Министерство образования Республики Беларусь  
Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования  
Кафедра инженерной психологии и эргономики  
Дисциплина: Основы конструирования программ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе  
на тему  
РАЗРАБОТКА ПРОГРАММЫ УЧЕТА ЗАРАБОТНОЙ ПЛАТЫ СОТРУДНИКОВ

Выполнил: студент группы 113871

Жур Матвей Дмитриевич  
Проверил:

Рыбак Виктор Александрович

Минск 2022

*Белорусский государственный университет информатики и радиоэлектроники* *Факультет компьютерного проектирования*

«УТВЕРЖДАЮ»  
Заведующий кафедрой ПОИТ  
\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Лапицкая  
«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 года

ЗАДАНИЕ  
по курсовой работе  
студенту Журу М.Д.

1. Требования к программе .  
2. 2. Конструирование программы.  
2.1 Разработка модульной структуры программы .  
2.2 Выбор способа организации данных .  
2.3 Разработка перечня пользовательских функций программы.  
3. Разработка алгоритмов работы программы .  
3.1 Алгоритм функции main.  
3.2 Алгоритм функции ... (выбирается из перечня пользовательских  
функций).  
3.3 Алгоритм функции ... (выбирается из перечня пользовательских  
функций).  
4. Описание работы программы .  
4.1 Авторизация.  
4.2 Модуль администратора.  
4.3 Модуль пользователя.  
4.4 Исключительные ситуации.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.А. Рыбак  
Задание принял для исполнения \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.Д. Жур

**Содержание**

1. Требования к программе………………………………........…........…....….…..…  
2. 2. Конструирование программы………………………..…........…........…........….  
2.1 Разработка модульной структуры программы……….…........…........…........….  
2.2 Выбор способа организации данных……………………........…........…........….  
2.3 Разработка перечня пользовательских функций программы.…........…........….  
3. Разработка алгоритмов работы программы…………...…........…........…........….  
3.1 Алгоритм функции main…………………………………........…........…........….  
3.2 Алгоритм функции accessLogin…........…........…........…........…........…........….  
3.3 Алгоритм функции initValues…........…........…........…........…........…........…….  
4. Описание работы программы.…........…........…........…........…........…........…..…  
4.1 Авторизация.…........…........…........…........…........…........…........…........…........  
4.2 Модуль администратора.…........…........…........…........…........…........…........….  
4.3 Модуль пользователя.…........…........…........…........…........…........…........…..…  
4.4 Исключительные ситуации.…........…........…........…........…........…........…........

**1. Требования к программе**

В ходе курсовой работы требуется разработать программу учета выплат заработной платы сотрудникам предприятия.

Информация о сотрудниках предприятия содержит: ФИО сотрудника, название отдела, должность, размер заработной платы за месяц.

Индивидуальное задание: вычислить общую сумму выплат за месяц по каждому отделу, а также среднемесячный заработок сотрудников по каждому отделу.

Вывести список сотрудников, у которых зарплата ниже введенной 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.

В рамках данного этапа необходимо считать данные из файла с учетными записями пользователей следующего вида:

login;

password;

role (данное поле служит для разделения в правах администраторов и пользователей).

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).

Рекомендуется: выполнять проверку новых учетных записей на уни-  
кальность логина.

Вторым этапом работы программы является собственно работа с данными, которая становится доступной только после прохождения авторизации. Данные хранятся в отдельном файле и имеют вид, описанный подробно в каждом варианте к курсовой работе.

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.

Допускается создавать учетные записи для нескольких администраторов.

Работа с данными:

а) режим редактирования:

просмотр всех данных;

добавление новой записи;

удаление записи;

редактирование записи;

б) режим обработки данных:

выполнение индивидуального задания;

поиск данных (как минимум по трем различным параметрам);

сортировка (как минимум по трем различным параметрам).

Модуль пользователя включает подмодуль работы с данными со следую-  
щими функциональными возможностями:

просмотр всех данных;

выполнение индивидуального задания;

поиск данных (как минимум по трем различным параметрам);

сортировка (как минимум по трем различным параметрам).

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

В курсовой работе необходимо предусмотреть:

1. Обработку исключительных ситуаций:

введенные пользователем данные не соответствуют формату поля  
(например, символы в числовом поле);

введенные пользователем данные нелогичны (например, отрицательная  
цена товара);

файл с данными для чтения не существует;

ничего не найдено по результатам поиска;

номер удаляемой записи выходит за пределы массива/вектора.

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 и т. д.

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\_students) или «верблюжья нотация» (например, flagExit). 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.

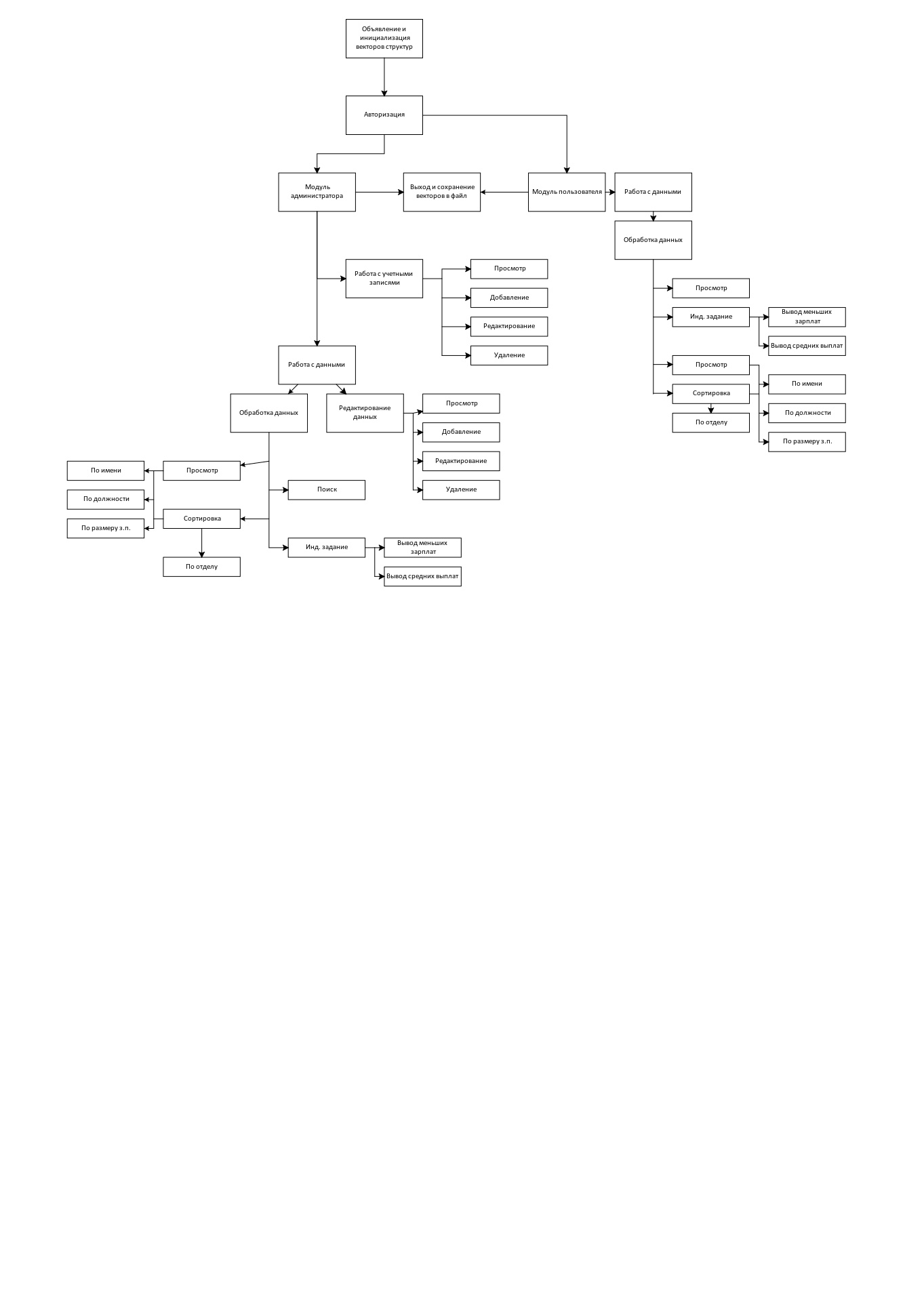
****

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 зависимости с какими привелегиями пользователь вошёл в программу, следует дальнейшая работа с данными, а при авторизации администратора, с аккаунтами.

**2.2 Выбор способа организации данных**

Данные организованы в соответствующие структуры. Акканунты хранятся в структуре UserAccounts, данные в структурах Values и working\_wage.

Структура UserAccounts используется для аккаунтов пользователей и содержит следующие поля:

- поле password — строка, пароль пользователя;

- поле login — строка, логин пользователя;

- поле role — целочисленная переменная, для определения привелегий аккаунта.

Структура Values используется для хранения информации и о работниках содержит такие поля, как:

- worker\_name — строка, имя работника;

- worker\_pos — строка, должность работника;

- montly\_wage — целочисленная переменная, средняя заработная плата работника;

- worker\_dep — целочисленная переменная, отдел работника.

Структура working\_wage используется для хранения заработных плат по месяцам и содержит поля:

- worker\_name — строка, имя работника;

- one\_month\_wage — целочисленный вектор, заработная плата работника за месяц.

Для объединения структур используются вектора, объявляемые в функции main. Дальнейшая работа с векторами ведётся с помощью указателей.

**2.3 Разработка перечня пользовательских функций программы**

Для работы программы предлагается разработать следующие функции:

Для проверки ввода:

int setIntValue(); - установка значений целочисленных переменных;

Для работы с аккаунтами:

void addAccount(std::vector<acc::UserAccounts>\*); - создание нового акканута;

void deleteAccount(std::vector<acc::UserAccounts>\*); - удаление аккаунта;

void editAccount(std::vector<acc::UserAccounts>\*); изменение аккаунта;

void openAccount(std::vector<acc::UserAccounts>\*); просмотр списка аккаунтов;

void saveAccountsToFile(std::vector<acc::UserAccounts>); - сохранение аккаунтов в файл.

Для работы с данными о заработных платах:

int averageWorkWage(std::vector<val::working\_wage>\*, std::string); - расчёт средней заработной платы;

void setWorkWage(std::vector<val::working\_wage>\*, std::string); - установка заработной платы для новых работников;

void openWorkWage(std::vector<val::working\_wage>\*, std::string); - просмотр заработной платы работника;

void changeWorkWage(std::vector<val::working\_wage>\*, std::string, std::string); - изменение заработной платы работника;

void deleteWorkWage(std::vector<val::working\_wage>\*,std::string); - удаление заработных плат работника.

Для работы с данными о работниках:

void addValue(std::vector<val::Values>\*); - добавить данные о работнике;

void deleteValue(std::vector<val::Values>\*); - удалить данные о работнике;

void editValue(std::vector<val::Values>\*); - изменить данные о работнике;

void openValue(std::vector<val::Values>\*); - просмотреть список работников;

void saveValuesToFile(std::vector<val::Values>); - сохранить данные о работниках в файл;

void saveWagesToFile(std::vector<val::working\_wage>); - сохранить заработные платы по месяцам в файл;

void set\_work\_wage\_pointer\_values(std::vector<val::working\_wage>\*); - установить указатель на структуру заработных плат;

std::vector<val::working\_wage>\* get\_work\_wage\_pointer\_values(); - получить указатель на структуру заработных плат.

Для сортировки данных о работниках:

void valuesSort(std::vector<val::Values>\* , int); - функция сортировки.

Для поиска данных о работниках:

void findName (std::vector<val::Values>\*); - для поиска по имени работника;

void findDep(std::vector<val::Values>\*); - для поиска по отделу работника;

void findWage(std::vector<val::Values>\*); - для поиска по заработной плате работника.

Для выполнения индивидуального задания:

void allWageInDepartment(std::vector<val::Values>\*); - расчет средних заработных выплат;

void lowWageWorkers(std::vector<val::Values>\*); - просмотр данных о сотрудниках с наименьшей заработной платой.

Для первоначальной инициализации структур:

std::vector<val::working\_wage> initOneMonthWage(); - считывание и инициализация структуры заработных плат;

std::vector<acc::UserAccounts> initSize(); - считывание и инициализация структуры аккаунтов;

std::vector<val::Values> initValues(); - считывание и инициализация структуры работников.

Для проверки входа:

std::vector<acc::UserAccounts>::iterator login\_input(std::vector<acc::UserAccounts>\*); - для проверки логина;

std::vector<acc::UserAccounts>::iterator password\_input(std::vector <acc::UserAccounts> ::iterator); - для проверки пароля.

Для доступа к меню и организацию режима работы:

bool adminMenu(std::vector<acc::UserAccounts>\*); - для доступа к меню администратора;

bool userMenu(std::vector<val::Values>\*); - для доступа к меню пользователя;

bool logon(std::vector<acc::UserAccounts>::iterator,std::vector<acc:: UserAccounts>\*, std::vector<val::Values>\*); - проверка привелегий и вывод функций соответствующего меню.

Для шифрования пароля предлагается общедоступная библиотека хеширования md5 и соответствующая функция:

MD5(const std::string& text); - для хеширования пароля.

Для вывода текстовой информации:

void showAdminMenu(int); - вывод меню администратора;

void showUserMenu(); - вывод меню пользователя;

void showSortMenu(); - вывод меню сортировки;

void showFindMenu(); - вывод меню поиска;

void showLoginMenu(); - вывод меню авторизации;

void showIndividualMenu(); - вывод меню индивидуального задания.

**3. Разработка алгоритмов работы программы**

**3.1 Алгоритм функции** **main**

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

1) объявление структур данных;

2) инициализация структур данных;

3) установка указателей;

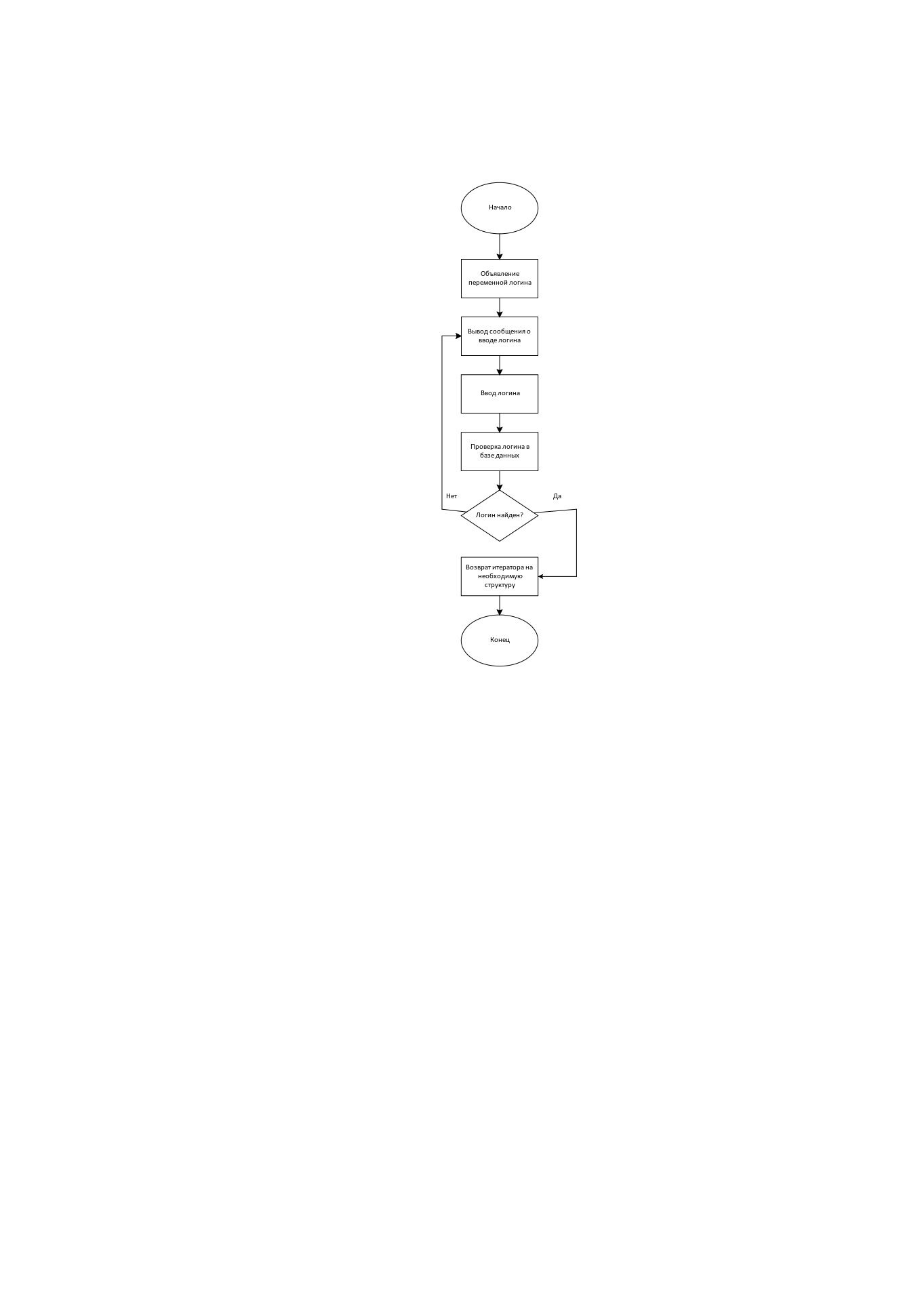
4) вызов функции авторизации пока пользователь не захочет выйти;

5) сохранение структур в файлы.

Алгоритм функции main представлен на рисунке 3.1.

Рисунок 3.1 — алгоритм функции main.

**3.2 Алгоритм функции login\_input**

****

Алгоритм функции login\_input предоставлен на рисунке 3.2.

Рисунок 3.2 - алгоритм функции login\_input

Исходя из требований к программе, в ней необходимо предусмотреть авторизацию по логину и паролю для разграничения прав доступа администраторов и пользователей. Исходя из этого, необходимо разработать функцию, которая бы проверяла логин на существование в базе аккаунтов.

Данная функция предназначена для поиска логина, вводимого с клавиатуры пользователем, в структуре аккаунтов. В случае, если логин найден, данная функция возвращает итератор на данную структуру.

**3.3 Алгоритм функции initValues**

Исходя из требований к программе, в ней необходимо предусмотреть сохранение и открытие информации из файла. Для открытия файла структур работников разработана функция инициализации initValues.

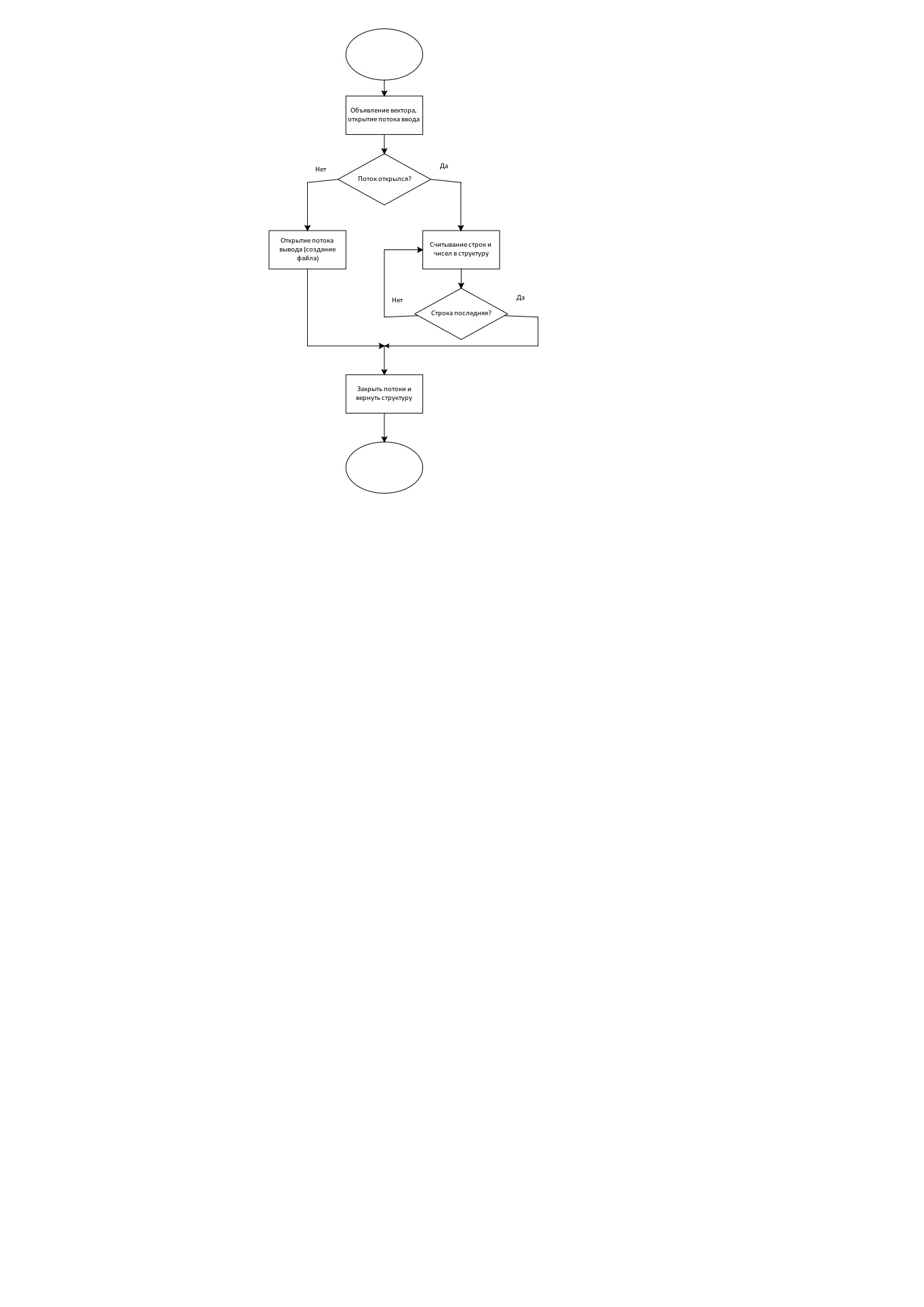
Алгоритм данной функции предоставлен на рисунке 3.3.

Рисунок 3.3 — алгоритм функции initValues

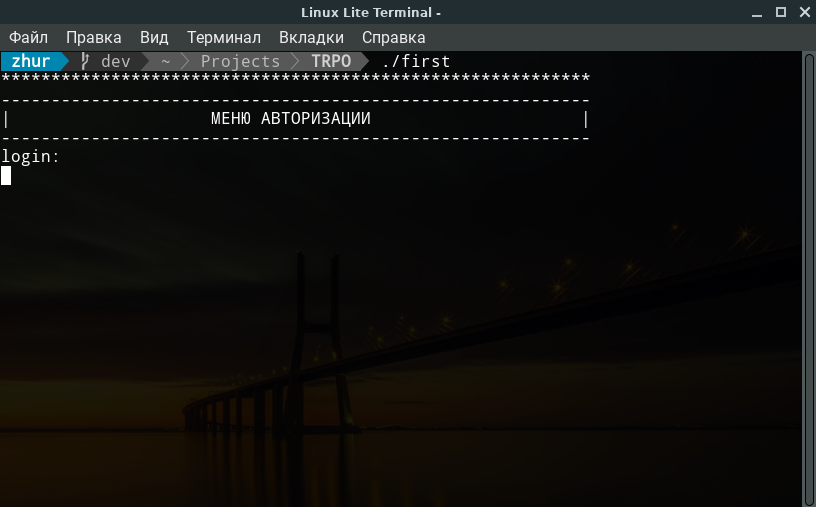
В ходе этой функции программа считывает информацию из файла, затем записывает информацию в структуру программы. Также данная функция обрабатывает некотороые исключительные ситуации, такие как файл не существует или не был открыт.

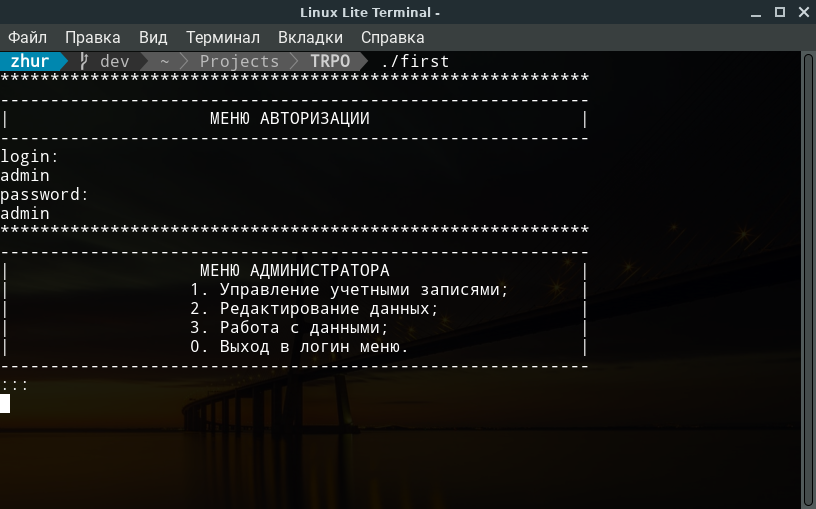
**4. Описание работы программы**

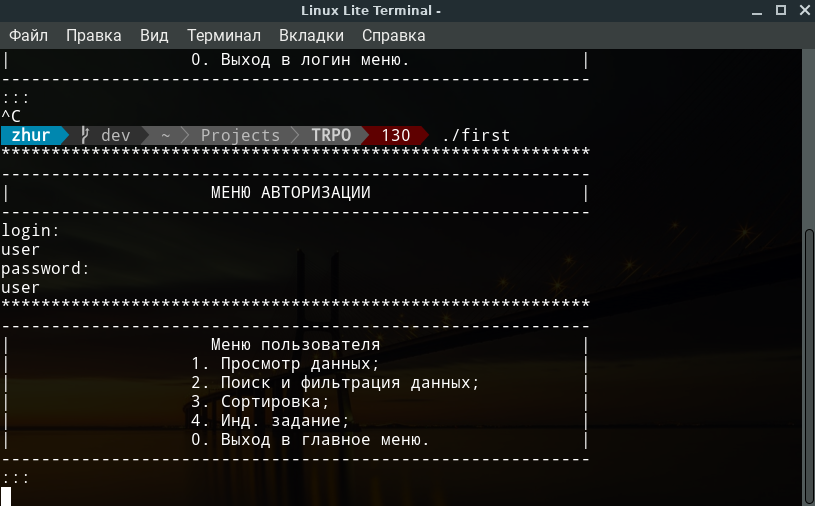
**4.1 Авторизация**

Процесс авторизации состоит из ввода логина и пароля пользователя. После этого программа проверяет, есть ли такой логин, если он есть, то просит ввести пароль, при удачном вводе пароля программа проверит уровень привелегий пользователя и выведет на экран соответствующее меню администратора либо пользователя.

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 работниках, а также работа с данными. Для доступа к пунктам меню используется ввод соответствующих цифр с клавиатуры.

Аккаунты считываются из файла в начале программы и записываются в файл в конце её выполнения.

С аккаунтами можно выполнять такие операции, как:

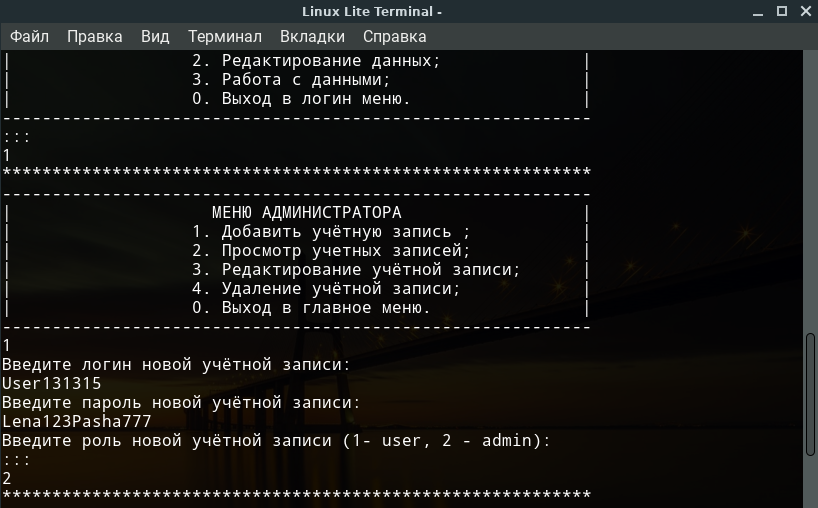
- добавление нового аккаунта;

- удалить существующий аккаунт;

- редактировать существующий аккаунт;

- просмотреть все аккаунты.

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

Рисунок 4.4 — Процесс создания нового аккаунта.

Аккаунт сохранится в файл только после выхода из программы, поэтому после создания аккаунта необходимо выйти, чтобы сохранить аккаунты в файл, однако перед выходом создавать можно столько аккаунтов, сколько необходимо для работы.

На рисунке 4.5 представлен txt файл с данными аккаунтов до добавления.



Рисунок 4.5 — аккаунты в файле до создания.

На рисунке 4.6 представлен txt файл с данными аккаунтов после создания нового аккаунта и выходе из программы, соответственно сохранения его в файл.

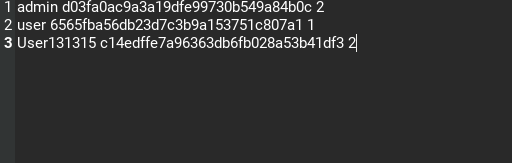


Рисунок 4.6 — аккаунты в файле после создания нового аккаунта и сохранения.

Данные о работниках хранятся в двух файлах с записями о самих работниках и о их заработной плате по месяцам.

С данными о работниках можно выполнять такие операции, как:

- добавление нового работника;

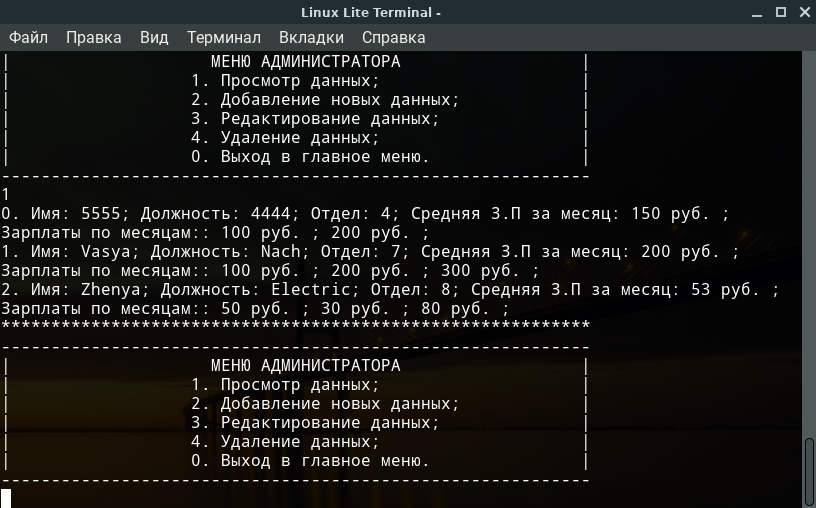
- удаление существующего работника;

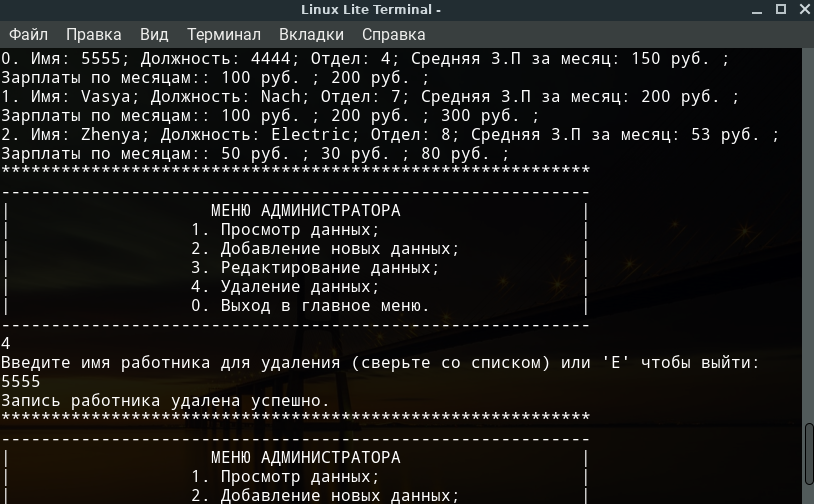
- редактирование существующего работника;

- просмотр всех работников.

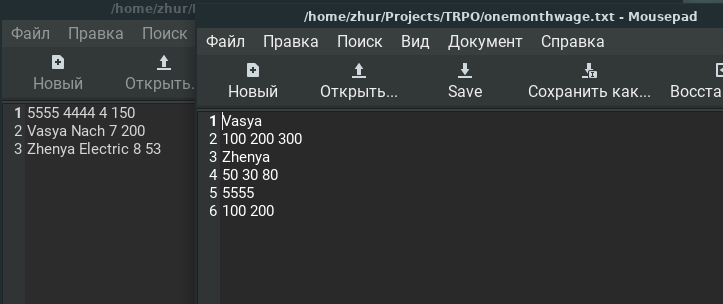
В качестве примера удалим, добавим и отредактируем данные.

На рисунках 4.7 и 4.8 представлен процесс удаления существующего работника.

Рисунок 4.7 — работники до удаления.

Рисунок 4.8 — процесс удаления работника.

На рисунках 4.9 и 4.10 представлен файл с работниками до и после удаления.

Рисунок 4.9 — работники до удаления.

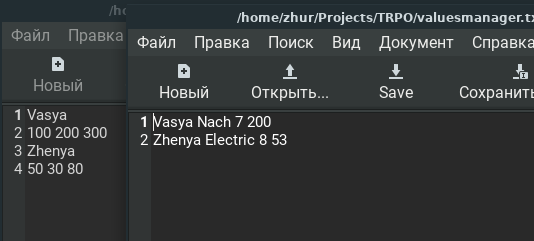
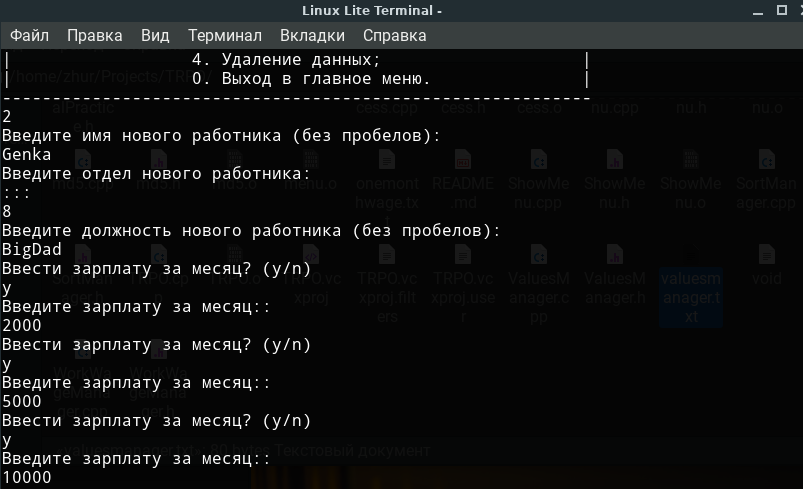
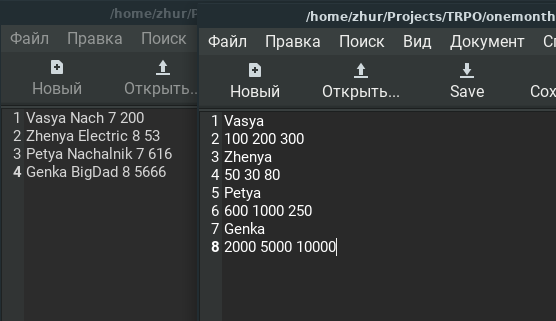


Рисунок 4.10 — работники после удаления.

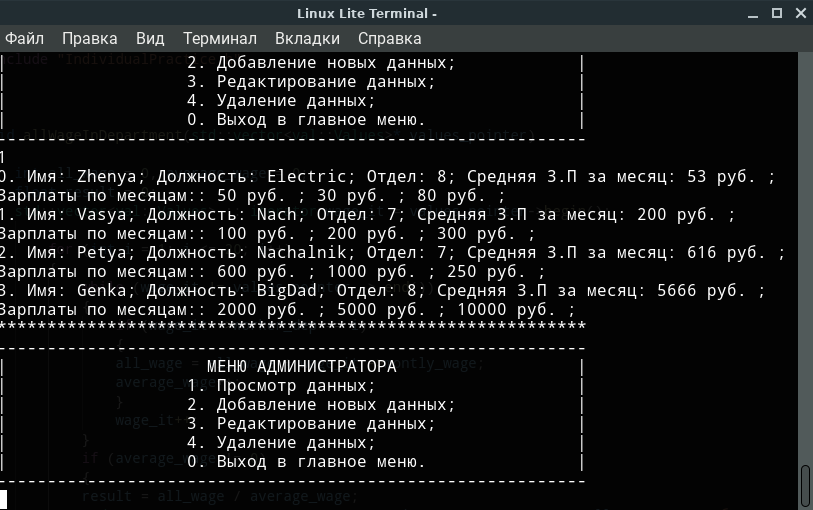
Рисунок 4.11— добавление нового работника.

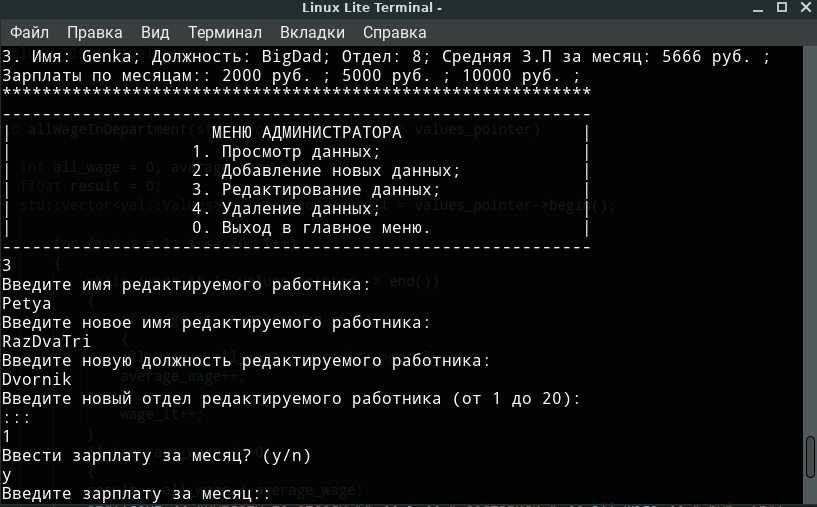
Зарплата вводится по месяцам и хранится в отдельном файле, зарплату по месяцам после ввода удалить или изменить нельзя, а только добавить за следующий месяц, процесс вычисления средней заработной платы осуществляется на основе ранее введенных данных и данных, вводимых после.

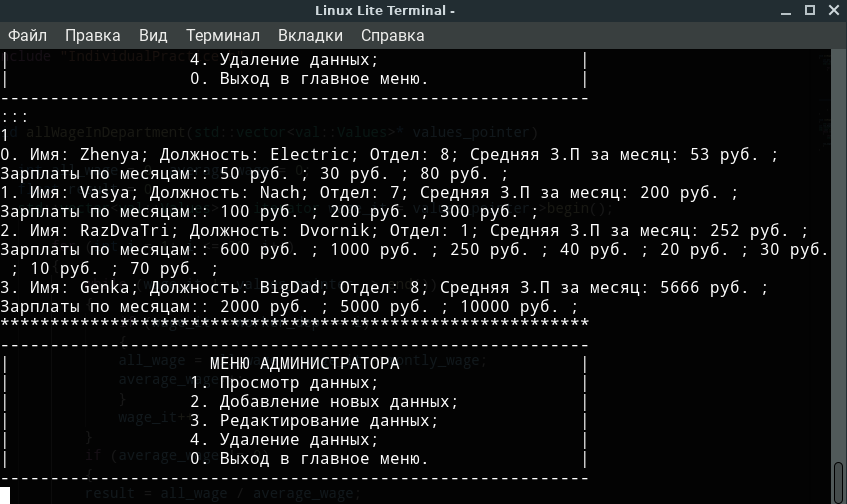
На рисунке 4.8 представлен txt файл с данными о работниках.

Рисунок 4.12 — данные о работниках в файле.

На рисунках 4.13, 4.14 и 4.15 представлен процесс редактирования записи о работнике.

Рисунок 4.13 — работники до редактирования.

Рисунок 4.14 — процесс редактирования работника.

Рисунок 4.15 — после редактирования работника.

Как мы видим, после добавления зарплат по месяцам, старые зарплаты остаются, и расчёт средней заработной платы работника осуществляется по всем зарплатам, что у него были.

**4.3 Модуль пользователя**

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

- поиск работников по имени, должности и размеру заработной платы;

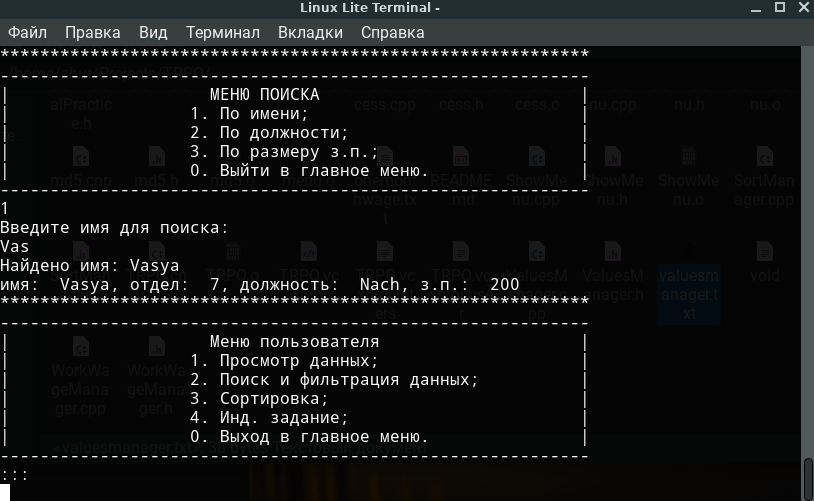
- сортировка по имени, отделу, должности и размеру заработной платы;

- просмотр данных работников;

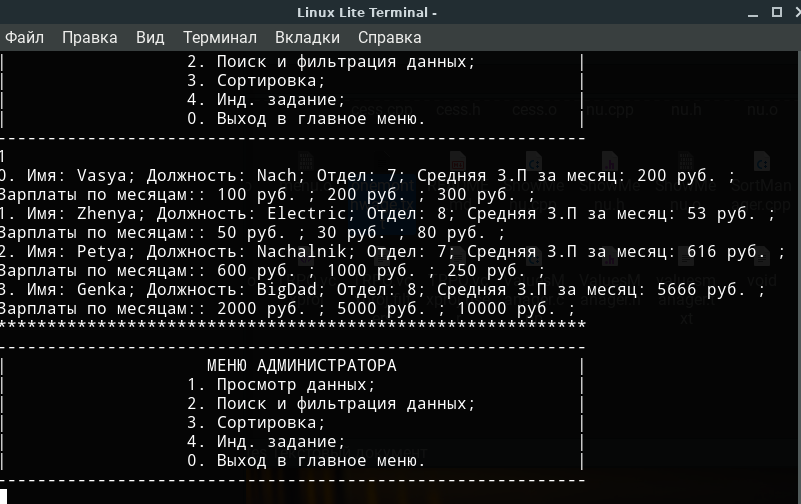
- выполнение индивидуального задания.

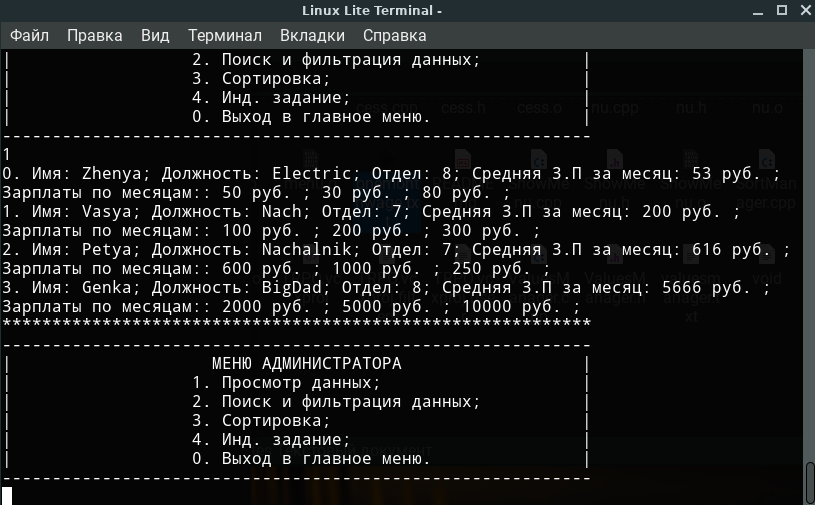
Данный функционал доступен как пользователю, так и администратору.

На рисунке 4.16 представлен процесс поиска по имени рабоника.

Рисунок 4.16 — поиск работника по имени.

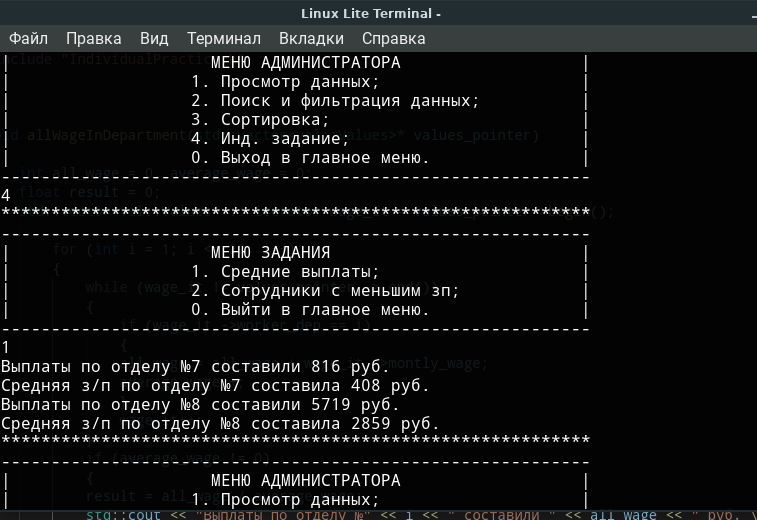
На рисунках 4.17 и 4.18 представлен процесс сортировки по размеру заработной платы.

Рисунок 4.17 — не сортированные данные о работниках.

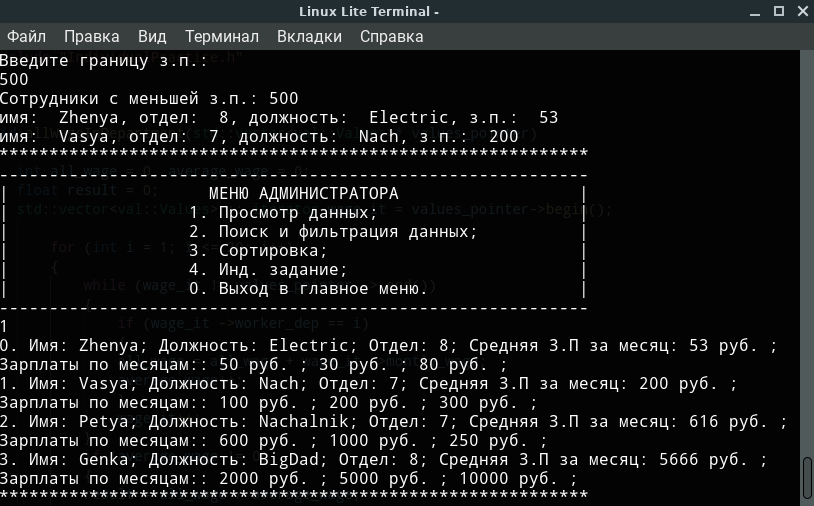
Рисунок 4.17 — сортированные данные о работниках по размеру заработной платы.

Для выполнения индивидуального задания необходимо перейти в соответствующий пункт меню. Процесс выполнения индивидуального задания, а именно вычисление средних выплат и заработных плат сотрудников, а также отображение сотрудников с заработной платой меньше, чем введённая с клавиатуры.

На рисунке 4.18 представлена информация о средних выплатах.

Рисунок 4.18 — средние выплаты и заработные платы по отделам.

На рисунке 4.19 представлена информация о работниках с наименьшей заработной платой, введённой с клавиатуры.

Рисунок 4.19 — сотрудники с заработной платой меньше, чем введённая с клавиатуры.

**4.4 Исключительные ситуации**

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

В числе исключительных ситуаций находятся:

- неправильный ввод числовой переменной;

- отсутствие файла данных аккаунтов, рабочих;

- отсутствие в файле аккаунтов логинов;

- ввод недействительного пункта меню.

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

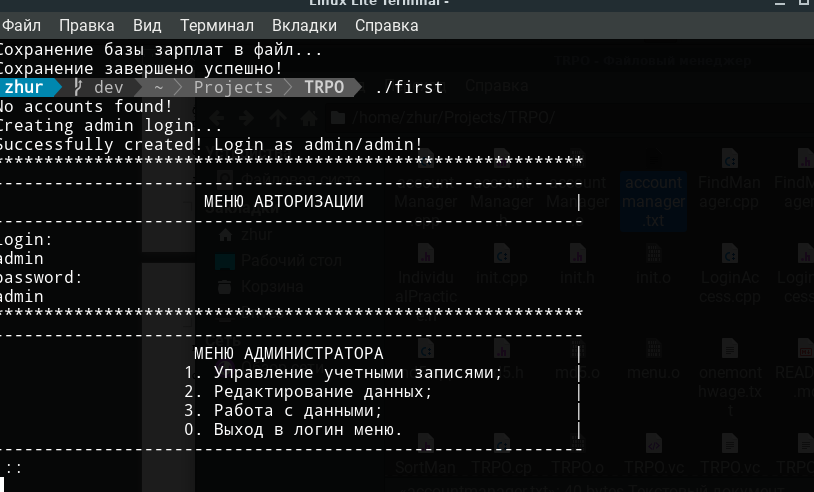
В случае неправильного ввода числовой переменной предусмотрена функция, проверяющая вводимые с клавиатуры данные, и если в них присутствует что-то кроме цифр, то потребуется ввести данные ещё раз.

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. 4.20).

Рисунок 4.20 — отсутствие логинов в файле аккаунтов.

Как можно заметить, программа ответила, что аккаунтов нет, и создала логин admin с паролем admin, под которым можно войти.