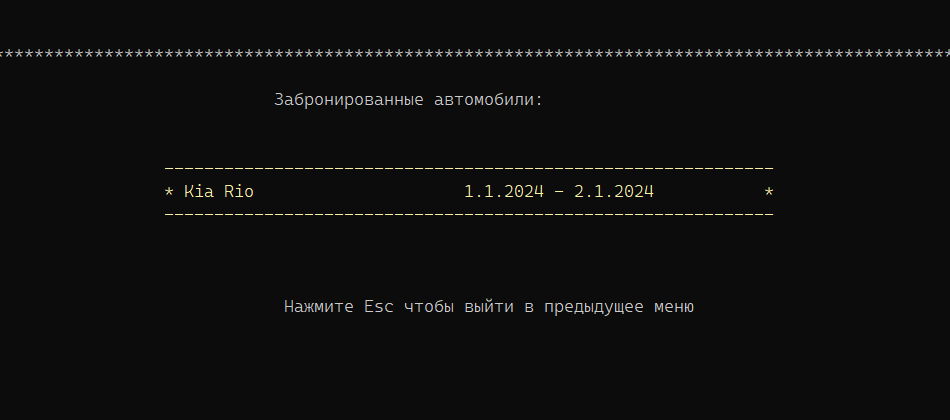


Рисунок 8.12 – Реализация вывода всех забронированных пользователем автомобилей

Меню администратора можно увидеть на рисунке 8.13. В нем реализованы возможности добавления нового автомобиля в программу, удаления записи из программы, добавления нового администратора в программу, удаление конкретной брони пользователя, уведомления пользователя о каком-либо действии (рисунок 8.13, рисунок 8.14.).



Рисунок 8.13 – Реализация меню администратора

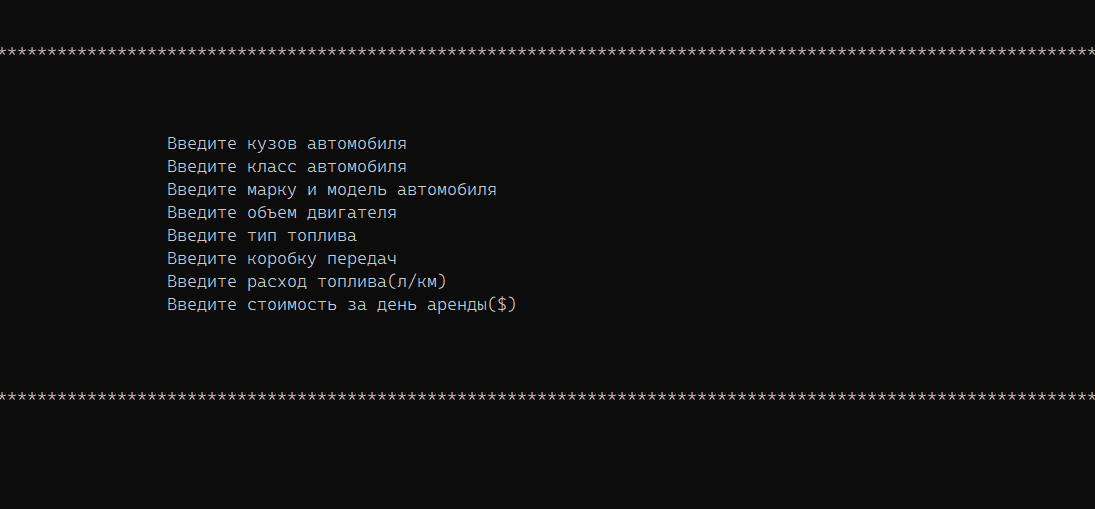


Рисунок 8.14 – Реализация добавления нового автомобиля

# Заключение

В заключении данной работы можно отметить успешное выполнение поставленных задач и достижение цели – разработка автоматизированной системы аренды автомобилей.

В ходе исследования процесса аренды автомобилей были выявлены неидеальные системы бронирования авто. Это вдохновило на разработку функциональной модели основного процесса, которая учитывает потребности и ожидания как арендодателей транспортных средств, так и покупателей.

Алгоритмы классов и пользовательских методов были тщательно спроектированы, обеспечивая эффективное взаимодействие между различными компонентами системы. Это включает в себя удобные способы выбора автомобиля, времени аренды и т.д.

Программная реализация системы была выполнена с учетом современных технологий и стандартов безопасности, обеспечивая стабильную работу и защиту личной информации пользователей.

Разработанная система не только повышает конкурентоспособность отдельных арендодателей, но и улучшает удовлетворенность клиентов, предоставляя им удобные и интуитивно понятные средства приобретения автомобилей. Такие инновационные решения в области автоматизации не только оптимизируют процессы бронирования авто, но и создают уникальный опыт для посетителей.

Обобщая, можно сказать, что выполненная курсовая работа не только соответствует поставленным задачам, но и приносит значительную ценность в области автоматизированных систем аренды автомобилей.

# Список использованных источников

[1] Шилдт Г. Полный справочник по C++ — 4-е изд. — М.: Вильямс,  
2011. — 800 с.

[2] Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл.  
Пер. с англ. – М. : Издательство «Русская редакция», 2010. — 896 стр.

[3] Metanit [Электронный ресурс]. – Режим доступа: <https://metanit.com/>.

[4] Habr [Электронный ресурс]. – Режим доступа: <https://habr.com/>.

[5] Васильев А.Н. Программирование на С++ в примерах и задачах. ООО «Издательство «Эксмо», 2021. – 364 с.

[6] Гилберт Стивен, Макартни Билл. Самоучитель Visual C++ 6 в примерах. - К. : ООО "ТИД ДС", 2003. - 496с.

[7] Холзнер С. Visual C++: Учебный курс. СПб: Питер, 2000. - 576с.

[8] CyberForum [Электронный ресурс]. – Режим доступа: https://www.cyberforum.ru/.

[9] Доусон, М. Изучаем C++ через программирование игр / М. Доусон. - СПб.: Питер, 2017. - 576 c.

[10] Шилдт, Герберт С++ для начинающих. Шаг за шагом / Герберт Шилдт. - М.: ЭКОМ Паблишерз, 2010. - 640 c.

# Приложение А

**(обязательное)**

**Листинг программы с комментариями**

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("chcp 1251");

keybd\_event(VK\_MENU, 0x38, 0, 0);

keybd\_event(VK\_RETURN, 0x1c, 0, 0);

keybd\_event(VK\_RETURN, 0x1c, KEYEVENTF\_KEYUP, 0);

keybd\_event(VK\_MENU, 0x38, KEYEVENTF\_KEYUP, 0);

User new\_user;

Admin new\_admin;

Client new\_client;

int state;

while (1)

{

clear();

switch (menu(MainMenu)) {

case 1:

new\_user.register\_();

gotoxy(80, 12);

cout << "Вы были успешно зарегистрированы" << endl;

gotoxy(80, 13);

system("pause");

clear();

coutMenu();

break;

case 2:

state = new\_user.login\_();

gotoxy(80, 12);

cout << "Успешная авторизация" << endl;

gotoxy(80, 13);

system("pause");

clear();

switch (state) {

case 1:

new\_client.setdata(new\_user);

while (state != 6) {

coutUserFunct();

gotoxy(80, 2);

new\_client.getreservations();

cout << "Добро пожаловать " << new\_client.getname();

gotoxy(75, 3);

cout << "Это система по аренде автомобилей!" << endl;

state = menu(UserMenu);

switch (state) {

case 1:new\_client.choosecar();

clear();

break;

case 2:new\_client.cout\_reservations();

clear();

break;

case 3:new\_client.changeAccountinfo();

clear();

break;

case 4:new\_client.notifications();

clear();

break;

case 5:new\_client.change\_reservation();

clear();

break;

case 6:

break;

}

}

break;

case 2:

new\_admin.setdata(new\_user);

while (state != 6) {

coutAdmFunct();

gotoxy(80, 2);

cout << "Добро пожаловать " << new\_admin.getname();

gotoxy(80, 3);

cout << "Это меню администратора!" << endl;

state = menu(AdmMenu);

switch (state) {

case 1:new\_admin.addCar();

clear();

break;

case 2:new\_admin.coutCars();

clear();

break;

case 3:new\_admin.changeAccountinfo();

clear();

break;

case 4:new\_admin.cout\_reservations();

clear();

break;

case 5:new\_admin.addAdmin();

break;

case 6:break;

}

}

}

break;

case 3:endProgram();

return 0;

break;

}

}

return 0;

}

void Client::choosecar() {

int result=1;

while (1)

{

clear();

All\_cars\_user(this->cars);

int i;

Reservation new\_reservation;

Date new\_date1,new\_date2;

i = escmenu(UserCars);

Sleep(100);

if (i == 123) break;

else if (i >= 0) {

system("cls");

cout << cars[i] << endl;

gotoxy(60, 3);

system("echo Нажмите ENTER, чтобы подтвердить выбор или любую клавишу чтобы вернуться && pause > nul");

if (GetAsyncKeyState(VK\_RETURN) & 0x8000) {

gotoxy(60, 22);

cout << "Введите дату получения транспорта" << endl;

gotoxy(130, 22);

int y = 23;

result = 1;

for (int j{}; j < all\_reservations.size(); j++) {

if (cars[i] == all\_reservations[j].getcar()) {

if (y==23) {

cout << "Забронированные даты:" << endl;

y--;

}

gotoxy(130, y+=2);

cout << all\_reservations[j].returnAcquisitiondate() << " - " << all\_reservations[j].returnReturnDate() << endl;

}

}

while (result == 1) {

gotoxy(60, 23);

cout << "Введите дату XX.XX.XXXX ";

gotoxy(73, 23);

result = new\_date1.input();

if (result == 1) {

gotoxy(60, 22);

cout << "Вы ввели неверную дату, попробуйте еще раз" << endl;

continue;

}

}

result = 1;

gotoxy(60, 24);

cout << "Введите данные возврата транспорта" << endl;

while (result == 1) {

gotoxy(60, 25);

cout << "Введите дату XX.XX.XXXX ";

gotoxy(73, 25);

result = new\_date2.input();

if (result == 1 || new\_date2 == new\_date1) {

gotoxy(60, 24);

cout << "Вы ввели неверную дату, попробуйте еще раз" << endl;

result = 1;

continue;

}

if (new\_date2 < new\_date1) {

gotoxy(60, 24);

cout << "Вы умеете возвращаться назад во времени? Попробуйте еще раз" << endl;

result = 1;

continue;

}

if (new\_date2 - new\_date1 > 10) {

gotoxy(60, 24);

cout << "Арендовать машину можно не более чем на 10 дней" << endl;

result = 1;

continue;

}

}

gotoxy(60, 26);

cout << "Услуга будет стоить " << User::cars[i].getcost() \* (new\_date2 - new\_date1) << " $";

gotoxy(60, 27);

system("echo Продолжить? Enter - да, любая клавиша - отказаться && pause > nul");

if (GetAsyncKeyState(VK\_RETURN) & 0x8000) {

new\_reservation.setdata(this->getlogin(),cars[i],new\_date1,new\_date2);

bool permission = true;

for (int j = 0; j < all\_reservations.size(); j++) {

if (cars[i].getname() == all\_reservations[j].getcarname()) {

if (new\_date1 > all\_reservations[j].returnReturnDate() || new\_date2 < all\_reservations[j].returnAcquisitiondate()) {

permission = true;

}

else permission = false;

}

}

if (permission == true) {

all\_reservations.push\_back(new\_reservation);

cars[i].book();

gotoxy(60, 28);

cout << "Успешное бронирование!";

gotoxy(60, 29);

system("pause");

}

else

{

gotoxy(60, 28);

cout << "Такая дата уже занята, попробуйте другое время" << endl;

gotoxy(60, 29);

system("pause");

}

}

}

}

}

}

void Admin::addCar() {

system("cls");

Car temp;

vsl::gotoxy(0, 5);

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

gotoxy(60, 9);

cout << "Введите кузов автомобиля" << endl;

gotoxy(60, 10);

cout << "Введите класс автомобиля" << endl;

gotoxy(60, 11);

cout << "Введите марку и модель автомобиля" << endl;

gotoxy(60, 12);

cout << "Введите объем двигателя" << endl;

gotoxy(60, 13);

cout << "Введите тип топлива" << endl;

gotoxy(60, 14);

cout << "Введите коробку передач" << endl;

gotoxy(60, 15);

cout << "Введите расход топлива(л/км)" << endl;

gotoxy(60, 16);

cout << "Введите стоимость за день аренды($)" << endl;

vsl::gotoxy(0, 20);

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

string value;

float data;

gotoxy(86, 9);

getline(cin, value);

temp.setbody(value);

gotoxy(86, 10);

cin.clear();

getline(cin, value);

temp.settype(value);

gotoxy(95, 11);

cin.clear();

getline(cin, value);

temp.setname(value);

gotoxy(86, 12);

cin >> data;

cin.clear();

cin.ignore(1024, '\n');

temp.setcapacity(data);

gotoxy(86, 13);

getline(cin, value);

temp.setfuel\_type(value);

gotoxy(86, 14);

getline(cin, value);

temp.settransmission(value);

gotoxy(91, 15);

cin >> data;

temp.setfuel\_use(data);

gotoxy(98, 16);

cin.clear();

cin.ignore();

cin >> data;

cin.ignore();

temp.setcost(data);

cars.push\_back(temp);

ofstream fout;

fout.open("C:\\Users\\USER\\Desktop\\baby\_work\\car\_rental\_system\\info\\cars.txt", ios::app);

temp.saveToFile(fout);

fout.close();

}

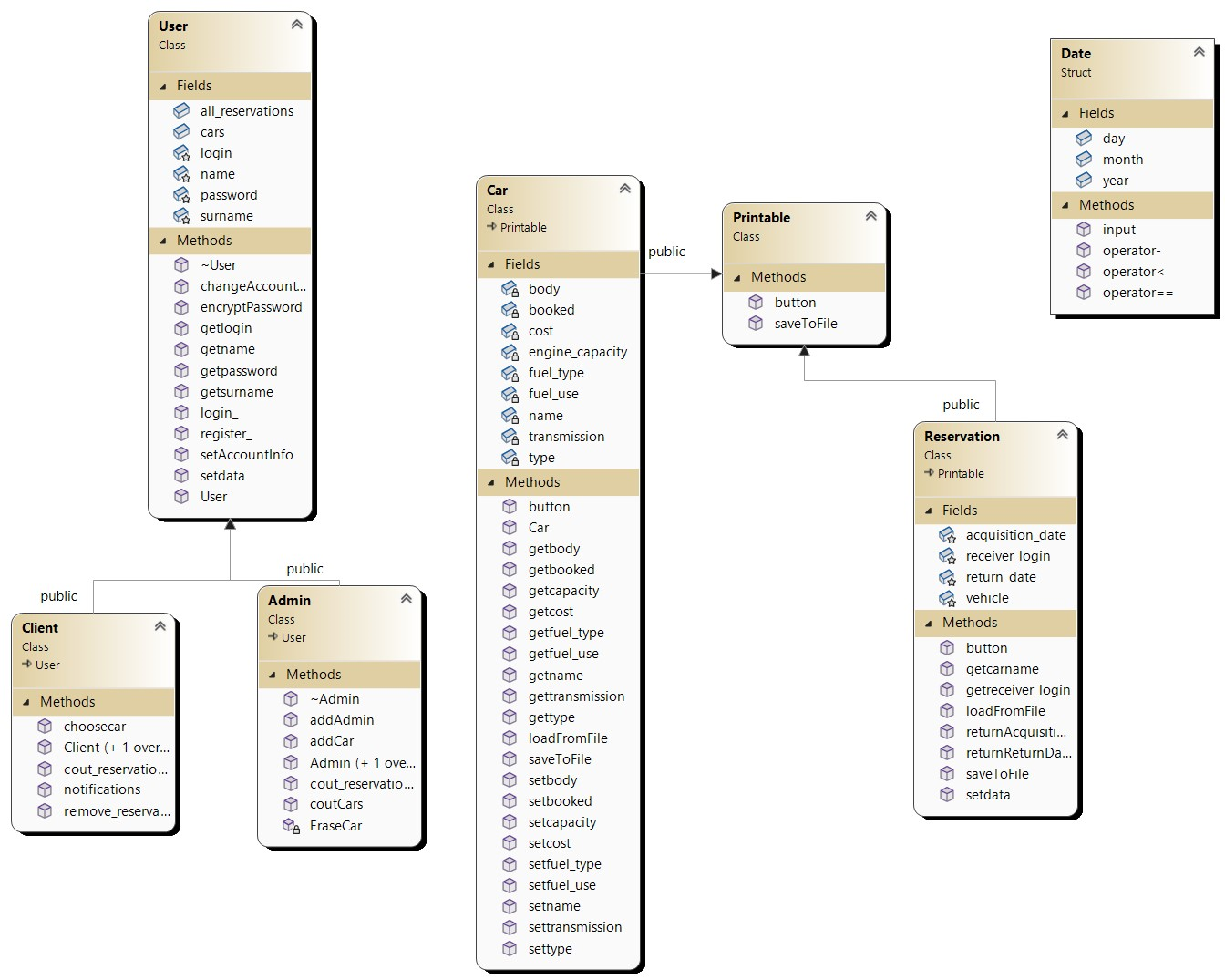
# Приложение Б

**Проверка на заимствование в системе «Антиплагиат»**

# Приложение В

(обязательное)

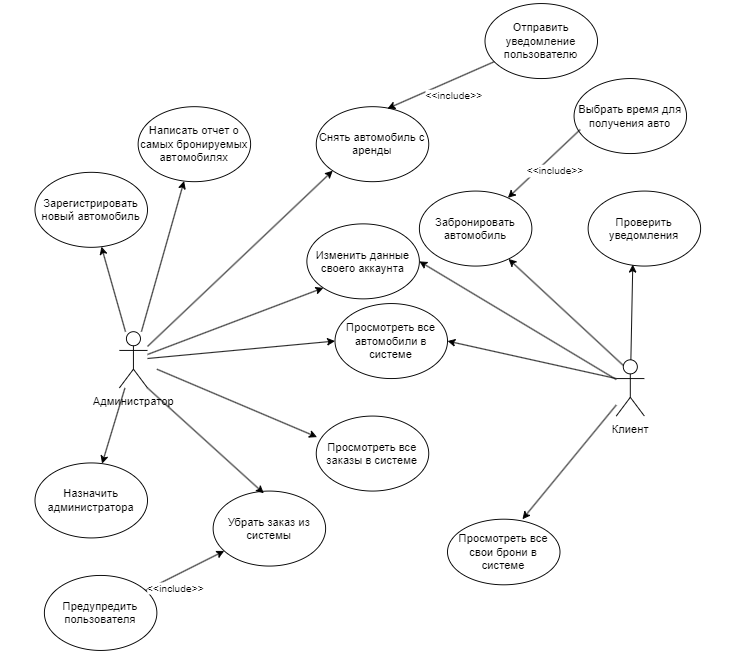
Диаграмма классов



# Приложение Г

(обязательное)

Диаграмма вариантов использования



# Приложение Д

(обязательное)

Блок-схемы алгоритмов

Изображение выглядит как текст, диаграмма, зарисовка, План

Автоматически созданное описание

Рисунок Д.1 – Блок-схема алгоритма функции main

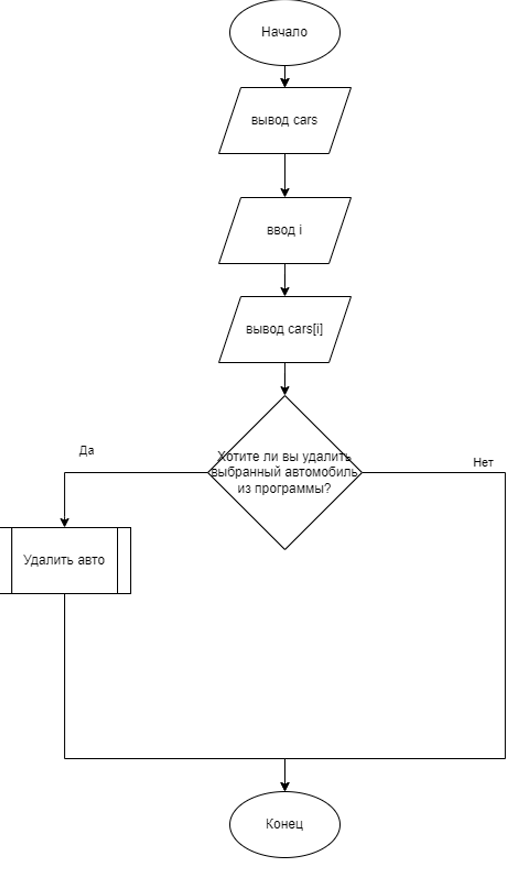


Рисунок Д.2 – Алгоритм метода Admin::coutCars()

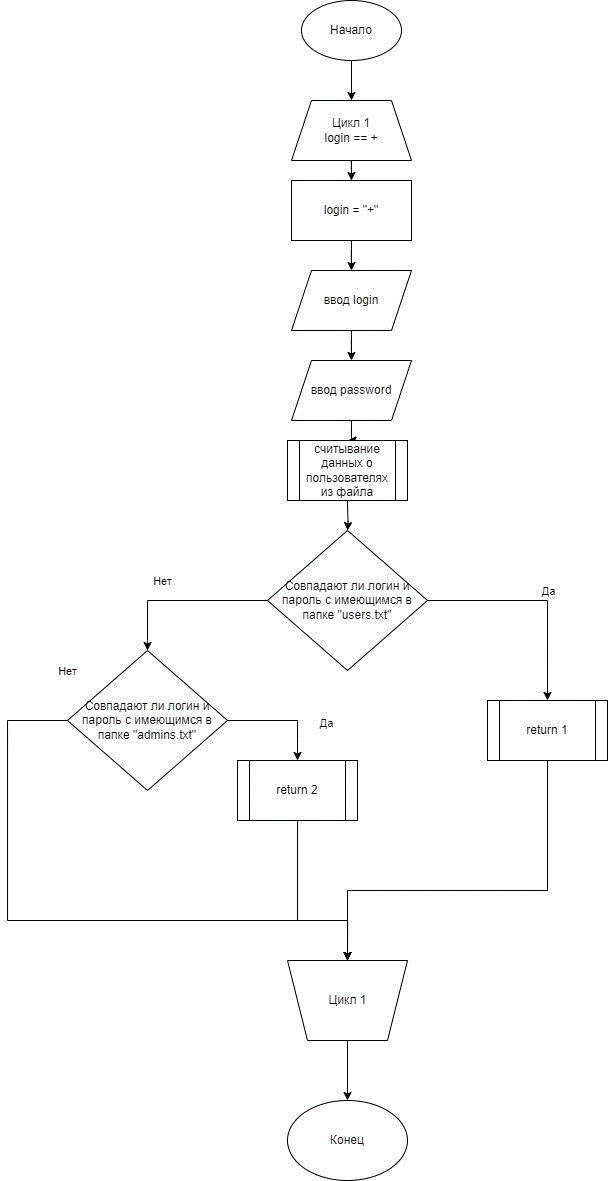


Рисунок Д.3 – Алгоритм метода User::login()