# 1. ТРЕБОВАНИЯ К ПРОГРАММЕ

В ходе курсовой работы требуется разработать программу учета переговоров абонентов сотовой связи

Оператор сотовой связи хранит информацию о разговорах своих абонентов: номер абонента, ФИО абонента, указание принадлежности вызова к исходящему или входящему, номер исходящего или входящего вызова, дата звонка, время звонка, продолжительность разговора, тариф одной минуты. Индивидуальное задание: вывести по каждому абоненту за требуемый период времени: перечень входящих и исходящих вызовов, общее время входящих вызовов, общее время исходящих вызовов, общую сумму на исходящие вызовы (требуемый период времени вводится с клавиатуры).

Общее для всех вариантов задание: реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя.

Исходные данные:

1. Тема курсовой работы: «программа учёта выплат заработной платы сотрудникам предприятия».

2. Язык программирования С++.

3. Среда разработки Microsoft Visual Studio.

4. Вид приложения – консольное.

5. Парадигма программирования – процедурная.

6. Способ организации данных – структуры (struct).

7. Способ хранения данных – файлы.

8. Каждая логически завершенная подзадача программы должна быть реализована в виде отдельной функции.

9. Построение программного кода должно соответствовать соглашению о  
коде «С++ Code Convention».

10. К защите курсовой работы представляются: консольное приложение и пояснительная записка.

11. Текст пояснительной записки оформляется в соответствии со стандартом предприятия СТП 01–2017.

Функциональные требования к разработке программы приведены ниже.

Первым этапом работы программы является авторизация – предостав-  
ление прав доступа.

Целесообразно выполнять хеширование пароля с «солью» («соль» – случайная строка, специально сгенерированная для данной учетной записи).

И «соль», и результат применения хеш-функции к паролю с «солью» хранятся в базе данных. В этом случае структрура учетных записей пользователей будет иметь вид:

- key;

- login;

- salted\_hash\_password (результат хеширования пароля с «солью»);

- salt («соль»);

- role;

- subscriberNumber (Номер абонента);

- subscriberName (ФИО абонента);

После ввода пользователем своих персональных данных (логина и пароля) и сверки со считанной из файла информацией необходимо предусмотреть возможность входа: в качестве администратора и в качестве пользователя.

Дополнительно по соображениям безопасности в форме авторизации  
целесообразно содержать пароль захешированным.

Регистрация новых пользователей осуществляется администратором в  
режиме работы с учетными записями пользователей (т. е. администратор сам создает для пользователей аккаунты).

Также выполняется проверка новых учетных записей на уни-  
кальность логина.

Вторым этапом работы программы является собственно работа с данными, которая становится доступной только после прохождения авторизации.

Для работы с данными должны быть предусмотрены два функциональных модуля: модуль администратора и модуль пользователя.

Модуль администратора включает следующие подмодули (с указанием  
функциональных возможностей):

- Управление учетными записями пользователей:

- просмотр всех учетных записей;

- добавление новой учетной записи;

- редактирование учетной записи;

- удаление учетной записи.

- Управление информацией о звонках:

- просмотр информации о звонках;

- добавление записи звонка;

- редактирование информации звонка;

- удаление записи о звонке.

Допускается создавать учетные записи для нескольких администраторов.

Модуль пользователя включает подмодуль для просмотра всех записей с информацией о звонках.

Также администратору и пользователю доступен режим обработки данных:

- выполнение индивидуального задания: вывод всех звонков за определенный период, общей суммы исходящих звонков, общего времени исходящих, входящих звонков;

- поиск данных: по ФИО, по номеру записи звонка, по номеру абонента, по номеру вызова;

- сортировка: по ФИО, по продолжительности вызова, по дате вызова.

Для реализации перечисленных модулей/подмодулей необходимо создавать меню с соответствующими пунктами.

В курсовой работе необходимо предусмотреть:

1. Обработку исключительных ситуаций:

* неправильный ввод числовой переменной;
* некорректные ввод данных при редактировании записей в векторах структур;
* проверка уникальности ввода определенных полей вектора структур: логин пользователя, номер телефон пользователя;
* отсутствие файла данных аккаунтов;
* администратор не может удалить собственную запись;
* отсутствие в файле аккаунтов логинов;
* ввод недействительного пункта меню.

2. Возможность возврата назад (навигация).

3. Запрос на выполнение необратимых действий, а именно подтверждение  
удаления вида «Вы действительно хотите удалить файл (запись)?».

4. Обратную связь с пользователем, например вывод сообщения об успешности удаления/редактирования записи.

Требования к программной реализации состоят из следующих пунктов:

1. Все переменные и константы должны иметь осмысленные имена в рамках тематики варианта курсовой работы. Переменным рекомендуется присваивать имена, состоящие из букв нижнего регистра; для формирования составного имени используется нижнее подчеркивание (например, number\_of\_userss) или «верблюжья нотация» (например, updateUsersFile). Константам рекомендуется присваивать имена, состоящие из букв верхнего регистра (например, SIZE\_ARR\_OF\_\_ACCOUNTS, FILE\_OF\_ACCOUNTS).

2. Имена функций должны быть осмысленными, начинаться с буквы нижнего регистра, строиться по принципу глагол + существительное (например,  
addAccount, findStudentBySurname). Если функция выполняет проверку и возвращает результат типа bool, то ее название должно начинаться с глагола is (например, isNumberNumeric, isLoginUnique).

3. Не допускается использование оператора прерывания goto.

4. Код не должен содержать неименованных числовых констант («магиче-  
ских» чисел), неименованных строковых констант (например, имен файлов и др.).

Подобного рода информацию следует представлять как глобальные константы. По правилам качественного стиля программирования тексты всех информационных сообщений, выводимых пользователю в ответ на его действия, также оформляются как константы.

5. Код необходимо комментировать (как минимум в части объявления  
структур, массивов/векторов, прототипов функций, нетривиальной логики).

6. Код не должен дублироваться – для этого существуют функции!

7. Одна функция решает только одну задачу (например, не допускается в  
одной функции считывать данные из файла и выводить их на консоль – это две  
разные функции). При этом внутри функции возможен вызов других функций.

8. Выполнение операций чтения/записи в файл должно быть сведено к минимуму (т. е. после однократной выгрузки данных из файла в массив/вектор дальнейшая работа ведется с этим массивом/вектором, а не происходит многократное считывание данных из файла в каждой функции).

9. Следует избегать глубокой вложенности условных и циклических кон-  
струкций: вложенность блоков должна быть не более трех.

10. Следует избегать длинных функций: текст функции должен умещаться на один экран.

11. Следует выносить код логически независимых модулей в отдельные .cpp файлы и подключать их с помощью заголовочных .h файлов.

# 2. КОНСТРУИРОВАНИЕ ПРОГРАММЫ

## 2.1 Разработка модульной структуры программы

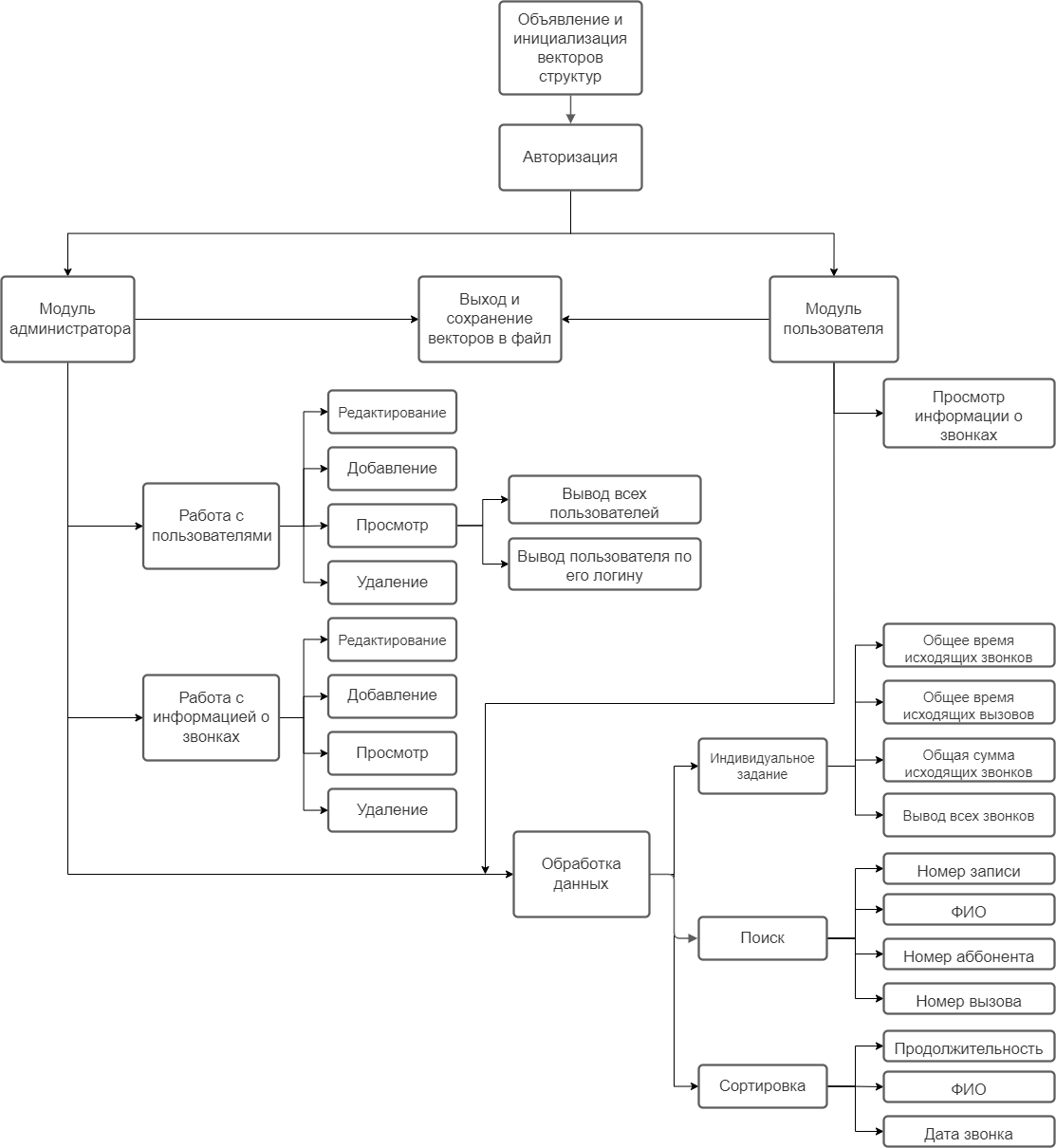
Модульная структура программы предоставлена на рисунке 2.1. 

 Рис. 2.1 — модульная структура программы.

При входе в программу, программа объявляет структуры, необходимые для дальнейшей работы и инициализирует их из файлов.

После инициализации программа предоставляет авторизацию, вход по логину и паролю.

После авторизации в программе, пользователь, в зависимости от своей роли, получает доступ к определенным функциям. Администратору предоставляется возможность работы с пользователями и информацией о звонках, в то время как пользователь имеет доступ только для просмотра информации о звонках. Обе роли также имеют доступ к меню "обработка данных", которое включает следующие функции: просмотр индивидуального задания, поиск информации о звонках и сортировку данных.

## 2.2 Выбор способа организации данных

Данные организованы в соответствующие структуры. Аккаунты пользователей хранятся в структуре User, данные о звонках в структуре CallRecords.

Структура User используется для хранения информации о аккаунтах пользователей и содержит следующие поля:

* поле key – целочисленная переменная, представляющая идентификатор записи – первичный ключ;
* поле saltedHashPassword — строка, хешированный пароль пользователя;
* поле salt - строка, соль для хеширования пароля;
* поле login — строка, логин пользователя;
* поле role — целочисленная переменная, для определения привелегий пользователя;
* поле subscriberNumber ­– целочисленная переменная, номер мобильного телефона абонента;
* поле subscriberName – строка, ФИО абонента.

Структура CallRecord используется для хранения информации о звонках и содержит такие поля, как:

* key - целочисленная переменная, представляющая идентификатор записи – первичный ключ;
* subscriberNumber - целочисленная переменная, хранящая номер абонента, внешний ключ структуры User;
* subscriberName - строковая переменная, содержащая ФИО абонента, внешний ключ структуры User;
* isOutgoingCall - логическая переменная (флаг), указывающая, является ли вызов исходящим;
* callNumber - целочисленная переменная, представляющая номер вызова;
* callDate - строковая переменная, содержащая дату звонка;
* callTime - строковая переменная, содержащая время звонка;
* callDuration - вещественная переменная, указывающая продолжительность разговора в секундах;
* tariffPerMinute - вещественная переменная, указывающая тариф на одну минуту разговора;

## 2.3 Разработка перечня пользовательских функций программы

Для работы программы предлагается разработать следующие функции:

Для чтение данных из файлов:  
 static vector<User> getUserArray(std::string filename); - создание вектора структуры <User>

static vector<CallRecord> getDataArray(std::string filename); - создание вектора структуры <CallRecord>

Для работы с учетными данными пользователей:

static void checkFile(string filename, string fileInfo); - проверка на существование файла

static void updateUsersFile(vector<User> users, const std::string& filename, string Operation); - обновление информации о пользователях

static int generateMobileNumber(); - генерация случайного номера мобильного телефона

static void createUser(const std::string& filename); - создание пользователя

static string getFullFIO(); - ввод полного ФИО пользователя

static void ChangeFIO(string filename, string oldFIO, string newFIO); - смена ФИО пользователя

static void ChangeNumber(string filename, string FIO, int newPhone, int oldPhone); - смена номера телефона пользоватея

static void updateUser(string filename); - меню редактирования пользователя

static void deleteUser(string filename, string username); - меню удаления пользователя

static void viewAllUsers(string filename); - просмотр всех пользователей

static void findForLoginViewUser(string filename); - поиск и вывод информации о пользователе по его логину  
 Для работы с информацией о звонках:

static double getCallTariff(const std::string dateTimeString); - получение тарифа текущего звонка

static void createData(const std::string& filename); - создание записи о звонке

static void viewData(const std::string& filename); - просмотр всех звонков

static bool isValidDate(int day, int month, int year); - проверка корректности ввода даты

static string inputDate(); - ввод даты

static string inputTime(); - ввод времени

static void updateCallRecord(string filename); - меню обновление информации о звонках

static void updateDataFile(vector<CallRecord> data, const std::string& filename, string Operation); - обновление файла с информации о звонках

static void deleteCallRecord(string filename); - удаление информации о звонках  
 Для хеширование паролей:

static string hashPasswordWithSaltMethod(const std::string& password, const std::string& salt) – генерация пароля с «солью»

static string generate\_salt(int length) – генерация «соли»

Для индивидуального задания:

static void Search(); - поиск по списку звонков

static void IndividualTasksInterface(); - меню индивидуального задания

static void Sort(); - меню для сортировки данных

# 3. Разработка алгоритмов работы программы

## 3.1 Алгоритм функции main

Алгоритм функции main представлен на рисунке 3.1.

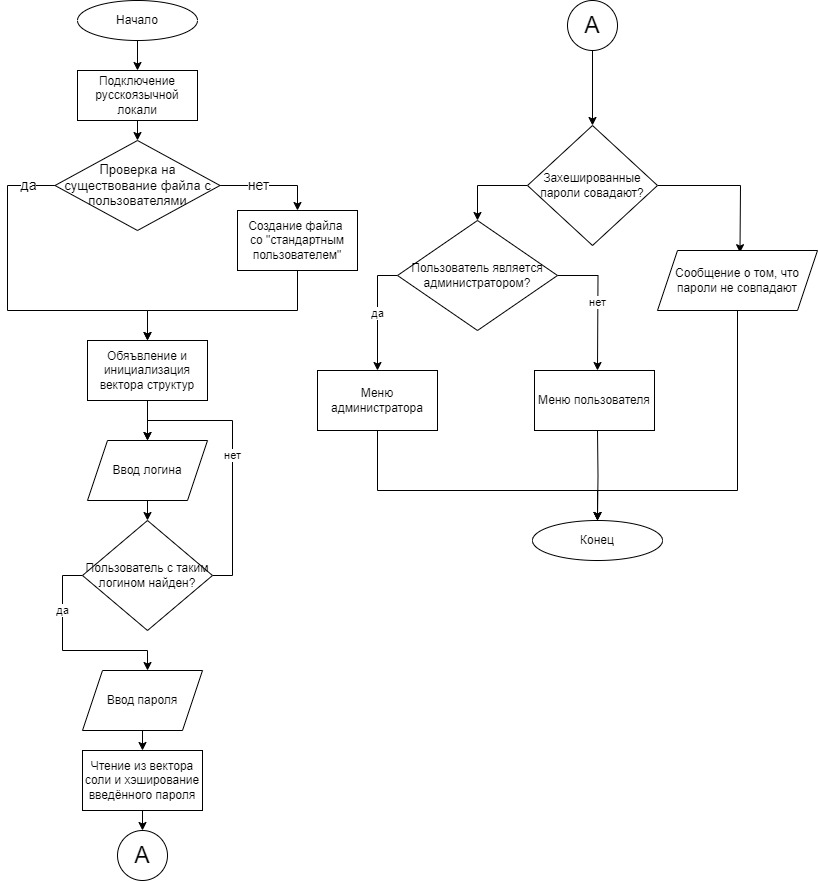


Рисунок 3.1 — алгоритм функции main.

Функция main предназначена для объявления структур данных и вызова остальных функций. Её алгоритм работы включает в себя:

1) подключение русскоязычной локалии;

2) объявление и инициализация вектора структур;

3) авторизация пользователя;

4) переход в меню пользователя или администратора.

## 3.2 Алгоритм функции updateUser

Алгоритм функции updateUser предоставлен на рисунке 3.2.

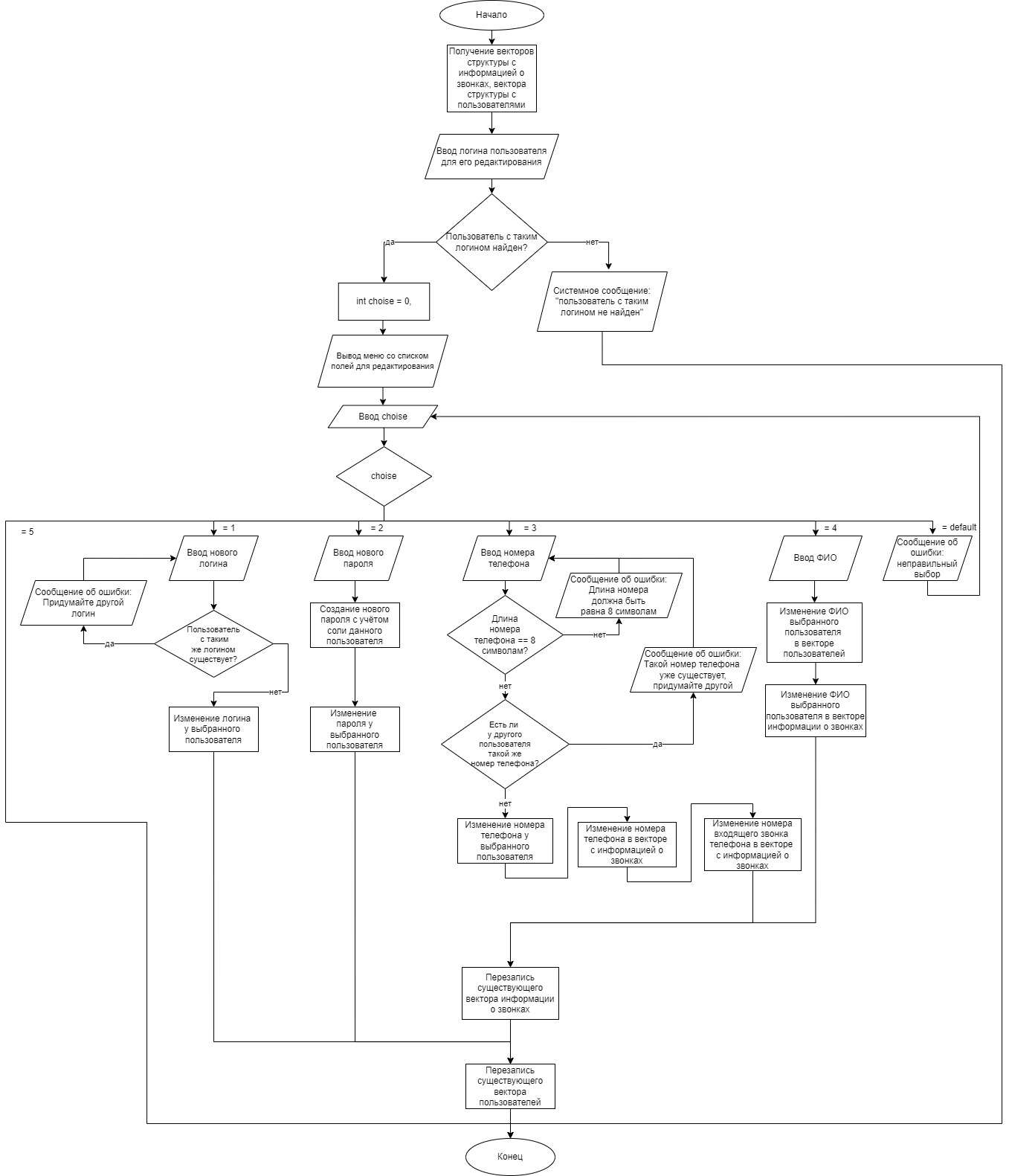


Рисунок 3.2 ­­­– алгоритм функции updateUser

Исходя из требований к программе, в ней необходимо предусмотреть редактирование учетной записи пользователя по логину. Исходя из этого, необходимо разработать функцию, которая бы проверяла логин пользователя на существование в векторе пользователей и давала возможность администратору для редактирование его: ФИО, номера телефона, логина и пароля.

Алгоритм работы функции updateUser включает в себя:

1. Ввод логина пользователя для поиска;
2. поиск пользователя с указанным логином в векторе пользователей;
3. если пользователь найден, предоставление возможности редактирования различных параметров;
4. для каждого параметра производится валидация и соответствующие изменения в векторе пользователей;
5. при редактировании номера телефона или ФИО происходят соответствующие изменения в файле с информацией о звонках.;
6. запись обновленных данных в файл с пользователями;
7. вывод сообщения об ошибке, если указанный логин не найден в системе.

Общий смысл функции updateUser заключается в том, что она позволяет удобно и безопасно администратору системы редактировать учетные записи пользователей. При редактировании логина, пароля, номера телефона или ФИО происходят соответствующие изменения в векторе пользователей, а также в файле с информацией о звонках.

## 3.3 Алгоритм функции individualTasksInterface

Исходя из требований к программе, в ней необходимо предусмотреть выполнение индивидуального задания: вывести по каждому абоненту за требуемый период времени: перечень входящих и исходящих вызовов, общее время входящих вызовов, общее время исходящих вызовов, общую сумму на исходящие вызовы (требуемый период времени вводится с клавиатуры).

Исходя из этого, необходимо разработать функцию, которая бы выводила информацию за введенный пользователем период времени по звонкам, вывести общее время исходящих и входящих звонков, а также возможность администратору для редактирование его: ФИО, номера телефона, логина и пароля.

Алгоритм работы функции individualTasksInterface включает в себя:

1. ввод начальной даты для фильтрации;
2. ввод конечной даты для фильтрации;
3. проверка на корректность обе даты;
4. поиск записей с информацией о звонках в диапазоне между двух дат;
5. если совпадения по фильтру найдены – вывод меню с выбором опции: вывода всех звонков в заданном промежутке дат, общего времени входящих и исходящих звонков, общей суммы входящих зконков
6. иначе выводит сообщение о том, что в заданном диапазоне не найдено ни одной записи

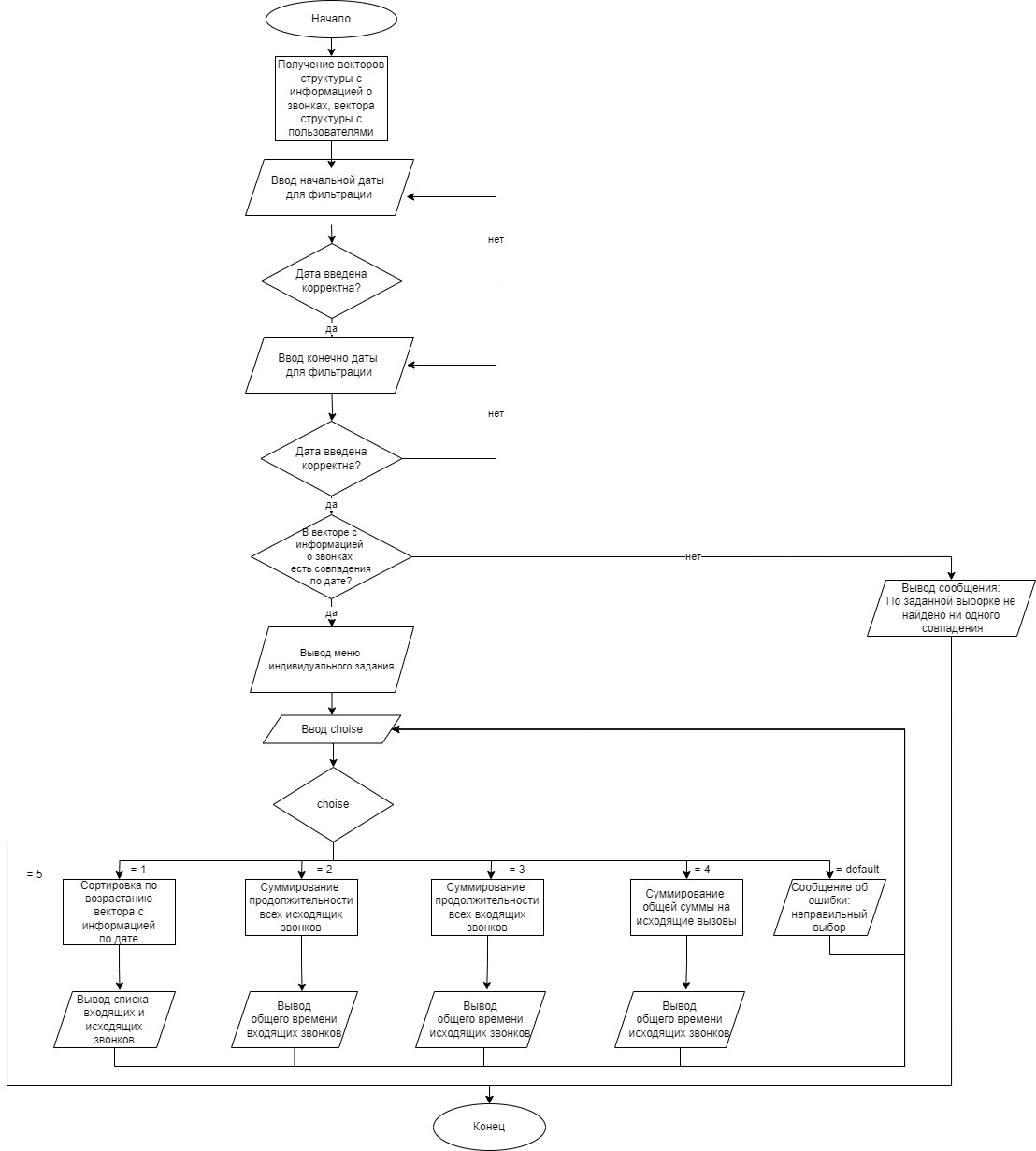
Алгоритм данной функции предоставлен на рисунке 3.3

Рисунок 3.3 — алгоритм функции initValues

Таким образом общий смысл функции заключается в том, что она позволяет пользователю фильтровать и анализировать данные о звонках, основываясь на заданных временных интервалах, и предоставляет опций для выполнения задач индивидуального задания.

# 4. ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

## 4.1 Авторизация

Процесс авторизации состоит из ввода логина и пароля пользователя. После этого программа проверяет, есть ли такой логин. Если логин найден, то программа просит ввести пароль. При правильном вводе пароля программа проверяет роль учетной записи и выводит на экран соответствующее меню администратора или пользователя.

Важно отметить, что для повышения безопасности пароль хэшируется с использованием соли. Хэширование пароля с солью является одним из основных методов защиты паролей от несанкционированного доступа. При хэшировании пароля с солью, случайная уникальная строка, называемая солью, добавляется к паролю перед хэшированием. При процессе авторизации, введенный пользователем пароль также хэшируется с использованием той же соли, и затем полученный хэшированием пароль сравнивается с сохраненным паролем в файле. Если пароли совпадают, то пароль считается правильным, и пользователю предоставляется доступ к соответствующему меню.

На рисунках 4.1, 4.2 и 4.3 представлен процесс авторизации.

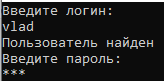


Рисунок 4.1 — авторизация в программу.

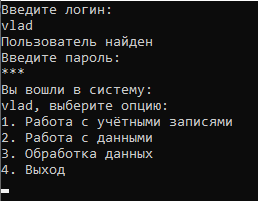


Рисунок 4.2 — вход под администратором.

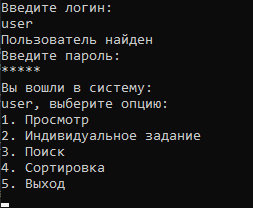


Рисунок 4.3 — вход под пользователем.

## 4.2 Модуль администратора

В модуле администратора имеются функции управления учётными записями, записями о звонках, а также работы с данными. Для доступа к пунктам меню используется ввод соответствующих цифр с клавиатуры.

С аккаунтами можно выполнять такие операции, как:

- добавление нового пользователя;

- удалить существующего пользователя;

- редактировать существующего пользователя;

- просмотреть всех пользователей.

В качестве примера добавим новый аккаунт в файл аккаунтов (рис. 4.4).

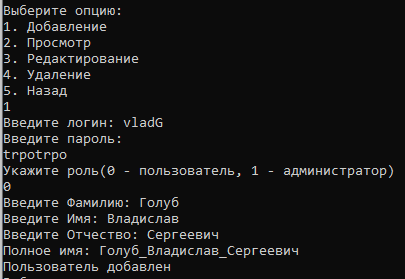


Рисунок 4.4 — Процесс создания нового аккаунта.

Новый пользователь сразу сохраниться в файл после добавления его в вектор пользователей. На рисунке 4.5 представлена работа функции вывода информации о пользователе по его логину.

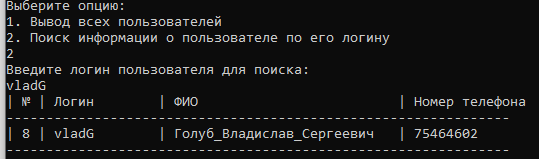


Рисунок 4.5 — Вывод информации о пользователе.

В программе предусмотрен вывод информации о всех пользователях. На рисунке 4.6 представлена работа функции вывода информации о всех пользователях.

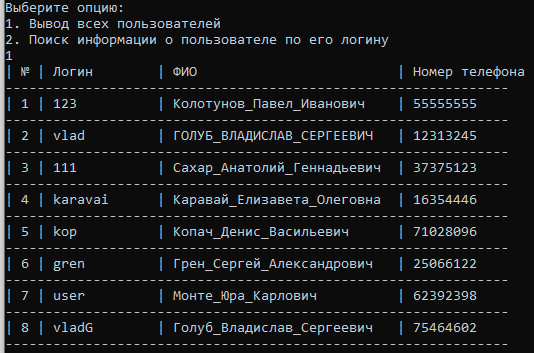


Рисунок 4.6 — Вывод информации о всех пользователях.

В программе предусмотрена возможность редактирования данных пользователей: логин, пароль, номер телефона и ФИО. При изменении ФИО или номера телефона пользователя, их найденные совпадения в файле с информацией о звонках также будут изменены на новые значения. Также при редактировании логина и номера телефона, новые значения проверяются на уникальность

На рисунке 4.7 представлен процесс редактирования логина пользователя.

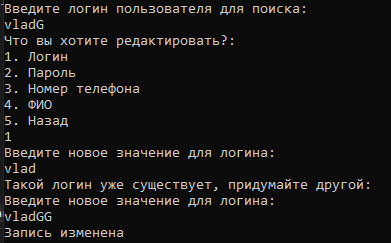


Рисунок 4.7 — Вывод информации о всех пользователях.

На рисунке 4.8 представлен процесс удаления существующего пользователя,

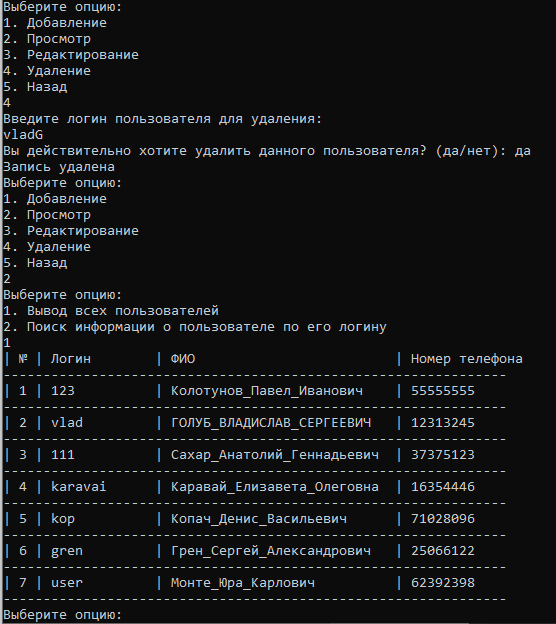


Рисунок 4.8 — Процесс удаления пользователя.

С данными о звонках можно выполнять такие операции, как:

- добавление новой записи о звонке;

- удаление существующей записи о звонке;

- редактирование существующей записи о звонке;

- просмотр всех звонков.

В качестве примера добавим, отредактируем и удалим запись о звонке.

На рисунке 4.9 представлен процесс добавления записи о звонке.

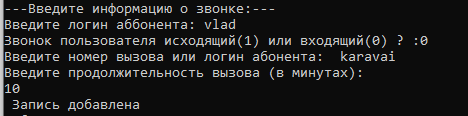


Рисунок 4.9 — Процесс добавления пользователя.

На рисунке 4.10 представлен вывод всех звонков.

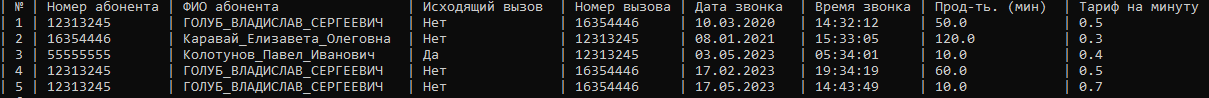


Рисунок 4.10 — Вывод всех звонков.

На рисунке 4.11 представлен процесс редактирования продолжительности звонка

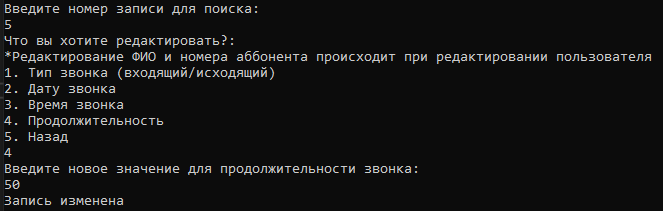


Рисунок 4.11 — Редактирование продолжительности звонка

На рисунке 4.12 представлен процесс редактирования даты звонка

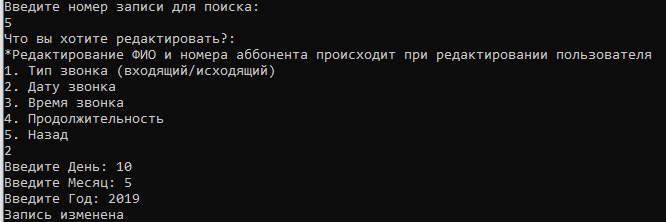


Рисунок 4.12 — Редактирование даты звонка

На рисунке 4.13 представлен вывод всех звонков, в том числе отредактированная запись

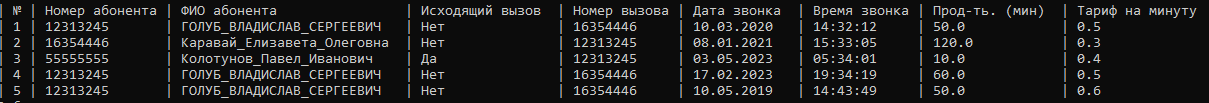


Рисунок 4.13 — Редактирование даты звонка

На рисунке 4.14 представлен процесс удаления пользователя.

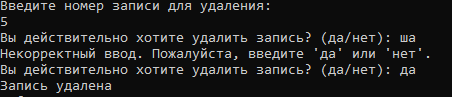


Рисунок 4.14 — Удаление записи о звонке

## 4.3 Модуль пользователя

В модуле пользователя представлена возможность работы с данными работников, доступны такие функции, как:

- просмотр всех записей с информацией по звонкам;

- выполнение индивидуального задания;

- поиск записи по: номеру записи, ФИО, номеру абонента или номеру звонка;

- сортировка по: ФИО абонента, дате звонка, продолжительности звонка.

Данный функционал доступен как пользователю, так и администратору.

На рисунке 4.15 представлено главное меню пользователя

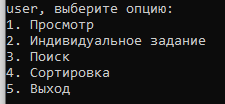


Рисунок 4.15 — главное меню пользователя

На рисунках 4.16 и 4.17 представлен процесс сортировки по продолжительности звонка и по дате звонка.

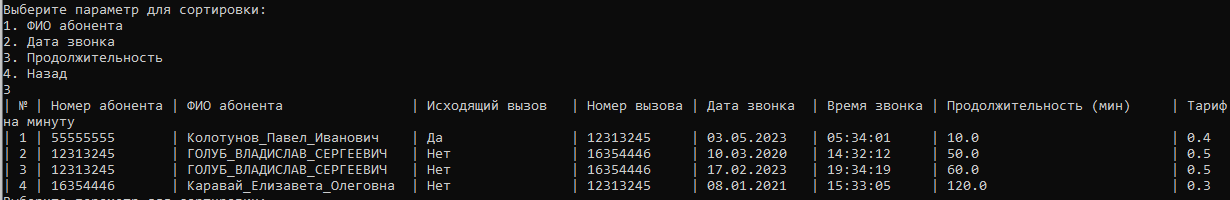


Рисунок 4.16 — сортировка по продолжительности звонка.

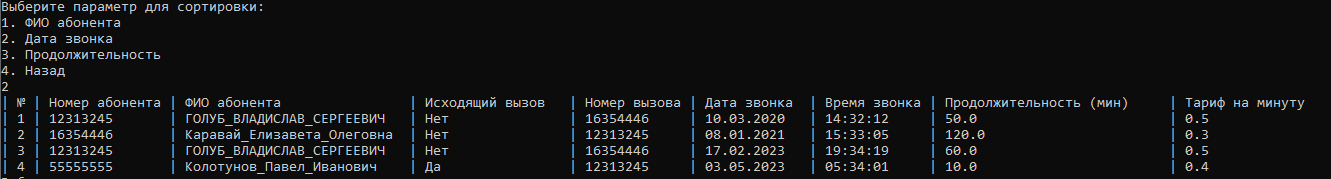


Рисунок 4.17 — сортировка по дате звонка

На рисунке 4.18 представлено меню индивидуального задания: вывести по каждому абоненту за требуемый период времени: перечень входящих и исходящих вызовов, общее время входящих вызовов, общее время исходящих вызовов, общую сумму на исходящие вызовы (требуемый период времени вводится с клавиатуры).

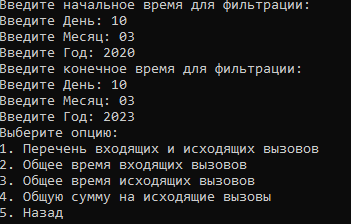


Рисунок 4.18 — меню индивидуального задания.

На рисунке 4.19 представлен пример выполнения индивидуального задания: вывод перечня входящих и исходящих звонков.

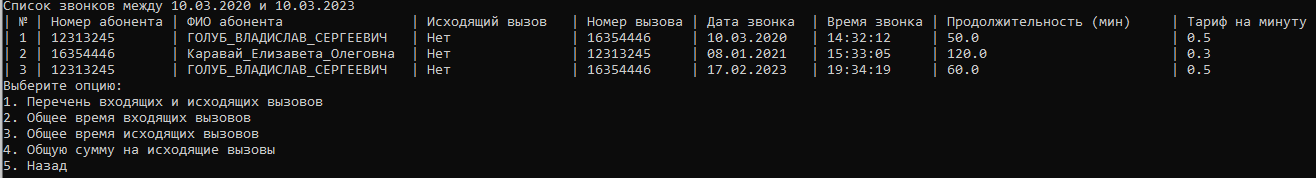


Рисунок 4.19 — перечень входящих и исходящих звонков

На рисунках 4.20 и 4.21 представлен вывод общего времени на исходящие и входящие вызовы.



Рисунок 4.20 — перечень входящих и исходящих звонков



Рисунок 4.21 — перечень входящих и исходящих звонков

На рисунке 4.22 представлена общая сумма на исходящие вызовы.



Рисунок 4.22 — перечень входящих и исходящих звонков

На рисунке 4.23 представлено меню поиска информация о звонках.

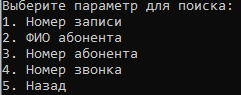


Рисунок 4.23 — меню поиска информации о звонках

Поиск может осуществляться по следующим параметрам: номер записи, ФИО и номер абонента, а также номер звонка. На рисунках 4.24 и 4.25 представлены поиски по номеру и ФИО абонента.

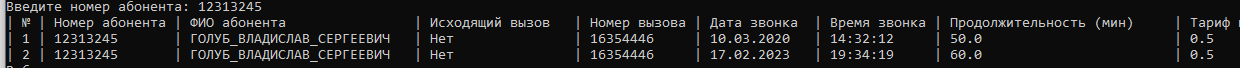


Рисунок 4.24 — поиск по номеру абонента

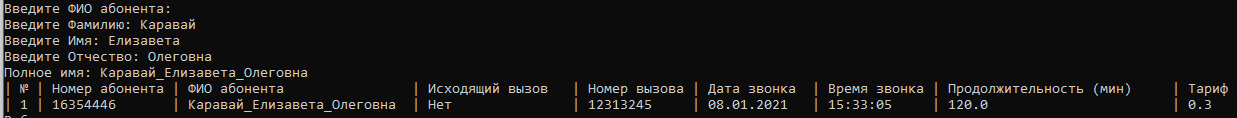


Рисунок 4.25 — поиск по ФИО абонента

## 4.4 Исключительные ситуации

В ходе выполнения и работы программы могут возникать ситуации, когда пользователь действует не совсем правильно, например, вводит в числовые переменные буквы. Для таких случаев предусмотрены обработки исключительных ситуаций.

В числе исключительных ситуаций находятся:

- неправильный ввод числовой переменной;

- некорректные ввод данных при редактировании записей в векторах структур;

- проверка уникальности ввода определенных полей вектора структур: логин пользователя, номер телефон пользователя;

- отсутствие файла данных аккаунтов;

- администратор не может удалить собственную запись;

- отсутствие в файле аккаунтов логинов;

- ввод недействительного пункта меню.

Для того, чтобы программа корректно продолжала свою работу, необходимо данные ситуации обработать.

В случае неправильного ввода числовой переменной предусмотрена функция, проверяющая вводимые с клавиатуры данные, и если в них присутствует что-то кроме цифр, то потребуется ввести данные ещё раз.

В случае отсутствия файлов данных предусмотрена обработка, которая создаст недостающие файлы.

В случае отсутствия аккаунтов в файле, программа внесёт коррективы и добавит первоначальный логин admin/admin.

При вводе недействительного пункта меню программа проигнорирует ввод, и будет ожидать правильного ввода.

В качестве примера приведена ситуация, когда файл аккаунтов пуст, и создаётся первый аккаунт (рис. 4.26).

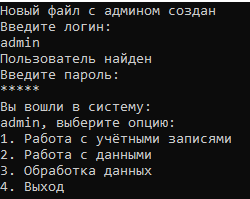
****

Рисунок 4.26 – создание первоначального логина admin/admin.

Как можно заметить, программа сообщила, что создала новый файл с аккаунтом администратора, потому что в системе отсутствовал файл с пользователями и создала запись с логином admin и паролем admin, под которым можно войти.

На рисунке 4.27 предоставлен пример инициализации исключительного события ввода неправильного пункта меню.

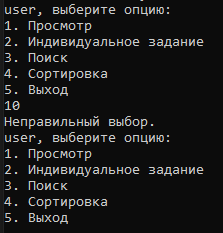


Рисунок 4.27 – создание первоначального логина admin/admin.

На рисунке 4.28 предоставлен пример проверки уникальности и корректности при редактировании номера телефона пользователя.

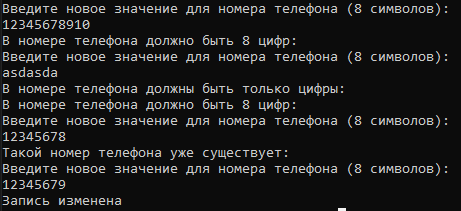


Рисунок 4.28 – проверки уникальности и корректности номера телефона

На рисунке 4.29 предоставлен пример исключительной ситуации, когда администратор пытается удалить учётную запись, под которой он авторизовался



Рисунок 4.29 – попытка удалить собственную учётную запись