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 ВЫПЛАТ ЗАРАБОТНОЙ ПЛАТЫ СОТРУДНИКАМ ПРЕДПРИЯТИЯ

Выполнил: студент гр014302
Невейков Андрей Сергеевич

Проверил: Раднёнок Антон Леонидович

Минск 2021

Оглавление

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

Разработать программу учета выплат заработной платы сотрудникам предприятия.

Сведения о сотрудниках предприятия содержат: Ф.И.О. сотрудника, название отдела, должность, размер заработной платы за месяц.

Вывести список сотрудников, у которых зарплата ниже введенной с клавиатуры. Вычислить общую сумму выплат за месяц по каждому отделу, а также среднемесячный заработок сотрудников по каждому отделу.

Реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя, как минимум три вида поиска, как минимум три вида сортировки.

Исходные требования к курсовой работе

1. Язык программирования C++.
2. Среда разработки Microsoft Visual Studio.
3. Вид приложения – консольное.
4. Парадигма программирования – процедурная.
5. Способ организации данных – структуры (struct).
6. Способ хранения данных – файлы.
7. Каждая логически завершенная подзадача программы должна быть реализована в виде отдельной функции.
8. Построение программного кода должно соответствовать соглашению о коде «C++ Code Convention».
9. К защите курсовой работы представляются: консольное приложение и пояснительная записка.

Функциональные требования к курсовой работе

Функциональные требования к курсовой работе

Первым этапом работы программы является авторизация предоставление прав. В рамках данного этапа необходимо считать данные базы данных, содержащей учетные записи 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 (в этом случае, например, role = 1),

- в качестве пользователя (в этом случае, например, role = 0).

Если база данных с учетными записями пользователей не существует, то необходимо её программно создать и записать учетные данные администратора.

Регистрация новых пользователей осуществляется самим пользователем путем ввода желаемых логина и пароля и ожидания подтверждения администратором новой учетной записи. Для реализации этого способа в структуре учетных записей пользователей необходимо предусмотреть дополнительное поле condition:

- login;
- password;
- role;
- condition (данное поле служит для подтверждения или блокировки администратором учетных записей). По умолчанию condition = 0 при попытке зарегистрироваться; далее администратор меняет значение на condition = 1 и тем самым подтверждает новую учетную запись: пользователь может осуществить вход в систему.

Вторым этапом работы программы является собственно работа с данными, которая становится доступной только после прохождения авторизации. Данные хранятся в отдельном файле.

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

Модуль администратора включает следующие подмодули**:**

1. Управление учетными записями пользователей:
   - ✓ просмотр всех учетных записей;
   - ✓ редактирование учетной записи;
   - ✓ удаление учетной записи.
2. Работа с данными:
   - ✓ просмотр всех данных;
   - ✓ добавление новой записи;
   - ✓ редактирование записи;
   - ✓ удаление записи;
   - ✓ вывод списка сотрудников, у которых зарплата ниже введенной с клавиатуры.
   - ✓ Сортировка данных (по трем различным параметрам)
     - По фамилии
     - По отделу
     - По зарплате
   - ✓ Поиск данных (по трем различным параметрам):
     - По фамилии сотрудника
     - По должности сотрудника
     - Сотрудника с наибольшей зарплатой

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

- ✓ просмотр всех данных;
- ✓ добавление новой записи;
- ✓ редактирование записи;
- ✓ удаление записи;
- ✓ вывод списка сотрудников, у которых зарплата ниже введенной с клавиатуры.
- ✓ вывод информации по отделам
- ✓ Сортировка данных (по трем различным параметрам)
  - По фамилии
  - По отделу
  - По зарплате
- ✓ Поиск данных (по трем различным параметрам):
  - По фамилии сотрудника
  - По должности сотрудника
  - Сотрудника с наибольшей зарплатой

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

Предусмотреть:

1. обработку исключительных ситуаций:
   A. имя пользователя или пароль не верны;
   B. запись с указанным идентификатором не найдена;
   C. пользователь с таким именем уже существует;
   D. ведённые данные не соответствуют формату поля;
2. возможность возврата назад (навигация);
3. вывод сообщения об успешности создания/создания файла/записи.

Требования к программной реализации

1. Все переменные и константы должны иметь осмысленные имена в рамках тематики варианта к курсовой работе.
2. Имена функций должны быть осмысленными и строится по принципу «глагол + существительное». Если функция выполняет какую-либо проверку и возвращает результат типа bool, то ее название должно начинаться с глагола is (например, isFileExist, isUnicLogin).
3. Код не должен содержать неименованных числовых констант (так называемых «магических» чисел), неименованных строковых

констант (например, имен файлов и др.). Подобного рода информацию следует выносить в глобальные переменные с атрибутом const. По правилам хорошего стиля программирования тексты всех информационных сообщений, выводимых пользователю в ответ на его действия, также оформляются как константы.

4. Код необходимо комментировать (как минимум в части нетривиальной логики).
5. Код не должен дублироваться – для этого существуют методы и функции.
6. 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.
7. Следует избегать длинных функций и глубокой вложенности: текст функции должен умещаться на один экран, а вложенность блоков и операторов должна быть не более трёх.

2. КОНСТРУИРОВАНИЕ ПРОГРАММЫ

Реализация программы будет осуществляться на языке C++ в среде Microsoft Visual Studio 2019. Программа будет компилироваться и использоваться в операционных системах семейства Microsoft Windows.

2.1 Разработка структуры программы

Согласно требованиям к программе, необходимо наличие исполняемой программы, которая работает с модулем администратора и модулем пользователя.

На рисунке 1 показаны основные модули программы.

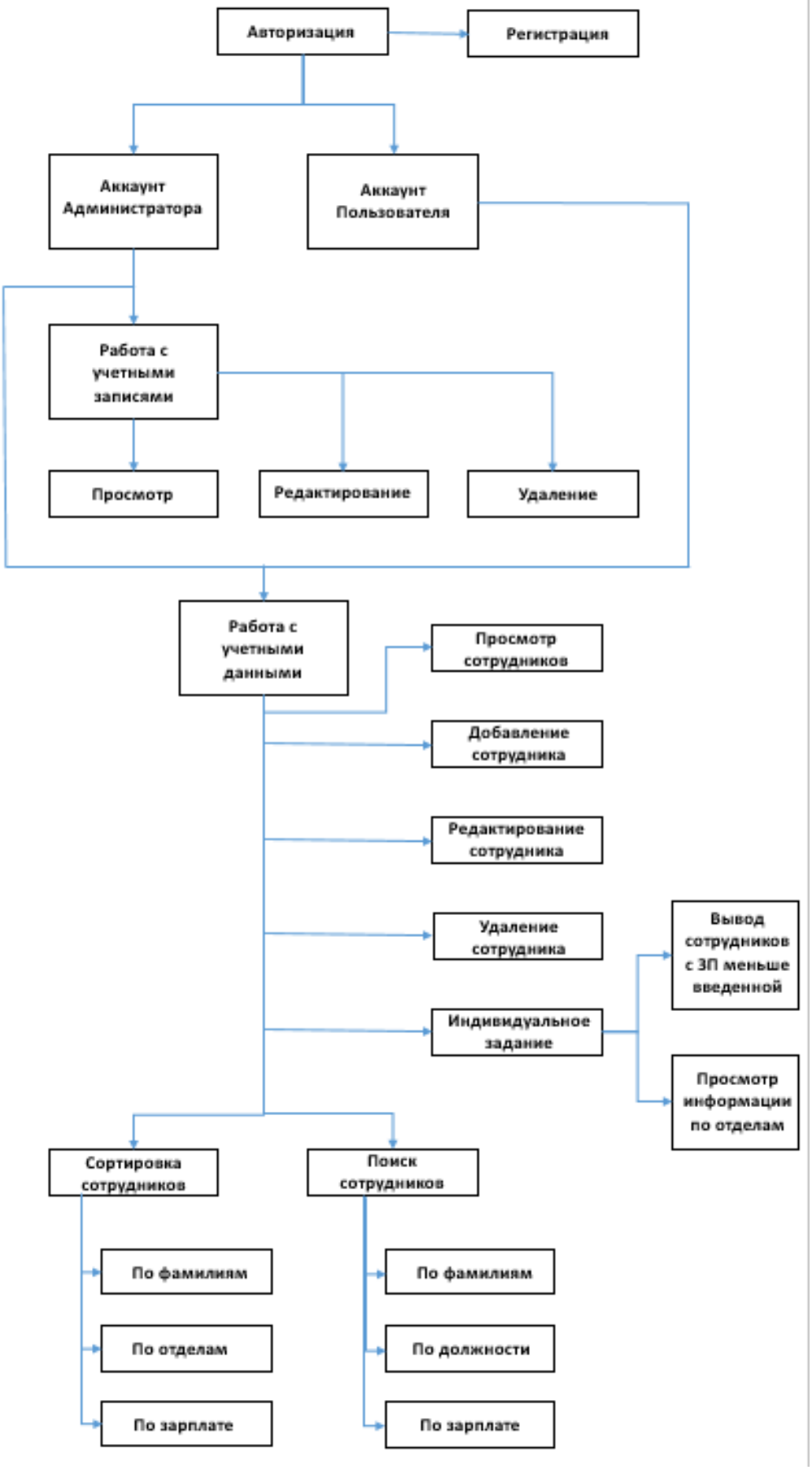

Рисунок 1. Модули программы

Функция аутентификации пользователя заключается в проверке существования в данных пользователей введённого логина и соответствующего ему пароля. Авторизация пользователя подразумевает получение его роли из данных и предоставление ему соответствующих возможностей.

После успешной авторизации пользователя создаётся пользовательская сессия с соответствующими привилегиями, согласно роли пользователя. Сессия администратора имеет полный доступ к модулям управления данных, а сессия пользователя, в свою очередь, имеет доступ только к модулю управления данных сотрудников.

Программа подразумевает наличие консольного пользовательского интерфейса.

2.2 Выбор способа организации данных

В качестве выбора способа описания входных данных приводится описание типов struct:

❖ для учетных записей пользователей:

```
struct Accounts      // Структура аккаунтов
{
	string login;
	string password;
	int status; // 0 - пользователь | 1 - администратор
	int activation; // 0 - деактивирован | 1 - активирован
};
```

❖ для информации о сотрудниках предприятия:

```
struct Employee      // Структура сотрудников
{
	string Name;
	string Surname;
	string Patronymic;
	string Department;
	string Position;
	int Salary;
};
```

❖ для информации об отделах предприятия:

```
struct Department // Структура отделов
{
	string Department_name; // Название отдела
	int Employees_amount = 0; // Количество сотрудников отдела
	int Total_department_salary = 0; // ЗП всего отдела
```

```
};
```

Структура Аккаунты содержит следующие поля:

- login → имя аккаунта пользователя.
- password → пароль аккаунта
- status → статус (админ/пользователь)
- activation → состояние аккаунта(активирован/деактивирован)

Структура Сотрудники содержит следующие поля:

Name → имя сотрудника
Surname → фамилия сотрудника
Patronymic → отчество сотрудника
Department → отдел сотрудника
Position → должность сотрудника
Salary → зарплата сотрудника

Для работы со структурами будет использоваться массив в качестве способа объединения входных данных

Область видимости входных данных локальная, поскольку, чем больше информации скрыто, тем меньше нужно удерживать в уме в определенный момент, и, тем ниже вероятность появления ошибки.

Программа работает с данными, которые находятся в 2(двух) файлах:

1. Файл, где хранятся данные учетных записей
2. Файл, где хранятся данные о сотрудника

2.3 Разработка перечня пользовательских функций программы

// Accounts

```
void Start_program(); // Функция из main для запуска программы
void User_menu(); // Меню для пользователя | Переход
void Admin_menu(Accounts* arr_of_accounts, int& number_of_accounts); // Меню для админа
[Упр.аккаунтами/сотрудниками]
void Add_accounts(Accounts* arr_of_accounts, int& number_of_accounts); // Добавление
аккаунтов
void Del_accounts(Accounts* arr_of_accounts, int& number_of_accounts); // Для удаление
аккаунтов
void Show_accounts(Accounts* arr_of_accounts, int number_of_accounts); // Вывод аккаунтов
void Check_user_status(Accounts* arr_of_accounts, int& number_of_accounts, int role); //
Проверка кто авторизировался
void Read_file_accounts(Accounts* arr_of_accounts, int number_of_accounts); //
Чтение|Перенос аккаунтов из файла
void Write_file_accounts(Accounts* arr_of_accounts, int number_of_accounts); // Записать
аккаунтов в файл
void Admin_accounts_menu(Accounts* arr_of_accounts, int& number_of_accounts); //
Управление аккаунтами
void Check_accounts_number(int& number_of_accounts); // Определение количества аккаунтов
в файле
void Check_login_and_password(Accounts* arr_of_accounts, int& number_of_accounts); //
Проверка логина и пароля
Accounts* Authorization_menu(Accounts* arr_of_accounts, int& number_of_accounts); // Меню
авторизации [Вход/Регистрация]
Accounts*      Memory_reallocation_for_accounts(Accounts*      arr_of_accounts,      int&
number_of_accounts, int m); // Перевыделение памяти
```

// Employees

```
// ОСНОВНЫЕ ФУНКЦИИ_____________________________________________________________
_____________________________________

void User_menu(); // Меню пользователя
void Check_employees_file(int& number_of_Employees); // Определение количества
сотрудников
void Show_employees(Employee* arr_of_Employees, int number_of_Employees); // Вывод
сотрудников
void Add_employees(Employee* arr_of_Employees, int& number_of_Employees); // Добавление
сотрудников
void Delete_employees(Employee* arr_of_Employees, int& number_of_Employees); // Удаление
сотрудников
void Write_to_employees_file(Employee* arr_of_Employees, int number_of_Employees); //
Запись в файл
void Read_from_employees_file(Employee* arr_of_Employees, int number_of_Employees); //
Чтение из файла
void Updating_of_employees_information(Employee* arr_of_Employees, int&
number_of_Employees); // Обновление информации о сотрудниках

Employee* Memory_reallocation_for_employees(Employee* arr_of_Employees, int&
number_of_Employees, int m); // Перевыделение памяти
Employee* Menu_employees(Employee* arr_of_Employees, int& number_of_Employees); // Меню
для выбора действия


// ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ_______________________________________________________
_____________________________________

void Menu_individual_task(Employee* arr_of_Employees, int number_of_Employees); // Выбор
индивидуального задания
void Individual_task_salary_info(Employee* arr_of_Employees, int number_of_Employees); //
Индивидуальное задание №1: Вывод сотрудников с ЗП < введенной с клавиатуры
void Individual_task_departments_info(Employee* arr_of_Employees, int
number_of_Employees); // Индивидуальное задание №2: Инфа по отделам | Общая ЗП | Средняя
ЗП

Department* Individual_task_department_memory_reallocation(Department* arr_of_dep, int&
number_of_dep, int m); // Индивидуальное задание №2.1: Перевыделение памяти


// СОРТИРОВКА___________________________________________________________________
_____________________________________

Employee* Sorting_by_Surname(Employee* arr_of_Employees, int number_of_Employees); //
Сортировка №1: По фамилиям
Employee* Sorting_by_department(Employee* arr_of_Employees, int number_of_Employees); //
Сортировка №2: По отделам
Employee* Sorting_by_the_salary(Employee* arr_of_Employees, int number_of_Employees); //
Сортировка №3: По зарплате
Employee* Selection_sorting_menu(Employee* arr_of_Employees, int number_of_Employees); //
Выбор сортировки


// ПОИСК________________________________________________________________________
_____________________________________

void Find_surname(Employee* arr_of_Employees, int number_of_Employees); // Поиск №1: По
фамилии
```

```
void Find_position(Employee* arr_of_Employees, int number_of_Employees); // Поиск №2: По должности
void Find_max_salary(Employee* arr_of_Employees, int number_of_Employees); // Поиск №3: По максимальной ЗП
void Show_result_of_finding(Employee* arr_of_Employees, int number_of_Employees, int i); // Вывод для поиска
void  Menu_of_finding_employees(Employee*  arr_of_Employees,  int  number_of_Employees);  // Выбор поиска
```

// Protection

```
int Protection(int a, int b, int& n); // Защита от букв и чисел [ от a до b] | С переменной n
```

// Signal

```
void Save_it(); // Новые данные успешно сохранены!
void Set_color(int text, int bg); // Изменение цвета
void Ask_for_saving(); // Хотите сохранять?
void Account_not_found(); // Такого аккаунта нет в системе
void Password_incorrect(string TempLogin, bool& flagPassword, int& amount); // Выводит ошибку если несколько раз ввели не тот пароль [Вход/Регистрация]
```

3. РАЗРАБОТКА АЛГОРИТМОВ ПРОГРАММЫ

3.1 Алгоритм функции main

Функция main является точкой входа в программу, вызывает основные функции инициализации, также функцию с авторизацией и дальнейшей обработкой данных. На рисунке 2 показана блок-схема алгоритма функции main.

Рисунок 2 - Блок-схема алгоритма функции Start_program

3.2 Алгоритм функции Show_accounts

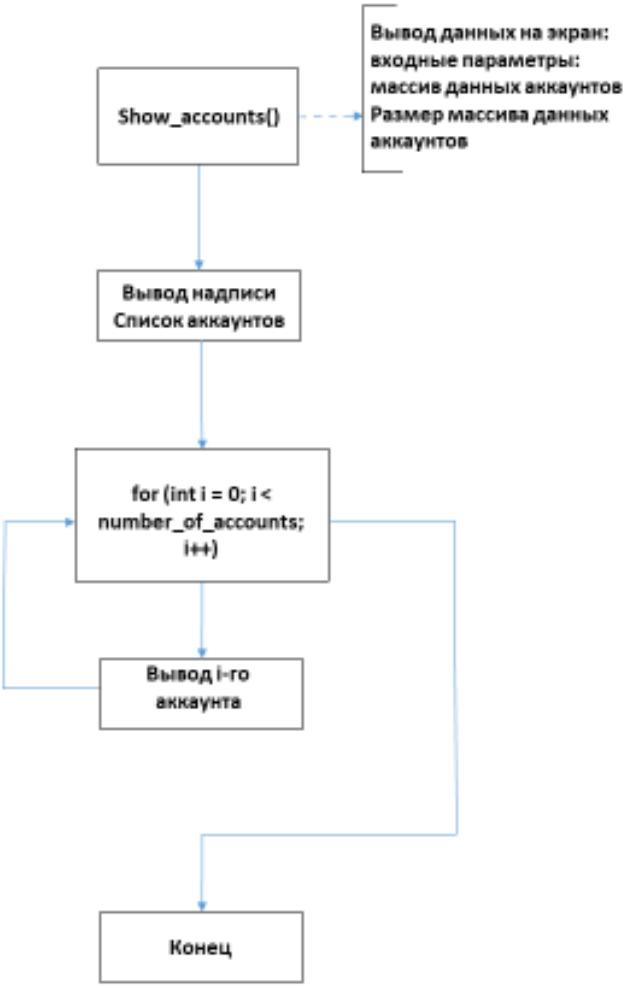

Рисунок 3 - Блок-схема алгоритма функции show_accounts

3.3 Алгоритм функции User_menu

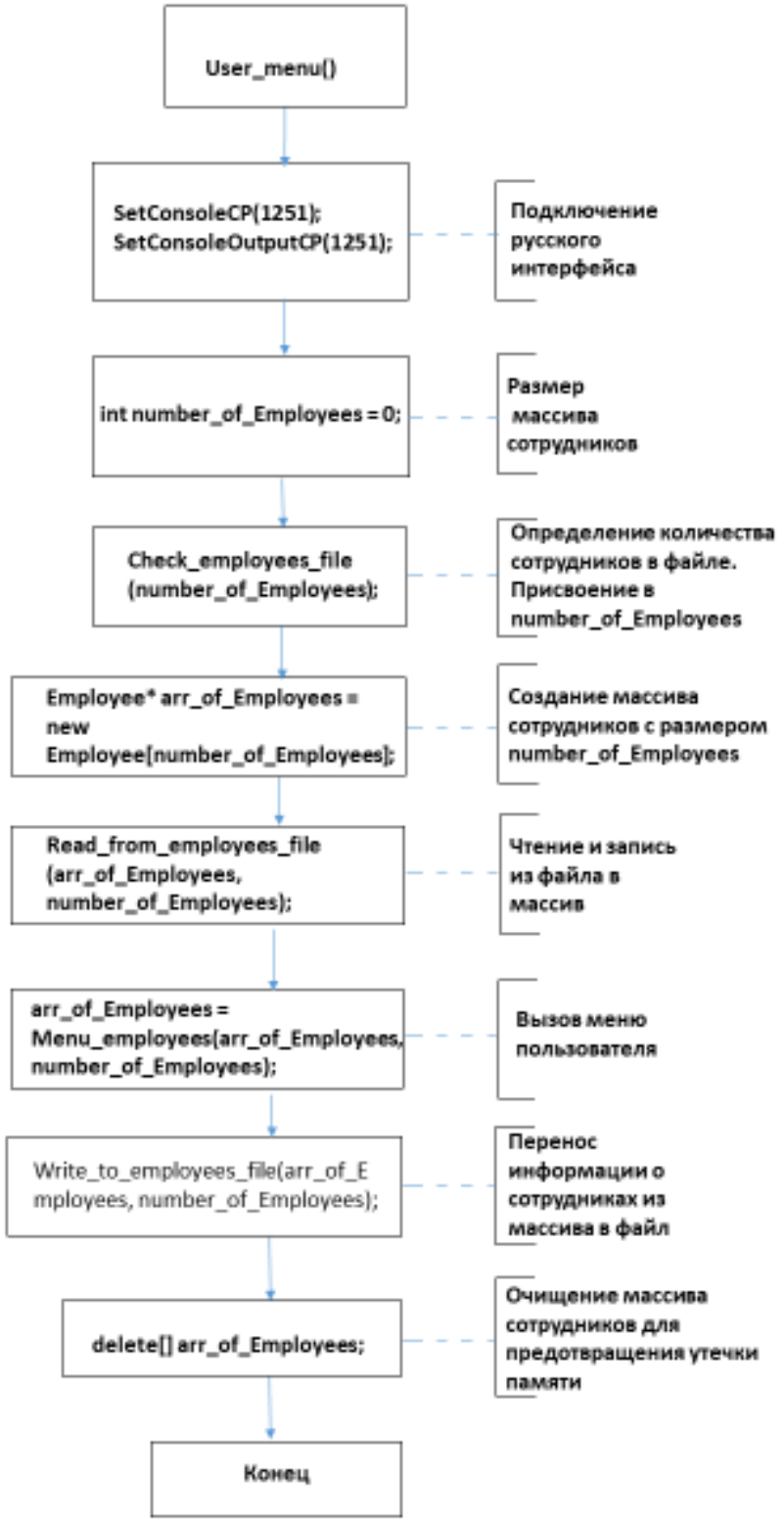

Рисунок 4 - Блок-схема алгоритма функции User_menu()

4. ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

При запуске программа печатает приветствие и пытается найти и открыть файл с данными аккаунтов. Если открытие файла не увенчалось успехом, то программа автоматически создаст новый файл.
В новом файле будет создана первая учетная запись для администратора.
Имя файла прописано в коде программы: `FILE_INFO_ACCOUNTS = "Accounts.txt";`
Если файл успешно открыт, то программа считает данные в массив.

4.1 Авторизация

При успешном открытии файла с данными аккаунтов, существует выбор между входом в учетную запись или же регистрация нового аккаунта.

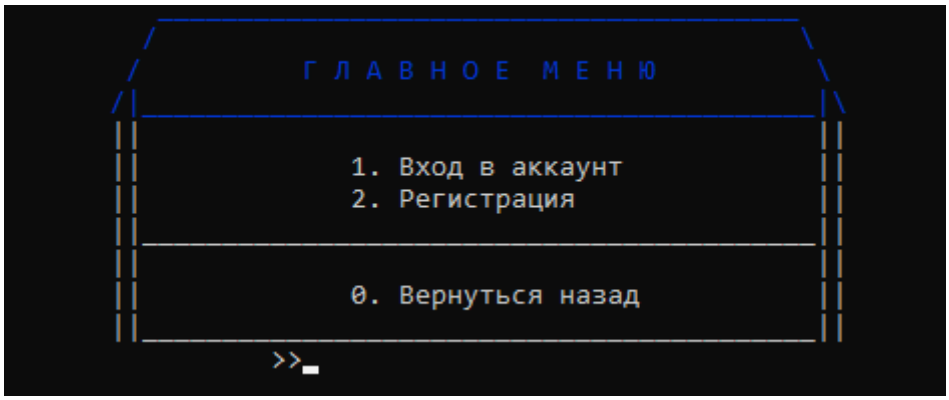

Рисунок 5 – Начало работы программы

При выборе пункта №2 откроется окно регистрации, где пользователь должен будет указать желаемый логин и пароль. Ввод пароля запрашивает дважды с целью предотвращения ошибок при создании новой учетной записи. Если данные указаны верно, пользователь увидит сообщение о том, что его данные успешно сохранены.

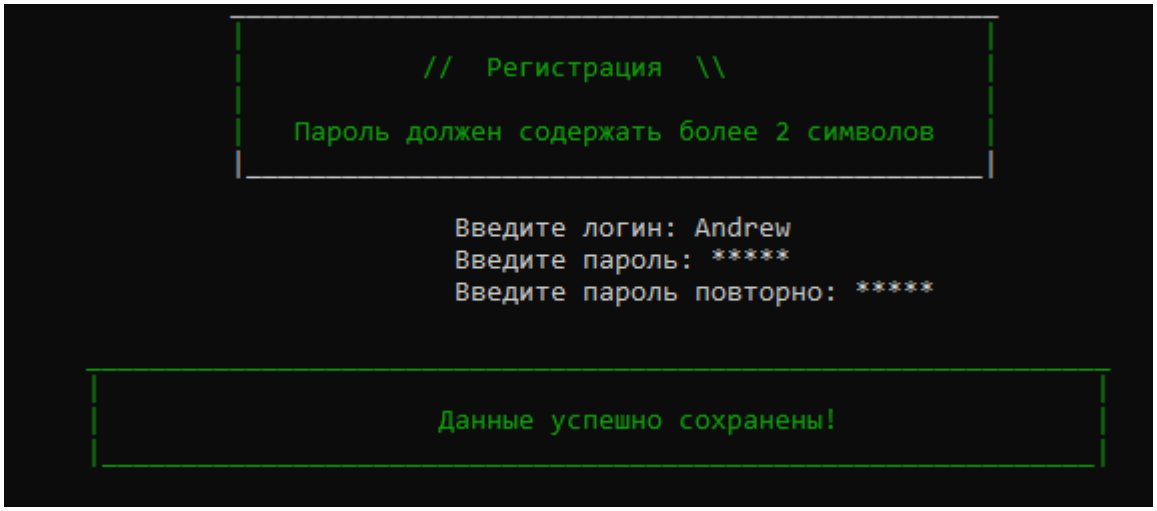

Рисунок 6 – Регистрация нового аккаунта

При выборе пункта №1 откроется окно для входа в учетную запись, где нужно указать логин и пароль. Введенные данные сверяются с данными, которые хранятся в массиве.

Если данный аккаунт ещё не активирован, система выдаст ошибку

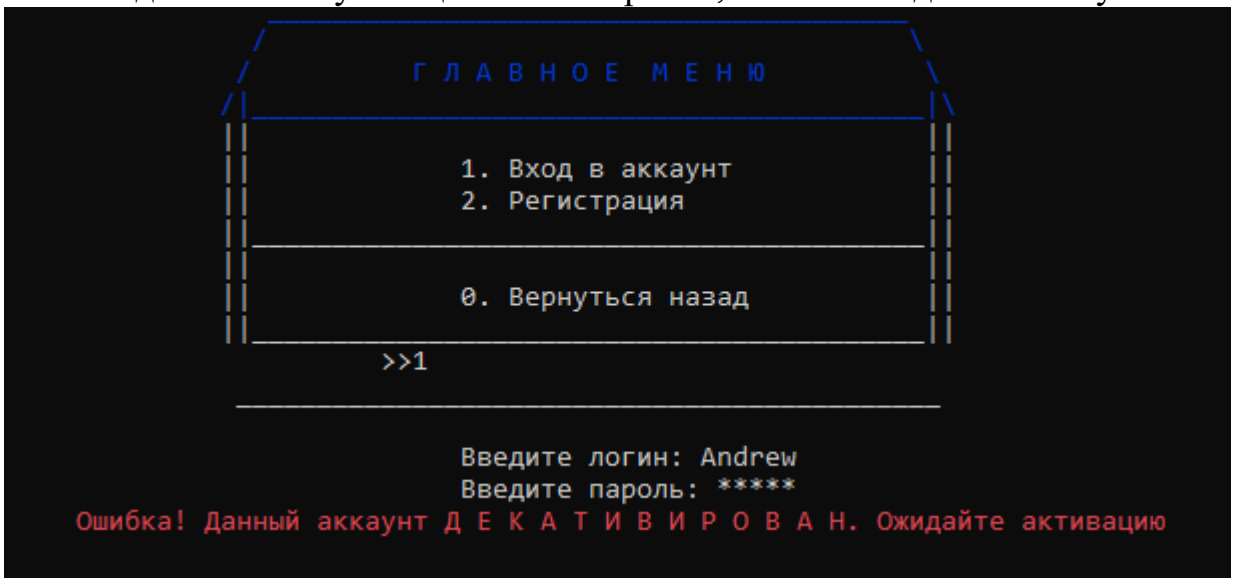

Рисунок 7 – Авторизация

4.2 Модуль администратора

При входе в учетную запись администратора выводится соответствующие сообщение и появляется функционал для работы.

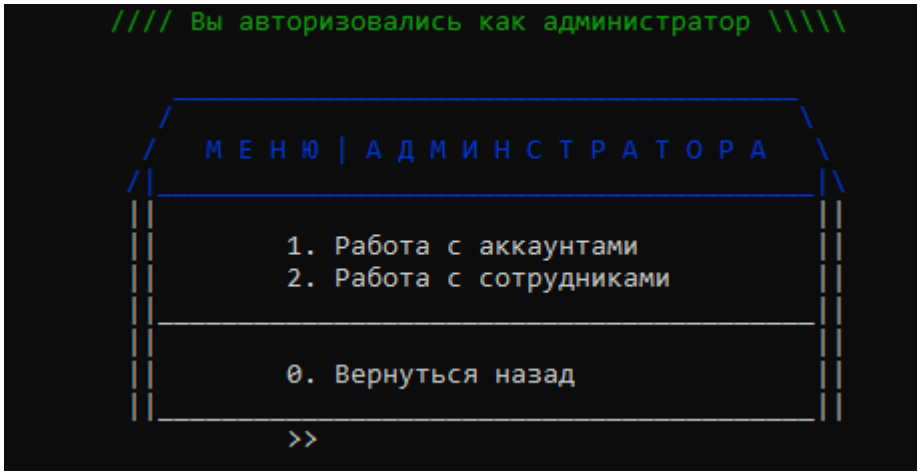

Рисунок 8 – Вход для администратора

При выборе пункта №1 (Работа с аккаунтами) открывается соответствующие меню:

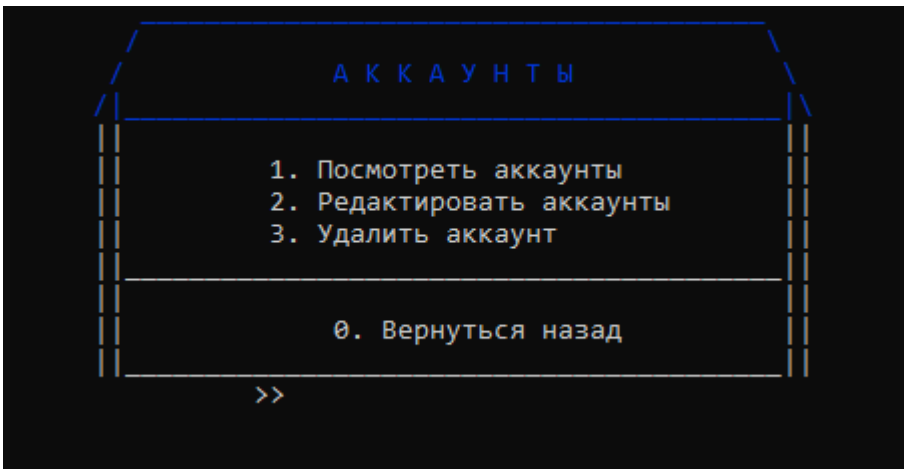

Рисунок 9 – меню для работы с аккаунтами

Пункт №1 (Просмотр аккаунтов)

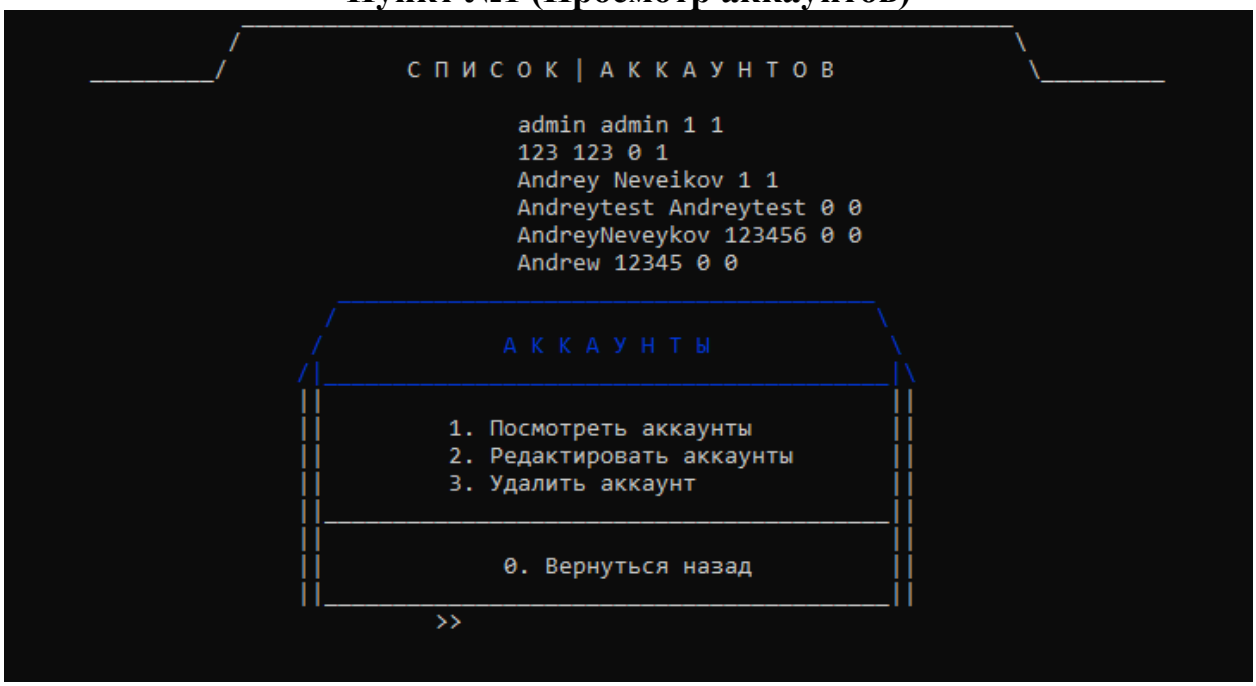

Рисунок 10 – просмотр аккаунтов

Пункт №2 (Редактирование аккаунтов)

При редактировании аккаунтов выводится список всех аккаунтов, после чего система запрашивает ввода логина аккаунта над которым нужно произвести какие-либо корректировки, после чего выбрать то, что нужно изменить. При успешном изменении данных будет выведено сообщение о том, что данные успешно сохранены.

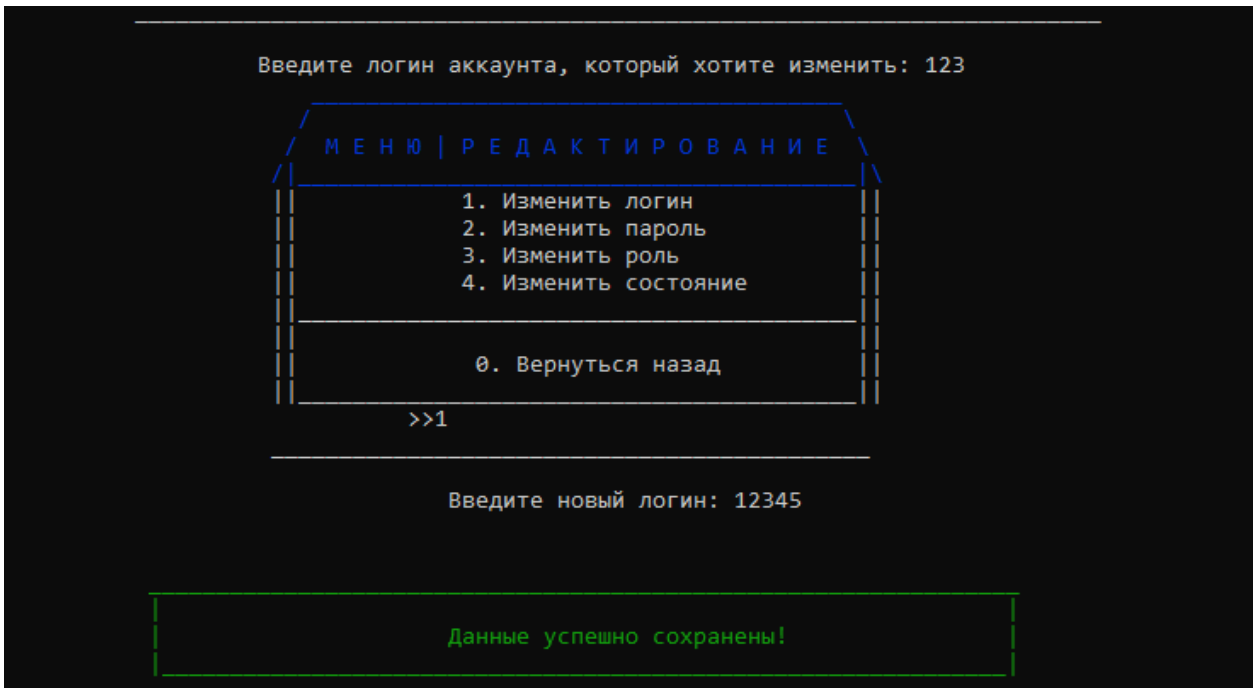

Рисунок 11 – редактирование аккаунтов

Пункт №3 Удаление аккаунта

При выборе пункта с удалением аккаунта выводится список аккаунтов после чего система запрашивает логин аккаунта, который нужно удалить.

Если такой аккаунт существует, то система запрашивает подтверждение т.к. данные в случае удаления уже нельзя будет восстановить.

Если удаление аккаунта подтверждается, то выводится надпись, что данные успешно обновлены, т.е. аккаунт удален из системы.

От удаления можно отказаться и действие будет отменено.

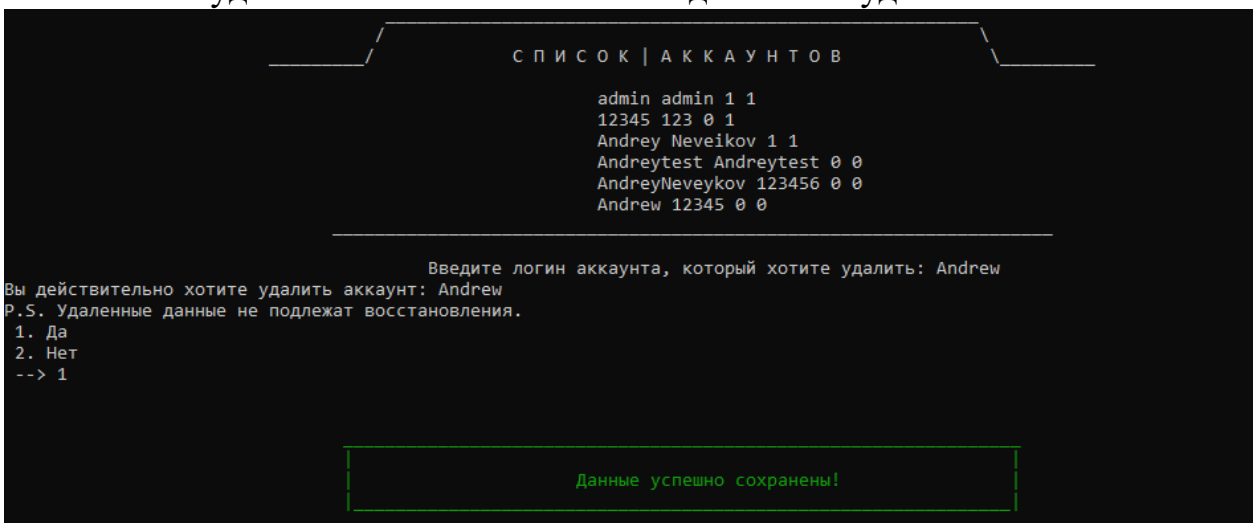

Рисунок 12 – удаление аккаунта.

Редактирование состояния

Администратор лично активирует аккаунты.

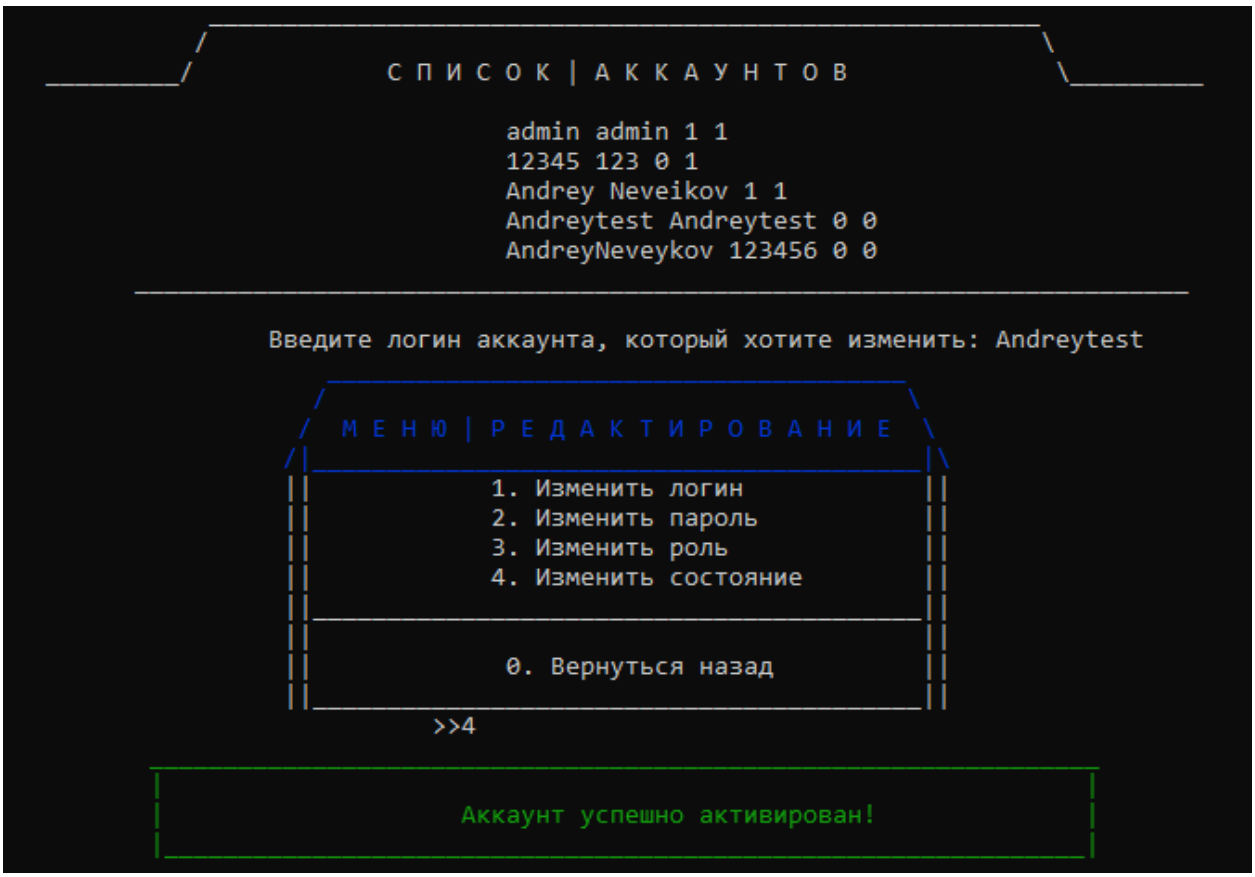

Рисунок 13 – активация аккаунта

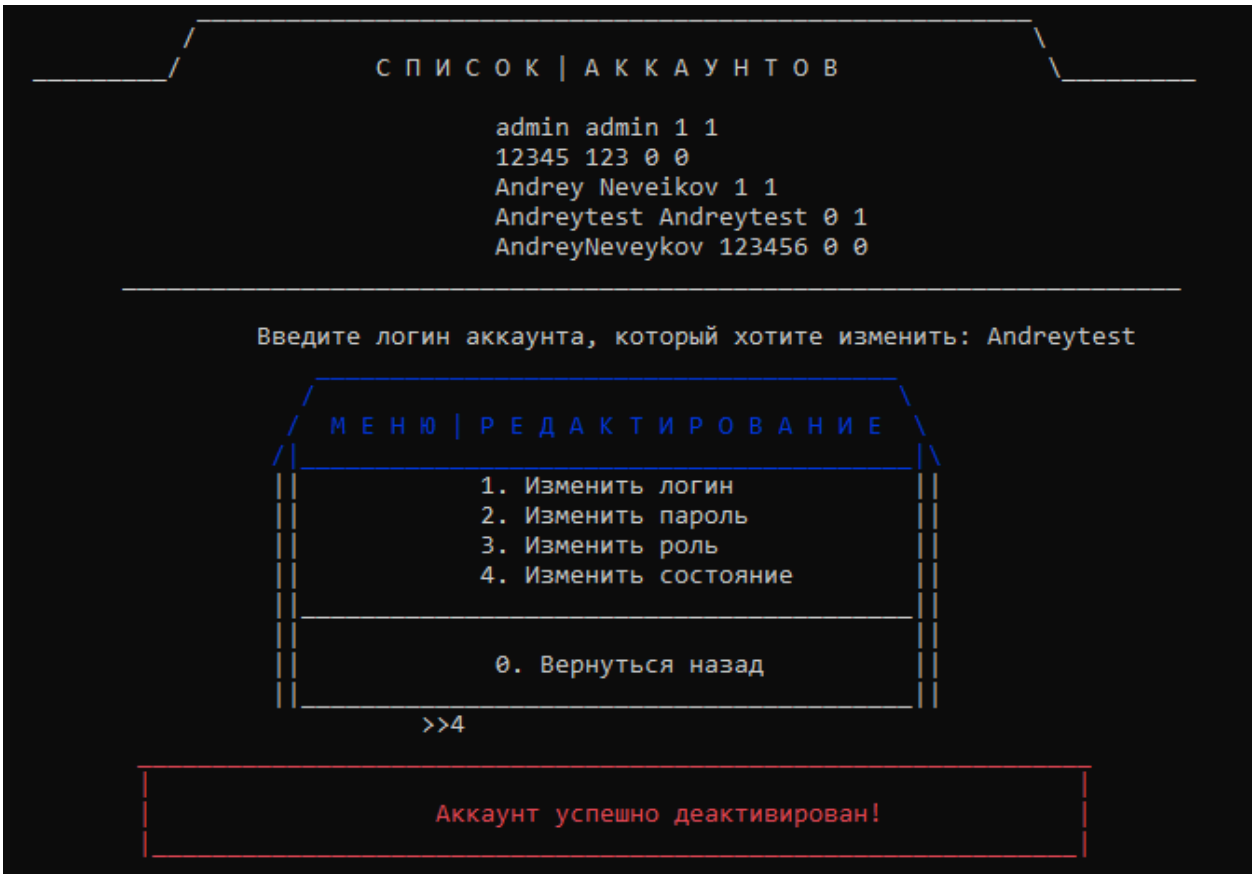

Рисунок 14 – деактивация

1. Изменение роли

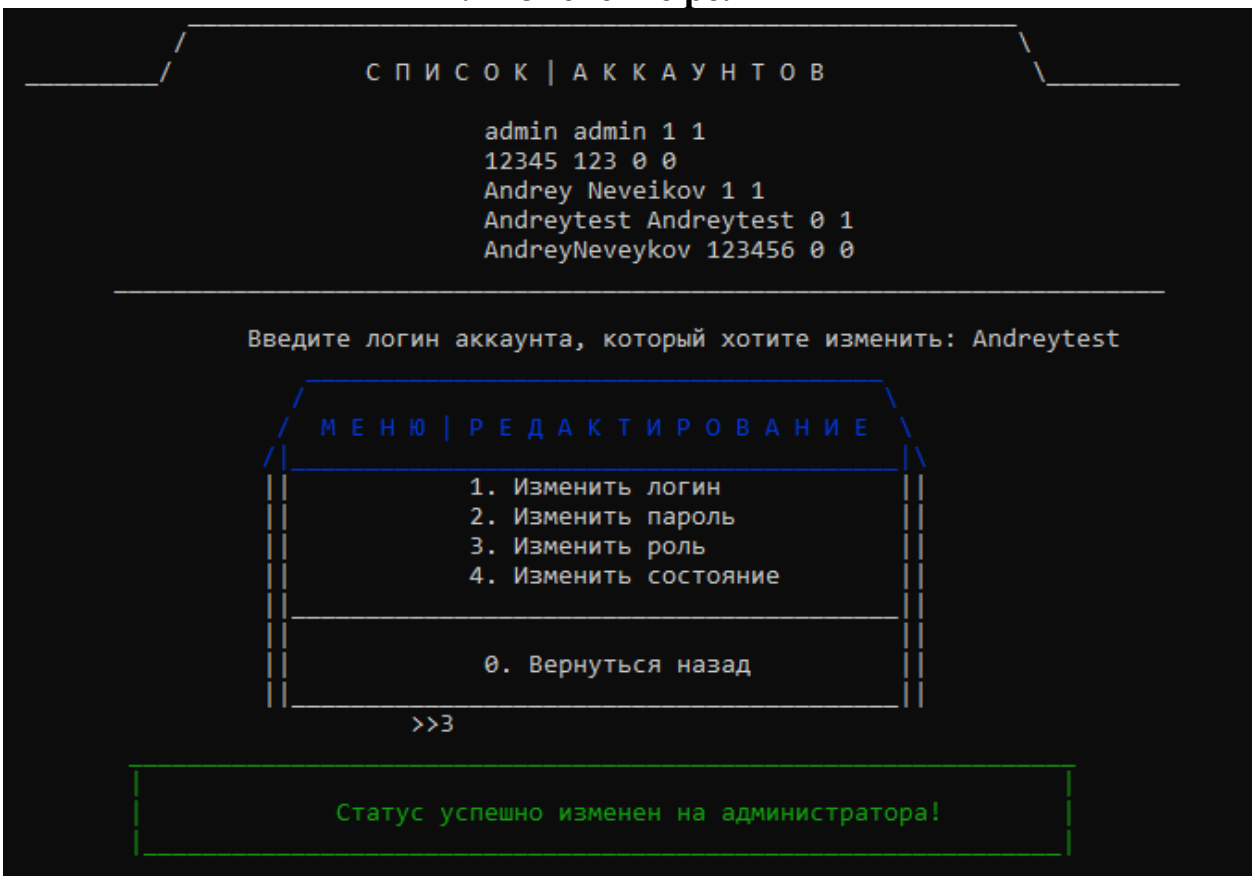

Рисунок 15 – права администратора

2. Изменение роли

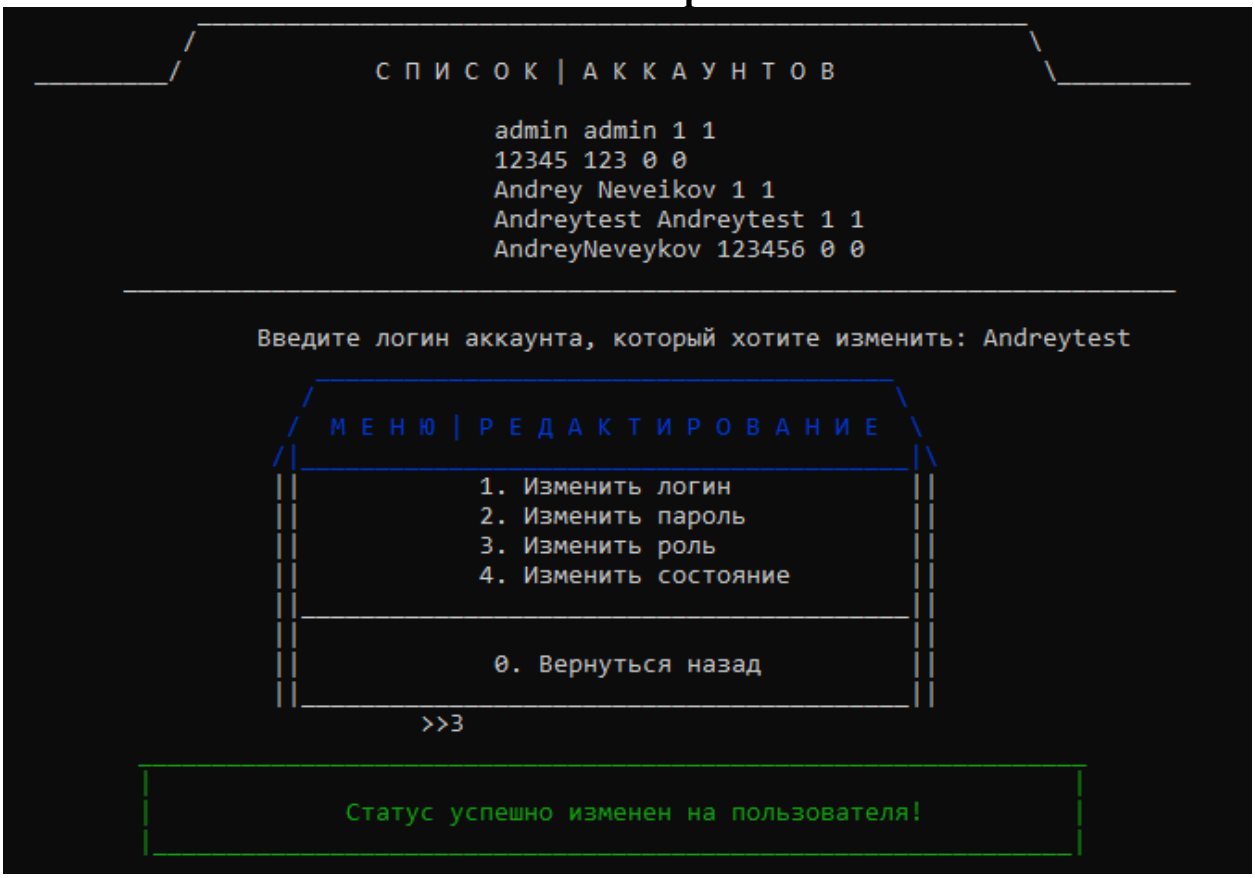

Рисунок 16 – права пользователя

Исключительные ситуации:

Для аккаунта администратор установлена дополнительная защита. Если при запросе на удаление будет введен аккаунта админа, возникнет ошибка.

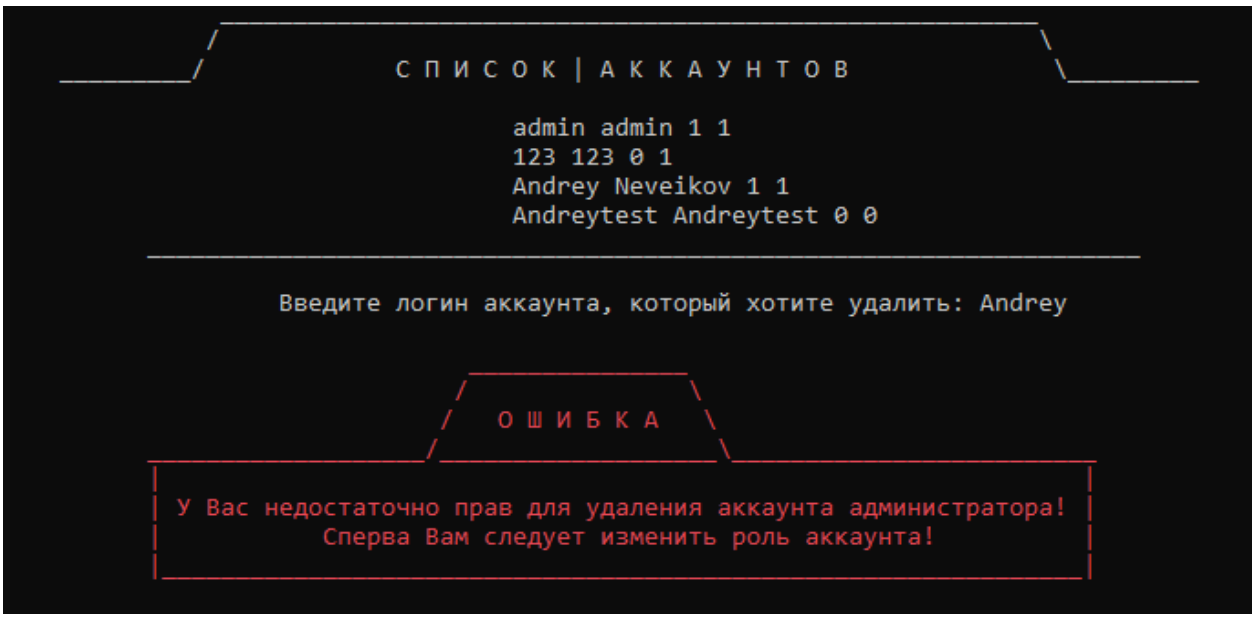

Рисунок 17 – ошибка при попытке удаления администратора

Если будет введен аккаунт, которого нету в системе, то появится сообщение о том, что такого аккаунта не существует.

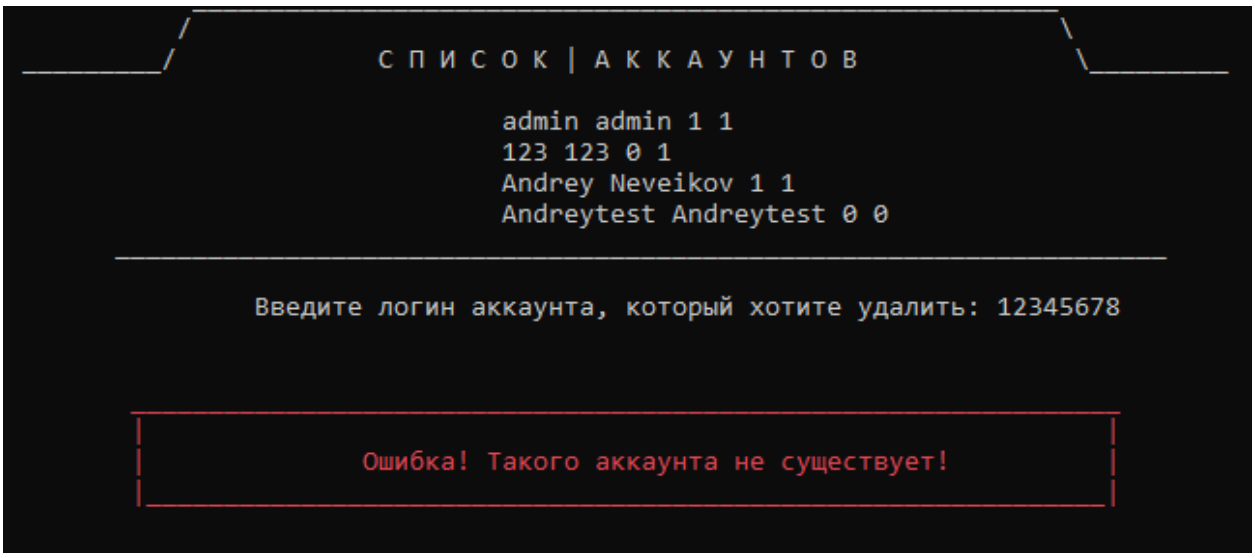

Рисунок 18 – если аккаунта нету в системе

4.3 Модуль пользователя

Если аккаунт активирован, будет выведено приветствие для пользователя

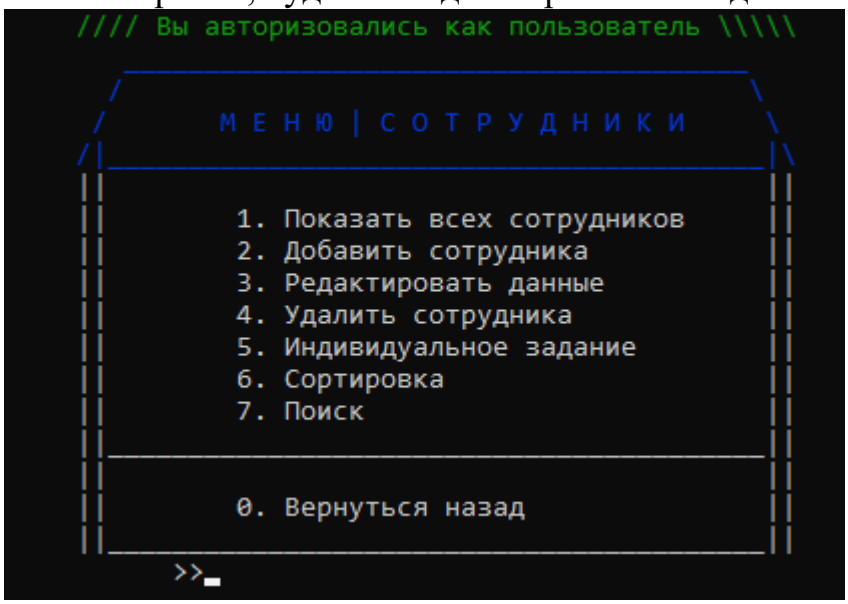

Рисунок 19 – вход для пользователя

При выборе пункта №3 откроется меню редактирования аккаунта, где нужно будет указать фамилию сотрудника, данные которого нужно отредактировать. После выбора того, что нужно отредактировать мы вводим соответствующие изменения. После изменения появится сообщение об успешном обновлении данных.

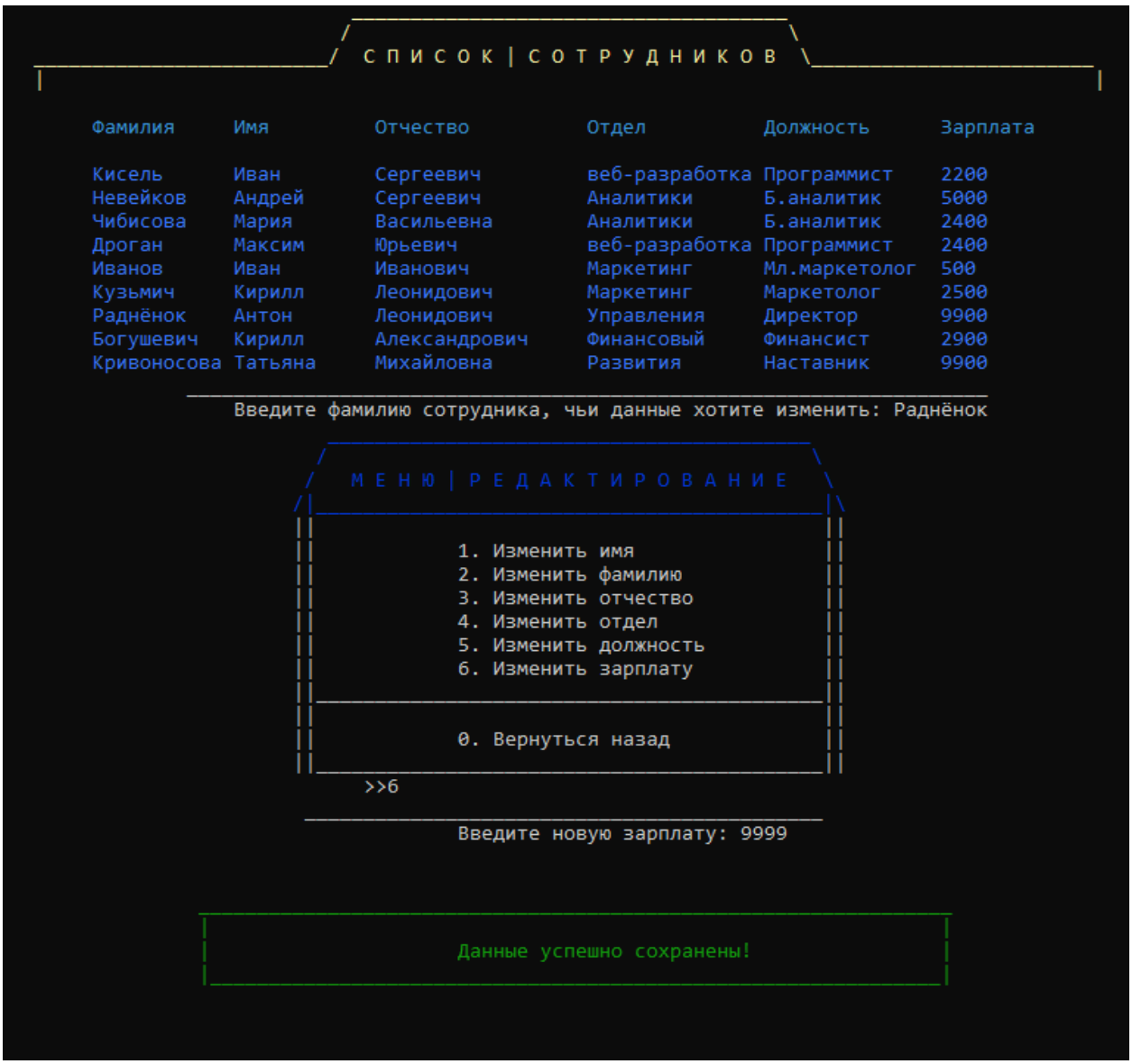

Рисунок 20 – редактирование данных сотрудников

При выборе №4 (Удаление сотрудника)

Выводится список существующих сотрудников, после чего нужно указать фамилию сотрудника.

Если такой сотрудник есть в базе данных → система запросит подтверждение для удаления т.к. данные после удаления восстановить будет невозможно.

Если подтверждение получено, то сотрудник будет удален и появится сообщение.

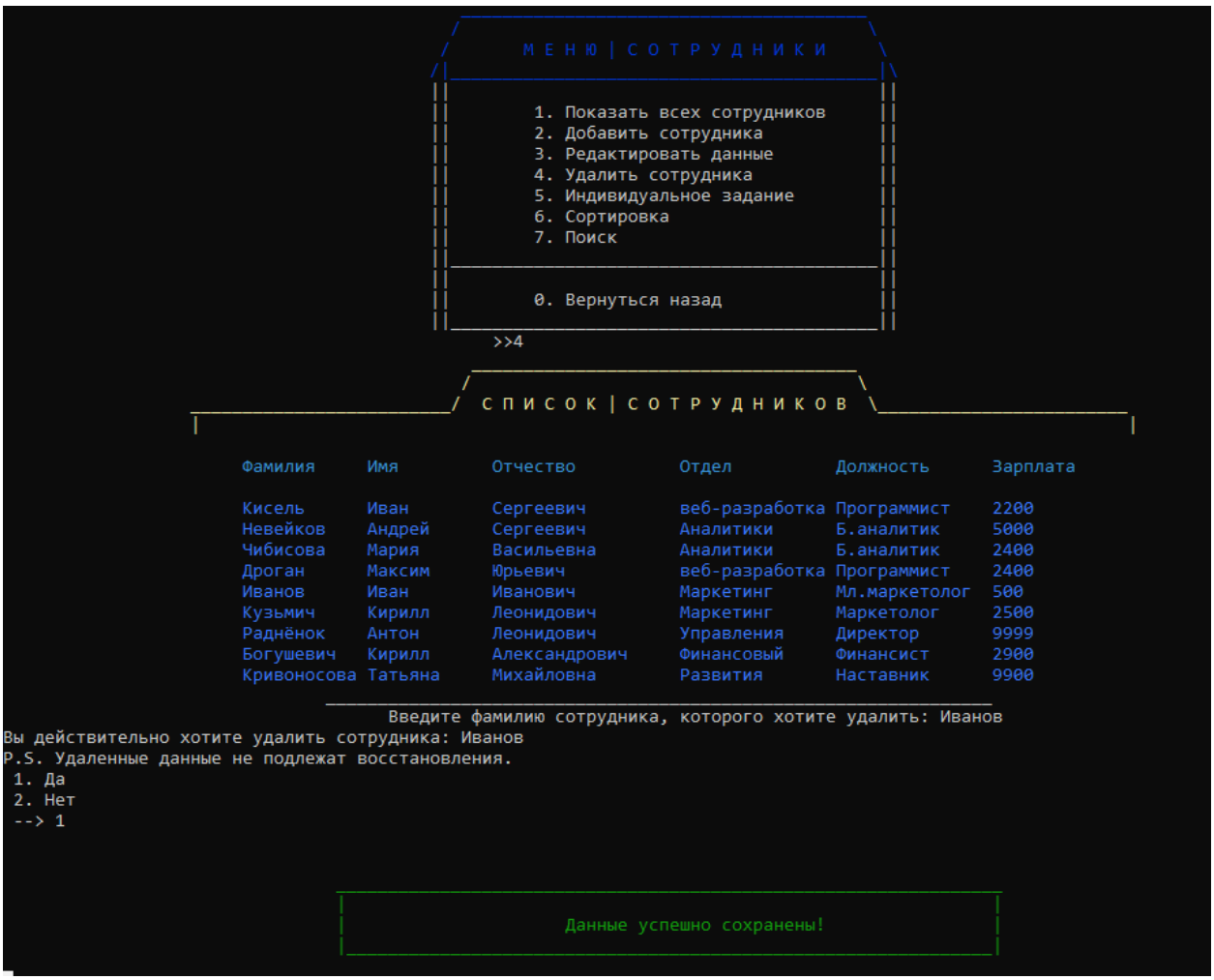

Рисунок 21 – удаление сотрудника

Пункт №5 Индивидуальное задание

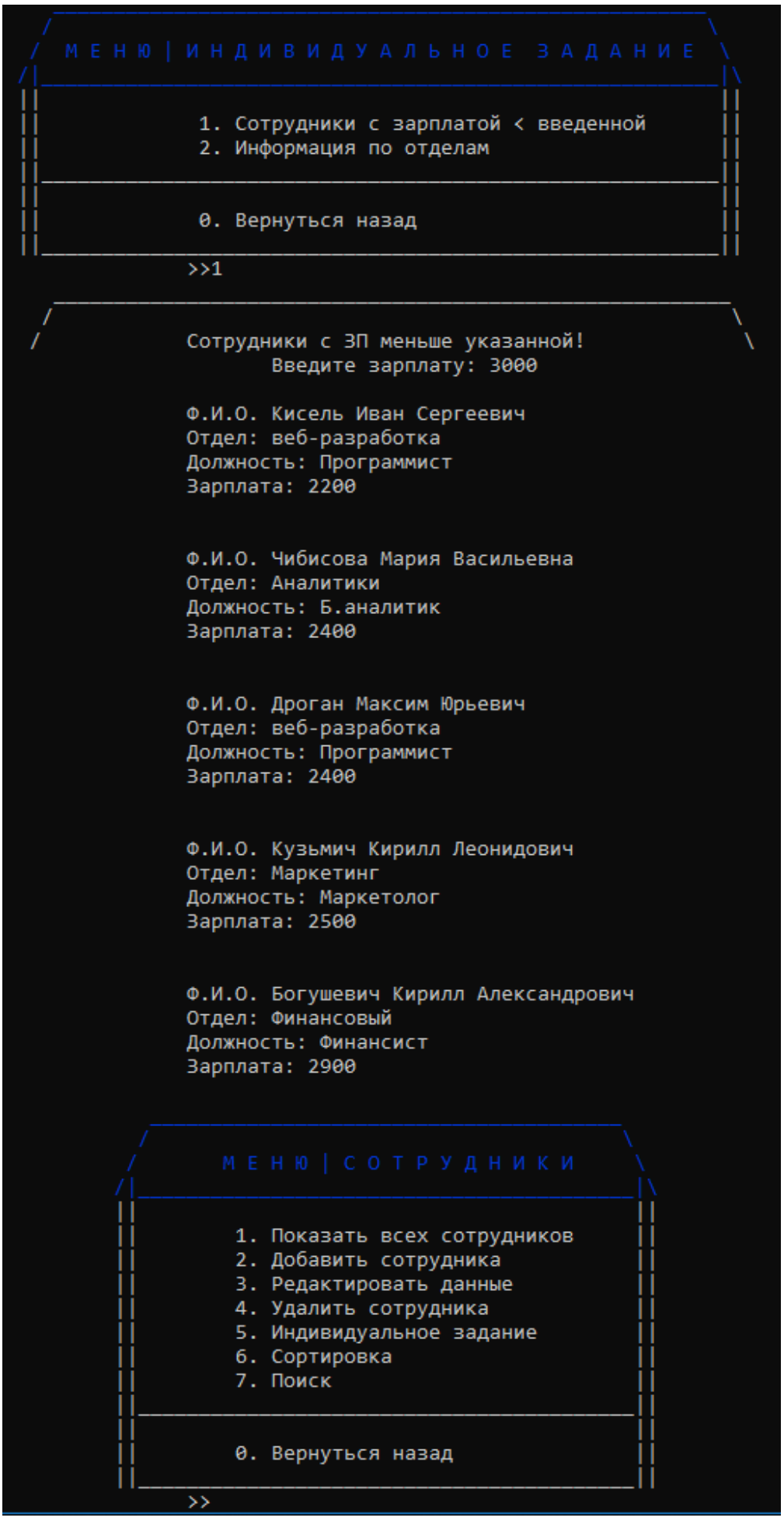
МЕНЮ | ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
1. Сотрудники с зарплатой < введенной
2. Информация по отделам
0. Вернуться назад
>>1
Сотрудники с ЗП меньше указанной!
Введите зарплату: 3000
Ф.И.О. Кисель Иван Сергеевич
Отдел: веб-разработка
Должность: Программист
Зарплата: 2200
Ф.И.О. Чибисова Мария Васильевна
Отдел: Аналитики
Должность: Б.аналитик
Зарплата: 2400
Ф.И.О. Дроган Максим Юрьевич
Отдел: веб-разработка
Должность: Программист
Зарплата: 2400
Ф.И.О. Кузьмич Кирилл Леонидович
Отдел: Маркетинг
Должность: Маркетолог
Зарплата: 2500
Ф.И.О. Богушевич Кирилл Александрович
Отдел: Финансовый
Должность: Финансист
Зарплата: 2900
МЕНЮ | СОТРУДНИКИ
1. Показать всех сотрудников
2. Добавить сотрудника
3. Редактировать данные
4. Удалить сотрудника
5. Индивидуальное задание
6. Сортировка
7. Поиск
0. Вернуться назад
>>

Рисунок 22 – вывод сотрудник с ЗП меньше введенной.

Пункт №5.2 Вывод информации по отделам


МЕНЮ | ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ
1. Сотрудники с зарплатой < введенной
2. Информация по отделам
0. Вернуться назад
>>2
Отдел: веб-разработка
Сотрудников в отделе: 2
Общая зарплата: 4600
Средняя ЗП: 2300
Отдел: Аналитики
Сотрудников в отделе: 2
Общая зарплата: 7400
Средняя ЗП: 3700
Отдел: Маркетинг
Сотрудников в отделе: 1
Общая зарплата: 2500
Средняя ЗП: 2500
Отдел: Управления
Сотрудников в отделе: 1
Общая зарплата: 9999
Средняя ЗП: 9999
Отдел: Финансовый
Сотрудников в отделе: 1
Общая зарплата: 2900
Средняя ЗП: 2900
Отдел: Развития
Сотрудников в отделе: 1
Общая зарплата: 9900
Средняя ЗП: 9900
МЕНЮ | СОТРУДНИКИ
1. Показать всех сотрудников
2. Добавить сотрудника
3. Редактировать данные
4. Удалить сотрудника
5. Индивидуальное задание
6. Сортировка
7. Поиск
0. Вернуться назад
>>

Рисунок 23 – вывод информации по отделам

Пункт №6.1 Меню сортировки

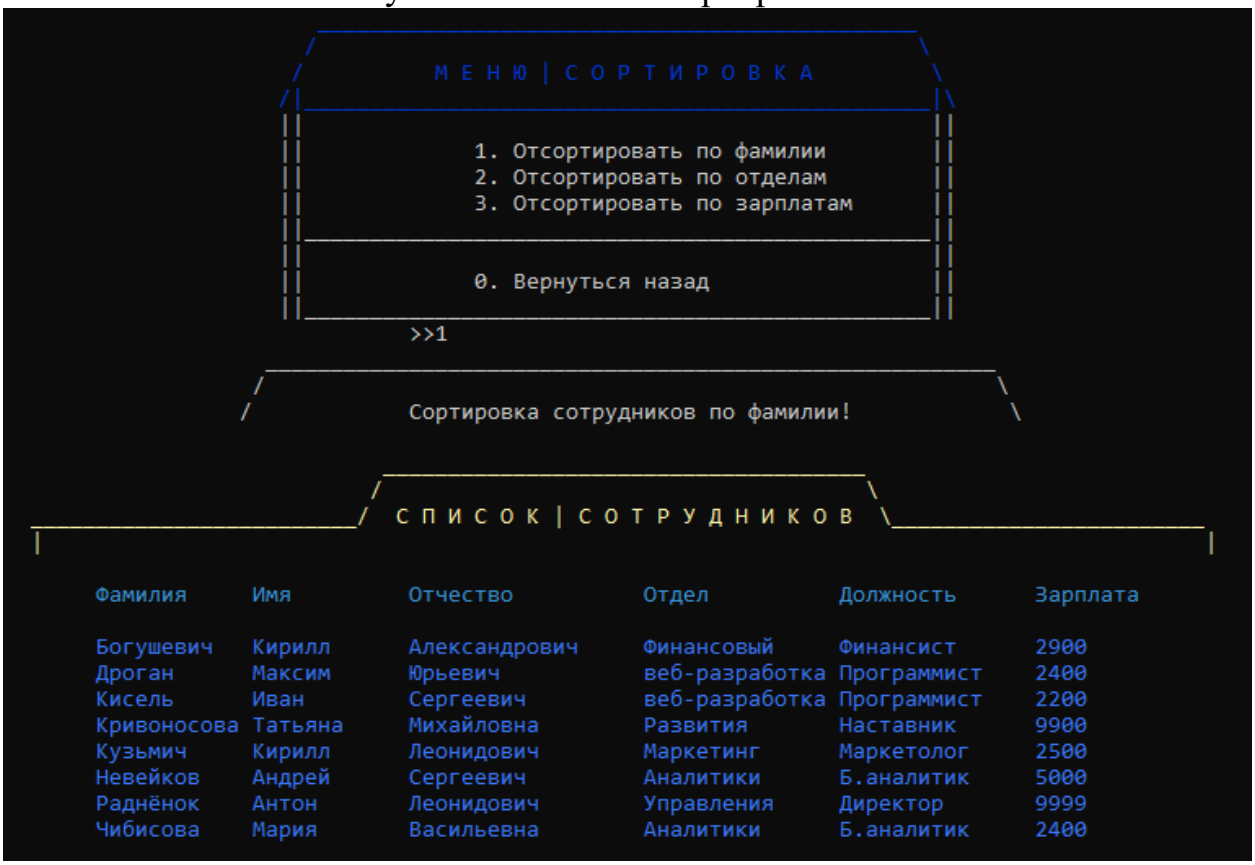

Рисунок 24 – сортировки по фамилии

Пункт №6.3

После успешной сортировки система спросит нужно ли сохранять обновленные данные или же стоит оставить те, что были до этого.

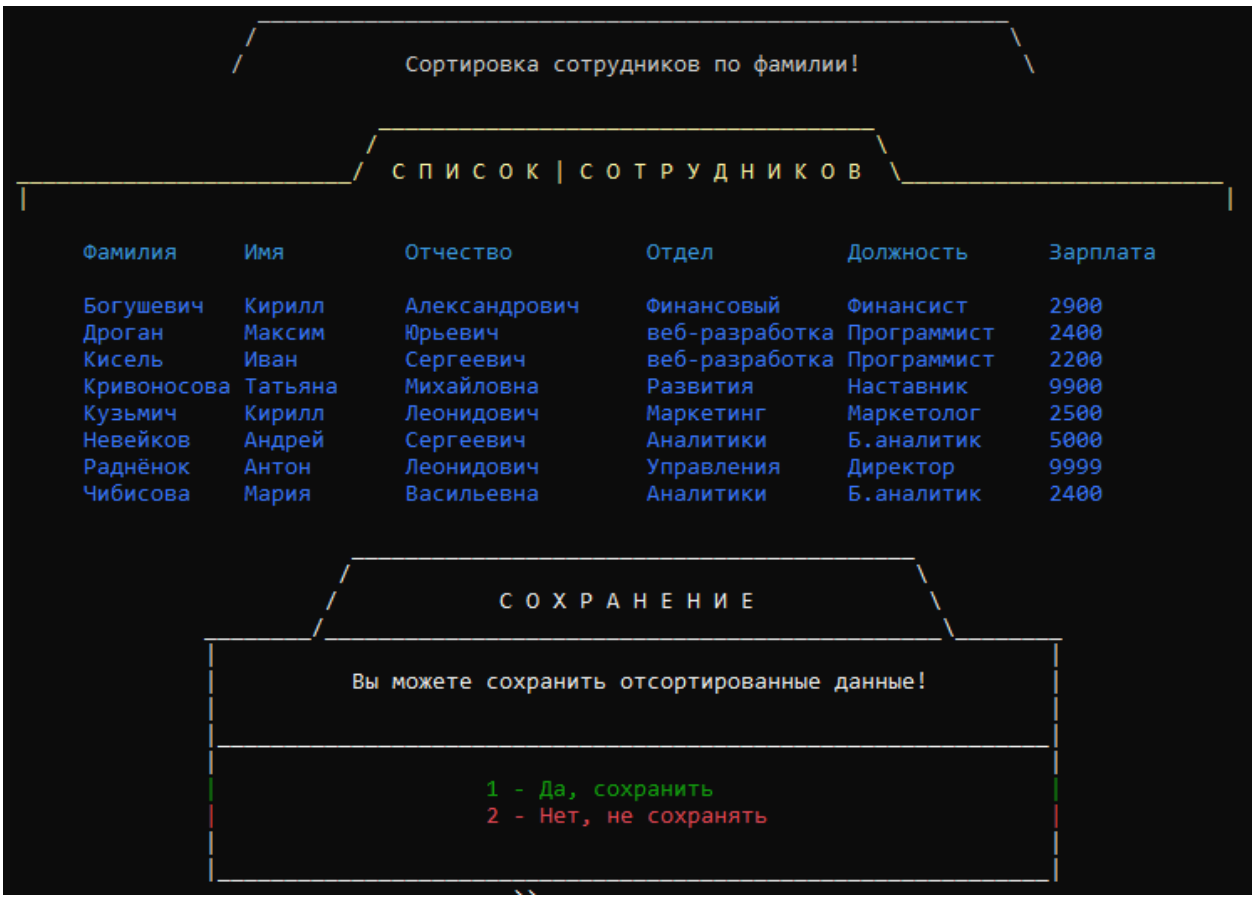

Рисунок 25 – сортировка по зарплатам

№6.2 Сортировка по отделам

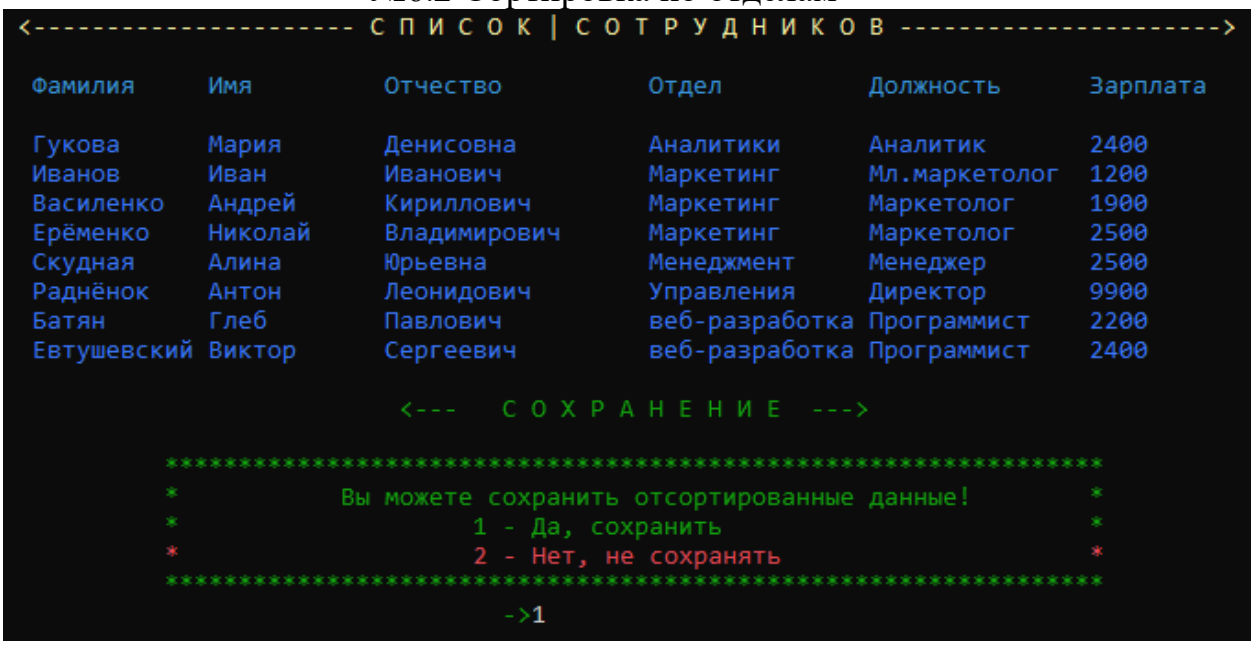

Рисунок 26 – сортировка по отделам

Пункт №7 (Поиск по фамилии)

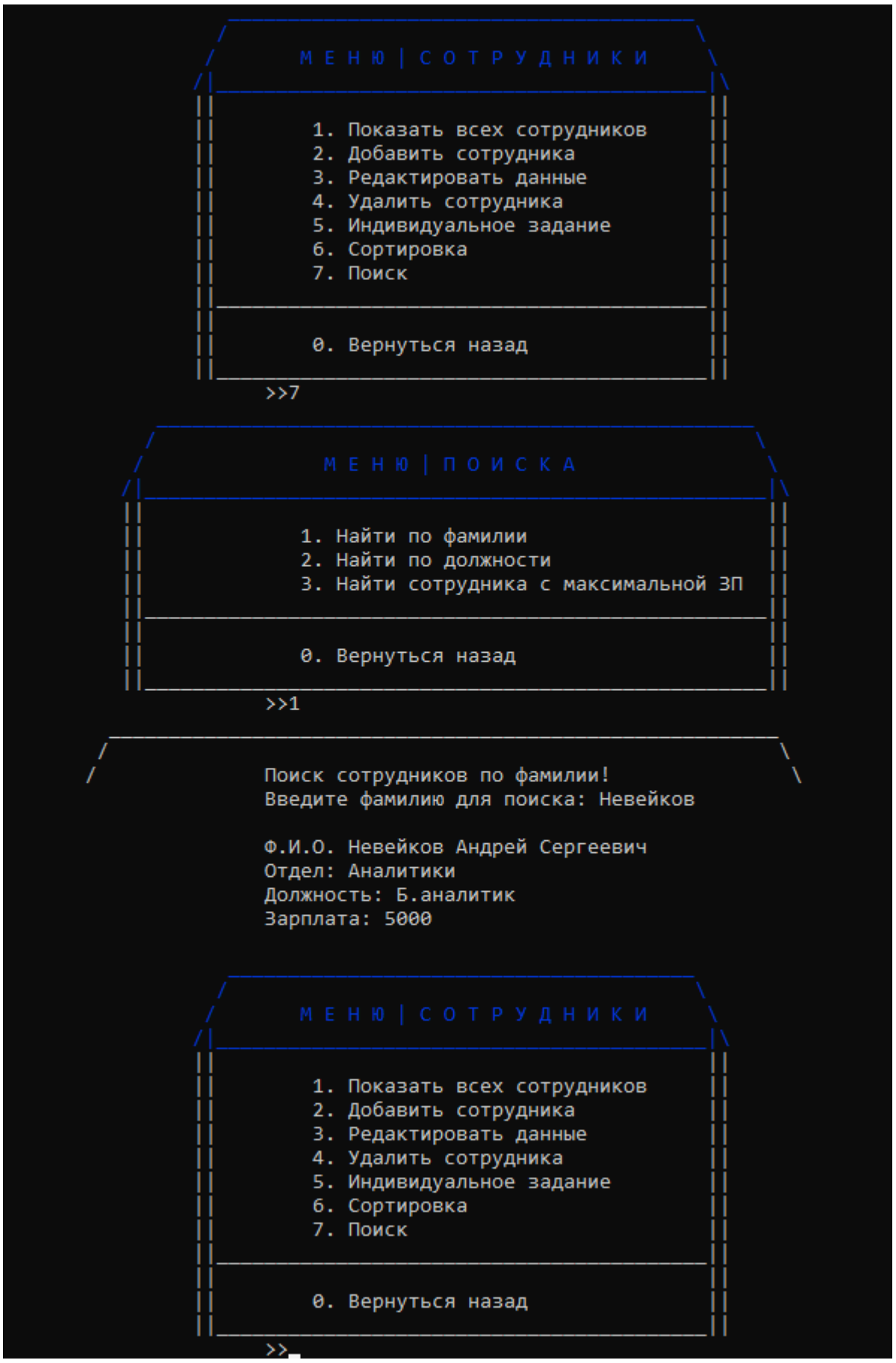

Рисунок 27 – поиск по фамилии

Пункт №7.1 (Поиск по должности)

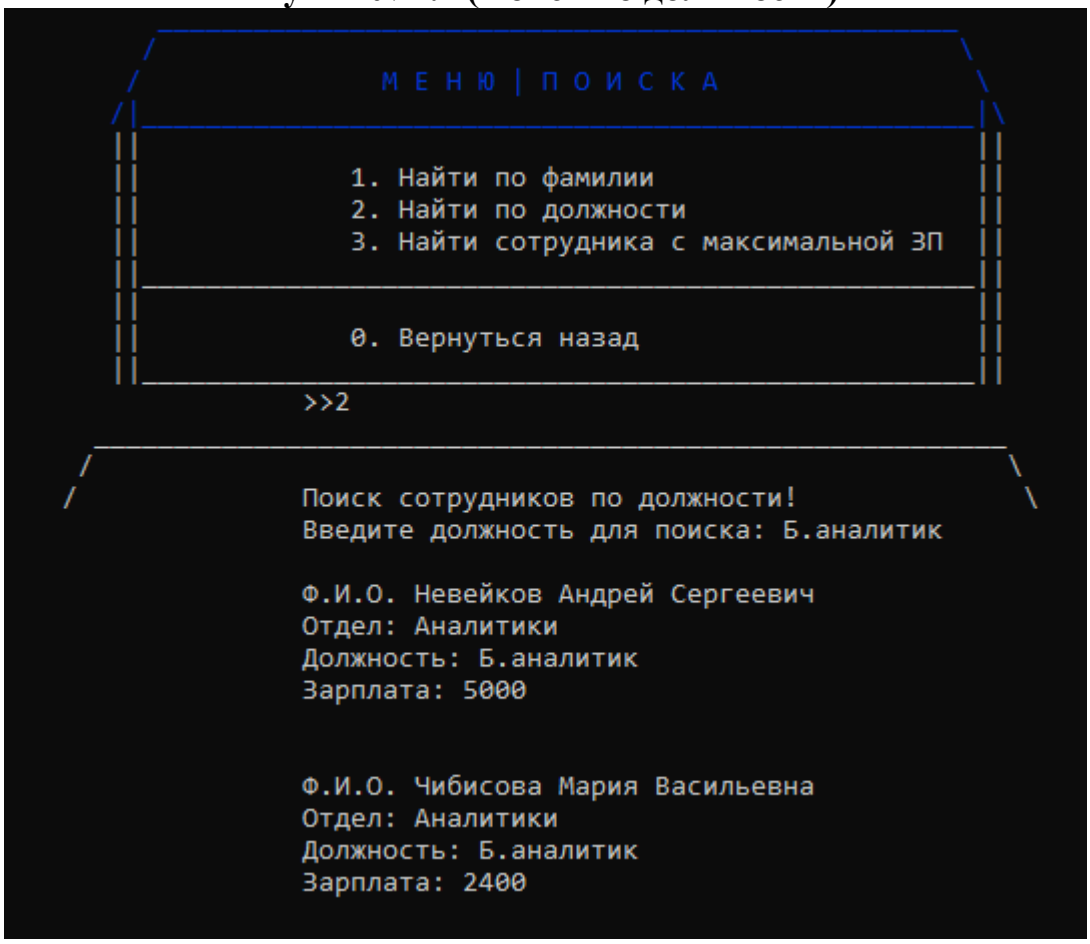

Рисунок 28 – поиск по должности

Пункт №7.3 (Поиск по наибольшей зарплате)

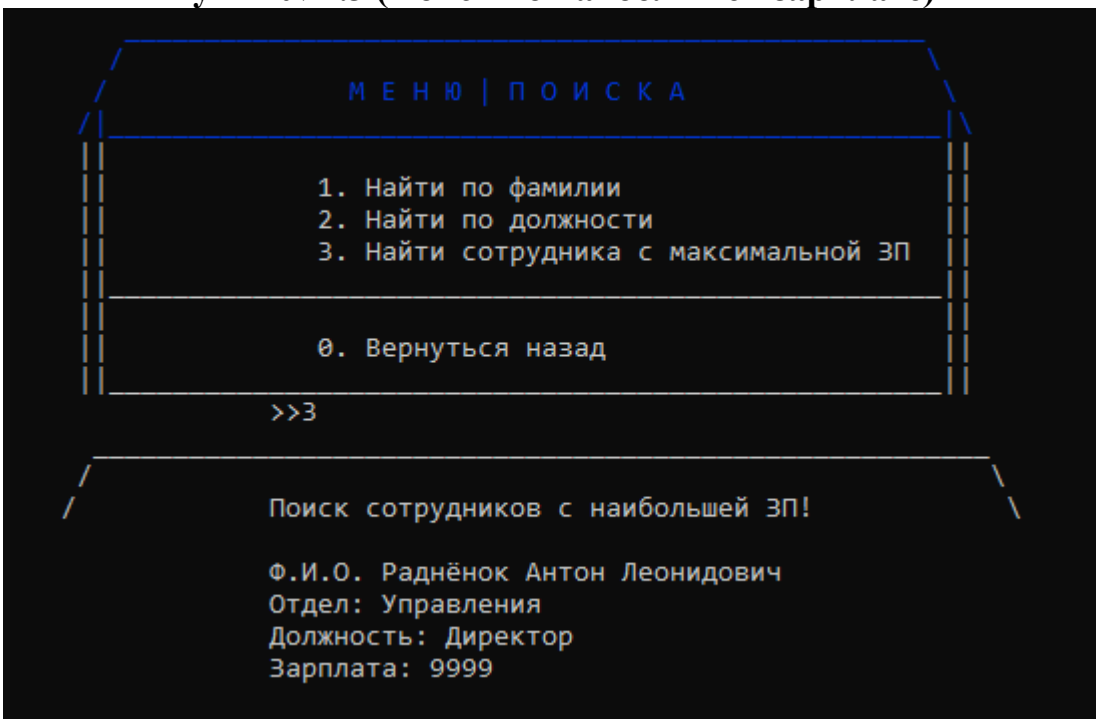

Рисунок 29 – поиск по самой высокой ЗП

Исключительные ситуации:

Предотвращение поломки программы путем ввода некорректных значений:

Защита от символов/превышение диапазона

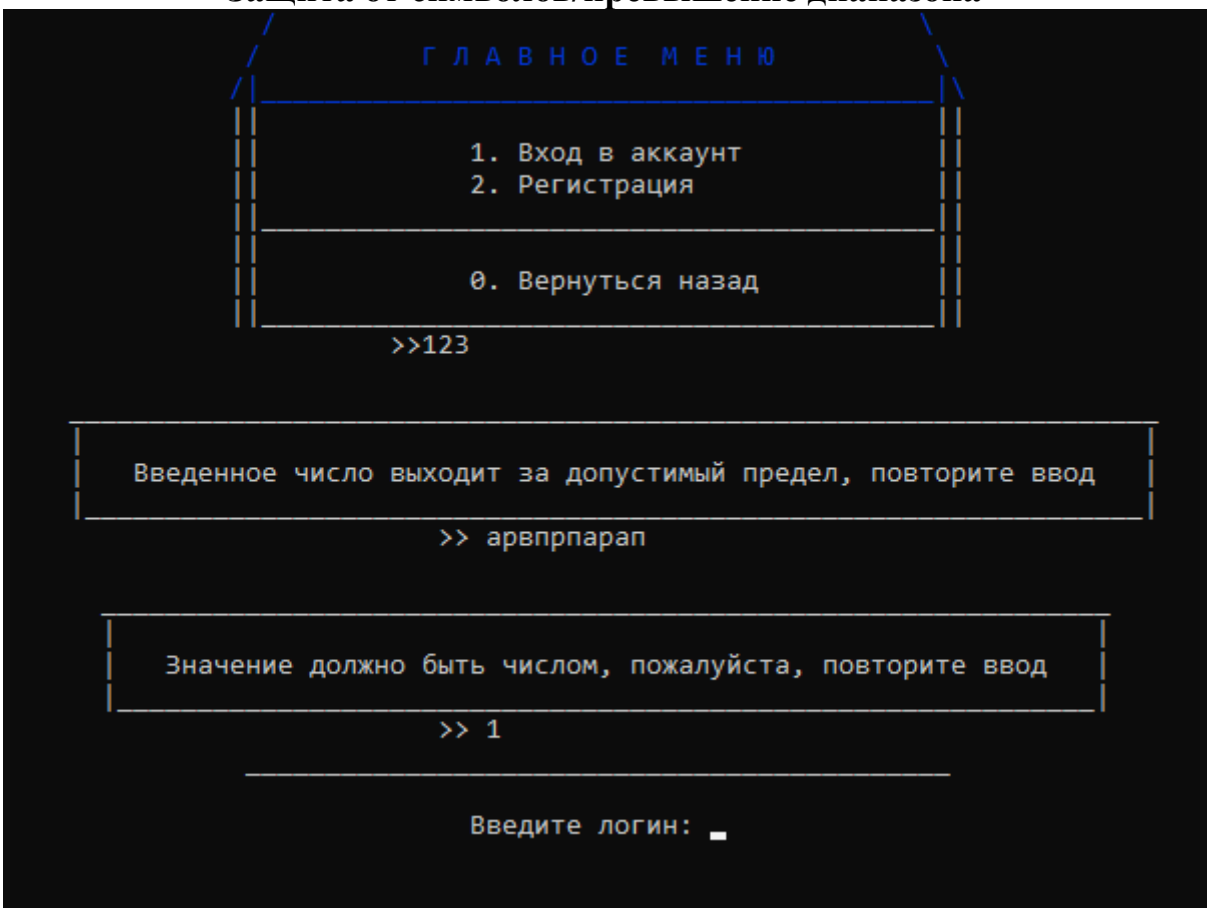

Рисунок 30 – защита от символов и нарушения диапазона

Неверный логин:

Если 3 раза будет введен неверный логин, система выдаст ошибку

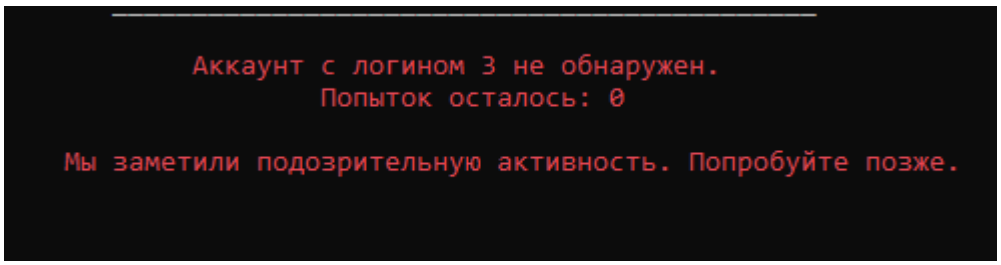

Рисунок 31 – если неверный логин

Неверный пароль

Если 3 раза будет введен неверный пароль, система предложит изменить логин.

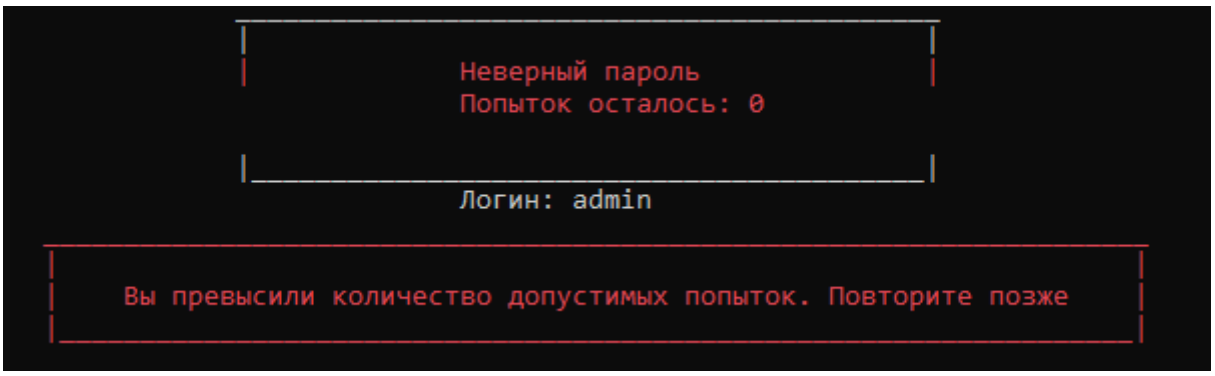

Рисунок 32 – если неверный пароль

Регистрация. Пароль должен быть больше 3(трех) символов

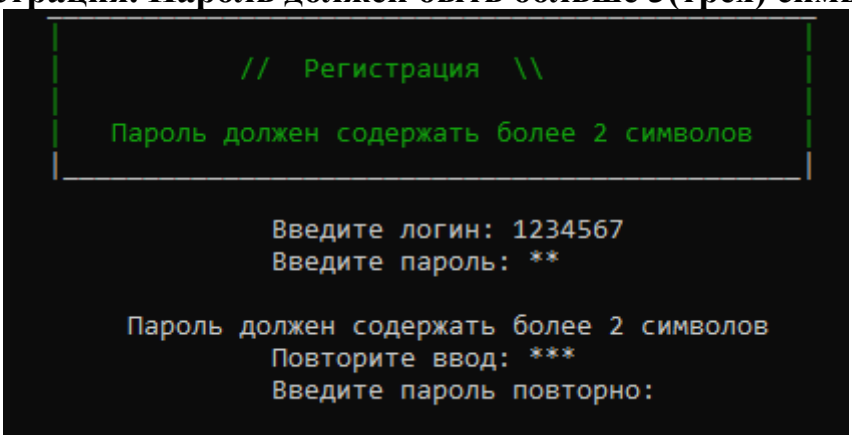

Рисунок 33 – пароль > 3 символов

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг кода с комментариями

Файл main.cpp

```
#include "Accounts.h"

int main()
{
	Start_program(); // Первая функция, которяа запускает программу
	return 0;
}
```

Accounts.h

```
#pragma once // Чтобы не произошел повторный запуск
#include <iostream> // cout
#include <string> // Строки
#include <fstream> // Работа с файлами
#include <iomanip> // Для setw

using namespace std;
const string FILE_INFO_ACCOUNTS = "Accounts.txt"; // Файл для аккаунтов

struct Accounts		// Структура аккаунтов
{
	string login;
	string password;
	int status; // 0 - пользователь | 1 - администратор
	int activation; // 0 - деактивирован | 1 - активирован
};

void Start_program(); // Функция из main для запуска программы
void User_menu(); // Меню для пользователя | Переход
void Admin_menu(Accounts* arr_of_accounts, int& number_of_accounts); // Меню для админа
[Упр.аккаунтами/сотрудниками]
```

```
void Add_accounts(Accounts* arr_of_accounts, int& number_of_accounts); // Добавление
аккаунтов
void Del_accounts(Accounts* arr_of_accounts, int& number_of_accounts); // Для удаление
аккаунтов
void Show_accounts(Accounts* arr_of_accounts, int number_of_accounts); // Вывод аккаунтов
void Check_user_status(Accounts* arr_of_accounts, int& number_of_accounts, int role); //
Проверка кто авторизировался
void Read_file_accounts(Accounts* arr_of_accounts, int number_of_accounts); //
Чтение|Перенос аккаунтов из файла
void Write_file_accounts(Accounts* arr_of_accounts, int number_of_accounts); // Записать
аккаунтов в файл
void Admin_accounts_menu(Accounts* arr_of_accounts, int& number_of_accounts); //
Управление аккаунтами
void Check_accounts_number(int& number_of_accounts); // Определение количества аккаунтов
в файле
void Check_login_and_password(Accounts* arr_of_accounts, int& number_of_accounts); //
Проверка логина и пароля

Accounts* Authorization_menu(Accounts* arr_of_accounts, int& number_of_accounts); // Меню
авторизации [Вход/Регистрация]
Accounts* Memory_reallocation_for_accounts(Accounts* arr_of_accounts, int&
number_of_accounts, int m); // Перевыделение памяти
```

Accounts.cpp

```
#include "Accounts.h"
#include "Protection.h"
#include "Signal.h"

// Функция из main для запуска
программы_____
_____
void Start_program()
{
    SetConsoleCP(1251); // Ввод из консоли/редактора кода
    SetConsoleOutputCP(1251); // Вывод на консоль

    int number_of_accounts = 0;
    Check_accounts_number(number_of_accounts); // Подсчитывает
сколько аккаунтов в файле и обновляем переменную
    Accounts* array_of_accounts = new Accounts[number_of_accounts]; //
Выделение память для массива структур аккаунтов
    Read_file_accounts(array_of_accounts, number_of_accounts); //
Функция чтения из файла | Запись в массивы
    array_of_accounts = Authorization_menu(array_of_accounts,
number_of_accounts); // 1 - вход | 2 - регистрация

    Write_file_accounts(array_of_accounts, number_of_accounts); //
Записывает новые данные в файл
```

```
    delete[] array_of_accounts; // Очистка информации | для
предотвращения утечки памяти
}

// Определение количества
сотрудников_____________________________________________________
_____________________________
void Check_accounts_number(int& number_of_accounts)
{
    string temp; // Временная переменная для считывания данных
    ifstream fin(FILE_INFO_ACCOUNTS, ios::in); // Чтение данных из
файла
    if (!fin.is_open()) // Если первый запуск --> 1 аккаунт
    {
        number_of_accounts = 1;
    }
    else
    {
        while (!fin.eof()) // Возвращает флаг конца файла |
Подсчитываем сколько аккаунтов
        {
            fin >> temp >> temp >> temp >> temp;
            number_of_accounts++;
        }
    }
    fin.close(); // Закрываем файл
}

// Чтение аккаунтов из
файла___________________________________________________________
_____________________________
void Read_file_accounts(Accounts* arr_of_accounts, int
number_of_accounts)
{
    ifstream fin(FILE_INFO_ACCOUNTS, ios::in); // Для чтения аккаунтов
    if (!fin.is_open()) // Если первый запуск --> создаем аккаунт админа
    {
        Set_color(2, 0);
        cout << "
_____________________________________________________  " << endl;
        cout << "                        |                                    |  " <<
endl;
        cout << "                        |       Аккаунт админа >> Логин: admin
Пароль: admin       |  " << endl;
```

```
            cout << "
|_____|  " << endl;
            Set_color(7, 0);
            arr_of_accounts[0].login = "admin";
            arr_of_accounts[0].password = "admin";
            arr_of_accounts[0].status = 1;
            arr_of_accounts[0].activation = 1;
    }

    else
    {
            int number_of_accounts = 0; // Обнуляем переменную
            while (!fin.eof())
            {
                    fin >> arr_of_accounts[number_of_accounts].login >>
arr_of_accounts[number_of_accounts].password >>
                            arr_of_accounts[number_of_accounts].status >>
arr_of_accounts[number_of_accounts].activation;
                    number_of_accounts++;
            }
    }
    fin.close();
}

// Запись аккаунтов в
файл_____
_____
void Write_file_accounts(Accounts* arr_of_accounts, int
number_of_accounts)
{
    ofstream fout(FILE_INFO_ACCOUNTS, ios::out); // Сохранение в файл
    for (int i = 0; i < number_of_accounts; i++)
    {
            fout << arr_of_accounts[i].login << " " <<
arr_of_accounts[i].password << " " << arr_of_accounts[i].status << " " <<
arr_of_accounts[i].activation;
            if (i < number_of_accounts - 1)
                    fout << endl;
    }
    fout.close();
}
```

```
// Вывод(просмотр)
аккаунтов_____
_____
void Show_accounts(Accounts* arr_of_accounts, int number_of_accounts)
{
    cout << "
_____                " << endl;
    cout << "                        /                        \\          " <<
endl;
    cout << "                 _________/          С П И С О К | А К К А У Н Т О В
\\_________   " << endl << endl;
    for (int i = 0; i < number_of_accounts; i++)
    {
            cout << "                                    " <<
                    arr_of_accounts[i].login << " " <<
arr_of_accounts[i].password << " " <<
                    arr_of_accounts[i].status << " " <<
arr_of_accounts[i].activation << endl;;
    }
}

// Добалвение
аккаунтов_____
_____
void Add_accounts(Accounts* arr_of_accounts, int& number_of_accounts)
{
    system("cls");
    cout << "
_____" << endl;
    Set_color(2, 0);
    cout << "                        |                        |" << endl;
    cout << "                        |      //  Регистрация  \\\\          |" <<
endl;
    cout << "                        |                        |" << endl;
    cout << "                        |  Пароль должен содержать более 2
символов  |" << endl;
    Set_color(7, 0);
    cout << "
|_____|" << endl << endl;
    cout << "                        Введите логин: ";
    string TempLogin, TempPassword, TempPasswordRepeat; //
Временный логин/пароль/повторный пароль
    bool flagLogin = true, flagCheckPassword = true, flagPassword = true; //
Флаги для регулирования циклов
```

```
    cin >> TempLogin; // Вводим желаемый логин
    int amount = 3;

    while (flagLogin) // пока true | Проверяем есть ли в программе такой логин
    {
        for (int i = 0; i < number_of_accounts; i++)
        {
            if (TempLogin == arr_of_accounts[i].login) // Если нашли такой же
            {
                system("cls");
                cout << "
_____________________________________________" << endl;
                Set_color(12, 0);
                cout << "                                Аккаунт с логином " << TempLogin << " уже существует" << endl;
                Set_color(7, 0);
                cout << "
_____________________________________________" << endl << endl;
                cout << "                                        Введите логин: ";
                cin >> TempLogin; // Просим ввести заново
            }
        }
        flagLogin = false; // Если ошибок нету --> меняем флаг для выхода из цикла
    }
    arr_of_accounts[number_of_accounts - 1].login = TempLogin; // Записываем логин | -1 т.к. индексирование с 0

    // Пароль
    while (flagPassword)
    {
        cout << "                                    Введите пароль: ";
        TempPassword = Inputing_password(); // Записываем пароль
        while (flagCheckPassword) // Проверяем можно ли такой пароль
        {
            if (TempPassword.size() < 3) // Если пароль < 3 символов --> выводим ошибку
            {
                cout << endl << endl;
```

```
                cout << "                    Пароль должен содержать более 2 символов        " << endl;
                cout << "                                Повторите ввод: ";
                TempPassword = Inputing_password(); // Если < 3
            }
            else
            {
                flagCheckPassword = false; // Ошибок нет --> выходим из цикла
            }
        }
        cout << endl;
        cout << "                            Введите пароль повторно: ";
        TempPasswordRepeat = Inputing_password(); // Для повторного ввода пароля
        if (TempPassword == TempPasswordRepeat) // Если совпали
        {
            arr_of_accounts[number_of_accounts - 1].password = TempPassword; // Записываем пароль
            Save_it();
            flagPassword = false; // Выходим из цикла записи пароля
        }
        else
        {
            Password_incorrect(TempLogin, flagPassword, amount); // В Signal | Выводит ошибку если несколько раз ввели не тот пароль
        }
    }
    arr_of_accounts[number_of_accounts - 1].status = 0; // Роль - пользователь
    arr_of_accounts[number_of_accounts - 1].activation = 0; // Аккаунт деактивирован
    cout << endl;
}

// корректировки аккаунтов_____________________________________________________________

void Change_in_accounts(Accounts* arr_of_accounts, int& number_of_accounts) // Корректировки в аккаунтах | Для админа
{
```

```
    system("cls");
    Show_accounts(arr_of_accounts, number_of_accounts); // Выводим все
аккаунты
    string TempLogin; // Строка для внесения логина
    bool flag = true;
    int i = 0;
    cout << "
_____" << endl <<
endl;
    cout << "                              Введите логин аккаунта, который
хотите изменить: ";
    cin >> TempLogin;

    while (flag && i < number_of_accounts) // Цикл для поиска нужного
аккаунта
    {
            if (TempLogin == arr_of_accounts[i].login) // Как только нашли
подходящий логин меняем flagPassword на false
            {
                    flag = false;
            }
            else
            {
                    i++; // До того как не найдем нужный аккаунт
            }
    }

    if (flag) // Если нету аккаунта
    {
            Account_not_found(); // Выводит что такого аккаунта нет в
системе
    }
    else // Если аккаунт найден --> спрашиваем, что нужно заменить
    {

            int sw;
            Set_color(1, 0); // Синий | Цвет текста | Цвет фона
            cout << "                                    _____
" << endl;
            cout << "                                  /                    \\       " <<
endl;
            cout << "                                 / М Е Н Ю | Р Е Д А К Т И Р О В А
Н И Е  \\       " << endl;
```

```
            cout << "
/|_____________________________________|\\       " << endl;
            Set_color(7, 0); // Светло-серый
            cout << "                                ||         1. Изменить логин          ||
" << endl;
            cout << "                                ||         2. Изменить пароль         ||
" << endl;
            cout << "                                ||         3. Изменить роль           ||
" << endl;
            cout << "                                ||         4. Изменить состояние
||        " << endl;
            cout << "
||_____________________________________||        " << endl;
            cout << "                                ||                                    ||        " <<
endl;
            cout << "                                ||          0. Вернуться назад        ||
" << endl;
            cout << "
||_____________________________________||        " << endl;
            cout << "                                 >>";

            Protection(0, 4, sw);
            switch (sw)
            {
            case 1: // Изменение логина
                  cout << "
_____________________________________" << endl << endl;
                  cout << "                                    Введите новый
логин: ";
                  cin >> arr_of_accounts[i].login;
                  Save_it(); // Новые данные успешно сохранены
                  break;

            case 2: // Изменение пароля
                  cout << "
_____________________________________" << endl << endl;
                  cout << "                                    Введите новый
пароль: ";
                  cin >> arr_of_accounts[i].password;
                  Save_it(); // Новые данные успешно сохранены
                  break;

            case 3: // Изменение статуса
                  if (arr_of_accounts[i].status) // true (1) т.е. админ
```

```
                {
                        arr_of_accounts[i].status = 0; // Изменяем на
пользователя
                        Set_color(2, 0);
                        cout << "
_____" << endl;
                        cout << "                        |
|" << endl;
                        cout << "                        |          Статус успешно
изменен на пользователя!        |" << endl;
                        cout << "
|_____|" << endl;
                        Set_color(7, 0);
                        Sleep(1000);
                        system("cls");
                        break;
                }
                if (!arr_of_accounts[i].status) // false (0) т.е. пользователь
                {
                        arr_of_accounts[i].status = 1; // Изменяем на
админа
                        Set_color(2, 0);
                        cout << "
_____" << endl;
                        cout << "                        |
|" << endl;
                        cout << "                        |          Статус успешно
изменен на администратора!       |" << endl;
                        cout << "
|_____|" << endl;
                        Set_color(7, 0);
                        Sleep(1000);
                        system("cls");
                        break;
                }
        case 4: // Изменение состояния
                if (arr_of_accounts[i].activation) // Если аккаунт
активирован
                {
                        arr_of_accounts[i].activation = 0; // Деактивируем
                        Set_color(12, 0); // Светло-зеленый
                        cout << "
_____" << endl;
```

```
                        cout << "                        |
|" << endl;
                        cout << "                        |              Аккаунт
успешно деактивирован!          |" << endl;
                        cout << "
|_____________________________________________________|" << endl;
                        Set_color(7, 0);
                        Sleep(1000);
                        system("cls");
                        break;
                }
                if (!arr_of_accounts[i].activation) // Если аккаунт НЕ
активирован
                {
                        arr_of_accounts[i].activation = 1; // Активируем
                        Set_color(2, 0); //Зеленый
                        cout << "
_____________________________________________________" << endl;
                        cout << "                        |
|" << endl;
                        cout << "                        |               Аккаунт
успешно активирован!           |" << endl;
                        cout << "
|_____________________________________________________|" << endl;
                        Set_color(7, 0);
                        Sleep(1000);
                        system("cls");
                        break;
                }
        case 0: break;
        }
    }
}

// Выбор действия в меню
авторизации_____________________________________________________________
_____________________________
Accounts* Authorization_menu(Accounts* arr_of_accounts, int&
number_of_accounts)
{
    int sw;
    bool flag = true;
    while (flag)
    {
```

```
            Set_color(1, 0); // Синий | Цвет текста | Цвет фона
            cout << "                                  _____________________________________
" << endl;
            cout << "                                /                                    \\        " <<
endl;
            cout << "                                /        Г Л А В Н О Е  М Е Н Ю
\\        " << endl;
            cout << "
/|_____________________________________|\\       " << endl;
            Set_color(7, 0); // Светло-серый
            cout << "                                ||                                    ||        " <<
endl;
            cout << "                                ||          1. Вход в аккаунт          ||
" << endl;
            cout << "                                ||          2. Регистрация             ||
" << endl;
            cout << "
||_____________________________________||       " << endl;
            cout << "                                ||                                    ||        " <<
endl;
            cout << "                                ||          0. Вернуться назад         ||
" << endl;
            cout << "
||_____________________________________||       " << endl;
            cout << "                                       >>";
            Protection(0, 2, sw);
            switch (sw)
            {
            case 1: // Вход в аккаунт
                  Check_login_and_password(arr_of_accounts,
number_of_accounts); // Проверяем данные для входа
                  break;

            case 2: // Регистрация аккаунта
                  arr_of_accounts =
Memory_reallocation_for_accounts(arr_of_accounts, number_of_accounts,
number_of_accounts + 1); // Перевыделение памяти
                  Add_accounts(arr_of_accounts, number_of_accounts); //
Запись информации
                  break;

            case 0:
                  flag = false;
```

```
                break;
            }
    }
    return arr_of_accounts;
}

// Проверка логина и
пароля_____
_____
void Check_login_and_password(Accounts* arr_of_accounts, int&
number_of_accounts)
{
    bool findLogin = true, errorLogin = true, flagPassword = true, flagGlobal
= true;
    int login, amount = 3;
    string TempLogin, TempPassword;

    while (findLogin) // Цикл для поиска логина
    {
            cout << "
_____" << endl << endl;
            cout << "                                    Введите логин: ";

            cin >> TempLogin; // Просим ввести логин
            for (int i = 0; i < number_of_accounts; i++)
            {
                    if (TempLogin == arr_of_accounts[i].login) // Если такой
логин существует
                    {
                            login = i; // Запоминаем индекс
                            findLogin = false; // Выходим из цикла т.к. нашли
логин
                            errorLogin = false; // Отключаем следующую
запись
                    }
            }
            if (errorLogin) // Сработает если логин не будет найден
            {
                    system("cls");
                    amount--; // Если ввели неверный логин уменьшаем
переменную
                    cout << "
_____" << endl << endl;
                    Set_color(12, 0);
```

```
                cout << "                                Аккаунт с логином " <<
TempLogin << " не обнаружен.     " << endl;
                cout << "                                    Попыток осталось: "
<< amount << "            " << endl;
                Set_color(7, 0);

                if (amount == 0) // Если значение 0 --> выходим из
цикла
                {
                        Set_color(12, 0);
                        cout << endl << "                        Мы заметили
подозрительную активность. Попробуйте позже.      " << endl;
                        Sleep(1500);
                        system("cls");
                        findLogin = false;
                        flagPassword = false; // Отключаем следующий
цикл
                }
        }
    }
    amount = 3; // Обновляем amout
    while (flagPassword)
    {
        cout << "                                Введите пароль: ";
        TempPassword = Inputing_password(); // Для ввода пароля
        if (TempPassword == arr_of_accounts[login].password) // Если
пароль подошел к аккаунту
        {
                if (arr_of_accounts[login].activation) // Если состояние
1(true) --> проверяем роль
                {
                        Check_user_status(arr_of_accounts,
number_of_accounts, arr_of_accounts[login].status); // Проверяем кто
зашел
                }
                else // Если 0 (false) --> значит аккаунт деактивирован
                {

                        Set_color(12, 0);
                        cout << endl << "                    Ошибка! Данный
аккаунт Д Е К А Т И В И Р О В А Н. Ожидайте активацию" << endl <<
endl;
                        Sleep(1500);
                        system("cls");
```

```
                }
                flagPassword = false;
            }
            else // Если пароль не совпал
            {
                Password_incorrect(TempLogin, flagPassword, amount); //
В Signal | Выводит ошибку если несколько раз ввели не тот пароль
            }
    }
}

// Перевыделение
памяти_____
_____

Accounts* Memory_reallocation_for_accounts(Accounts* arr_of_accounts,
int& number_of_accounts, int m) // m принимает значение на 1 больше
{
    Accounts* temp_arr = new Accounts[m]; // Выделяем память

    for (int i = 0; i < number_of_accounts; i++) // Переносим старые
данные
    {
        temp_arr[i].login = arr_of_accounts[i].login;
        temp_arr[i].password = arr_of_accounts[i].password;
        temp_arr[i].status = arr_of_accounts[i].status;
        temp_arr[i].activation = arr_of_accounts[i].activation;
    }

    delete[]arr_of_accounts; // Удаляем старый массив структур
    number_of_accounts = m; // Обновляем информацию о количестве
    arr_of_accounts = temp_arr; // Переносим информацию

    return arr_of_accounts; // Возвращаем обновленный массив
}

// Проверка роли. Выбор вывода меню: Админ |
Пользователь_____
_____
void Check_user_status(Accounts* arr_of_accounts, int&
number_of_accounts, int role)
{
    if (role == 1) // Если админ
    {
```

```
            system("cls");
            Set_color(2, 0);
            cout << "                              //// Вы авторизовались как
администратор \\\\\\\\\\          " << endl << endl;
            Set_color(7, 0);
            Admin_menu(arr_of_accounts, number_of_accounts); //
Включаем меню админа
    }

    else
            if (role == 0) // Если пользователь
            {
                    system("cls");
                    Set_color(2, 0);
                    cout << "                                  //// Вы авторизовались как
пользователь \\\\\\\\\\          " << endl;
                    Set_color(7, 0);
                    User_menu(); // Переход в функционал пользователя
            }
}

// Меню админа для выбора
действия_____________________________________________________________
_____________
void Admin_menu(Accounts* arr_of_accounts, int& number_of_accounts)
{
    int sw;
    bool flag = true;
    while (flag)
    {
            Set_color(1, 0); // Синий | Цвет текста | Цвет фона
            cout << "                                  ____________________________________
" << endl;
            cout << "                                 /                                    \\        " <<
endl;
            cout << "                                 /   М Е Н Ю | А Д М И Н С Т Р А Т О
Р А   \\       " << endl;
            cout << "
/|_____________________________________|\\       " << endl;
            Set_color(7, 0); // Светло-серый
            cout << "                                 ||                                    ||        " <<
endl;
            cout << "                                 ||       1. Работа с аккаунтами
||       " << endl;
```

```
            cout << "                              ||       2. Работа с сотрудниками
||       " << endl;
            cout << "
||_____________________________________||       " << endl;
            cout << "                              ||                              ||       " <<
endl;
            cout << "                              ||       0. Вернуться назад           ||
" << endl;
            cout << "
||_____________________________________||       " << endl;
            cout << "                               >>";

            Protection(0, 2, sw);
            switch (sw)
            {
            case 1: // Работа с аккаунтами
                  system("cls");
                  Admin_accounts_menu(arr_of_accounts,
number_of_accounts); // Меню для управление аккаунтами
                  break;

            case 2: // Работа с сотрудниками
                  system("cls");
                  User_menu();
                  break;

            case 0:
                  system("cls");
                  flag = false; // Для выхода из цикла
                  break;
            }
    }

}

// Меню админа для работы с
аккаунтами_____________________________________________________________
_____________________
void Admin_accounts_menu(Accounts* arr_of_accounts, int&
number_of_accounts)
{
    int sw;
    bool flag = true;
    while (flag)
```

```
    {
        Set_color(1, 0); // Синий | Цвет текста | Цвет фона
        cout << "                                    _____________________________________
" << endl;
        cout << "                                  /                                    \\       " <<
endl;
        cout << "                                  /          А К К А У Н Т Ы           \\
" << endl;
        cout << "
/|_____________________________________|\\      " << endl;
        Set_color(7, 0); // Светло-серый
        cout << "                                  ||                                    ||       " <<
endl;
        cout << "                                  ||       1. Посмотреть аккаунты
||      " << endl;
        cout << "                                  ||       2. Редактировать аккаунты
||      " << endl;
        cout << "                                  ||       3. Удалить аккаунт            ||
" << endl;
        cout << "
||_____________________________________||      " << endl;
        cout << "                                  ||                                    ||       " <<
endl;
        cout << "                                  ||         0. Вернуться назад         ||
" << endl;
        cout << "
||_____________________________________||      " << endl;
        cout << "                                   >>";

        Protection(0, 3, sw);
        switch (sw)
        {
        case 1: // Просмотр аккаунтов
            system("cls");
            Show_accounts(arr_of_accounts, number_of_accounts);
            break;

        case 2: // Корректировки аккаунтов
            Change_in_accounts(arr_of_accounts, number_of_accounts);
            break;

        case 3: // Удаление аккаунта
            system("cls");
            Show_accounts(arr_of_accounts, number_of_accounts);
```

```
                Del_accounts(arr_of_accounts, number_of_accounts);
                break;

        case 0: // Для выхода из while
                flag = false;
                system("cls");
                break;
        }
    }

}

// Для удаления
аккаунта_____
_____
void Del_accounts(Accounts* arr_of_accounts, int& number_of_accounts)
{
    string delLogin; // Для логина аккаунта, который хотим удалить
    cout << "
_____" << endl <<
endl;
    cout << "                    Введите логин аккаунта, который
хотите удалить: ";

    cin >> delLogin;
    int i = 0, del = 0;
    bool flag = true;

    while (flag && i <= number_of_accounts)
    {
        if (arr_of_accounts[i].login == delLogin) // Поиск аккаунта
        {
            flag = false; // Если нашли --> выходим из цикла
        }
        else
        {
            i++;
        }
    }

    if (flag) // Если аккаунта нет
    {
        Account_not_found();
    }
```

```
    else // Если аккаунт есть
    {
            if (arr_of_accounts[i].status == 1) // Защита чтобы нельзя было
удалить аккаунт админа
            {
                    Set_color(12, 0);
                    cout << endl;
                    cout << "                                        _____________" << endl;
                    cout << "                                      /             \\
" << endl;
                    cout << "                                     /   О Ш И Б К А   \\
" << endl << endl;
                    cout << "
_____________________/_____________________\\_____________________" << endl;
                    cout << "                          |                                    |"
<< endl;
                    cout << "                          | У Вас недостаточно прав для
удаления аккаунта администратора! |" << endl;
                    cout << "                          |         Сперва Вам следует
изменить роль аккаунта!         |" << endl;
                    cout << "
|_____________________________________________________________|" << endl;
                    Set_color(7, 0);
                    Sleep(2000);
                    system("cls");
            }
            else
            {
                    cout << "Вы действительно хотите удалить аккаунт: " <<
arr_of_accounts[i].login << endl;
                    cout << "P.S. Удаленные данные не подлежат
восстановления." << endl;
                    cout << " 1. Да" << "\n 2. Нет" << "\n --> ";
                    Protection(1, 2, del);
                    if (del == 1)
                    {
                            for (int j = i; j < number_of_accounts - 1; j++) // i ->
номер аккаунта | Смещение в конец удаленного аккаунта
                            {
                                    arr_of_accounts[j] = arr_of_accounts[j + 1];
                            }
                            number_of_accounts--; // Уменьшаем количество
аккаунтов
                            Save_it(); // Данные успешно сохранены
```

```
                }
                else
                {
                        cout << "Удаление отменено" << endl;
                }
            }
        }
    }
```

Employees.h

```
#pragma once
#include <iostream>
#include <string>
#include <fstream>
#include <iomanip>

using namespace std;
const string FILE_INFO_EMPLOYEES = "Employee.txt";

struct Employee      // Структура сотрудников
{
        string Name;
        string Surname;
        string Patronymic;
        string Department;
        string Position;
        int Salary;
};

struct Department // Структура отделов
{
        string Department_name; // Название отдела
        int Employees_amount = 0; // Количество сотрудников отдела
        int Total_department_salary = 0; // ЗП всего отдела
};

// ОСНОВНЫЕ
ФУНКЦИИ_____________________________________________________________________________
_____________________________

void User_menu(); // Меню пользователя
void Check_employees_file(int& number_of_Employees); // Определение количества
сотрудников
void Show_employees(Employee* arr_of_Employees, int number_of_Employees); // Вывод
сотрудников
void Add_employees(Employee* arr_of_Employees, int& number_of_Employees); // Добавление
сотрудников
void Delete_employees(Employee* arr_of_Employees, int& number_of_Employees); // Удаление
сотрудников
void Write_to_employees_file(Employee* arr_of_Employees, int number_of_Employees); //
Запись в файл
void Read_from_employees_file(Employee* arr_of_Employees, int number_of_Employees); //
Чтение из файла
void Updating_of_employees_information(Employee* arr_of_Employees, int&
number_of_Employees); // Обновление информации о сотрудниках

Employee* Memory_reallocation_for_employees(Employee* arr_of_Employees, int&
number_of_Employees, int m); // Перевыделение памяти
```

```
Employee* Menu_employees(Employee* arr_of_Employees, int& number_of_Employees); // Меню для выбора действия


// ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ_____________________________________________________________________________

void Menu_individual_task(Employee* arr_of_Employees, int number_of_Employees); // Выбор индивидуального задания
void Individual_task_salary_info(Employee* arr_of_Employees, int number_of_Employees); // Индивидуальное задание №1: Вывод сотрудников с ЗП < введенной с клавиатуры
void Individual_task_departments_info(Employee* arr_of_Employees, int number_of_Employees); // Индивидуальное задание №2: Инфа по отделам | Общая ЗП | Средняя ЗП

Department* Individual_task_department_memory_reallocation(Department* arr_of_dep, int& number_of_dep, int m); // Индивидуальное задание №2.1: Перевыделение памяти


// СОРТИРОВКА_____________________________________________________________________________

Employee* Sorting_by_Surname(Employee* arr_of_Employees, int number_of_Employees); // Сортировка №1: По фамилиям
Employee* Sorting_by_department(Employee* arr_of_Employees, int number_of_Employees); // Сортировка №2: По отделам
Employee* Sorting_by_the_salary(Employee* arr_of_Employees, int number_of_Employees); // Сортировка №3: По зарплате
Employee* Selection_sorting_menu(Employee* arr_of_Employees, int number_of_Employees); // Выбор сортировки


// ПОИСК_____________________________________________________________________________

void Find_surname(Employee* arr_of_Employees, int number_of_Employees); // Поиск №1: По фамилии
void Find_position(Employee* arr_of_Employees, int number_of_Employees); // Поиск №2: По должности
void Find_max_salary(Employee* arr_of_Employees, int number_of_Employees); // Поиск №3: По максимальной ЗП
void Show_result_of_finding(Employee* arr_of_Employees, int number_of_Employees, int i); // Вывод для поиска
void Menu_of_finding_employees(Employee* arr_of_Employees, int number_of_Employees); // Выбор поиска
```

Employees.cpp

```
#include "Employees.h"
#include "Protection.h"
#include "Signal.h"

// Структура работников >>  ФИО  |       Отдел        | Должность  | ЗП
// Структура отделов    >> Отдел | Кол-во сотрудников | Общая ЗП
// ИДЗ >> Сумма выплат по отделам | Средняя ЗП по отделам | Вывести сотрудников с ЗП ниже введенной
// Стартовое меню пользователя для перехода из файла с аккаунтами
```

```
void User_menu()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    int number_of_Employees = 0;
    Check_employees_file(number_of_Employees); // Определение размера
массива
    Employee* arr_of_Employees = new Employee[number_of_Employees]; //
Выделяем новую память
    Read_from_employees_file(arr_of_Employees, number_of_Employees); //
Чтение файла и перенос данных в массивы
    arr_of_Employees = Menu_employees(arr_of_Employees,
number_of_Employees); // Меню сотрудников

    Write_to_employees_file(arr_of_Employees, number_of_Employees); //
Записываем сотрудников
    delete[] arr_of_Employees;
}

// Определение количества
сотрудников_____
_____

void Check_employees_file(int& number_of_Employees)
{
    string temp;
    ifstream fin(FILE_INFO_EMPLOYEES, ios::in); // Открыть файл для
чтения
    if (!fin.is_open())
    {
        number_of_Employees = 1;
    }
    else
    {
        while (!fin.eof()) // Пока не дойдем до коцна файла
        {
            fin >> temp >> temp >> temp >> temp >> temp >> temp;
            number_of_Employees++;
        }
    }
    fin.close();
}
```

```
// Чтение и запись информации в массивы
структур_____
_____

void Read_from_employees_file(Employee* arr_of_Employees, int
number_of_Employees)
{
    ifstream fin(FILE_INFO_EMPLOYEES, ios::in);
    if (!fin.is_open()) // Если пусто
    {
            Set_color(12, 0);
            cout << "
_____
" << endl;
            cout << "                        |                                   |"
<< endl;
            cout << "                        |          Список сотрудников пуст!
|" << endl;
            cout << "                        |      Пожалуйста, добавьте первого
сотрудника!        |" << endl;
            cout << "
|_____|"
<< endl;
            Set_color(7, 0);
            Add_employees(arr_of_Employees, number_of_Employees); //
Запускаем функцию добавления сотрудников
    }
    else // Если сотрудники уже есть в файле
    {
            int i = 0;
            while (!fin.eof()) // Считываем всю информацию из файла
            {
                    fin >> arr_of_Employees[i].Name >>
arr_of_Employees[i].Surname >> arr_of_Employees[i].Patronymic >>
                            arr_of_Employees[i].Department >>
arr_of_Employees[i].Position >> arr_of_Employees[i].Salary;
                    i++;
            }
    }
    fin.close();
}
```

```
// Запись в файл_____________________________________________________
_____________________________________________________

void Write_to_employees_file(Employee* arr_of_Employees, int number_of_Employees)
{
    ofstream fout(FILE_INFO_EMPLOYEES, ios::out); // Для сохранение в файл
    for (int i = 0; i < number_of_Employees; i++)
    {
            fout << arr_of_Employees[i].Name << " " << arr_of_Employees[i].Surname << " " << arr_of_Employees[i].Patronymic << " " <<
                    arr_of_Employees[i].Department << " " << arr_of_Employees[i].Position << " " << arr_of_Employees[i].Salary;
            if (i < number_of_Employees - 1) // endl пока не дойдем до последнего сотрудника
                    fout << endl;
    }
    fout.close();
}

// Меню сотрудников (Выбор действия)_____________________________________
_____________________________________

Employee* Menu_employees(Employee* arr_of_Employees, int& number_of_Employees)
{
    bool flag = true; // Для остановки цикла
    while (flag)
    {

            Set_color(1, 0); // Синий | Цвет текста | Цвет фона
            cout << "                                                  _____________________________________________          " << endl;
            cout << "                                                 /                                             \\        " << endl;
            cout << "                                                /      М Е Н Ю | С О Т Р У Д Н И К И     \\       " << endl;
            cout << "                                                /|_____________________________________________|\\       " << endl;
            Set_color(7, 0); // Светло-серый
```

```
            cout << "                                ||                                ||       "
<< endl;
            cout << "                                ||       1. Показать всех
сотрудников    ||       " << endl;
            cout << "                                ||       2. Добавить сотрудника
||       " << endl;
            cout << "                                ||       3. Редактировать данные
||       " << endl;
            cout << "                                ||       4. Удалить сотрудника
||       " << endl;
            cout << "                                ||       5. Индивидуальное задание
||       " << endl;
            cout << "                                ||       6. Сортировка                ||
" << endl;
            cout << "                                ||       7. Поиск                     ||
" << endl;
            cout << "
||_____________________________________________||       " << endl;
            cout << "                                ||                                ||       "
<< endl;
            cout << "                                ||       0. Вернуться назад
||       " << endl;
            cout << "
||_____________________________________________||       " << endl;
            cout << "                                        >>";

            int sw;
            Protection(0, 7, sw); // Защита на ввода от букв и диапазон [0-7]
            switch (sw)
            {
            case 1: // Вывод инфу о сотрудниках
                  system("cls");
                  Show_employees(arr_of_Employees, number_of_Employees);
                  break;
            case 2: // Добавление новых сотрудников
                  arr_of_Employees =
Memory_reallocation_for_employees(arr_of_Employees,
number_of_Employees, number_of_Employees + 1); // перераспределение
памяти т.к. мы добавляем сотрудников
                  Add_employees(arr_of_Employees, number_of_Employees); //
Для добавления
                  break;
            case 3: // Обновление информации о сотрудниках
```

```
                Updating_of_employees_information(arr_of_Employees,
number_of_Employees);
                break;
            case 4: // Удаление
                Show_employees(arr_of_Employees, number_of_Employees);
                Delete_employees(arr_of_Employees, number_of_Employees);
                break;
            case 5: // ИДЗ
                Menu_individual_task(arr_of_Employees,
number_of_Employees);
                break;
            case 6: // Сортировка
                arr_of_Employees =
Selection_sorting_menu(arr_of_Employees, number_of_Employees);
                break;
            case 7: // Поиск
                Menu_of_finding_employees(arr_of_Employees,
number_of_Employees);
                break;
            case 0:
                system("cls");
                flag = false;
                break; // Если хотят выйти меняем флаг на false
            }
    }
    return arr_of_Employees;
}

// ПОКАЗАТЬ ВСЕХ
СОТРУДНИКОВ_____________________________________________
_____________________________________________________________
___

void Show_employees(Employee* arr_of_Employees, int
number_of_Employees)
{
    Set_color(14, 0);
    cout << "
_____________________________________________                    " << endl;
    cout << "                                  /                    \\
" << endl;
    cout << "                 _____________________________/  С П И С О К | С О Т
Р У Д Н И К О В  \\_____________________________ " << endl;
```

```
    cout << "                |
| " << endl << endl;
    Set_color(3, 0);
    cout << "                    " <<
            setw(12) << left << "Фамилия" <<
            setw(12) << left << "Имя" <<
            setw(18) << left << "Отчество" <<
            setw(15) << left << "Отдел" <<
            setw(15) << left << "Должность" <<
            setw(15) << left << "Зарплата" << endl;

    cout << endl;
    for (int i = 0; i < number_of_Employees; i++)
    {
            Set_color(9, 0);
            cout << "                    " <<
                    setw(12) << left << arr_of_Employees[i].Surname <<
                    setw(12) << left << arr_of_Employees[i].Name <<
                    setw(18) << left << arr_of_Employees[i].Patronymic <<
                    setw(15) << left << arr_of_Employees[i].Department <<
                    setw(15) << left << arr_of_Employees[i].Position <<
                    setw(15) << left << arr_of_Employees[i].Salary << endl;
    }
    Set_color(7, 0);
}

// Перевыделение
памяти__________________________________________________________________
_____________________________________________________

Employee* Memory_reallocation_for_employees(Employee*
arr_of_Employees, int& number_of_Employees, int m)
{

    Employee* temp_arr = new Employee[m]; // Выделяем память под массив
структур
    for (int i = 0; i < number_of_Employees; i++) // Переносим старые данных
    {
            temp_arr[i].Name = arr_of_Employees[i].Name;
            temp_arr[i].Surname = arr_of_Employees[i].Surname;
            temp_arr[i].Patronymic = arr_of_Employees[i].Patronymic;
            temp_arr[i].Department = arr_of_Employees[i].Department;
            temp_arr[i].Position = arr_of_Employees[i].Position;
            temp_arr[i].Salary = arr_of_Employees[i].Salary;
```

```
    }

    delete[]arr_of_Employees; // Удаляем старые данные
    number_of_Employees = m; // Обновляем количество сотрудников
    arr_of_Employees = temp_arr; // Переносим информацию
    return arr_of_Employees; // Возвращаем обновленную инфу
}

// ДОБАВЛЕНИЕ НОВОГО
СОТРУДНИКА_____________________________________________________
_____________________________________________

void Add_employees(Employee* arr_of_Employees, int&
number_of_Employees)
{
    Set_color(14, 0);
    cout << "
_____________________________________________________________________
" << endl;
    cout << "                        |                                    |" <<
endl;
    cout << "                        |          Добавление нового сотрудника!
|" << endl;
    cout << "
|____________________________________________________________________|"
<< endl;
    Set_color(7, 0);
    cout << "                                    Введите фамилию: ";
    cin >> arr_of_Employees[number_of_Employees - 1].Surname; // -1 т.к.
индексирование с 0
    cout << "                                    Введите имя: ";
    cin >> arr_of_Employees[number_of_Employees - 1].Name;
    cout << "                                    Введите отчество: ";
    cin >> arr_of_Employees[number_of_Employees - 1].Patronymic;
    cout << "                                    Введите отдел: ";
    cin >> arr_of_Employees[number_of_Employees - 1].Department;
    cout << "                                    Введите должность: ";
    cin >> arr_of_Employees[number_of_Employees - 1].Position;
    cout << "                                    Введите зарплату: ";
    Protection(0, 10000000000000000,
arr_of_Employees[number_of_Employees - 1].Salary);
    Save_it();
}
```

```
// РЕДАКТИРОВАТЬ ИНФОРМАЦИЮ О СОТРУДНИКЕ_____________________________________
_____________________________________

void Updating_of_employees_information(Employee* arr_of_Employees, int& number_of_Employees)
{
    system("cls");
    Show_employees(arr_of_Employees, number_of_Employees); // Выводим на экран всех сотрудников
    string temp; // Строка для фамилии
    bool flag = true;
    int i = 0;
    cout << "
_____________________________________________________________________________
____" << endl;
    cout << "                              Введите фамилию сотрудника, чьи данные хотите изменить: ";
    cin >> temp;
    while (flag && i < number_of_Employees) // Цикл для поиска фамилии | Пока flag true и пока не перебрали всех сотрудников
    {
        if (temp == arr_of_Employees[i].Surname) // Нашли совпадение --> прекращаем цикл
        {
            flag = false;
        }
        else // Иначе идём дальше
        {
            i++;
        }
    }

    if (flag) // Если ничего не нашли
    {
        Account_not_found(); // Учетной записи с такой фамилией нету
    }
    else // Если нашли | После цикла while --> спрашиваем что нужно изменить
    {

        Set_color(1, 0); // Синий | Цвет текста | Цвет фона
        cout << "
_____________________________________________          " << endl;
```

```
            cout << "                                /                                \\          "
<< endl;
            cout << "                               /   М Е Н Ю | Р Е Д А К Т И Р О
В А Н И Е   \\         " << endl;
            cout << "
/|_____|\\         " << endl;
            Set_color(7, 0); // Светло-серый
            cout << "                              ||                                ||         "
<< endl;
            cout << "                              ||          1. Изменить имя
||         " << endl;
            cout << "                              ||          2. Изменить фамилию
||         " << endl;
            cout << "                              ||          3. Изменить отчество
||         " << endl;
            cout << "                              ||          4. Изменить отдел
||         " << endl;
            cout << "                              ||          5. Изменить должность
||         " << endl;
            cout << "                              ||          6. Изменить зарплату
||         " << endl;
            cout << "
||_____||         " << endl;
            cout << "                              ||                                ||         "
<< endl;
            cout << "                              ||          0. Вернуться назад
||         " << endl;
            cout << "
||_____||         " << endl;
            cout << "                                    >>";

            int sw;
            Protection(0, 6, sw);
            switch (sw)
            {
            case 1:
                  cout << "
_____" << endl;
                  cout << "                                          Введите новое имя:
";
                  cin >> arr_of_Employees[i].Name;
                  Save_it(); // Новые данные успешно сохранены
                  break;
            case 2:
```

```
                cout << "
_____________________________________________" << endl;
                cout << "                                Введите новую
фамилию: ";
                cin >> arr_of_Employees[i].Surname;
                Save_it();
                break;
        case 3:
                cout << "
_____________________________________________" << endl;
                cout << "                                Введите новое
отчество: ";
                cin >> arr_of_Employees[i].Patronymic;
                Save_it();
                break;
        case 4:
                cout << "
_____________________________________________" << endl;
                cout << "                                Введите новый
отдел: ";
                cin >> arr_of_Employees[i].Department;
                Save_it();
                break;
        case 5:
                cout << "
_____________________________________________" << endl;
                cout << "                                Введите новую
должность: ";
                cin >> arr_of_Employees[i].Position;
                Save_it();
                break;
        case 6:
                cout << "
_____________________________________________" << endl;
                cout << "                                Введите новую
зарплату: ";
                cin >> arr_of_Employees[i].Salary;
                Save_it();
                break;

        case 0:  break;
        }
    }
}
```

```
// УДАЛЕНИЕ
СОТРУДНИКА_____
_____
__

void Delete_employees(Employee* arr_of_Employees, int&
number_of_Employees)
{
    string delEmployee;
    cout << "
_____
" << endl;
    cout << "                              Введите фамилию сотрудника, которого
хотите удалить: ";
    cin >> delEmployee;
    int i = 0;
    bool flag = true;
    while (flag && i <= number_of_Employees) // Цикл для поиска нужного
аккаунта
    {
            if (arr_of_Employees[i].Surname == delEmployee)
            {
                    flag = false; // Если нашли - изменяем flag
            }
            else
            {
                    i++;
            }
    }

    if (flag) // Если не нашли
    {
            Account_not_found(); // Такой учетной записи нет (выводим
сообщение)
    }
    else // Если нашли
    {
            int del = 0;
            cout << "Вы действительно хотите удалить сотрудника: " <<
arr_of_Employees[i].Surname << endl;
            cout << "P.S. Удаленные данные не подлежат восстановления."
<< endl;
            cout << " 1. Да" << "\n 2. Нет" << "\n --> ";
```

```
            Protection(1, 2, del);
            if (del == 1)
            {
                for (int j = i; j < number_of_Employees - 1; j++) // i -> номер
аккаунта | Смещение в конец удаленного аккаунта
                {
                    arr_of_Employees[j] = arr_of_Employees[j + 1];
                }
                number_of_Employees--; // Уменьшаем количество
сотрудников
                Save_it(); // Данные успешно сохранены
            }
            else
            {
                cout << "Удаление отменено" << endl;
            }
    }
}


// ИНДИВИДУАЛЬНОЕ
ЗАДАНИЕ_____
_____


// Выбор ИДЗ

void Menu_individual_task(Employee* arr_of_Employees, int
number_of_Employees)
{

    Set_color(1, 0); // Синий | Цвет текста | Цвет фона
    cout << "
_____                    " <<
endl;
    cout << "                        /                                    \\        "
<< endl;
    cout << "                       /  М Е Н Ю | И Н Д И В И Д У А Л Ь Н О Е
З А Д А Н И Е  \\        " << endl;
    cout << "
/|_____|\\        "
<< endl;
    Set_color(7, 0); // Светло-серый
```

```
    cout << "                          ||                                        ||        "
<< endl;
    cout << "                          ||          1. Сотрудники с зарплатой <
введенной     ||        " << endl;
    cout << "                          ||          2. Информация по отделам
||        " << endl;
    cout << "
||_____________________________________________________________||        "
<< endl;
    cout << "                          ||                                        ||        "
<< endl;
    cout << "                          ||          0. Вернуться назад                ||
" << endl;
    cout << "
||_____________________________________________________________||        "
<< endl;
    cout << "                                  >>";

    int sw;
    Protection(0, 2, sw);
    switch (sw)
    {
    case 1:
        Individual_task_salary_info(arr_of_Employees,
number_of_Employees); break;
    case 2:
        Individual_task_departments_info(arr_of_Employees,
number_of_Employees); break;
    case 0:
        break;
    }
}

// ИДЗ №1 (Вывод сотрудников с ЗП меньше введенной с
клавиатуры)_____________________________________________________
_____________________

void Individual_task_salary_info(Employee* arr_of_Employees, int
number_of_Employees) // С ЗП меньше указанной
{
    cout << "
_____________________________________________________  " <<
endl;
```

```
    cout << "                          /                                    \\  " <<
endl;
    cout << "                         /          Сотрудники с ЗП меньше
указанной!          \\ " << endl;
    cout << "                                    Введите зарплату: ";
    int FindSalary;
    cin >> FindSalary;
    for (int i = 0; i < number_of_Employees; i++)
    {
        if (arr_of_Employees[i].Salary < FindSalary) // Если зарплата
меньше введенной --> выводим инфу о сотруднике
        {
            Show_result_of_finding(arr_of_Employees,
number_of_Employees, i);
        }
    }
}

// ИДЗ №2 (Информация по отделам | Общая ЗП | Средняя ЗП |
Сотрудники)_____
_____

void Individual_task_departments_info(Employee* arr_of_Employees, int
number_of_Employees)
{
    int number_of_dep = 1;
    Department* arr_of_dep = new Department[number_of_dep]; // Создаю 1
объект структуры изначально | Отдел | Сотрудники | ЗП
    // Берем 1 сотрудника и сверяем его инфу со всеми отделами
    for (int i = 0; i < number_of_Employees; i++) // Прокрутка сотрудников
    {
        for (int j = 0; j <= number_of_dep; j++) // Прокрутка отделов
        {
            if (arr_of_Employees[i].Department ==
arr_of_dep[j].Department_name) // Если отдел сотрудника совпал с одним
из уже существующих отедлов
            {
                arr_of_dep[j].Total_department_salary +=
arr_of_Employees[i].Salary; // ЗП сотрудника добавляем к общей ЗП отдела
                arr_of_dep[j].Employees_amount++; // Обновляем
количество сотрудников
                break;
            }
```

```
                if (j == number_of_dep) // Если так --> либо это первый сотрудинк | либо нету такого отдела
                {
                        if (number_of_dep == 0) // Добавляем сотрудника сразу без выделения новой памяти
                        {
                                arr_of_dep[j].Department_name = arr_of_Employees[i].Department; // Заносим название отдела
                                arr_of_dep[j].Employees_amount++; // Обновляем количество сотрудников в отделе
                                arr_of_dep[j].Total_department_salary += arr_of_Employees[i].Salary; // Добавляем зарплату сотрудника к общей по отделу
                        }
                        else // Выделяем новую память
                        {
                                arr_of_dep = Individual_task_department_memory_reallocation(arr_of_dep, number_of_dep, number_of_dep + 1); // Расширяем память
                                arr_of_dep[j].Department_name = arr_of_Employees[i].Department;
                                arr_of_dep[j].Employees_amount++;
                                arr_of_dep[j].Total_department_salary += arr_of_Employees[i].Salary;
                        }
                        break;
                }
        }
    }

    for (int k = 1; k < number_of_dep; k++) // Вывод информации по отделу
    {
        cout << "                                Отдел: " << arr_of_dep[k].Department_name << endl;
        cout << "                                Сотрудников в отделе: " << arr_of_dep[k].Employees_amount << endl;
        cout << "                                Общая зарплата: " << arr_of_dep[k].Total_department_salary << endl;
        cout << "                                Средняя ЗП: " << arr_of_dep[k].Total_department_salary / arr_of_dep[k].Employees_amount << endl << endl;
    }
    delete[]arr_of_dep; // Удаляем инфу после использования
}
```

```
// ИДЗ №2 Перевыделение памяти для
отделов_____
_____

Department* Individual_task_department_memory_reallocation(Department*
arr_of_dep, int& number_of_dep, int m)
{

    Department* temp_arr = new Department[m];
    for (int i = 0; i < number_of_dep; i++) // Перенос информации
    {
            temp_arr[i].Department_name = arr_of_dep[i].Department_name;
            temp_arr[i].Employees_amount = arr_of_dep[i].Employees_amount;
            temp_arr[i].Total_department_salary =
arr_of_dep[i].Total_department_salary;
    }

    delete[]arr_of_dep; // Удаляем старый массив
    number_of_dep = m; // Обновляем размер
    arr_of_dep = temp_arr; // Переносим информацию
    return arr_of_dep; // Возвращаем обновленные данные
}



// МЕТОДЫ
СОРТИРОВКИ_____
_____


// Выбор сортировки
Employee* Selection_sorting_menu(Employee* arr_of_Employees, int
number_of_Employees)
{
    int sw;
    Set_color(1, 0); // Синий | Цвет текста | Цвет фона
    cout << "
_____                  " << endl;
    cout << "                              /                              \\          " <<
endl;
    cout << "                             /         М Е Н Ю | С О Р Т И Р О В К А
\\         " << endl;
```

```
    cout << "
/|_____________________________________________________|\\       " << endl;
    Set_color(7, 0); // Светло-серый
    cout << "                              ||                                ||       " <<
endl;
    cout << "                              ||          1. Отсортировать по фамилии
||       " << endl;
    cout << "                              ||          2. Отсортировать по отделам
||       " << endl;
    cout << "                              ||          3. Отсортировать по зарплатам
||       " << endl;
    cout << "
||_____________________________________________________||       " << endl;
    cout << "                              ||                                ||       " <<
endl;
    cout << "                              ||          0. Вернуться назад              ||
" << endl;
    cout << "
||_____________________________________________________||       " << endl;
    cout << "                                    >>";

    Protection(0, 3, sw);
    switch (sw)
    {
    case 1: arr_of_Employees = Sorting_by_Surname(arr_of_Employees,
number_of_Employees); break;
    case 2: arr_of_Employees = Sorting_by_department(arr_of_Employees,
number_of_Employees); break;
    case 3: arr_of_Employees = Sorting_by_the_salary(arr_of_Employees,
number_of_Employees); break;
    case 0:break;
    }
    return arr_of_Employees;
}

// Сортировка №1 (По
фамилиям)_____________________________________________________
_____________________________________________

Employee* Sorting_by_Surname(Employee* arr_of_Employees, int
number_of_Employees)
{
    Employee* temp_arr = new Employee[number_of_Employees]; //
Выделяем память
```

```
    for (int i = 0; i < number_of_Employees; i++) // Переносим информацию
    {
            temp_arr[i].Name = arr_of_Employees[i].Name;
            temp_arr[i].Surname = arr_of_Employees[i].Surname;
            temp_arr[i].Patronymic = arr_of_Employees[i].Patronymic;
            temp_arr[i].Department = arr_of_Employees[i].Department;
            temp_arr[i].Position = arr_of_Employees[i].Position;
            temp_arr[i].Salary = arr_of_Employees[i].Salary;

    }
    cout << "
_____________________________________________________________________________ " <<
endl;
    cout << "                              /                                \\     " <<
endl;
    cout << "                             /          Сортировка сотрудников по
фамилии!          \\    " << endl << endl;

    for (int i = 0; i < number_of_Employees - 1; i++)  // Сортировка от А до Я
    {
            for (int j = 0; j < number_of_Employees - i - 1; j++) // Отступаем на
один шаг
            {
                    if (temp_arr[j].Surname > temp_arr[j + 1].Surname) //
Сравнием по ASCII
                    {
                            swap(temp_arr[j], temp_arr[j + 1]); // Меняет местами
                    }
            }
    }
    Show_employees(temp_arr, number_of_Employees);

    int temp; // Временная переменная
    Ask_for_saving(); // Сохранять?

    Protection(1, 2, temp);
    switch (temp)
    {
    case 1: // Сохранить (Обновляем старый)
            delete[]arr_of_Employees;
            arr_of_Employees = temp_arr;
            break;
    case 2: // Оставить
            delete[] temp_arr; // Удаляем отсортированный
```

```
            break;
    }
    return arr_of_Employees;
}

// Сортировка №2 (По
отделам)_____
_____

Employee* Sorting_by_department(Employee* arr_of_Employees, int
number_of_Employees)
{

    Employee* arr_new = new Employee[number_of_Employees];
    for (int i = 0; i < number_of_Employees; i++)
    {
            arr_new[i].Name = arr_of_Employees[i].Name;
            arr_new[i].Surname = arr_of_Employees[i].Surname;
            arr_new[i].Patronymic = arr_of_Employees[i].Patronymic;
            arr_new[i].Department = arr_of_Employees[i].Department;
            arr_new[i].Position = arr_of_Employees[i].Position;
            arr_new[i].Salary = arr_of_Employees[i].Salary;

    }
    cout << "
_____                    " <<
endl;
    cout << "                          /                                    \\     " <<
endl;
    cout << "                          /          Сортировка сотрудников по
отделу!          \\    " << endl << endl;

    for (int i = 0; i < number_of_Employees - 1; i++) // Для пользователя 1-ый
сотрудинк --> 1 | Но в массиве 0 || От А до Я
    {
            for (int j = 0; j < number_of_Employees - i - 1; j++)
            {
                    if (arr_new[j].Department > arr_new[j + 1].Department) //
ASCII
                    {
                            swap(arr_new[j], arr_new[j + 1]);
                    }
            }
    }
```

```
    Show_employees(arr_new, number_of_Employees);

    int temp;
    Ask_for_saving(); // Сохранять?

    Protection(1, 2, temp);
    switch (temp)
    {
    case 1:
        delete[]arr_of_Employees; // Если нужно сохраниеть новые
значения --> удаляем старый массив
        arr_of_Employees = arr_new; // Заполняем массив новыми
значениями
        break;
    case 2:
        delete[] arr_new; // Если нет --> удаляем отсортированный
        break;
    }
    return arr_of_Employees;
}

// Сортировка №3 (По
зарплате)_____
_____

Employee* Sorting_by_the_salary(Employee* arr_of_Employees, int
number_of_Employees)
{

    Employee* arr_new = new Employee[number_of_Employees];
    for (int i = 0; i < number_of_Employees; i++)
    {
        arr_new[i].Name = arr_of_Employees[i].Name;
        arr_new[i].Surname = arr_of_Employees[i].Surname;
        arr_new[i].Patronymic = arr_of_Employees[i].Patronymic;
        arr_new[i].Department = arr_of_Employees[i].Department;
        arr_new[i].Position = arr_of_Employees[i].Position;
        arr_new[i].Salary = arr_of_Employees[i].Salary;

    }
    cout << "
_____   " <<
endl;
```

```
    cout << "                          /                                  \\  " <<
endl;
    cout << "                          /          Сортировка сотрудников по
зарплате!          \\ " << endl << endl;

    for (int i = 0; i < number_of_Employees - 1; i++) // От меньшей к большей
    {
        for (int j = 0; j < number_of_Employees - i - 1; j++)
        {
            if (arr_new[j].Salary > arr_new[j + 1].Salary)
            {
                swap(arr_new[j], arr_new[j + 1]);
            }
        }
    }
    Show_employees(arr_new, number_of_Employees);

    int temp;
    Ask_for_saving(); // Сохранять?

    Protection(1, 2, temp);
    switch (temp)
    {
    case 1: // Сохранить новый
        Save_it(); // Новые данные успешно сохранены
        delete[]arr_of_Employees;
        arr_of_Employees = arr_new;
        break;
    case 2: // Оставить без изменений
        delete[] arr_new;
        Save_it();
        break;
    }
    return arr_of_Employees;
}



// МЕТОДЫ
ПОИСКА_____________________________________________________
_____________________________________________________________


// Меню выбора поиска
```

```
void Menu_of_finding_employees(Employee* arr_of_Employees, int
number_of_Employees)
{
    Set_color(1, 0); // Синий | Цвет текста | Цвет фона
    cout << "
____________________________________________________            " << endl;
    cout << "                          /                                  \\        " <<
endl;
    cout << "                          /            М Е Н Ю | П О И С К А
\\        " << endl;
    cout << "
/|________________________________________________________|\\       " <<
endl;
    Set_color(7, 0); // Светло-серый
    cout << "                         ||                                  ||       " <<
endl;
    cout << "                         ||         1. Найти по фамилии              ||
" << endl;
    cout << "                         ||         2. Найти по должности
||       " << endl;
    cout << "                         ||         3. Найти сотрудника с
максимальной ЗП  ||       " << endl;
    cout << "
||________________________________________________________||       " <<
endl;
    cout << "                         ||                                  ||       " <<
endl;
    cout << "                         ||         0. Вернуться назад               ||
" << endl;
    cout << "
||________________________________________________________||       " <<
endl;
    cout << "                                >>";

    int sw;
    Protection(0, 3, sw);

    switch (sw)
    {
    case 1: Find_surname(arr_of_Employees, number_of_Employees); break; //
Поиск по фамилии
    case 2: Find_position(arr_of_Employees, number_of_Employees); break; //
Поиск по должности
```

```
    case 3: Find_max_salary(arr_of_Employees, number_of_Employees);
break; // Поиск по максимальной зарплате
    case 0: break; // Выход
    }
}

// Поиск №1 (По
фамилии)_____
_____

void Find_surname(Employee* arr_of_Employees, int number_of_Employees)
{
    string FindSurname; // Для ввода фамилии
    bool flag = true;
    cout << "
_____    " <<
endl;
    cout << "                          /                                    \\  " <<
endl;
    cout << "                         /          Поиск сотрудников по фамилии!
\\ " << endl;
    cout << "                                    Введите фамилию для поиска: ";
    cin >> FindSurname;
    for (int i = 0; i < number_of_Employees; i++)
    {
        if (FindSurname == arr_of_Employees[i].Surname) // Если нашли --
> выводим
        {
            Show_result_of_finding(arr_of_Employees,
number_of_Employees, i);
            flag = false; // --> отклюачем следующий if
        }
    }
    if (flag)
    {
        Account_not_found(); // Такого аккаунта нет в системе
    }

}

// Поиск №2 (По
должности)_____
_____
```

```
void Find_position(Employee* arr_of_Employees, int number_of_Employees)
{
    string SearchPositio; // Для ввода должности
    cout << "
_____________________________________________________________    " <<
endl;
    cout << "                          /                                  \\   " <<
endl;
    cout << "                          /          Поиск сотрудников по должности!
\\  " << endl;
    cout << "                                     Введите должность для поиска: ";
    cin >> SearchPositio;
    bool flag = true;
    for (int i = 0; i < number_of_Employees; i++)
    {
            if (arr_of_Employees[i].Position == SearchPositio)
            {
                    Show_result_of_finding(arr_of_Employees,
number_of_Employees, i);
                    flag = false; // --> отклюачем следующий if
            }
    }
    if (flag)
    {
            Account_not_found(); // Такого аккаунта нет в системе
    }
}

// Поиск №3 (По максимальной
ЗП)_____________________________________________________________
_____________________________

void Find_max_salary(Employee* arr_of_Employees, int
number_of_Employees)
{
    cout << "
_____________________________________________________________    " <<
endl;
    cout << "                          /                                  \\   " <<
endl;
    cout << "                          /          Поиск сотрудников с наибольшей
ЗП!          \\   " << endl;
    int MaxKol = 0;
    for (int i = 0; i < number_of_Employees; i++)
```

```
    {
        if (MaxKol < arr_of_Employees[i].Salary) // Если нашли зарплату
больше
        {
            MaxKol = arr_of_Employees[i].Salary; // Обновляем
значение MaxKol
        }
    }
    for (int i = 0; i < number_of_Employees; i++) // Какой сотрудник имеет
эту ЗП
    {
        if (arr_of_Employees[i].Salary == MaxKol) // Сверяем ЗП
сотрудника
        {
            Show_result_of_finding(arr_of_Employees,
number_of_Employees, i);
        }
    }

}

// Вывод результатов
поиска_____
_____

void Show_result_of_finding(Employee* arr_of_Employees, int
number_of_Employees, int i)
{
    cout << endl;
    cout << "                                    Ф.И.О. " <<
arr_of_Employees[i].Surname << " " << arr_of_Employees[i].Name << " " <<
arr_of_Employees[i].Patronymic << endl;
    cout << "                                    Отдел: " <<
arr_of_Employees[i].Department << endl;
    cout << "                                    Должность: " <<
arr_of_Employees[i].Position << endl;
    cout << "                                    Зарплата: " <<
arr_of_Employees[i].Salary << endl;
    cout << endl;
}
```

Protection.h

```
#pragma once
#include <iostream>

using namespace std;
```

```
int Protection(int a, int b, int& n); // Защита от букв и чисел [ от a до b] | С переменной n
string Inputing_password(); // Пароль
```

Protection.cpp

```
#include "Protection.h"

int Protection(int a, int b, int& n)
{
	cin >> n; // Ввод запрашиваемого значения
	while (true)
	{
		while ((!cin) || (cin.get() != '\n')) // Проверка на ввод буквы
		{
			cin.clear();
			cin.ignore(10000, '\n');
			cout << endl;
			cout << "
_____________________________________________________________________" << endl;
			cout << "                                                     |
|           " << endl;
			cout << "                                                     |   Значение должно быть числом, пожалуйста, повторите ввод    |           " << endl;
			cout << "
|_____________________________________________________________________|           " << endl;
			cout << "                                                                 >> ";
			cin >> n;
		}
		if (n >= a && n <= b) break; // Проверка на диапазон значений
		cout << endl;
		cout << "
_____________________________________________________________" << endl;
		cout << "                                    |
|           " << endl;
		cout << "                                    |   Введенное число выходит за допустимый предел, повторите ввод    |           " << endl;
		cout << "
|_____________________________________________________________|           " << endl;
		cout << "                                                  >> ";
		cin >> n;
	}
	return n; // Возвращаем запрашиваемое значение
}


// Ввод пароля_____________________________________________________________________________________

string Inputing_password()
{
	string password;
	int key_code = 0;
	while (true)
	{
		key_code = _getwch(); // Записываем код нажатой клавиши (любой)

		if (key_code == 13) // Чтобы "enter" не добавился к паролю [13 код клавиши enter]
		{
			break; // Остановка цикла после удачного ввода (нажатие enter)
		}
		else
```

```
                        if (key_code == 8) // [8 код клавиши backspace]
                        {
                                cout << (char)8 << ' ' << (char)8; // Смещение курсора влево
| Замена на ' ' | Ещё раз смещение курсора
                                if (!password.empty()) // Если строка не пустая, то можно
удалять. Если пусто >> ничего не делаем | empty >> true если пустая строка
                                        password.erase(password.length() - 1); // метод string
для удаления 1 символа
                        }
                        else
                        {
                                cout << "*"; // >> чтобы было видно действие
                                password += (char)key_code; // Чтобы добавить к строке
[password] (код с клавы) >> превращаем его в символ
                        }
        }
        return password;
    }
```

Signal.h

```
#pragma once
#include <iostream> // Ввод | Вывод >> cout
#include <Windows.h> // Для изменения цвета

using namespace std;

void Save_it(); // Новые данные успешно сохранены!
void Set_color(int text, int bg); // Изменение цвета
void Ask_for_saving(); // Хотите сохранять?
void Account_not_found(); // Такого аккаунта нет в системе
void Password_incorrect(string TempLogin, bool& flagPassword, int& amount); // Выводит
ошибку если несколько раз ввели не тот пароль [Вход/Регистрация]
```

Signal.cpp

```
#include "Signal.h"

void Set_color(int text, int bg)
{
        HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE); // Получение дескриптора вывода
на консоль
        SetConsoleTextAttribute(hStdOut, (WORD)((bg << 4) | text)); // Установка атрибутов
текста
}

// Сохранять
ли?_____
_____
void Ask_for_saving()
{
        Set_color(15, 0);
        cout << endl;
        cout << "
_____                                                  " << endl;
        cout << "                                                    /
\\           " << endl;
        cout << "                                                   /           С О Х Р А Н Е Н И Е
\\         " << endl;
        cout << "
________/_____________________________________________\\________" << endl;
        cout << "                                       |
| " << endl;
```

```
	cout << "                                        |          Вы можете сохранить
отсортированные данные!          | " << endl;
	cout << "                                        |
| " << endl;
	cout << "
|_____________________________________________________________| " << endl;
	cout << "                                        |
| " << endl;
	Set_color(2, 0);
	cout << "                                        |                  1 - Да, сохранить
| " << endl;
	Set_color(12, 0);
	cout << "                                        |                  2 - Нет, не
сохранять                      | " << endl;
	Set_color(15, 0);
	cout << "                                        |
| " << endl;
	cout << "
|_____________________________________________________________| " << endl;
	cout << "                                                      >>";
	Set_color(7, 0);
}

// Такого аккаунта не
существует!_____________________________________________________________________
_____________________
void Account_not_found()
{
	Set_color(12, 0);
	cout << endl << endl;
	cout << "
_____________________________________________________________" << endl;
	cout << "                                        |
|" << endl;
	cout << "                                        |             Ошибка! Такого аккаунта не
существует!          |    " << endl;
	cout << "
|_____________________________________________________________|" << endl;
	Set_color(7, 0);
	Sleep(1000);
	system("cls");
}

//Данные успешно
сохранены!_____________________________________________________________________
_____________________
void Save_it()
{
	Set_color(2, 0); // Зеленый
	cout << endl << endl;
	cout << "
_____________________________________________________________" << endl;
	cout << "                                        |
|" << endl;
	cout << "                                        |                   Данные успешно
сохранены!                |" << endl;
	cout << "
|_____________________________________________________________|" << endl;
	Set_color(7, 0); // Белый
	Sleep(1000);
	system("cls");
}
```

```
//Неверный
пароль!_____________________________________________________________________________
_____________________________________

// Передаём по адресу flag и amount чтобы они обновлялись в основной программе.
Изначально true | 3
void Password_incorrect(string TempLogin, bool& flagPassword, int& amount) // Если
несколько раз ввели не тот пароль --> выводим сообщение об ошибке
{
	amount--; // Уменьшаем перменную | Если amount дойдет до 0 --> останавливаем цикл
while
	system("cls");
	cout << "
_____________________________________________ " << endl;
	cout << "                                             |
| " << endl;
	Set_color(12, 0); // Красный
	cout << "                                             |              Неверный пароль
| " << endl;
	cout << "                                                            Попыток осталось:
" << amount << endl << endl;
	Set_color(7, 0); // Черный
	cout << "
|_____________________________________________| " << endl;
	cout << "                                                            Логин: " <<
TempLogin << endl;
	if (amount == 0)
	{
		Set_color(12, 0); // Светло-красный
		cout << "
_____________________________________________________________________ " << endl;
		cout << "                                    |
| " << endl;
		cout << "                                    |    Вы превысили количество
допустимых попыток. Повторите позже    | " << endl;
		cout << "
|_____________________________________________________________________| " << endl;
		Sleep(1500);
		system("cls");
		flagPassword = false; // Возвращаем значение false чтобы остановить цикл
	}
   }
```