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 учета деятельности косметического салона»**

БГУИР КР 1-40 05 01-02 024 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 272303 ТАРАБЕШ Аким Константинович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**СОДЕРЖАНИЕ**

[1 Анализ автоматизированной системы учета деятельности косметического салона 7](#_Toc153867794)

[1.1 Описание автоматизации учета деятельности косметического салона 7](#_Toc153867795)

[1.2 Обзор программ-аналогов по данной предметной области 7](#_Toc153867796)

[1.2.1 Салон красоты “Мильфей” 8](#_Toc153867800)

[1.2.2 Салон красоты “Манго 9](#_Toc153867801)

[1.3 Методы и алгоритмы решения поставленной задачи 9](#_Toc153867802)

[2 Функциональное моделирование на основе стандарта idef0 11](#_Toc153867803)

[3 структура используемых данных 15](#_Toc153867804)

[4 описание созданных программных конструкций 21](#_Toc153867805)

[5 разработка и описание диаграммы классов приложения 26](#_Toc153867806)

[6 разработка и описание диаграммы вариантов использования приложения 27](#_Toc153867807)

[7 блок-схема алгоритма всей программы и двух основных методов 28](#_Toc153867808)

[7.1 Блок-схема алгоритма функции SignIn 28](#_Toc153867809)

[7.2 Блок-схема алгоритма функции main 29](#_Toc153867810)

[7.3 Блок-схема алгоритма всей программы 29](#_Toc153867811)

[8 описание алгоритма запуска приложения, его использование, результаты работы программы, тестирование обработки ошибок 31](#_Toc153867812)

[8.1 Тестирование ошибок 31](#_Toc153867813)

[8.2 Алгоритм запуска приложения 31](#_Toc153867814)

[Приложение а 40](#_Toc153867815)

[Приложение б 65](#_Toc153867816)

[Приложение в 66](#_Toc153867817)

[Приложение г 67](#_Toc153867818)

[Приложение д 69](#_Toc153867819)

# **ВВЕДЕНИЕ**

Современное развитие индустрии красоты и ухода за собой неразрывно связано с прогрессивными технологиями, направленные на повышение качества обслуживания клиентов и автоматизацию процессов и задач.

Использования информационных технологий в сфере услуг обеспечивает функционирование косметических салонов на высшем уровне в следствии того, будет снижена нагрузка на руководящие должности. Потенциальный клиент сможет не стоять в очередях, а воспользовавшись системой эффективно осуществить поиск необходимой информации и оформления услуг.

Целью данного курсового проекта является разработка автоматизированной системы учета деятельности косметического салона.

Целесообразность работы обусловлена следующим фактором. В косметических салонах любой работник перегружен в связи с большим количеством клиентов, оформления отчетов и консультации посетителей. В связи с этим необходимым является использование в салоне автоматизированной системы, которая предоставляет необходимую информацию, обладает возможностью записи на любую интересующую услугу и эффективным учетом деятельности отдельным звеньев салона.

Салонный бизнес является достаточно новым видом деятельности, который в настоящее время активно развивается.

Таким образом, можно сделать вывод, что тема курсового проекта является достаточно актуальной в текущее время.

Задачи работы:

* ознакомиться с предметной областью;
* проанализировать процессы;
* разработать пояснительную записку и код.

Объектом исследования данной курсовой работы является косметический салон и его процессы, которые подлежат автоматизации.

# **Анализ автоматизированной системы учета деятельности косметического салона**

## **Описание автоматизации учета деятельности косметического салона**

Значения салона красоты в современном мире невозможно отнести к простому предоставлению косметических процедур. В современном обществе салоны красоты играют важную роль, предоставляя клиентам не только услуги по уходу за внешностью, но и различные процедуры для улучшения здоровья.

Услуги предлагаемые салонами можно разделить на 2 группы:

* меджевые услуги (парикмахерские, маникюр, педикюр)
* косметологические услуги (уход за кожей и здоровьем)

Данные услуги всегда принадлежат к числу более востребованных.

## **Обзор программ-аналогов по данной предметной области**

Для реализации актуального и конкурентоспособного продукта необходимо также исследовать приложения конкурентов.

В настоящее время существует множество специализированных программ автоматизации деятельности салона красоты, которые предназначены непосредственно для клиента и работников. Они обладают рядом достоинств:

* широкий спектр услуг;
* консультация по интересующей услуге;
* возможность просматривать отзывы о салоне;
* обратная связь.

Несмотря на ряд достоинств и большое количество возможностей, в большинство программ также имеет ряд недостатков:

* отсутствие конкретного прайс-листа;
* при записи на услугу, нет возможности записи к определенному специалисту.



## **Салон красоты “Мильфей”**

Салон красоты “Мильфей” предоставляет большой спектр косметических услуг (рисунок 1.2.1.1).



Рисунок 1.2.1.1 – Салон красоты “Мильфей”

Реализация заказа косметической услуги в салоне красоты “Мильфей” представлена следующим образом. Первоначально необходимо выбрать специалиста, дату и время и услугу (рисунок 1.2.1.2).

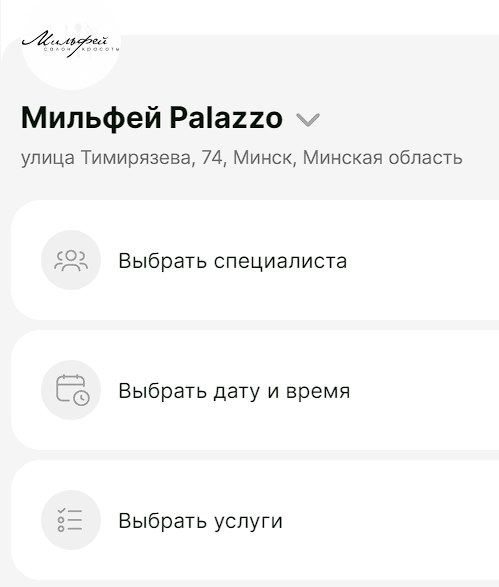


Рисунок 1.2.1.2 – Заказ услуги в салоне красоты “Мильфей”

Данный продукт дает возможность при записи на услугу не только выбор самой услуги, а также выбор интересующего специалиста и даты оказания услуги, что делает данный продукт весьма удобным в использовании.

## **Салон красоты “Манго”**

В салоне красоты “Манго” заказ реализован следующим образом. Для начала необходимо оставить заявку на услуги, после чего приходит ответ (рисунок 1.2.2.1).

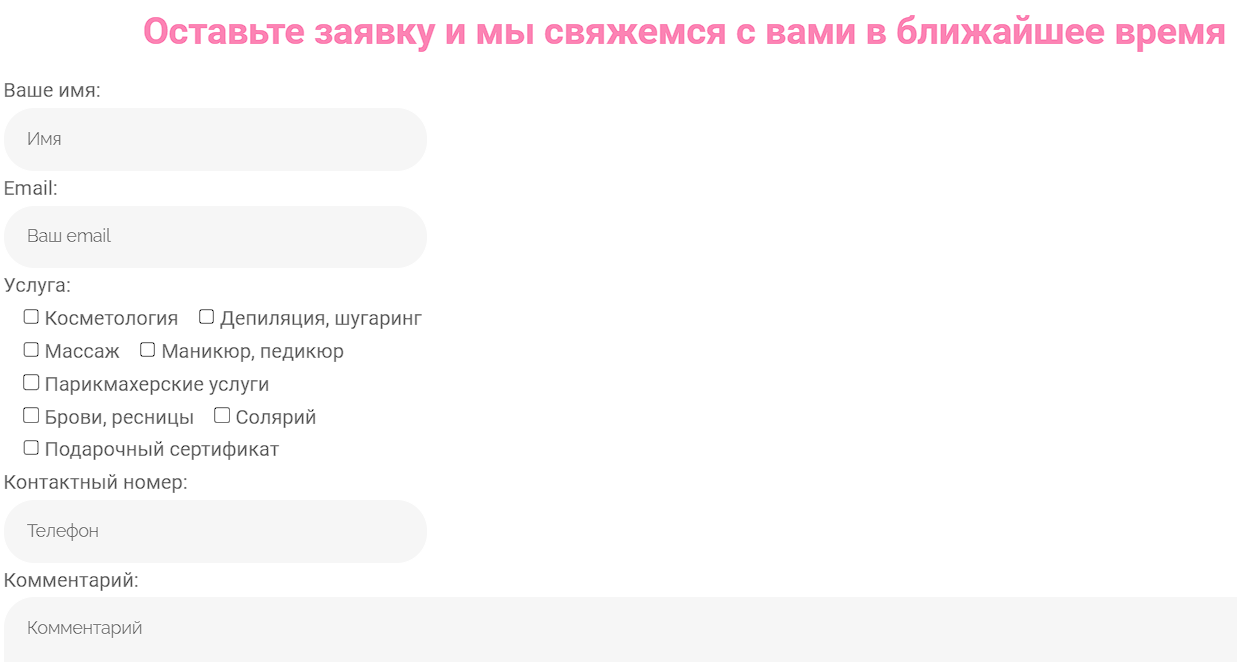


Рисунок 1.2.2.1 – Заказ услуги в салоне красоты “Манго”

Исходя из анализа данного продукта, можно сказать что он не удобен в использовании. Необходимо оставлять заявку на оказании услуги и ожидать обратной связи, что не всегда является желаемым для пользователя.

## **Методы и алгоритмы решения поставленной задачи**

Для автоматизации системы учета деятельности косметического салона можно использовать следующие методы и алгоритмы:

* создание удобного и интуитивно понятного интерфейса для заказа косметических услуг, который бы обеспечивал простоту использования для различных категорий пользователей;
* разработка алгоритма для расчета стоимости заказа и общей стоимости всех заказов;
* создание и управление файлами с данными, которые хранят информацию о заказах, услугах и данных пользователя;
* использование быстрых алгоритмов сортировок, поиска, фильтрации для уменьшения времени отклика программы;
* разработка архитектуры системы, определение сущностей системы и их взаимодействия между собой.

Таким образом были рассмотрены различные алгоритмы и методы для решения поставленной задачи.

# **Функциональное моделирование на основе стандарта IDEF0**

IDEF0 – нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции. К её особенностям можно отнести:

* использование контекстной диаграммы;
* поддерживает декомпозиции;
* доминирование;
* выделение четырех типов стрелок.

На рисунке 2.1 представлена функциональная модель IDEF0 процесса “Предоставить косметическую услугу”, которая отражает общий принцип работы салона красоты.

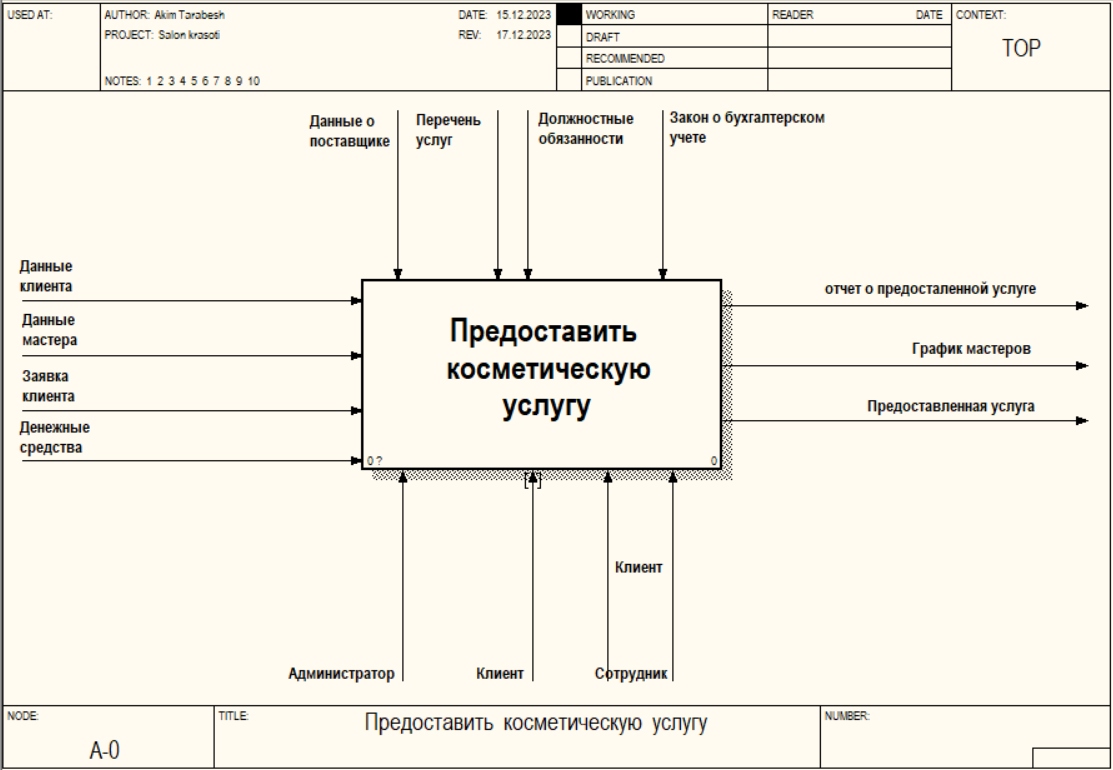


Рисунок 2.1 – Функциональная модель IDEF0 процесса “Предоставить косметическую услугу”

Для предоставления услуг требуется заявка от клиента на предоставление определенной косметической процедуры и его данные, данные о мастере, а также денежные средства. Ресурсами данного процесса являются сотрудник, администратор и клиент. В управлении выступают данные о поставщике, перечень услуг, должностные обязанности, закон о бухгалтерском учете. В результате выполнения процесса получим график мастеров, отчет о предоставленной услуге, предоставленная услуга.

Для более подробного анализа, рассмотрим декомпозицию функционального блока “предоставить косметическую услугу” (рисунок 2.2)

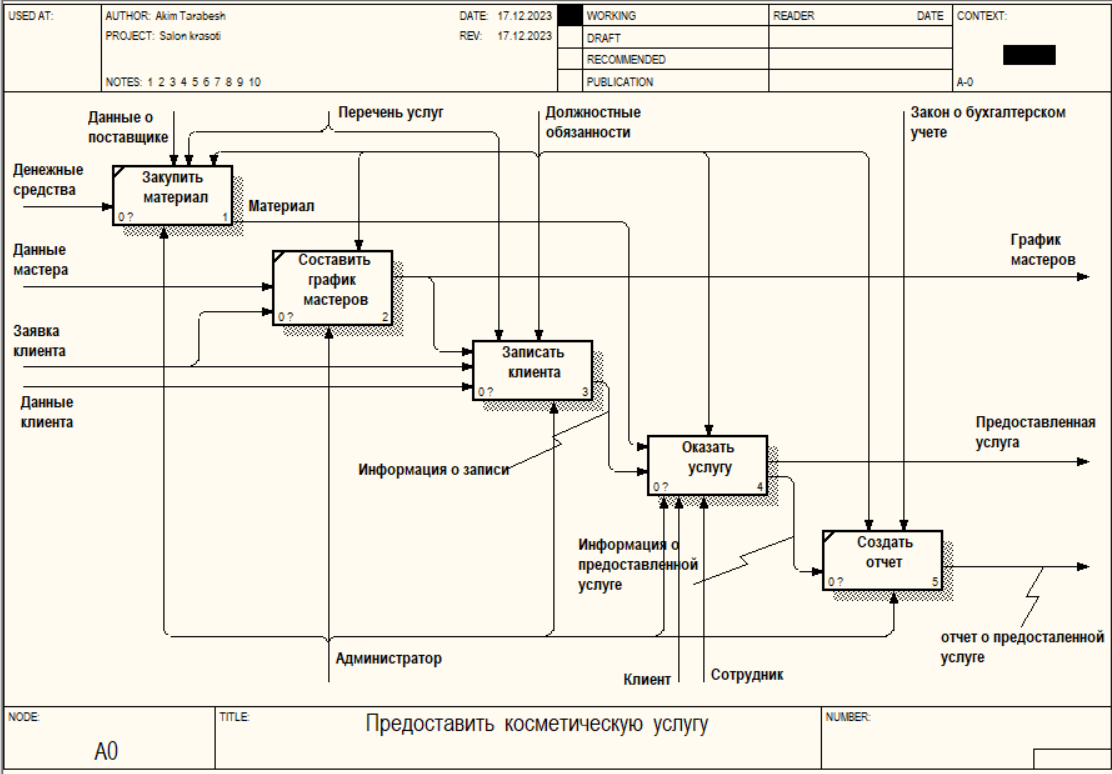


Рисунок 2.2 – Декомпозиция А-0 процесса “Предоставить косметическую услугу”

Модель отражает внутреннюю структуру процесса “Предоставить косметическую услугу”, исходя из которой можно увидеть, что она состоит из следующих подпроцессов: закупить материал, составить график мастеров, записать клиента, оказать услугу и создать отчет. Рассмотрим декомпозицию процесса Оказать услугу (рисунок 2.3).

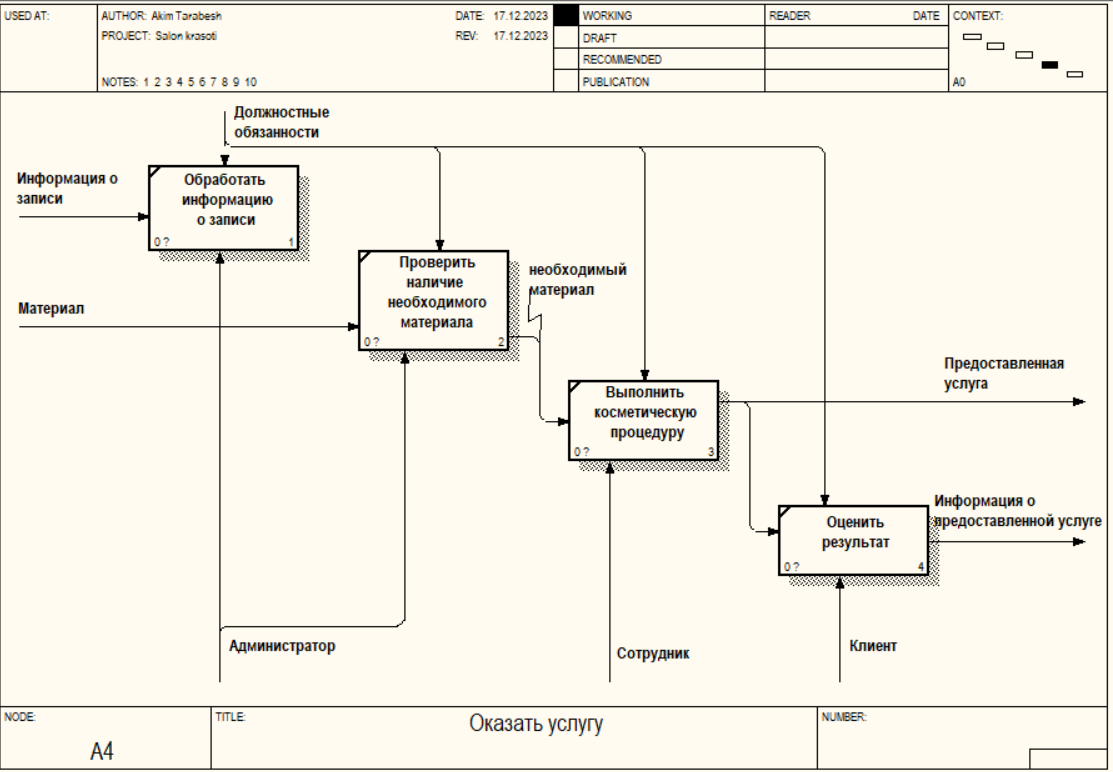


Рисунок 2.3 – Декомпозиция А-1 процесса “Оказать услугу”

В данной модели процесс “Оказать услугу” разбивается на под процессы: обработать информацию о записи, проверить наличие необходимого материала, выполнить косметическую процедуру, оценить результат.

Также рассмотрим декомпозицию процесса “Записать клиента”. (рисунок 2.4).

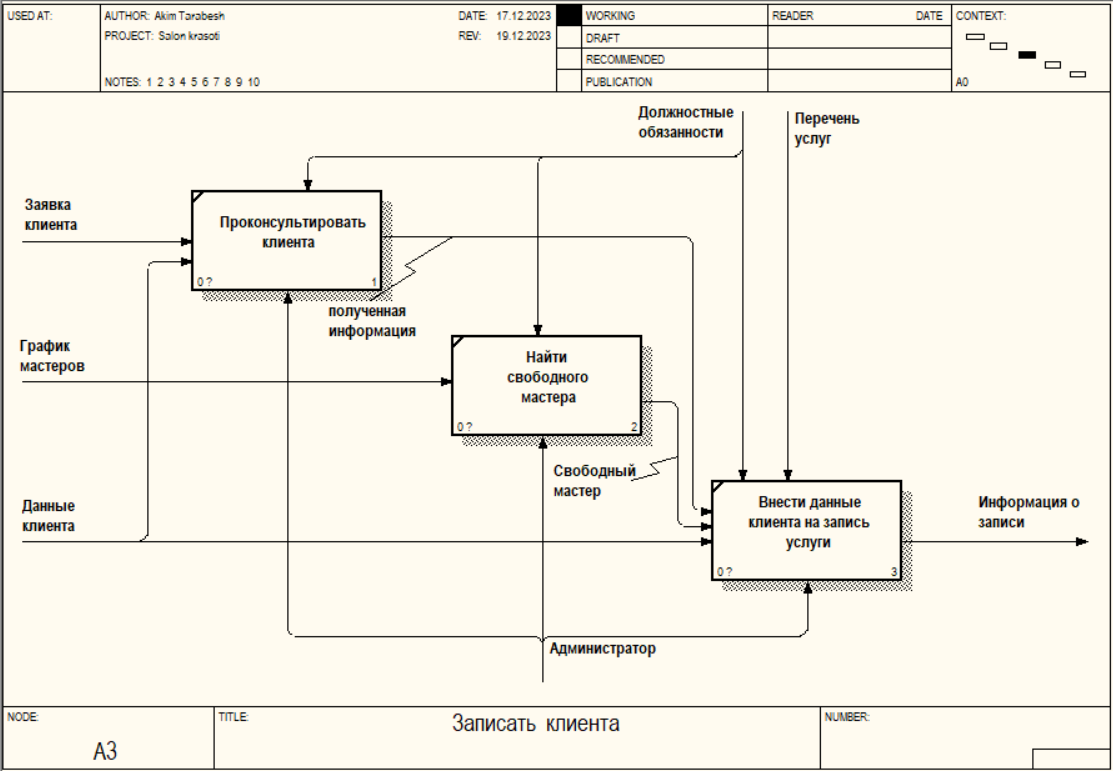


Рисунок 2.3 – Декомпозиция А-1 процесса “Записать клиента”

В данной модели процесс “ Записать клиента ” разбивается на под процессы: проконсультировать клиента, найти свободного мастера, внести данные клиента на запись услуги.

# **структура используемых данных**

Для организации данных были выбраны классы. Таблице приведенные ниже демонстрируют реализованные классы.

Таблица 3.1 – Переменные класса Account

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| login | string | Хранит логин пользователя системы |
| password | string | Хранит пароль пользователя системы |
| role | string | Хранит роль пользователя системы |

Таблица 3.2 – Переменные класса Person

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| name | string | Хранит имя пользователя системы |
| surname | string | Хранит фамилию пользователя системы |
| fname | string | Хранит отчество пользователя системы |

Таблица 3.3 – Переменные класса Calendar

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| currentDate | time\_t | Хранит значение текущей даты |
| status | vector<vector<bool>> | Хранит состояние(true/false) для определенной даты |
| dates | Vector<tm> | Хранит дату на которую может быть заказана услуга |

Таблица 3.4 – Переменные класса Employee

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| services | vector<Service> | Хранит услуги работника, которые необходимо выполнить |
| nameCustomer | vector<string> | Хранит имя клиента |
| surnameCustomer | vector<string> | Хранит фамилию клиента |
| patronymicCustomer | vector<string> | Хранит Отчество клиента |
| date | Calendar | Хранит график работы сотрудника |

Таблица 3.4 – Переменные класса Service

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| name\_service | string | Хранит значение названия услуги |

Таблица 3.4 – Переменные класса Service

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| price | float | Хранит цену за услугу |
| duration | int | Хранит длительность услуги в минутах |
| description | string | Хранит описание услуги(краткая информация) |

Таблица 3.5 – Переменные класса Order

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| total\_price | float | Хранит цену заказа |
| services | vector<Service> | Хранит набор услуг одного заказа |
| employyes | vector<Employee> | Хранит набор данных о сотруднике |
| timeService | vector<tm> | Хранит набор о дате и времени проведения определенной услуги |
| count\_line | int | Хранит количество услуг в данном заказе |

Таблица 3.6 – Переменные класса User

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| orders | vector<Order> | Хранит заказы пользователя |

Таблица 3.7 – Переменные класса Admin

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип данных | Назначение переменной |
| services | vector<Service> | Хранит услуги, предоставляемые салоном красоты |

Для разработки программы использовались методы. Ниже описаны классы и методы, относящиеся к соответствующему классу.

Методы класса Account:

* string getLogin();
* string getPassword();
* string getRole();
* void setRole(string role);
* void setLogin(string login);
* void setPassword(string password);
* void appUserDisplay();
* bool addLogin(vector<string> logins);
* virtual void addPassword();
* friend ifstream& operator>>(ifstream& file, Account& user).

Методы класса Calendar:

* void createCalendar();
* string getTimeString(int timeSlot) const;
* string getFormattedDate(const tm& date) const;
* tm printTime(int day, tm &time) const;
* tm printCalendar().

Методы класса Person:

* string getSurname();
* virtual void setPersonalData(int x, int y);
* virtual void getPersonalData(int x, int y).

Методы класса Employee:

* void addPassword();
* void setPersonalData(int x, int y);
* void getPersonalData(int x, int y);
* void displayEmployee();
* friend ostream& operator<<(ostream& cout, const Employee& obj).

Методы класса Order:

* void add\_Order(Service service, Employee employee, tm timeService);
* void delete\_orderInfo();
* void display\_order(int j);
* float getTotalPrice();
* friend ostream& operator<<(ostream& cout, const Order& obj).

Методы класса User:

* void setPersonalData(int x, int y);
* void getPersonalData(int x, int y);
* void add\_Service();
* void addPassword();
* void getService();
* void getEmployee();
* template<typename T>
* T addOrder(vector<T> obj);
* void addOrder(Service service, Employee employee, tm time);
* void setOrder();
* void displayOrder().

Методы класса Admin:

* void add\_Service();
* void getService();
* void addPassword().

Методы класса Service:

* Service addService();
* float getPrice();
* void displayService();
* friend ostream& operator<<(ostream& cout, const Service& obj);
* friend ifstream& operator>>(ifstream& file, Service& service).

Таким образ были рассмотрены структуры классов Account, Person, User, Admin, Employee, Service, Order, Calendar. С помощью данной структуры была разработана система автоматизации учета деятельности косметического салона. При разработке были задействованы основные принципы объектно-ориентированного программирования такие как: наследование, инкапсуляция и полиморфизм, что позволило создать гибкую и эффективную систему для автоматизации процессов учета деятельности косметического салона.

# **описание созданных программных конструкций**

Использованные конструкции были разработаны с учетом требований курсового проекта.

Реализация базовых принципов объектно-ориентированного программирования (инкапсуляции, наследования, полиморфизма) представлена на рисунках 4.1, 4.2, 4.3

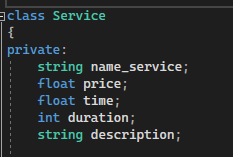


Рисунок 4.1 – Использование private полей как пример инкапсуляции



Рисунок 4.2 – Использование наследования



Рисунок 4.3 – Перегрузка функций как пример полиморфизма

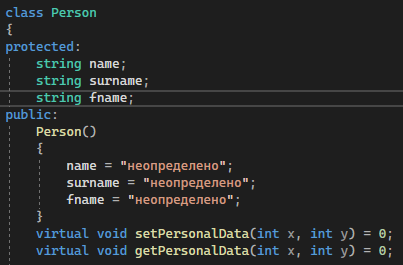
Использование абстрактных классов показано на рисунке 4.4

Рисунок 4.4 – Использование абстрактного класса

Использование передачи параметров по ссылке и по значению представлено на рисунках 4.5 и 4.6



Рисунок 4.5 – Передача параметров по ссылке



Рисунок 4.6 – Передача параметров по значению

Использование встроенных, пользовательских, дружественных и виртуальных функций представлено на рисунках 4.7, 4.8, 4.9, 4.10



Рисунок 4.7 – Использование встроенной функции

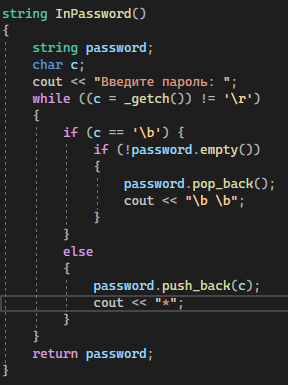


Рисунок 4.8 – Использование пользовательской функции

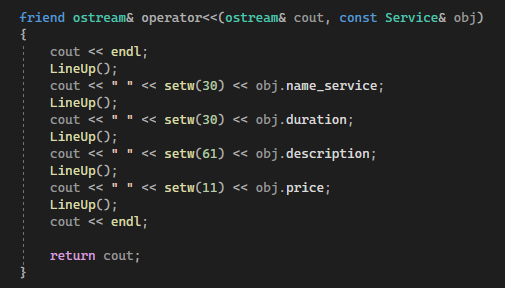


Рисунок 4.9 – Использование дружественной функции



Рисунок 4.10 – Использование виртуальных функций

Использование пространств имен встроенных и пользовательских представлено на рисунках 4.11 и 4.12.



Рисунок 4.11 – Использование встроенных пространственных имен



Рисунок 4.12 – Использование пользовательских пространственных имен

Реализация обработки ошибок программы (средствами языка си++) представлена на рисунке 4.13

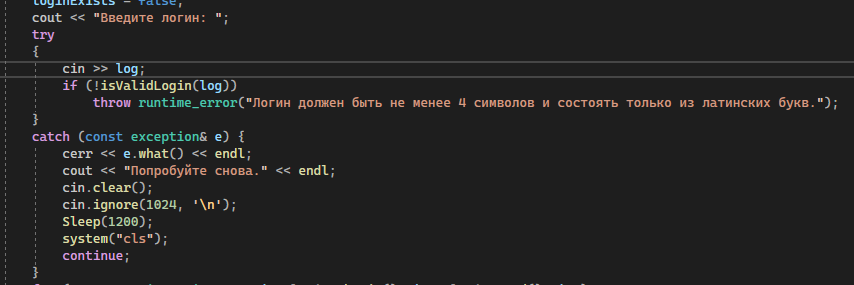


Рисунок 4.13 – Реализация обработки ошибки при вводе логина

Использование перегрузки методов и операторов, переопределение методов показана на рисунках 4.14, 4.15, 4.16

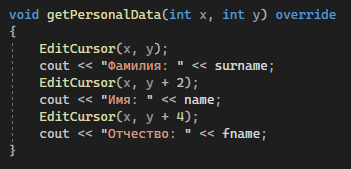


Рисунок 4.14 – Переопределение метода



Рисунок 4.15 – Перегрузка метода

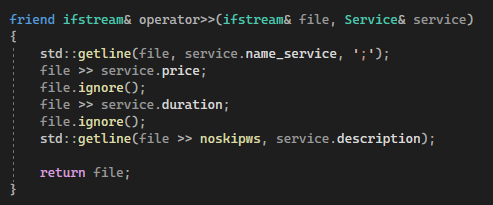


Рисунок 4.16 – Перегрузка оператора

Использование шаблонов классов и методов показано на рисунках 4.17 и 4.18



Рисунок 4.17 – Использование шаблона метода



Рисунок 4.18 – Использование шаблона класса

# **разработка и описание диаграммы классов приложения**

При разработке автоматизированной системы, ключевым этапом является проектирование её архитектуры с использованием диаграммы классов. Диаграмма классов (рисунок 5.1) предоставляет визуальное представление архитектуры системы, иллюстрируя классы и их взаимосвязи.

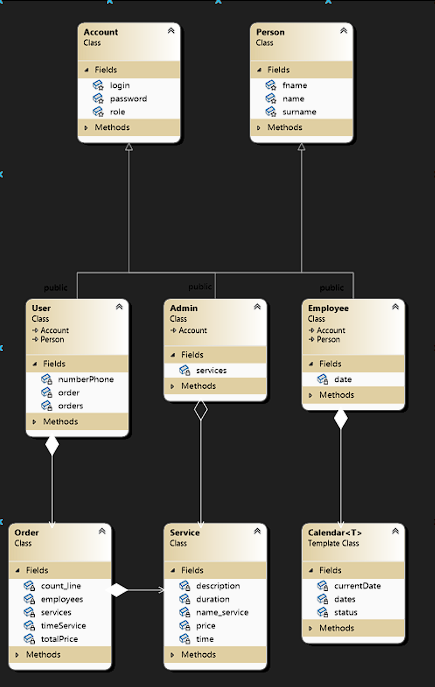


Рисунок 5.1 – Диаграмма классов

Из диаграммы классов видно, что классы User, Employee наследуются от классов Account и Person. Класс User связан с Order композицией, также как и Order связан с Service. Employee также композиция шаблонного класса Сalendar. А Admin связан с Service агрегацией.

# **разработка и описание диаграммы вариантов использования приложения**

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

В программе выделяется три основные роли (рисунок 6.1): сотрудник, администратор и пользователь, каждый из которых обладает собственным набором сценария использования приложения.

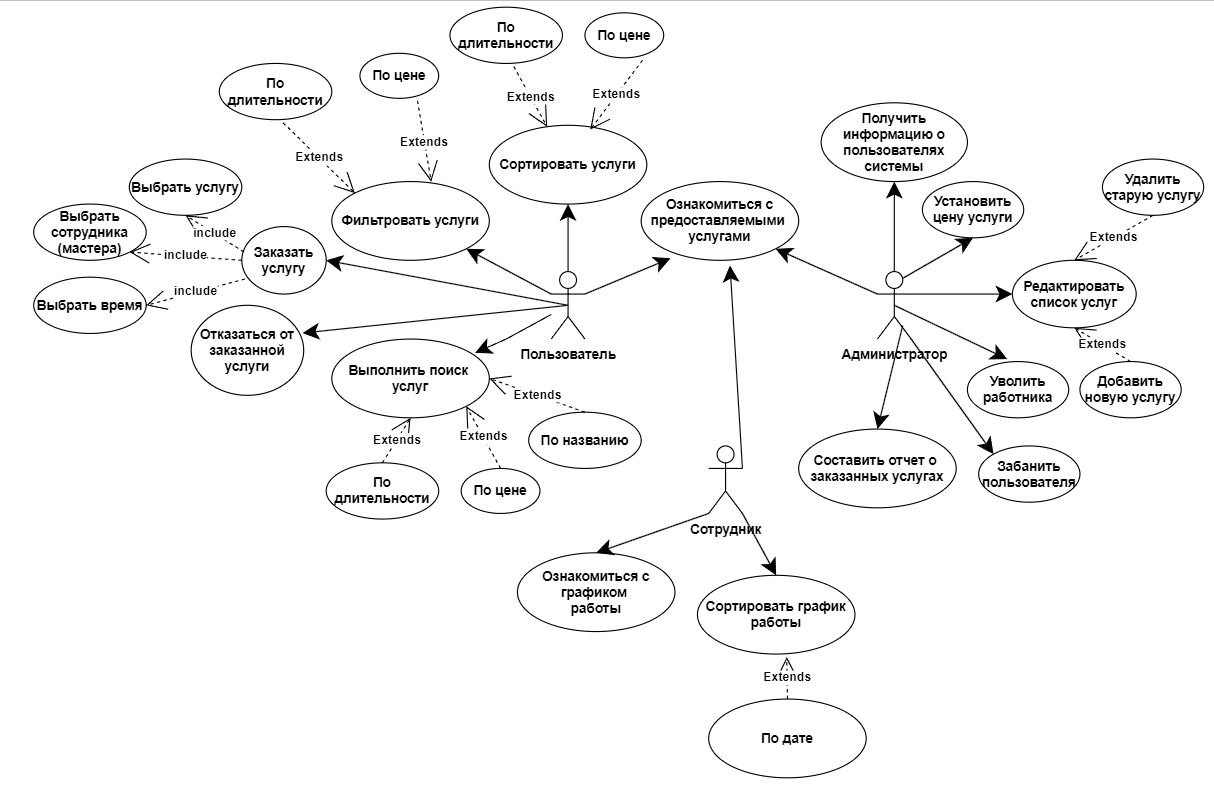


Рисунок 6.1 – Диаграмма вариантов использования

Диаграмма вариантов использования позволяет быстро определить все возможности системы относительно каждой роли, что гарантирует качественное понимания функционала системы и возможностей каждого пользователя.

# **блок-схема алгоритма всей программы и двух основных методов**

## **Блок-схема алгоритма функции SignIn**

Схемы алгоритмов являются важным инструментом при разработке программного обеспечения. Схемы алгоритмов облегчают визуализацию последовательности шагов, необходимых для выполнения конкретной задачи.

На рисунке 7.1.1 представлен алгоритм работы функции SignIn, использующей для авторизации пользователей.

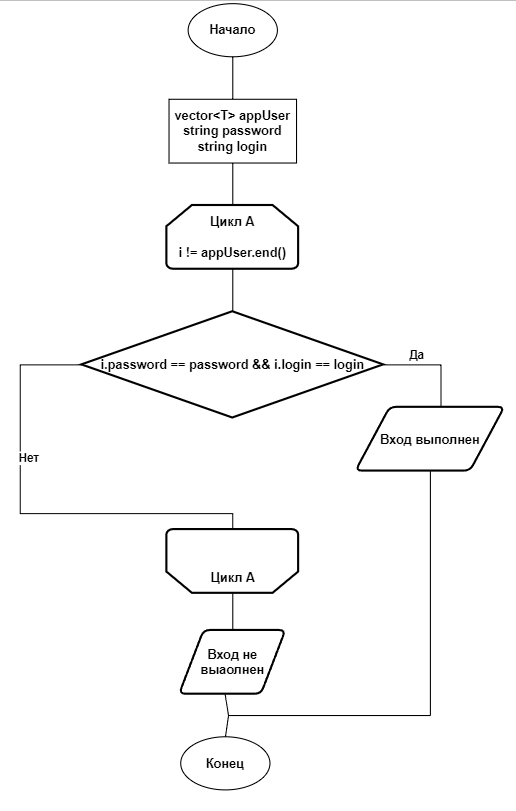


Рисунок 7.1.1 – Алгоритм функции SignIn

## **Блок-схема алгоритма функции main**

На рисунке 7.2.1 представлен алгоритм работы функции main, где пользователю предоставляется на выбор три действия: зарегистрироваться, авторизоваться или выйти из системы.

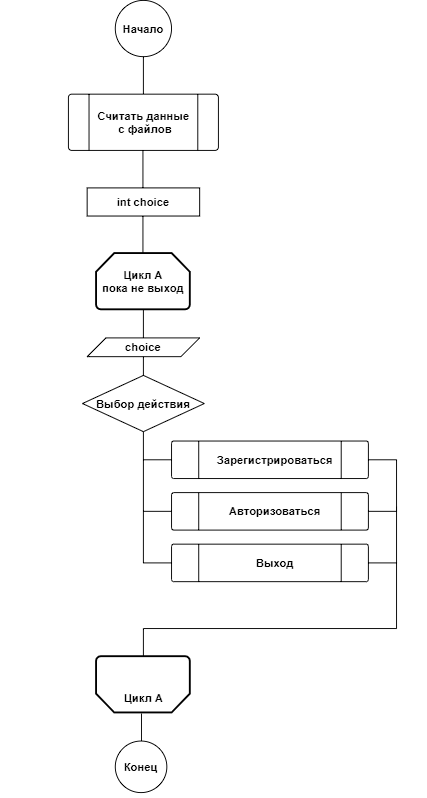


Рисунок 7.2.1 – Алгоритм функции main

## **Блок-схема алгоритма всей программы**

На рисунке 7.3.1 представлена блок-схема алгоритма всей программы.

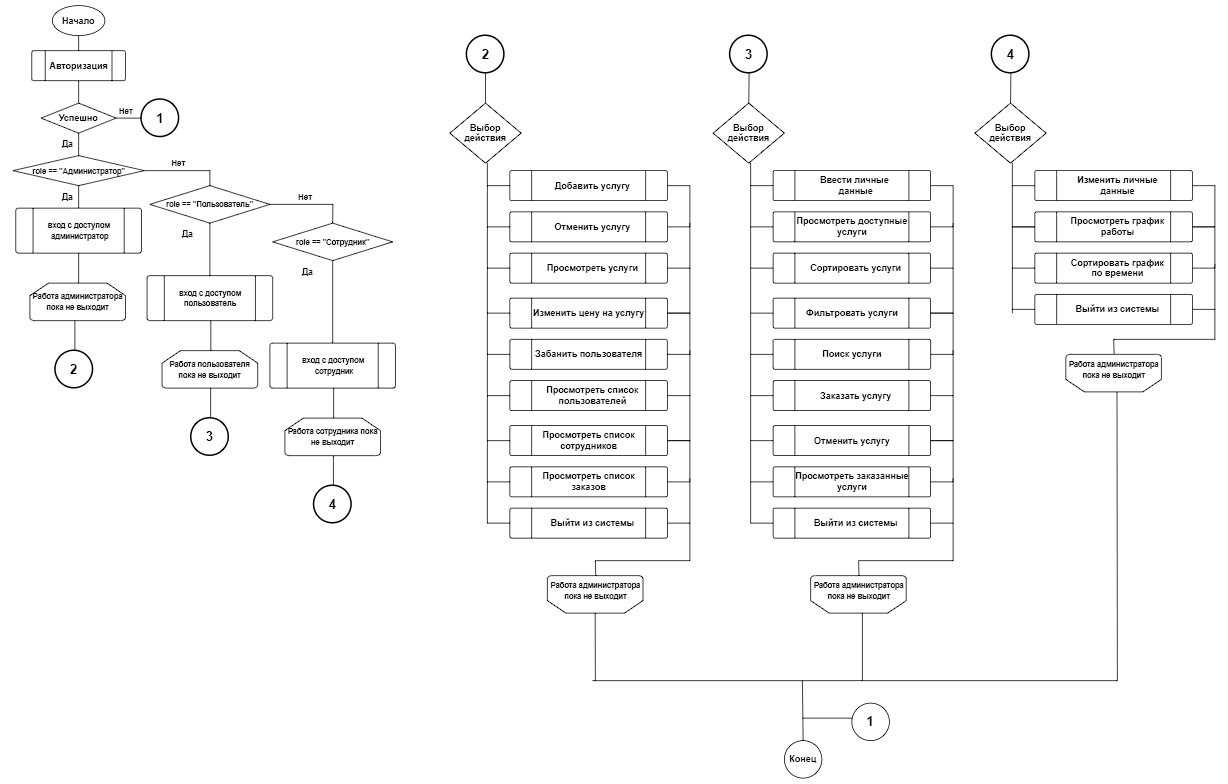


Рисунок 7.3.1 – Алгоритм работы всей программы

# **описание алгоритма запуска приложения, Его использование, результаты работы программы, тестирование обработки ошибок**

## **Тестирование ошибок**

В данной программе присутствует обработка данных. К примеру есть ограничения на ввод символов логина, количество символов должно быть от 4 и больше, а также должен состоять из латинских букв или цифр.(рисунок 8.1.1)

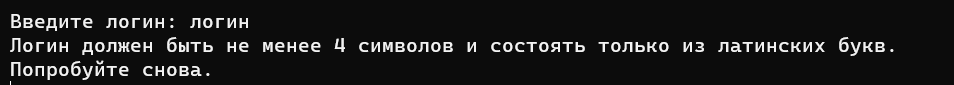


Рисунок 8.1.1 – Обработка данных ввода логина

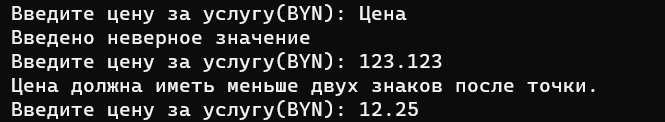


Рисунок 8.1.2 – Обработка данных ввода цены

## **Алгоритм запуска приложения**

После запуска программы пользователю предоставляется меню (рисунок 8.1), в котором он может выбрать одно из действий: Зарегистрироваться, Выйти, Войти.

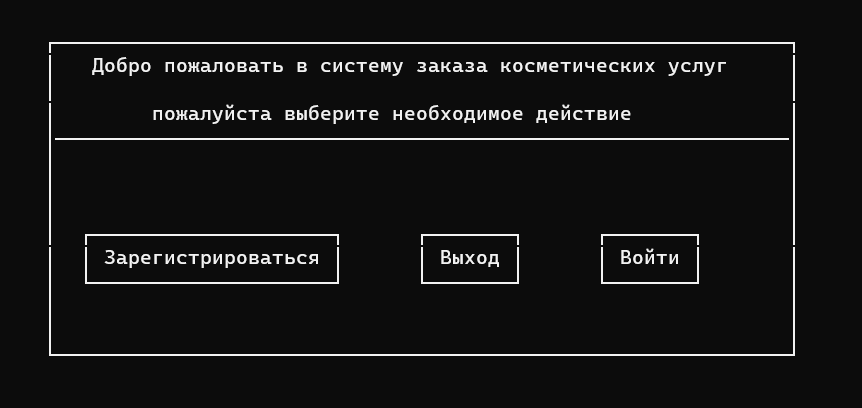


Рисунок 8.2.1 – Меню после запуска программы

При выборе регистрации, пользователю дается на выбор три роли (рисунок 8.2): Администратор, пользователь, сотрудник.

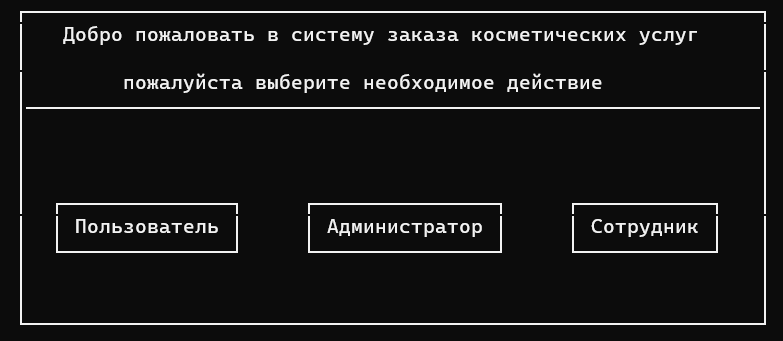


Рисунок 8.2.2 – Меню выбора роли

После выбора роли необходимо ввести логин и пароль (рисунок 8.3).

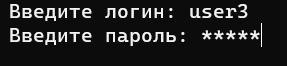


Рисунок 8.2.3 – Ввод логина и пароля

При вводе пароля, он отображается зашифровано в виде звездочек.

После авторизации под ролью пользователя предоставляется следующее меню(рисунок 8.2.4).

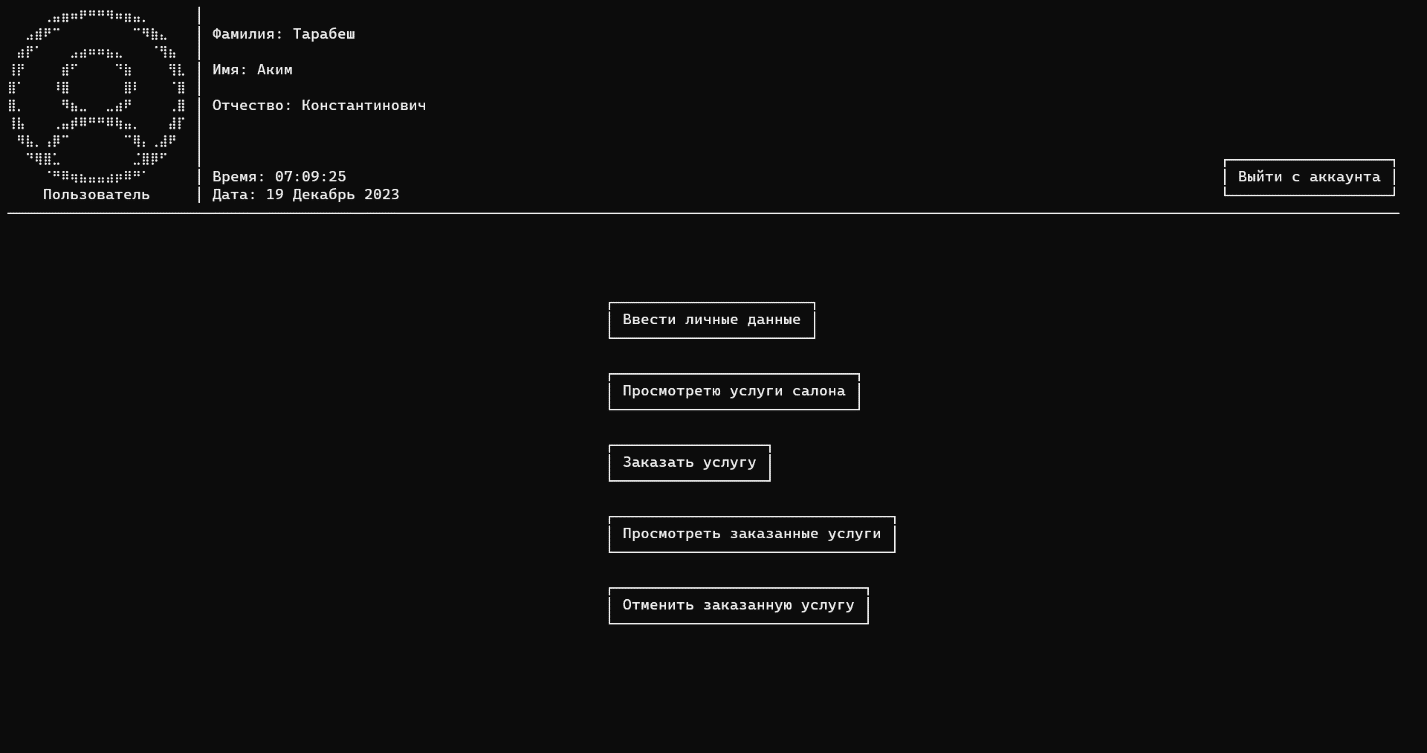


Рисунок 8.2.4 – Начальное меню пользователя

При выборе блока “Ввести личные данные” пользователю предоставляется возможность ввести личные данные (рисунок 8.2.5)

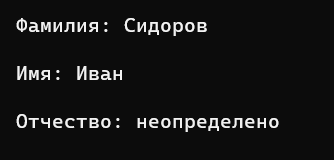


Рисунок 8.2.5 – Ввод личных данных

При выборе “Просмотреть услуги” пользователю выводятся доступные для заказа услуги, а также методы их сортировки по цене и длительности, поиска по цене, длительности и названии, а также фильтрации по цене.

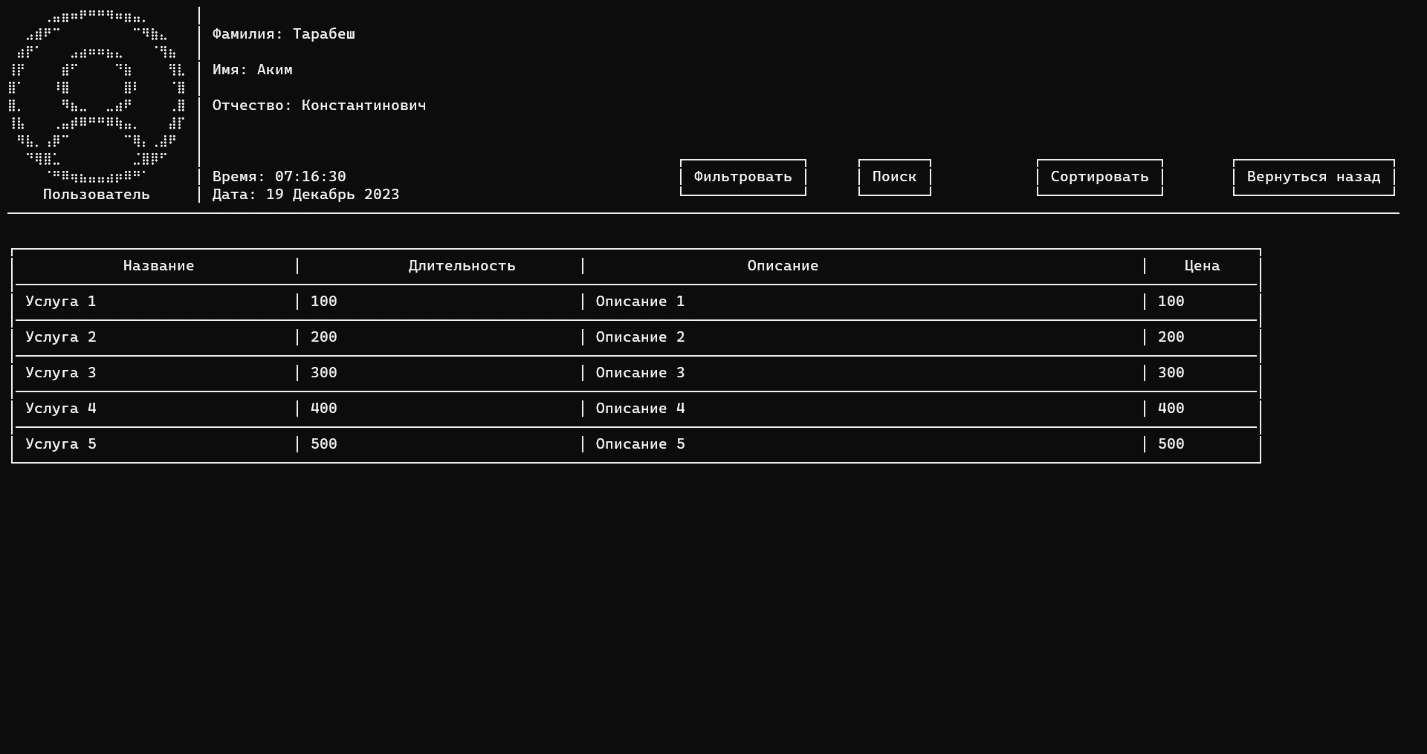


Рисунок 8.2.6 – Просмотр услуг салона

При выборе заказать услугу, пользователю предоставляется возможность выбора услуги из списка просмотра услуг и при выборе одной из предоставляется возможность выбора дня на который пользователь хочет заказать услугу, в течении недели от текущего дня (рисунок 8.2.7).



Рисунок 8.2.7 – Выбор дня заказанной услуги

Затем пользователю предоставляется возможность выбора времени, на которое он хочет заказать услугу (рисунок 8.2.8).



Рисунок 8.2.8 – Выбор времени заказанной услуги

И в заключении операции, пользователь выбирает свободного сотрудника. Не свободный сотрудник помечается красным цветом (рисунок 8.2.9).

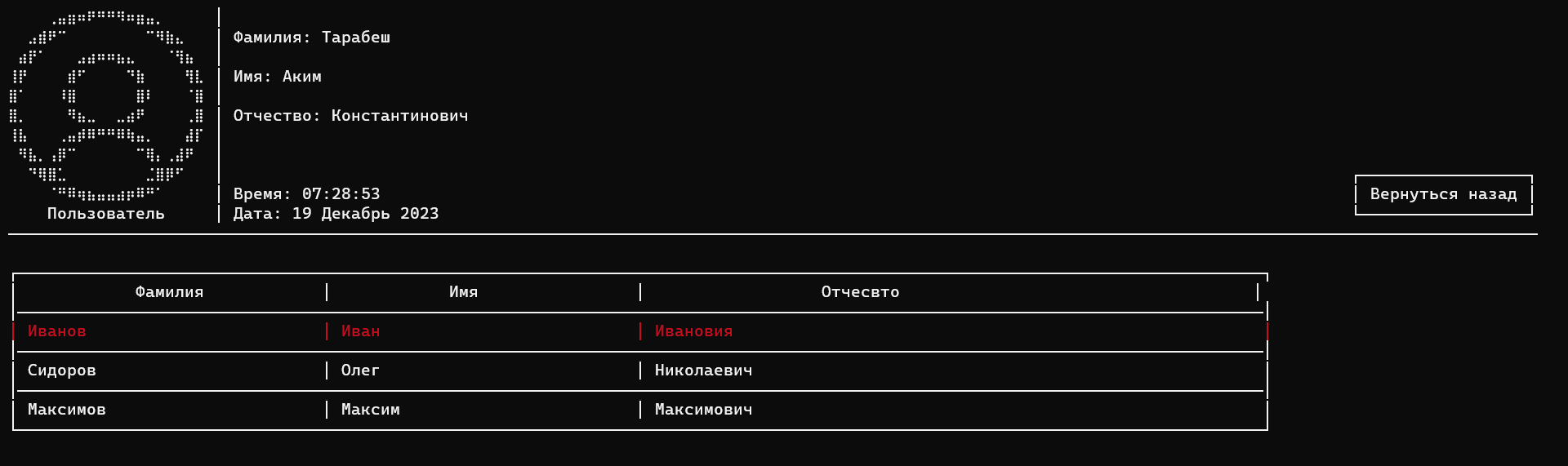


Рисунок 8.2.8 – Выбор свободного сотрудника

При выборе блока “Просмотреть заказанные услуги” пользователю выводятся его заказы и подсчитанная общая сумма заказов (рисунок 8.2.9)

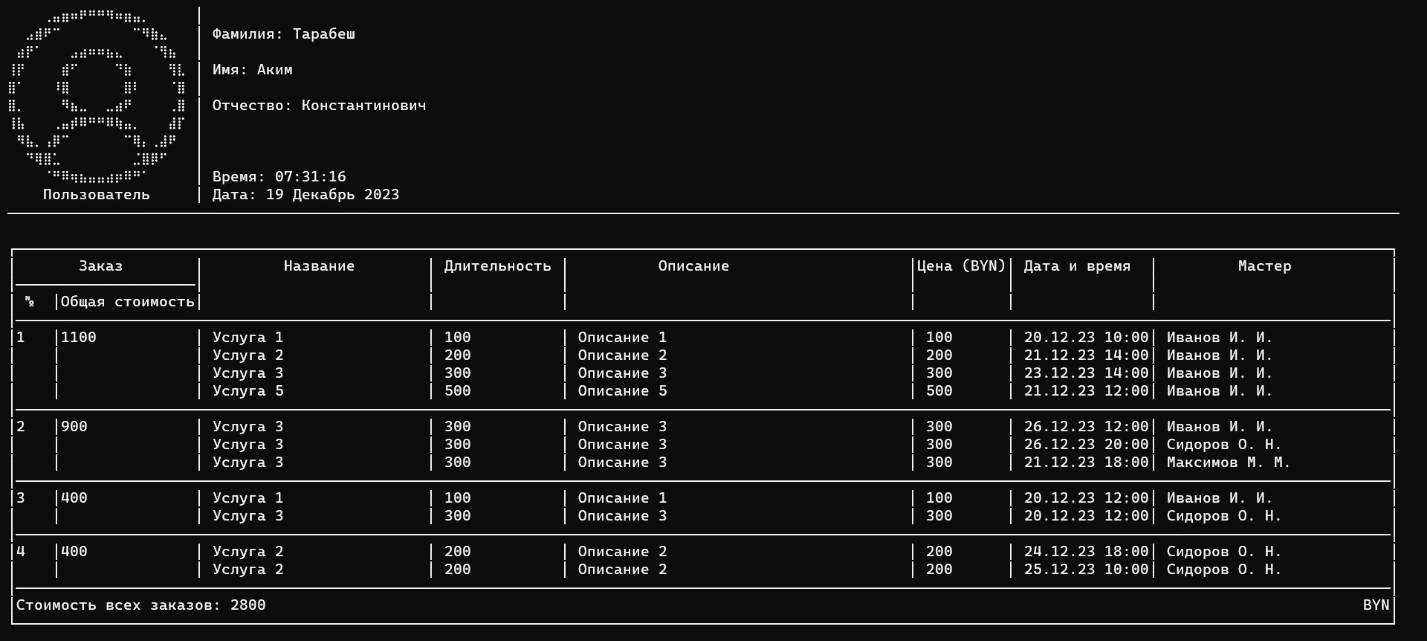


Рисунок 8.2.9 – Список заказанных услуг пользователем

При выборе отменить заказанную услугу, пользователю предоставляется возможность отмены заказа (рисунок 8.2.10).

Рисунок 8.2.10 – Список с отменённым заказом

При входе от имени сотрудника предоставляется меню из изменить личные данные и посмотреть график работы (рисунок 8.2.11)

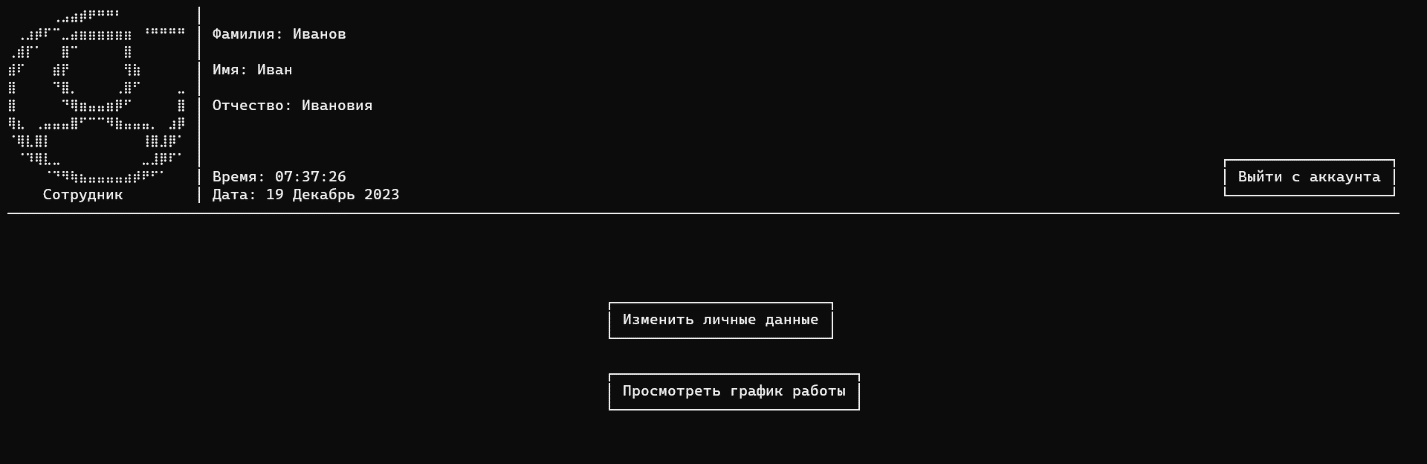


Рисунок 8.2.11 – Меню сотрудника

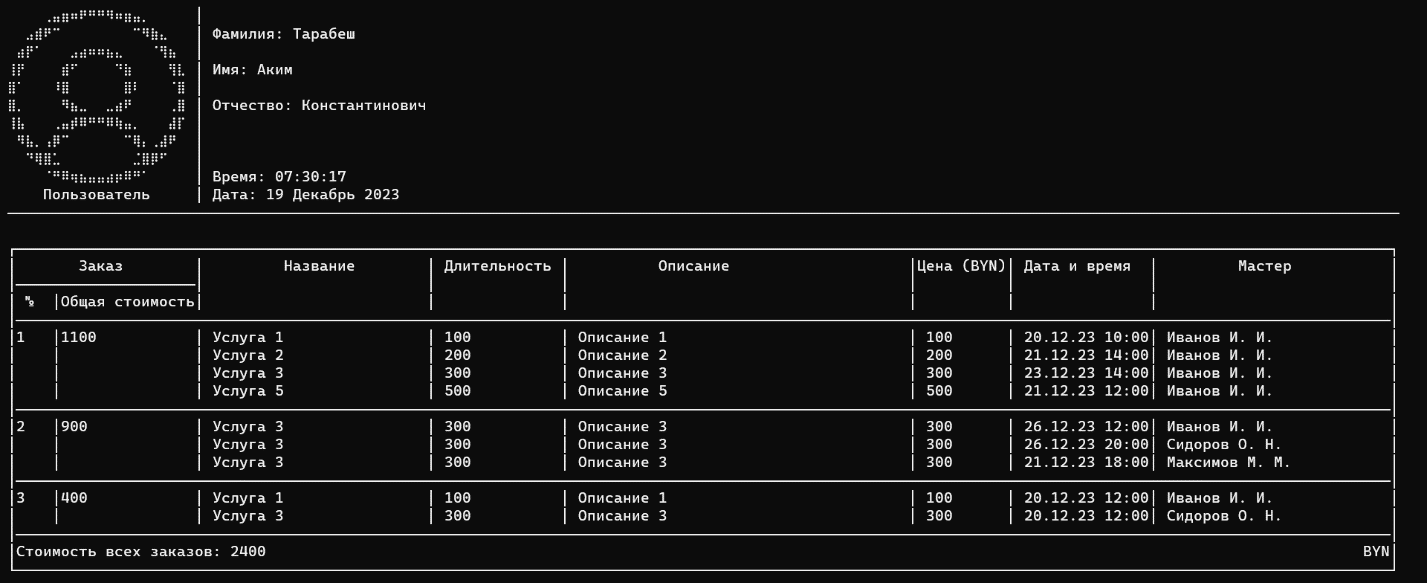
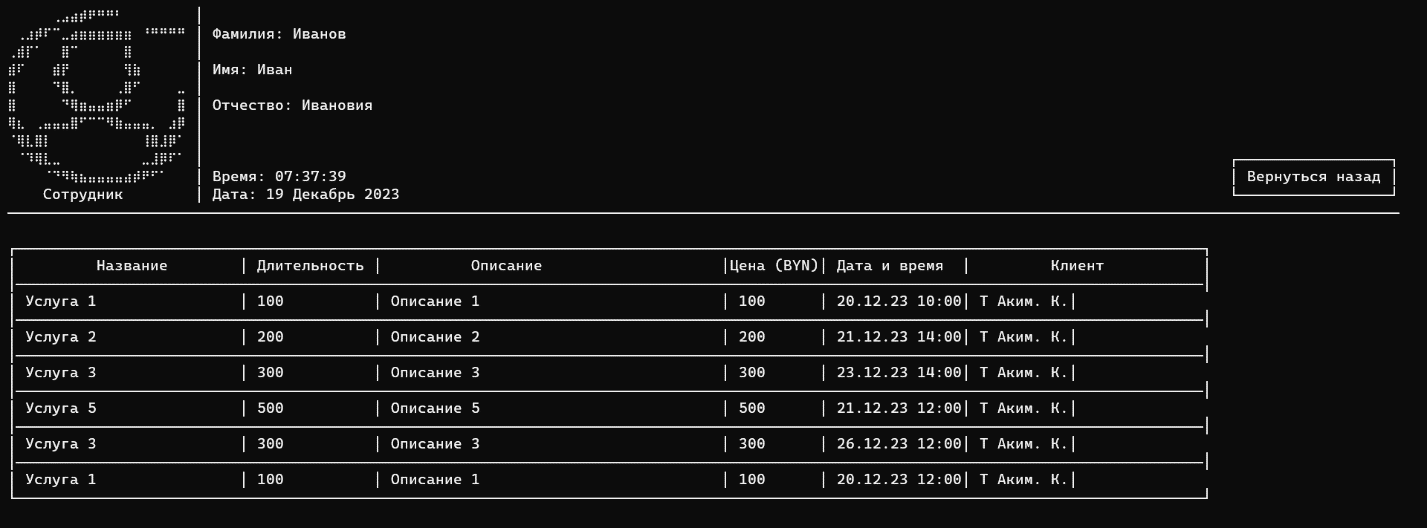


Рисунок 8.2.12 – Заказы сотрудника



Таким образом в результате разработки алгоритма запуска приложения и тестирования ошибок для учета деятельности косметического салона, был создан код на языке C++, который позволяет эффективно проводить учет заказанных услуг и работы салона в общих чертах. Алгоритм был разработан с учетом всех возможных ошибок, которые могут возникать при работе с данными, и был успешно протестирован на нескольких тестовых наборах данных. Полученные результаты показали, что код работает корректно и способен обрабатывать большие объемы информации. Таким образом, разработанный алгоритм является надежной и эффективной инструментом для учета деятельности косметического салона.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

[2] aka [Электронный ресурс]. – Режим доступа: https://aka.ms/GetM365.

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

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

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

[6] Бьерн Страуструп Язык программирования C++. Лекции и упражнения - М.: ЭКОМ Паблишерз, 2010. - 640 c.

[7] Роберт Мартин “Чистый код, создание, анализ и рефакторинг” 2023г – 464с

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Листинг кода**

main.cpp:

#include "definition.h"

template <class T>

void file\_write(T& obj, string name)

{

ofstream file;

file.open(name, ofstream::app);

if (!file.is\_open())

cout << "Ошибка открытия файла" << endl;

else

{

file << obj.getRole() << " " << obj.getLogin() << " " << obj.getPassword() << endl;

}

file.close();

}

template <class T>

void file\_write\_personDate(T& obj, string name)

{

ofstream file;

file.open(name, ofstream::app);

if (!file.is\_open())

cout << "Ошибка открытия файла" << endl;

else

{

file << obj.getRole() << " " << obj.getLogin() << " " << obj.getSurname() << " " << obj.getName() << " " << obj.getPatronymic() << endl;

}

file.close();

}

template <class T>

void file\_write\_Order(T& obj, string name)

{

ofstream file;

file.open(name, ofstream::app);

if (!file.is\_open())

cout << "Ошибка открытия файла" << endl;

else

{

file << obj;

}

file.close();

}

template void file\_write\_Order<User>(User&, string);

template void file\_write\_personDate<User>(User&, string);

template<class T>

bool signUp(vector<T>& appUsers, vector<string> logins)

{

if (!appUsers.back().addLogin(logins))

{

appUsers.pop\_back();

return false;

}

else

{

appUsers.back().addPassword();

return true;

}

}

int main()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO consoleInfo;

WORD saved\_attributes;

ConsoleColor();

HWND console = GetConsoleWindow();

ShowWindow(console, SW\_MAXIMIZE);

SetConsoleTextAttribute(hConsole, 15);

vector<string> logins;

vector<User> users;

vector<Admin> admins;

vector<Employee> employees;

vector<Service> services;

ifstream file\_;

file\_.open("service.txt", ios::in);

if (!file\_.is\_open())

cout << "Ошибка открытия файла" << endl;

else

{

Service service;

file\_.seekg(0, ios::beg);

while (file\_ >> service)

{

services.push\_back(service);

}

}

file\_.clear();

file\_.close();

ifstream file;

file.open("log.txt", ios::in);

if (!file.is\_open())

cout << "Ошибка открытия файла" << endl;

else

{

Account appUser;

file.seekg(0, ios::beg);

while (file >> appUser)

{

if (appUser.getRole() == "Пользователь")

{

User user;

user.setRole(appUser.getRole());

user.setLogin(appUser.getLogin());

user.setPassword(appUser.getPassword());

logins.push\_back(appUser.getLogin());

users.push\_back(user);

}

else if (appUser.getRole() == "Администратор")

{

Admin user;

user.setRole(appUser.getRole());

user.setLogin(appUser.getLogin());

user.setPassword(appUser.getPassword());

logins.push\_back(appUser.getLogin());

admins.push\_back(user);

}

else if (appUser.getRole() == "Сотрудник")

{

Employee user;

user.setRole(appUser.getRole());

user.setLogin(appUser.getLogin());

user.setPassword(appUser.getPassword());

logins.push\_back(appUser.getLogin());

employees.push\_back(user);

}

}

file.clear();

file.close();

}

ifstream file\_2;

file\_2.open("personalDate.txt", ios::in);

if (!file\_2.is\_open())

cout << "Ошибка открытия файла" << endl;

else

{

User user;

file\_2.seekg(0, ios::beg);

while (!file\_2.eof())

{

file\_2 >> user;

if (user.getRole() == "Пользователь")

{

for (vector<User>::iterator i = users.begin(); i != users.end(); i++)

{

if (i.operator->()->getLogin() == user.getLogin())

{

i.operator->()->setSurname(user.getSurname());

i.operator->()->setName(user.getName());

i.operator->()->setPatronymic(user.getPatronymic());

break;

}

}

}

else

{

for (vector<Employee>::iterator i = employees.begin(); i != employees.end(); i++)

{

if (i.operator->()->getLogin() == user.getLogin())

{

i.operator->()->setSurname(user.getSurname());

i.operator->()->setName(user.getName());

i.operator->()->setPatronymic(user.getPatronymic());

break;

}

}

}

}

file\_2.clear();

file\_2.close();

}

int choice;

do

{

system("cls");

choice = menu\_();

Sleep(100);

switch (choice)

{

case 1:

{

int sign = menu\_signUp();

SetConsoleTextAttribute(hConsole, 15);

if (sign == 1)

{

User user;

users.push\_back(user);

if (signUp(users, logins))

{

logins.push\_back(users.back().getLogin());

file\_write(users.back(), "log.txt");

}

}

else if (sign == 2)

{

Admin admin;

admins.push\_back(admin);

if (signUp(admins, logins))

{

logins.push\_back(admins.back().getLogin());

file\_write(admins.back(), "log.txt");

}

}

else if (sign == 3)

{

Employee employee;

employees.push\_back(employee);

if (signUp(employees, logins))

{

logins.push\_back(employees.back().getLogin());

employees.back().setPersonalData(0, 0);

file\_write\_personDate(employees.back(), "personalDate.txt");

file\_write(employees.back(), "log.txt");

}

}

break;

}

case 2:

EditCursor(0, 0);

for (vector<User>::iterator i = users.begin(); i != users.end(); i++)

{

cout << i.operator->()->getRole() << " " << i.operator->()->getLogin() << " " << i.operator->()->getPassword() << endl;

}

for (vector<Admin>::iterator i = admins.begin(); i != admins.end(); i++)

{

cout << i.operator->()->getRole() << " " << i.operator->()->getLogin() << " " << i.operator->()->getPassword() << endl;

}

for (vector<Employee>::iterator i = employees.begin(); i != employees.end(); i++)

{

cout << i.operator->()->getRole() << " " << i.operator->()->getLogin() << " " << i.operator->()->getPassword() << endl;

}

system("pause");

int choice\_ = 0;

vector<Admin>::iterator admin;

vector<User>::iterator user;

vector<Employee>::iterator employee;

string login;

string password;

char exit;

do

{

system("cls");

cout << "Введите логин: ";

cin >> login;

cin.ignore();

password = InPassword();

admin = signIn(admins, password, login);

if (admin != admins.end())

choice\_ = 1;

if (choice\_ == 0)

{

user = signIn(users, password, login);

if (user != users.end())

choice\_ = 2;

}

if (choice\_ == 0)

{

employee = signIn(employees, password, login);

if (employee != employees.end())

choice\_ = 3;

}

if (choice\_ == 0)

{

cout << endl << "Введен неверный логин или пароль." << endl << "Для выхода нажмите клавишу ESC, для продолжения любую другую клавишу" << endl;

exit = \_getch();

}

else

break;

} while (exit != 27);

ShowConsoleCursor(FALSE);

if (choice\_ == 0)

break;

switch (choice\_)

{

case 1:

{

menu\_Admin(\*admin, services, users, employees);

}

break;

case 2:

{

menu\_User(\*user, services, employees);

}

break;

case 3:

{

menu\_Employee(\*employee);

}

break;

}

break;

}

}while (choice != 3);

}

roles.h:

#pragma once

#include "service.h"

using namespace std;

template <class T>

extern void file\_write\_personDate(T& obj, string name);

extern Service add\_service(vector<Service>& services);

extern string InPassword();

extern bool isMouseClicked();

extern bool isMouseBox(int mouseX, int mouseY, int rectX, int rectY, string text);

template<class T>

extern void menu\_profile(T& obj);

extern HANDLE hConsole;

extern bool isValidLogin(string login);

class Account

{

protected:

string role;

string login;

string password;

public:

Account()

{

login = "Неизвестно";

password = "Неизвестно";

}

Account(string login, string password, string name, string surname)

{

this->login = login;

this->password = password;

}

string getLogin()

{

return login;

}

string getPassword()

{

return password;

}

string getRole()

{

return role;

}

void setRole(string role)

{

this->role = role;

}

void setLogin(string login)

{

this->login = login;

}

void setPassword(string password)

{

this->password = password;

}

void appUserDisplay()

{

cout << role << " " << login << " " << password << endl;

}

bool addLogin(vector<string> logins)

{

string log;

char exit = 0;

bool loginExists;

do

{

loginExists = false;

cout << "Введите логин: ";

try

{

cin >> log;

if (!isValidLogin(log))

throw runtime\_error("Логин должен быть не менее 4 символов и состоять только из латинских букв.");

}

catch (const exception& e) {

cerr << e.what() << endl;

cout << "Попробуйте снова." << endl;

cin.clear();

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

Sleep(1200);

system("cls");

continue;

}

for (vector<string>::iterator i = logins.begin(); i != logins.end(); i++)

{

if (log == \*i)

{

cout << "Данный логин занят, попробуйте снова." << endl;

loginExists = true;

break;

}

}

if (loginExists)

{

cout << endl << "Для выхода нажмите клавишу ESC, для продолжения любую другую клавишу" << endl;

exit = \_getch();

}

else

{

login = log;

return true;

}

} while (exit != 27);

return false;

}

virtual void addPassword()

{

this->password = InPassword();

}

friend ifstream& operator>>(ifstream& file, Account& user)

{

file >> user.role >> user.login >> user.password;

return file;

}

};

template<class T>

class Calendar

{

private:

time\_t currentDate;

array<array<bool, 6>, 7> status;

//array<array<string, 6>, 7> datesStr;

vector<T> dates;

int currDay;

int currTimeSlot;

public:

Calendar()

{

currentDate = time(0);

createCalendar();

}

int getCurrDay()

{

return currDay;

}

int getCurrTimeSlot()

{

return currTimeSlot;

}

void createCalendar()

{

currDay = 0;

currTimeSlot = 0;

for (int i = 0; i < 7; i++)

for (int j = 0; j < 6; j++)

status[i][j] = true;

dates.resize(7);

for (int day = 0; day < 7; ++day)

{

currentDate += 24 \* 60 \* 60;

dates[day] = \*localtime(&currentDate);

}

}

bool getStatus(int day, int timeSlot)

{

return status[day][timeSlot];

}

void setStatus(int day, int timeSlot, bool status)

{

if (status)

this->status[day][timeSlot] = true;

else

this->status[day][timeSlot] = false;

}

string getTimeString(int timeSlot) const

{

switch (timeSlot)

{

case 0: return "10:00";

case 1: return "12:00";

case 2: return "14:00";

case 3: return "16:00";

case 4: return "18:00";

case 5: return "20:00";

}

}

string getFormattedDate(const T& date) const

{

stringstream ss;

ss << put\_time(&date, "%d/%m/%Y");

return ss.str();

}

T printCalendar(int day, T &time)

{

int x1 = 0;

int y1 = 13;

int x, y;

while (true)

{

SetConsoleTextAttribute(hConsole, 15);

for (int i = 0; i < 6; i++)

{

getCursorPosition(x, y);

drawTextBox(getTimeString(i), x1 + 50 + i \* 10, y1);

if (isMouseBox(x, y, x1 + 50 + i \* 10 - 2, y1, getTimeString(i)))

{

SetConsoleTextAttribute(hConsole, 6);

drawTextBox(getTimeString(i), x1 + 50 + i \* 10, y1);

SetConsoleTextAttribute(hConsole, 15);

if (isMouseClicked())

{

system("cls");

time.tm\_hour = i \* 2 + 10;

time.tm\_min = 0;

time.tm\_mday = i;

currTimeSlot = i;

return time;

}

}

}

}

}

T printCalendar()

{

T time;

int x1 = 75;

int y1 = 13;

int x, y;

while (true)

{

SetConsoleTextAttribute(hConsole, 15);

y1 = 13;

for (int day = 0; day < 7; ++day)

{

getCursorPosition(x, y);

drawTextBox(getFormattedDate(dates[day]), x1, y1);

if (isMouseBox(x, y, x1 - 2, y1, getFormattedDate(dates[day])))

{

SetConsoleTextAttribute(hConsole, 6);

drawTextBox(getFormattedDate(dates[day]), x1, y1);

SetConsoleTextAttribute(hConsole, 15);

if (isMouseClicked())

{

system("cls");

Sleep(200);

time = printCalendar(day, time);

time.tm\_year = dates[day].tm\_year;

time.tm\_mon = dates[day].tm\_mon;

time.tm\_mday = dates[day].tm\_mday;

time.tm\_wday = day;

currDay = day;

//status[day][time.tm\_mday] = false;

return time;

}

}

y1 += 4;

}

}

}

};

class Person

{

protected:

string name;

string surname;

string patronymic;

public:

Person()

{

name = "неопределено";

surname = "неопределено";

patronymic = "неопределено";

}

string getSurname()

{

return surname;

}

string getName()

{

return name;

}

string getPatronymic()

{

return patronymic;

}

void setName(string name)

{

this->name = name;

}

void setSurname(string surname)

{

this->surname = surname;

}

void setPatronymic(string patronymic)

{

this->patronymic = patronymic;

}

virtual void setPersonalData(int x, int y) = 0;

virtual void getPersonalData(int x, int y) = 0;

};

class User;

class Order;

class Employee : public Account, public Person

{

private:

Calendar<tm> date;

vector<Service> services;

vector<tm> date\_services;

vector<string> nameCustomers;

vector<string> surnameCustomers;

vector<string> patronymicCustomers;

int count\_service;

public:

Employee()

{

role = "Сотрудник";

count\_service = 0;

}

Calendar<tm> getDate()

{

return date;

}

void setService(Service service)

{

services.push\_back(service);

count\_service++;

}

void setDateService(tm date\_service)

{

date\_services.push\_back(date\_service);

}

void setInfoUser(string nameCustomer, string surnameCustomer, string patronymicCustomer)

{

nameCustomers.push\_back(nameCustomer);

surnameCustomers.push\_back(surnameCustomer);

patronymicCustomers.push\_back(patronymicCustomer);

}

void setDateStatus(int day, int timeSlot, bool status)

{

date.setStatus(day, timeSlot, status);

}

bool getStatus(int day, int timeSlot)

{

return date.getStatus(day, timeSlot);

}

void addPassword() override

{

this->password = InPassword();

}

void add\_Service()

{

cout << "это делает сотрудник" << endl;

}

void setPersonalData(int x, int y) override

{

EditCursor(x, y);

cout << " ";

EditCursor(x, y);

cout << "Фамилия: ";

cin >> surname;

EditCursor(x, y + 2);

cout << " ";

EditCursor(x, y + 2);

cout << "Имя: ";

cin >> name;

EditCursor(x, y + 4);

cout << " ";

EditCursor(x, y + 4);

cout << "Отчество: ";

cin >> patronymic;

}

void getPersonalData(int x, int y) override

{

EditCursor(x, y);

cout << "Фамилия: " << surname;

EditCursor(x, y + 2);

cout << "Имя: " << name;

EditCursor(x, y + 4);

cout << "Отчество: " << patronymic;

}

void displayEmployee()

{

string name\_employee;

name\_employee = surname + " " + name[0] + "." + " " + patronymic[0] + ".";

cout << " " << setw(25) << name\_employee;

LineUp();

}

friend ostream& operator<<(ostream& cout, const Employee& obj)

{

cout << endl;

LineUp();

cout << " " << setw(30) << obj.surname;

LineUp();

cout << " " << setw(30) << obj.name;

LineUp();

cout << " " << setw(62) << obj.patronymic;

LineUp();

cout << endl;

return cout;

}

friend ifstream& operator>>(ifstream& file, Employee& employee)

{

file >> employee.role >> employee.login >> employee.surname >> employee.name >> employee.patronymic;

return file;

}

void displayDate()

{

LineUpUgol(1);

Line(133);

LineUpUgol(3);

cout << endl;

LineUp();

cout << setw(25) << left << " Название";

LineUp();

cout << setw(10) << left << " Длительность ";

LineUp();

cout << setw(38) << left << " Описание";

LineUp();

cout << setw(10) << left << "Цена (BYN)";

LineUp();

cout << setw(15) << left << " Дата и время";

LineUp();

cout << setw(26) << left << " Клиент";

LineUp();

for (int i = 0; i < count\_service; i++)

{

cout << endl;

LineUp();

Line(133);

LineUp();

cout << endl;

LineUp();

services[i].displayService();

wcout << " " << date\_services[i].tm\_mday << "." << date\_services[i].tm\_mon + 1 << "." << date\_services[i].tm\_year + 1900 - 2000 << " " << date\_services[i].tm\_hour << ":" << date\_services[i].tm\_min << "0";

LineUp();

cout << " " << nameCustomers[i][0] << " " << surnameCustomers[i] << ". " << patronymicCustomers[i][0] << ".";

//EditCursor(134, i + 18);

LineUp();

}

cout << endl;

LineUpUgol(2);

Line(133);

LineUpUgol(4);

/\*cout << services[i] << endl;

cout << surnameCustomers[i] << " " << nameCustomers[i] << " " << patronymicCustomers[i] << endl;

wcout << date\_services[i].tm\_mday << "." << date\_services[i].tm\_mon + 1 << "." << date\_services[i].tm\_year + 1900 << " " << date\_services[i].tm\_hour << ":" << date\_services[i].tm\_min + "0" << endl;\*/

}

};

class Order

{

private:

// set of OrderLine

float totalPrice;

vector<Service> services;

vector<Employee> employees;

vector<tm> timeService;

int count\_line;

public:

Order()

{

totalPrice = 0;

//service = nullptr;

//employye = nullptr;

//services

count\_line = 0;

}

void add\_Order(Service service, Employee employee, tm timeService)

{

//shared\_ptr<Service> new\_service(new Service[count\_line + 1]);

//shared\_ptr<Employee> new\_employee(new Employee[count\_line + 1]);

this->services.push\_back(service);

this->employees.push\_back(employee);

this->timeService.push\_back(timeService);

this->totalPrice = service.getPrice();

this->count\_line++;

}

void delete\_orderInfo()

{

count\_line = 0;

services.clear();

employees.clear();

timeService.clear();

}

int getCountLine()

{

return count\_line;

}

vector<Service> getService()

{

return services;

}

vector<Employee> getEmployee()

{

return employees;

}

vector<tm> getTimeService()

{

return timeService;

}

void display\_order(int j)

{

LineUp();

cout.width(4);

cout << j + 1;

LineUp();

totalPrice = 0;

for (int i = 0; i < this->count\_line; i++)

{

totalPrice += services[i].getPrice();

}

wcout << setw(15) << left << totalPrice;

LineUp();

for (int i = 0; i < this->count\_line; i++)

{

services[i].displayService();

wcout << " " << timeService[i].tm\_mday << "." << timeService[i].tm\_mon + 1 << "." << timeService[i].tm\_year + 1900 - 2000 << " " << timeService[i].tm\_hour << ":" << timeService[i].tm\_min << "0";

LineUp();

employees[i].displayEmployee();

if (i + 1 != count\_line)

{

LineUp();

cout.width(4);

cout << " ";

LineUp();

cout.width(15);

cout << " ";

LineUp();

}

}

cout << endl;

LineUp();

Line(154);

LineUp();

}

float getTotalPrice()

{

return totalPrice;

}

friend ostream& operator<<(ostream& cout, const Order& obj)

{

for (int i = 0; i < obj.count\_line; i++)

{

cout << obj.services[i] << " " << obj.totalPrice << " " << obj.employees[i] << " ";

}

return cout;

}

};

class User : public Account, public Person

{

private:

vector<Order> orders;

Order order;

public:

User()

{

role = "Пользователь";

}

/\*void add\_Service() override

{

order.add\_Order();

orders.push\_back(order);

}\*/

void setPersonalData(int x, int y) override

{

EditCursor(x, y);

cout << " ";

EditCursor(x, y);

cout << "Фамилия: ";

cin >> surname;

EditCursor(x, y + 2);

cout << " ";

EditCursor(x, y + 2);

cout << "Имя: ";

cin >> name;

EditCursor(x, y + 4);

cout << " ";

EditCursor(x, y + 4);

cout << "Отчество: ";

cin >> patronymic;

EditCursor(x, y + 6);

cout << " ";

EditCursor(x, y + 6);

}

void getPersonalData(int x, int y) override

{

EditCursor(x, y);

cout << "Фамилия: " << surname;

EditCursor(x, y + 2);

cout << "Имя: " << name;

EditCursor(x, y + 4);

cout << "Отчество: " << patronymic;

}

void addPassword() override

{

this->password = InPassword();

}

void getService()

{

LineUpUgol(1);

Line(139);

LineUpUgol(3);

cout << endl;

LineUp();

cout << setw(31) << left << " Название";

LineUp();

cout << setw(31) << left << " Длительность";

LineUp();

cout << setw(62) << left << " Описание";

LineUp();

cout << setw(12) << left << " Цена";

LineUp();

cout << endl;

LineUp();

Line(139);

LineUp();

}

void getEmployee()

{

LineUpUgol(1);

Line(127);

LineUpUgol(3);

cout << endl;

LineUp();

cout << setw(31) << left << " Фамилия";

LineUp();

cout << setw(31) << left << " Имя";

LineUp();

cout << setw(62) << left << " Отчесвто";

LineUp();

cout << endl;

LineUp();

Line(127);

LineUp();

}

template<typename T>

T addOrder(vector<T> &obj)

{

int x, y;

while (true)

{

Sleep(100);

SetConsoleTextAttribute(hConsole, 15);

drawTextBox("Вернуться назад", 137, 8);

getCursorPosition(x, y);

for (typename vector<T>::iterator i = obj.begin(); i < obj.end(); i++)

{

if (isMouse(x, y, 0, distance(obj.begin(), i) \* 2 + 15, 141))

{

SetConsoleTextAttribute(hConsole, 6);

EditCursor(0, distance(obj.begin(), i) \* 2 + 15);

cout << \*i;

if (isMouseClicked())

{

SetConsoleTextAttribute(hConsole, 15);

return \*i;

}

}

else

{

SetConsoleTextAttribute(hConsole, 15);

EditCursor(0, distance(obj.begin(), i) \* 2 + 15);

cout << \*i;

}

}

if (isMouseBox(x, y, 135, 8, "Вернуться назад"))

{

SetConsoleTextAttribute(hConsole, 4);

drawTextBox("Вернуться назад", 137, 8);

SetConsoleTextAttribute(hConsole, 15);

if (isMouseClicked())

{

system("cls");

Sleep(300);

T now;

return now;

}

}

}

}

void addOrder(Service service, Employee employee, tm time)

{

order.add\_Order(service, employee, time);

}

Order getOrder()

{

return order;

}

void setOrder()

{

orders.push\_back(order);

order.delete\_orderInfo();

}

void displayOrder()

{

LineUpUgol(1);

Line(154);

LineUpUgol(3);

cout << endl;

LineUp();

cout << setw(20) << left << " Заказ";

LineUp();

cout << setw(25) << left << " Название";

LineUp();

cout << setw(10) << left << " Длительность ";

LineUp();

cout << setw(38) << left << " Описание";

LineUp();

cout << setw(10) << left << "Цена (BYN)";

LineUp();

cout << setw(15) << left << " Дата и время";

LineUp();

cout << setw(26) << left << " Мастер";

LineUp();

cout << endl;

LineUp();

Line(20);

LineUp();

cout << setw(25) << " ";

LineUp();

cout << setw(14) << " ";

LineUp();

cout << setw(38) << " ";

LineUp();

cout << setw(10) << " ";

LineUp();

cout << setw(15) << " ";

LineUp();

cout << setw(26) << " ";

LineUp();

cout << endl;

LineUp();

cout << setw(4) << left << " №";

LineUp();

cout << setw(10) << left << "Общая стоимость";

LineUp();

cout << setw(25) << " ";

LineUp();

cout << setw(14) << " ";

LineUp();

cout << setw(38) << " ";

LineUp();

cout << setw(10) << " ";

LineUp();

cout << setw(15) << " ";

LineUp();

cout << setw(26) << " ";

LineUp();

cout << endl;

LineUp();

Line(154);

LineUp();

float all\_price = 0;

cout << endl;

int j = 0;

for (vector<Order>::iterator i = orders.begin(); i != orders.end(); i++)

{

i.operator->()->display\_order(j);

all\_price += i.operator->()->getTotalPrice();

j++;

}

LineUp();

wcout << "Стоимость всех заказов: " << setw(126) << left << all\_price << left << " BYN";

LineUp();

LineUpUgol(2);

Line(154);

LineUpUgol(4);

cout << endl << endl;

}

friend ifstream& operator>>(ifstream& file, User& user)

{

file >> user.role >> user.login >> user.surname >> user.name >> user.patronymic;

return file;

}

friend ofstream& operator<<(ofstream& file, User& user)

{

file << user.login << user.getOrder() << endl;

return file;

}

};

namespace adm {

class Admin : public Account

{

private:

vector<Service> services;

public:

Admin()

{

role = "Администратор";

}

void add\_Service() // Исправить добавление и тд

{

char exit;

Service service;

do

{

service.addService();

services.push\_back(service);

cout << endl << "Для выхода нажмите клавишу ESC, для продолжения любую другую клавишу" << endl;

exit = \_getch();

} while (exit != 27);

}

void getService()

{

LineUpUgol(1);

Line(139);

LineUpUgol(3);

cout << endl;

LineUp();

cout << setw(31) << left << " Название";

LineUp();

cout << setw(31) << left << " Длительность";

LineUp();

cout << setw(62) << left << " Описание";

LineUp();

cout << setw(12) << left << " Цена";

LineUp();

cout << endl;

LineUp();

Line(139);

LineUp();

}

void addPassword() override

{

this->password = InPassword();

}

void getPersonalData(int x, int y)

{

}

};

}

**service.h:**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include <vector>

#include <memory>

#include <conio.h>

#include <Windows.h>

#include <io.h>

#include <iomanip>

#include <fcntl.h>

#include <fstream>

#include <ctime>

#include <chrono>

#include <locale>

#include <array>

#include <unordered\_map>

#include <sstream>

#include <algorithm>

#include <memory>

extern HANDLE hConsole;

extern bool isMouse(int mouseX, int mouseY, int rectX, int rectY, int rectWidth);

extern BOOL ShowConsoleCursor(BOOL bShow);

using namespace std;

extern void Line(int a);

extern void LineUp();

extern void LineUpUgol(int a);

extern void drawTextBox(const string& text, int x, int y);

extern void EditCursor(int hight, int width);

extern bool isMouseClicked();

extern void getCursorPosition(int& x, int& y);

extern bool isMouseClickInRect(int mouseX, int mouseY, int rectX, int rectY, int rectWidth, int rectHeight);

class Service

{

private:

string name\_service;

float price;

float time;

int duration;

string description;

public:

Service()

{

name\_service = "Неизвестно";

price = 0;

time = 0;

}

Service addService()

{

ShowConsoleCursor(TRUE);

cout.setf(ios::left);

cout << setw(60) << " " << "Введите название услуги: ";

getline(cin, name\_service);

bool state = true;

do {

cout << setw(60) << " " << "Введите цену за услугу(BYN): ";

try {

cin >> price;

if (cin.fail())

throw exception("Введено неверное значение");

ostringstream oss;

oss << price;

string priceString = oss.str();

size\_t dotPosition = priceString.find('.');

size\_t count = priceString.length() - dotPosition - 1;

if (count > 2 && priceString.length() != count)

throw exception("Цена должна иметь меньше двух знаков после точки.");

state = true;

cin.clear();

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

}

catch (exception& error)

{

cerr << setw(60) << " " << error.what() << endl;

cin.clear();

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

state = false;

}

} while (!state);

do {

cout << setw(60) << " " << "Введите длительность услуги(Мин): ";

try {

cin >> duration;

if (cin.fail())

throw exception("Введено неверное значение");

state = true;

cin.clear();

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

}

catch (exception& error)

{

cerr << setw(60) << " " << error.what() << endl;

cin.clear();

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

state = false;

}

} while (!state);

cout << setw(60) << " " << "Введите описание услуги: ";

getchar();

getline(cin, description);

ofstream file;

file.open("service.txt", ofstream::app);

if (!file.is\_open())

cout << "Ошибка открытия файла" << endl;

else

{

file << name\_service << ";" << price << ";" << duration << ";" << description << endl;

}

file.close();

cout.unsetf(ios::left);

ShowConsoleCursor(FALSE);

return \*this;

}

static bool compPrice(Service& service1, Service& service2)

{

return service1.getPrice() < service2.getPrice();

}

static bool compDuration(Service& service1, Service& service2)

{

return service1.getDuration() < service2.getDuration();

}

bool search(string &obj)

{

return name\_service == obj;

}

bool search(float& obj)

{

return price == obj;

}

void setPrice(float price)

{

this->price = price;

}

bool search(int& obj)

{

return duration == obj;

}

float getPrice()

{

return price;

}

int getDuration()

{

return duration;

}

void displayService()

{

cout << " " << setw(24) << name\_service;

LineUp();

wcout << " " << setw(13) << left << duration;

LineUp();

cout << " " << setw(37) << description;

LineUp();

wcout << " " << setw(9) << left << price;

LineUp();

}

friend ostream& operator<<(ostream& cout, const Service& obj)

{

cout << endl;

LineUp();

cout << " " << setw(30) << obj.name\_service;

LineUp();

cout << " " << setw(30) << obj.duration;

LineUp();

cout << " " << setw(61) << obj.description;

LineUp();

cout << " " << setw(11) << obj.price;

LineUp();

cout << endl;

return cout;

}

friend ifstream& operator>>(ifstream& file, Service& service)

{

getline(file, service.name\_service, ';');

file >> service.price;

file.ignore();

file >> service.duration;

file.ignore();

getline(file >> noskipws, service.description);

return file;

}

};

# **ПРИЛОЖЕНИЕ Б**

**(обязательное)**

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

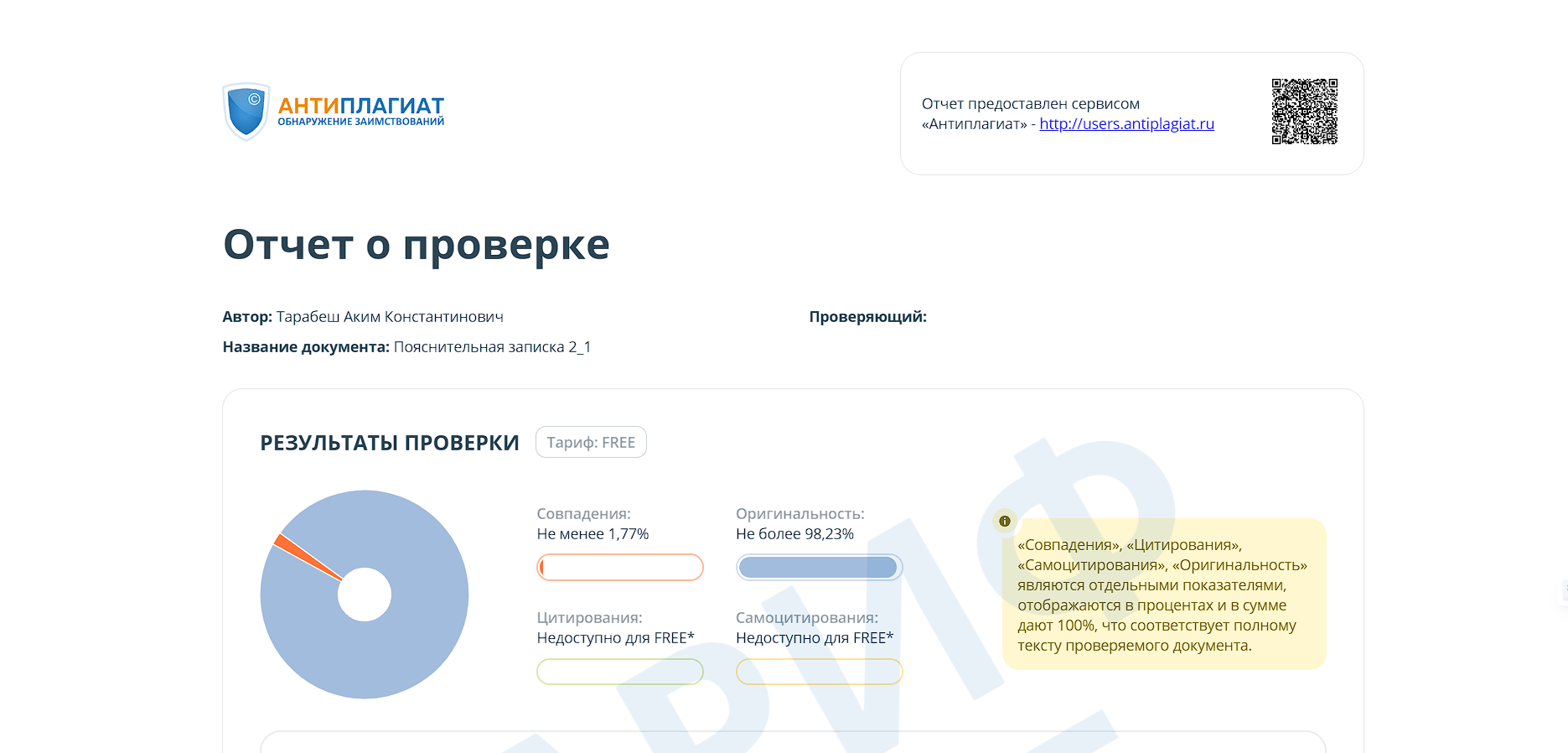


Рисунок Б.1 – Проверка на антиплагиат

# **ПРИЛОЖЕНИЕ В**

**(обязательное)**

**Диаграмма классов**

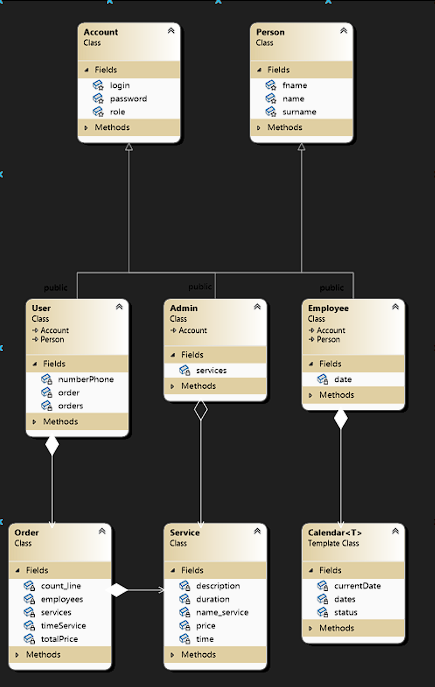


Рисунок В.1 – Диаграмма классов

# **ПРИЛОЖЕНИЕ Г**

**(обязательное)**

**Диаграмма вариантов использования**

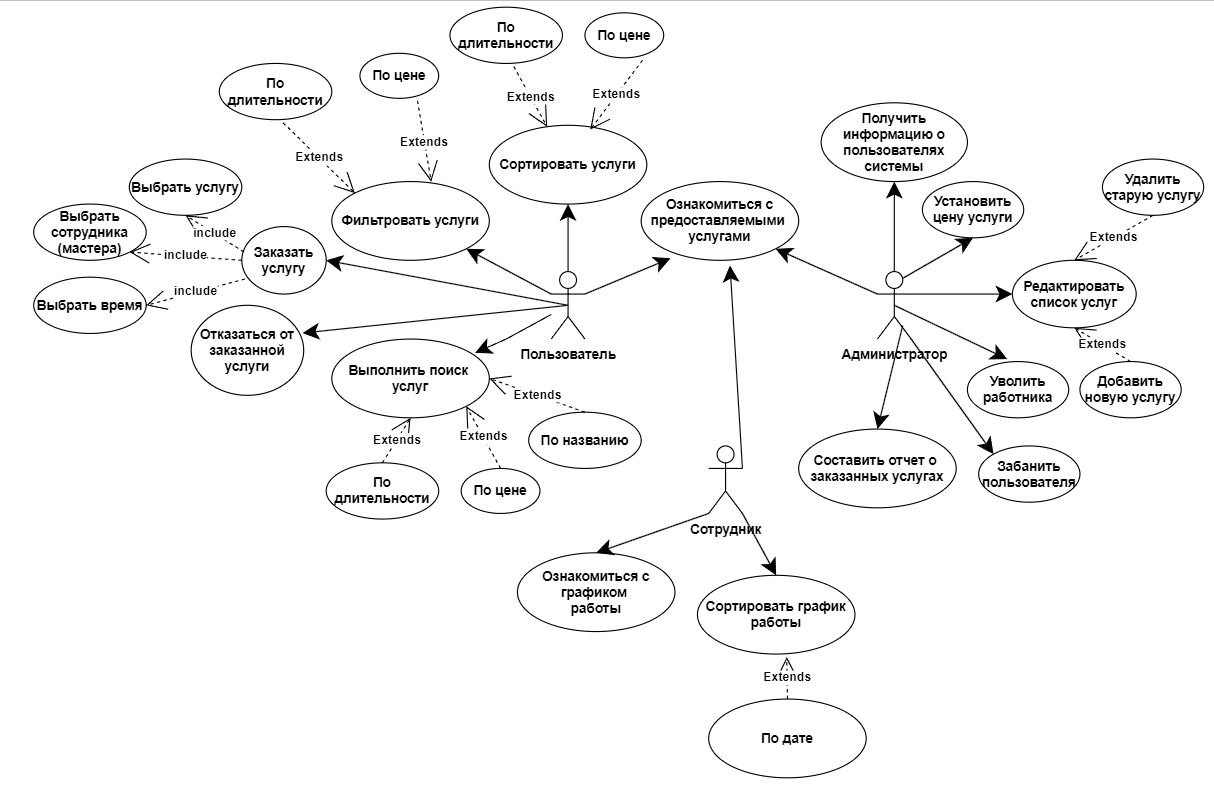


Рисунок Г.1 – Диаграмма вариантов использования

# **ПРИЛОЖЕНИЕ Д**

**(обязательное)**

**Блок-схемы алгоритмов**

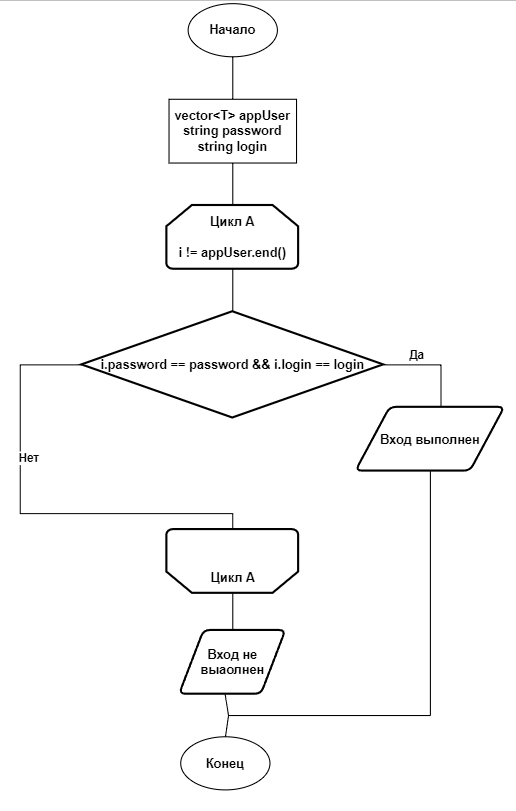


Рисунок Д.1 – Алгоритм функции SignIn

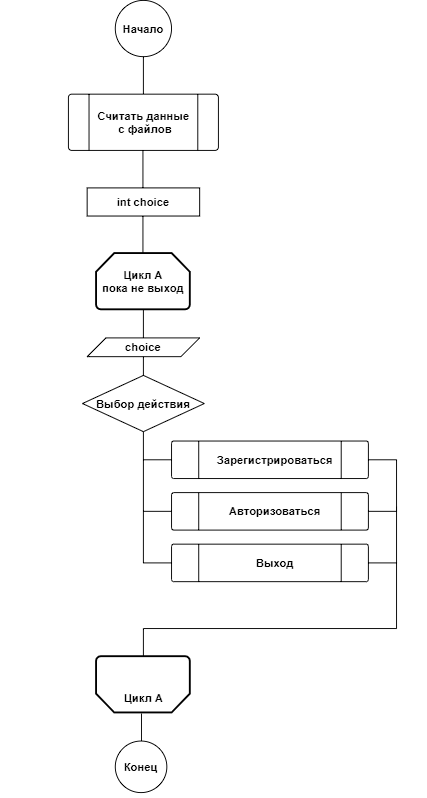


Рисунок Д.2 – Алгоритм функции main

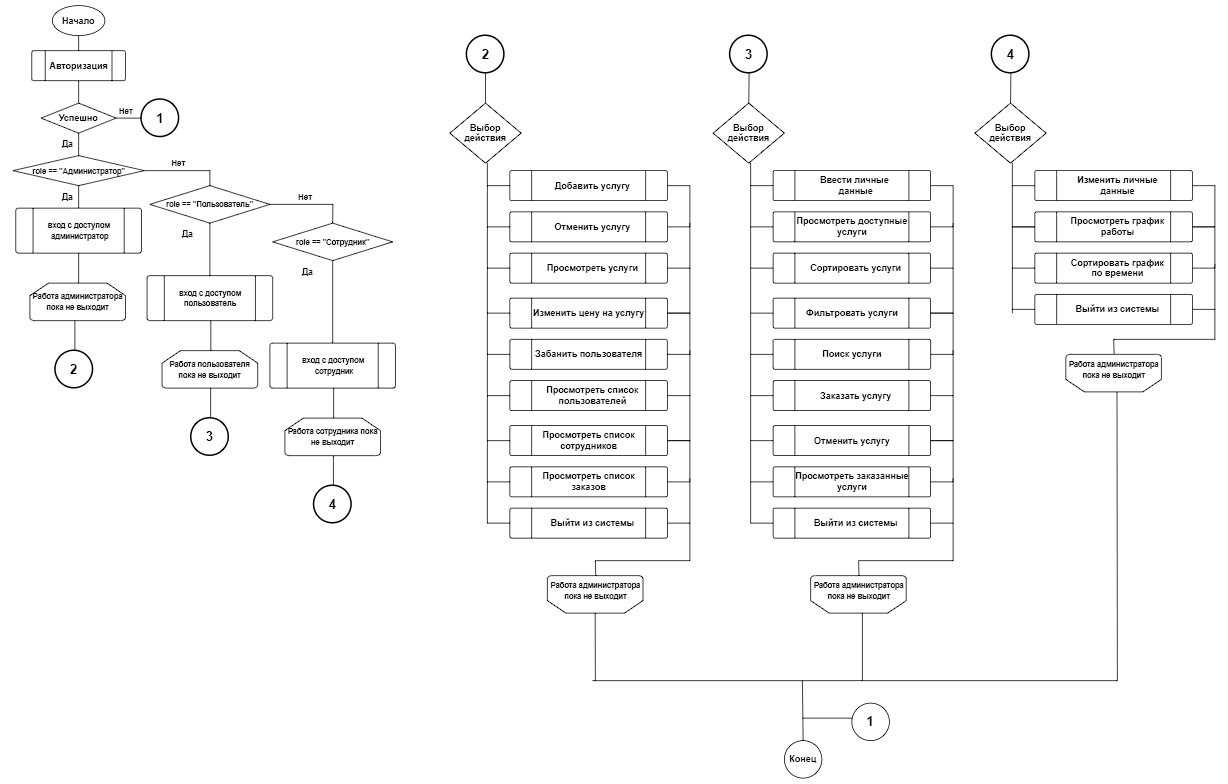


Рисунок Д.3 – Алгоритм работы всей программы