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**

**СТАЖА СОТРУДНИКОВ ПРЕДПРИЯТИЯ**

Выполнила: студентка группы 910901

Дащинская Л.А.

Проверила: Меженная М. М.

Минск 2020

**Содержание**

1. Требования к программе
2. Конструирование программы
   1. Разработка модульной структуры программы
   2. Выбор способа организации данных
   3. Разработка перечня пользовательских функций программы
3. Разработка алгоритмов работы программы
   1. Алгоритм функции main.
   2. Алгоритм функции individualTask.
   3. Алгоритм функции generateSalt.
4. Описание работы программы.
   1. Авторизация.
   2. Модуль администратора.
   3. Модуль пользователя.
   4. Исключительные ситуации.

Приложение

1. **Требования к программе**

**Задание**

Информация о сотрудниках предприятия содержит: Ф.И.О. сотрудника; дату рождения; название отдела; должность; дату начала работы.

Индивидуальное задание: вывести список сотрудников пенсионного возраста. Вывести список сотрудников в порядке убывания стажа.

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

**Исходные данные**

1. Тема курсовой работы «Разработка программы учета стажа сотрудников предприятия».
2. Язык программирования C++.
3. Среда разработки Microsoft Visual Studio.
4. Вид приложения — консольное.
5. Парадигма программирования — процедурная.
6. Способ организации данных — структуры (struct).
7. Способ хранения данных — файлы.
8. Каждая логически завершённая подзадача программы реализована в виде отдельной функции.

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

**Функциональные требования**

При первом запуске программы Вам необходимо создать аккаунт системного администратора, чтобы взаимодействовать с дальнейшим функционалом. В последующих случаях первым этапом работы программы будет являться авторизация, которая предоставит вам определённые права доступа. Происходит проверка наличия необходимых для работы файлов и создание, в случае их отсутствия.

**Для организации защиты данных, при вводе пароля происходит его маскировка символами «\*».**

**Пароль** в базе данных не храниться в открытом доступе, а проходит этап шифрования.

**Следующим этапом** работы программы является работа с данными, которая становится доступной после успешной авторизации.

Для работы с данными разработано два функциональных модуля: администратора, пользователя.

**1. Модуль администратора**

**1.1 Работа с базой данных «Учетные записи пользователей»:**

-Просмотр всех учетных записей;

-Добавление новой учетной записи;

-Редактирование учетной записи;

-Удаление учетной записи;

**1.2 Работа с базой данных «Информация о сотрудниках предприятия»:**

**-**Режим редактирования (добавить, просмотреть, редактировать, удалить данные);

**-**Режим обработки;

**2. Модуль пользователя**

**2.1 Работа с базой данных «Информация о сотрудниках предприятия»:**

-Просмотр данных;

-Выполнить индивидуальное задание;

-Сортировка информации;

-Поиск информации о сотрудников;

**В курсовой работе предусмотрены следующие случаи:**

1. Обработка исключительных ситуаций;
2. Обратная связь с пользователем;
3. Возможность вернуться назад;

**Требования к программной реализации**

1. Все переменные и константы имеют осмысленные имена в рамках тематики варианта курсовой работы. Переменным присвоены имена, состоящие из букв нижнего регистра; для формирования составного имени использовалась «верблюжья нотация». Константам присвоены имена, состоящие из букв верхнего регистра.

2. Имена функций также осмысленные, начинаются с буквы нижнего регистра. Функции строятся по типу «глагол + существительное».

3. Отсутствует оператор прерывания goto.

4. Код не содержит неименованных числовых констант, неименованных строковых констант. Такие константы представлены как глобальные в виде методов, возвращающие значение.

5. В коде отсутствуют дублирования.

6. Каждая функция выполняет определённую задачу. При этом имеется наличие одной функции в другой.

7. Выполнение операций чтения/записи сведено к минимуму.

8. Отсутствуют глубокие вложенности условных и циклических конструкций.

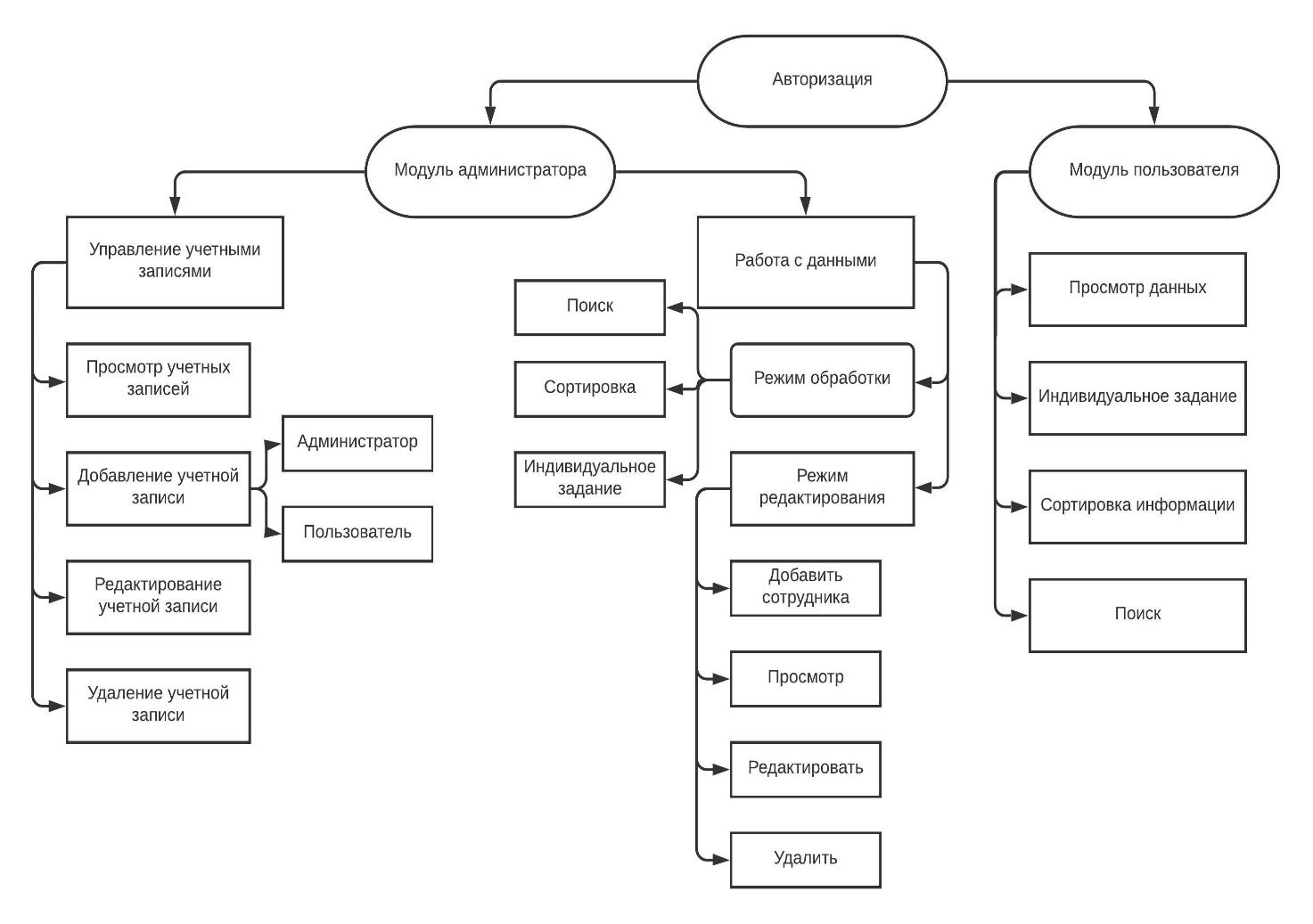
9. В большей степени отсутствуют длинные функции.

10. Имеются шаблонные функции.

11. Вынесен код логически независимых модулей в отдельные cpp-файлы и подключены они с помощью заголовочных h-файлов.

**2. Конструирование программы**

**2.1 Разработка модульной структуры программы**



*Структура программы*

**2.2 Выбор способа организации данных**

В качестве выбора способа описания входных данных были использованы структуры(struct):

**2.3 Разработка перечня пользовательских функций программы**

Курсовая работа состоит из одного главного cpp-файла и h-файла и 9 связок (h-файл+ cpp-файл):

**-main.cpp;**

**-startInfo.h;**

**-adminMenu.cpp + adminMenu.h;**

**-autorisation.cpp + autorisation.h;**

**-hashFunction.cpp + hashFunction.h;**

**-searchMenu.cpp + searchMenu.h;**

**-sortMenu.cpp + sortMenu.h;**

**-userMenu.cpp + userMenu.h;**

**-workWithAccontsAdmin.cpp + workWithAccontsAdmin.h;**

**- workWithDataAdmin.cpp + workWithDataAdmin.h;**

**- workWithFilesAdmin.cpp + workWithFilesAdmin.h;**

**startInfo.h**

struct autorisation

{

string login;

string passowrd;

bool role;

string salt;

};

struct date

{

int day;

int mounth;

int year;

};

struct company\_information

{

string fio;

char sex;

date birthday;

string department;

string occupation;

date begin\_work\_date;

};

struct encrypt\_element

{

char info;

char key;

};

const int NUMBER\_MOUNTH = 12;

const int NUMBER\_DAYS\_IN\_MONTH[NUMBER\_MOUNTH] = { 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 };

const int NUMBER\_DAYS\_IN\_YEAR = 365;

const int MAX\_DAYS = 31;

const int MIN\_DAYS = 0;

const int MAX\_YEARS = 2020;

const int MIN\_YEARS = 1920;

const int SENIOR\_MAN\_AGE = 62;

const int SENIOR\_WOMAN\_AGE = 57;

const int CURRENT\_YEAR = 2020;

**adminMenu.h**

void adminMenu(vector <autorisation>&, vector<company\_information>&, vector <encrypt\_element>); //меню администратора

**autorisation.h**

void autorisationMainFunction(vector <company\_information>&, vector<autorisation>&, vector <encrypt\_element>); //авторизация: ввод логина и пароля

bool isBufferWithNoChar(); //проверка на целочисленные значения

void choiseError(); //вывод сообщения об ошибке ввода

string inputStarsPassword(); //Пароль в звездочки и проверка на ввод латынью

void entranceInProgram(bool&, autorisation&, vector <encrypt\_element>, vector <autorisation>&, vector <company\_information>&); //вход в программу как администратор или пользователь

void behaviorAfterWorkInProgram(int&); //Повторный ввод данных и запуск авторизации по желанию пользователя либо завершение программы

bool isAutorisationCycleBreakOrContinue(autorisation&, string, vector <autorisation>&, vector <encrypt\_element>, int&, int&, int&); //цикл прерывания или продолжения авторизации

void accountNotExist(autorisation&, string, vector <autorisation>&, vector <encrypt\_element>, int&, int&, int&); //информирование, что аккаунта не существует и вариант продолжить или выйти

void errorDuringInputData(); //сообщение при повторном вводе данных для входа

void addFirstAdmin(string, string, string, vector <encrypt\_element>); //добавление первого администратора в файл

bool isFileExist(string); //проверка на существование файла

hashFunction.h

string generateSalt(); // Сгенерировать соль

string blendWithSalt(string, string); // Смешать с солью

void fillHashTable(vector <encrypt\_element>&); // Заполнение хеш-таблицы

char findInTable(char, vector <encrypt\_element>); // Найти в таблице и сопоставить с символом

string encryptCurrentPass(string, vector <encrypt\_element>, string); // Зашифровать текущий пароль

searchMenu.h

void searchMenu(vector<company\_information>); //меню по поиску

void searchForFIO(vector<company\_information>); //поиск по фио

void searchForSex(vector<company\_information>); //поиск по полу

void searchForBirthdayDate(vector<company\_information>); //поиск по дате рождения

void searchForDepartment(vector<company\_information>); //поиск по отделу

void searchForOccupation(vector<company\_information>); //поиск по должности сотрудника

void searchForBeginWorkDate(vector<company\_information>); //поиск по дате начала работы

sortMenu.h

void sortMenu(vector<company\_information>); //меню сортировки

void sortForFIO(vector<company\_information>); //сортировка по фамилии

void sortForSex(vector<company\_information>); //сортировка по полу

void sortForDepartment(vector<company\_information>); //сортировка по отделу

void sortForOccupation(vector<company\_information>); //сортировка по должности сотрудников

userMenu.h

void userMenu(vector<company\_information> data); //меню пользователя

workWithAccountsAdmin.h

void workWithAccountsAdmin(vector <autorisation>&, vector <encrypt\_element>); //меню работы с учетными записями пользователей

void deleteAccount(vector <autorisation>&, bool&, char\*); //удаление аккаунта

void redactAccount(vector <autorisation>&, vector <encrypt\_element>); //редактирование аккаунта

void viewAccounts(vector <autorisation>); //просмотр аккаунтов

void addNewAccount(vector <autorisation>&, vector <encrypt\_element>); //добавить новый аккаунт: либо админимстратор, либо пользователь

autorisation inputInfoAboutNewAccount(vector <encrypt\_element>, vector <autorisation>); //ввод информации в аккаунт

bool isTheSameLoginInVector(vector <autorisation>, string, int); //проверка на уникальность в процессе вериафикации

void addAdministrator(vector <autorisation>&, vector <encrypt\_element>); //добавить нового администратора

void addUser(vector <autorisation>&, vector <encrypt\_element>); //добавить нового пользователя

string inputLogin(); //ввод логина

bool isAllSymbolsCorrect(string); //проверка вводимых символов на диапазон

string inputUniqueLogin(vector <autorisation>); //проверка логина на оригинальность

workWithDataAdmin.h

char inputSex(); //ввод пола

string buildStringDate(date); //формирование даты

void showListOfWorkersForExperiances(vector<company\_information>); //вывод списка сотрудников в порядке убывания стажа

void showListOfSeniors(vector<company\_information>); //список сотрудников пенсионного возраста

date inputDate(); //ввод даты

bool isWorkContinue(); // обратная связь: хотите продолжить?

string inputWithUpRegister(); //если не прошло проверку на большую букву, то повторный ввод

void workWithDataAdmin(vector<company\_information>&); //меню режима работы с данными

void redactingModeAdmin(vector<company\_information>&); //меню режима редактирования данных

string inputString(); //состоит из проверки на большую букву

string reBuildStringDesign(string); //оформление на вывод

void addEmployeesData(vector<company\_information>&); //добавить информацию о сотрудниках

void viewData(vector<company\_information>); //просмотр данных

void redactData(vector<company\_information>&); //редактирование данных записи

void deleteData(vector<company\_information>&); //удаление данных записи

void processingModeAdmin(vector<company\_information>); //меню режима обработки данных

void individualTask(vector<company\_information>); //индивидуальное задание

bool IsFirstCharInUpperRegisterAndNotSpace(char); //проверка на первую большую букву и пробел

workWithFilesAdmin.h

void launchMainFunctionMenu(vector <autorisation>&, vector <company\_information>&, vector <encrypt\_element>&, string, string, string, string); //запустить главные функции: заполнение хеш-таблицы; добавление 1 админа; данные из файла в вектор; авторизация; из вектора в файл

void pushDataInFile(vector <company\_information>, string, vector <autorisation>, string); //записать все данные из векторов в файл

void fromVectorToFileACCOUNTS(vector <autorisation>, string); //запись данных из вектора в файл с аккаунтами

void fromVectorToFileDATA(vector <company\_information>, string); //запись данных из вектора в файл с данными

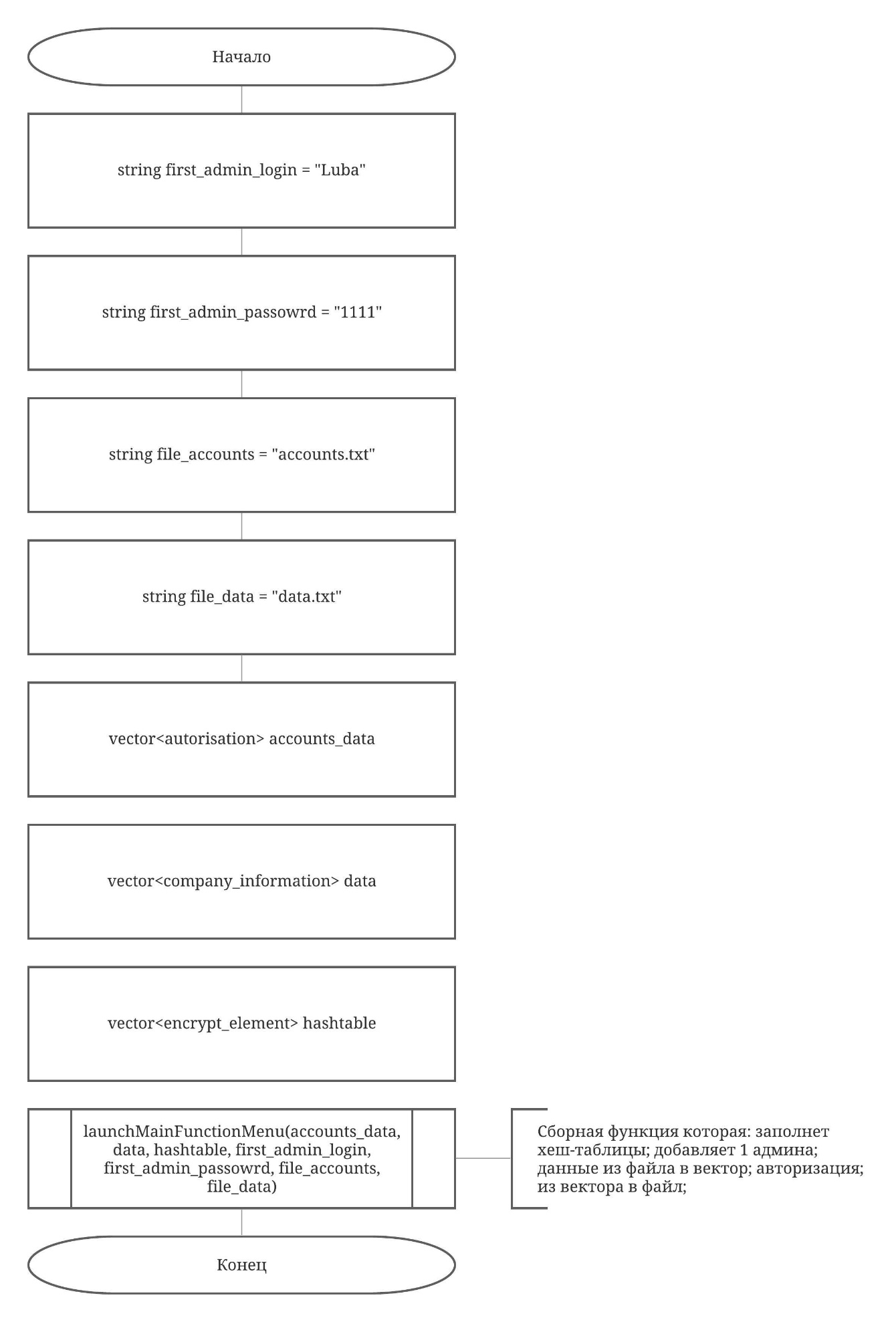
void copyInVector(vector<company\_information>&, string, vector <autorisation>&, string); //записать все данные из файла в вектор

void fromFileToVectorACCOUNTS(vector <autorisation>&, string); //запись данных из файла в вектор с аккаунтами

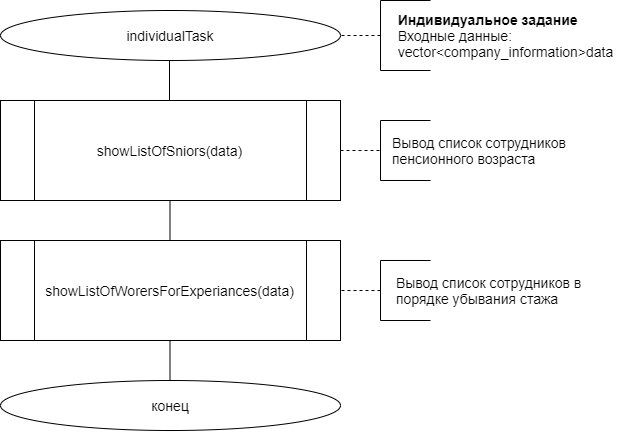
void fromFileToVectorDATA(vector<company\_information>&, string); //запись данных из файла в вектор с данными

**3. Разработка алгоритмов программы**

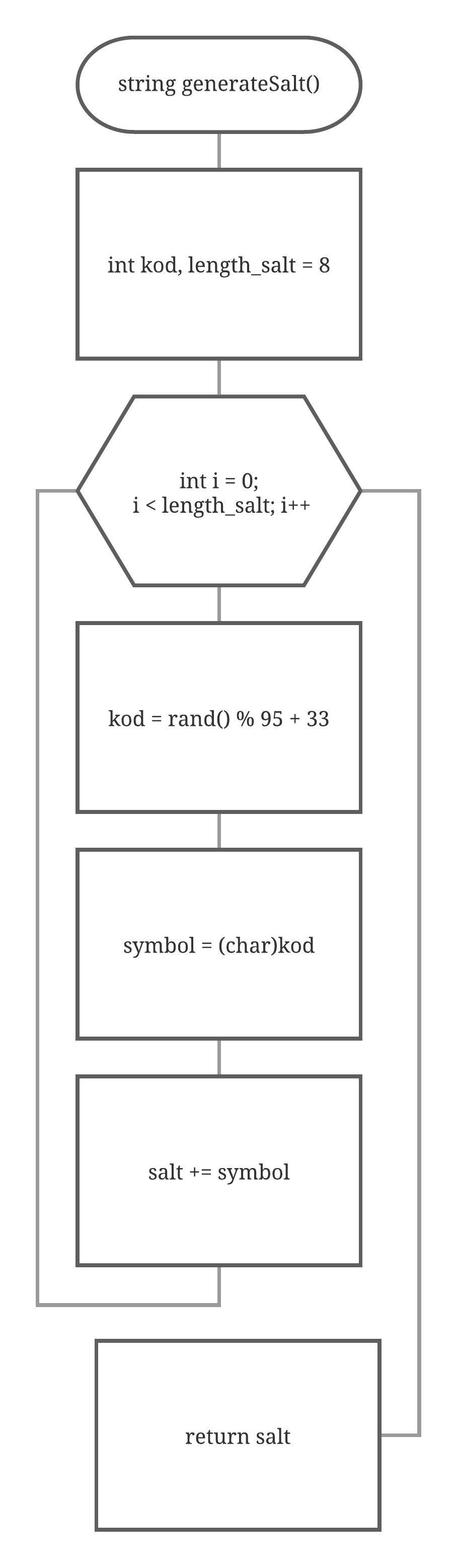
**3.1 Алгоритм функции main**

****

**3.2 Алгоритм функции individualTask.**

****

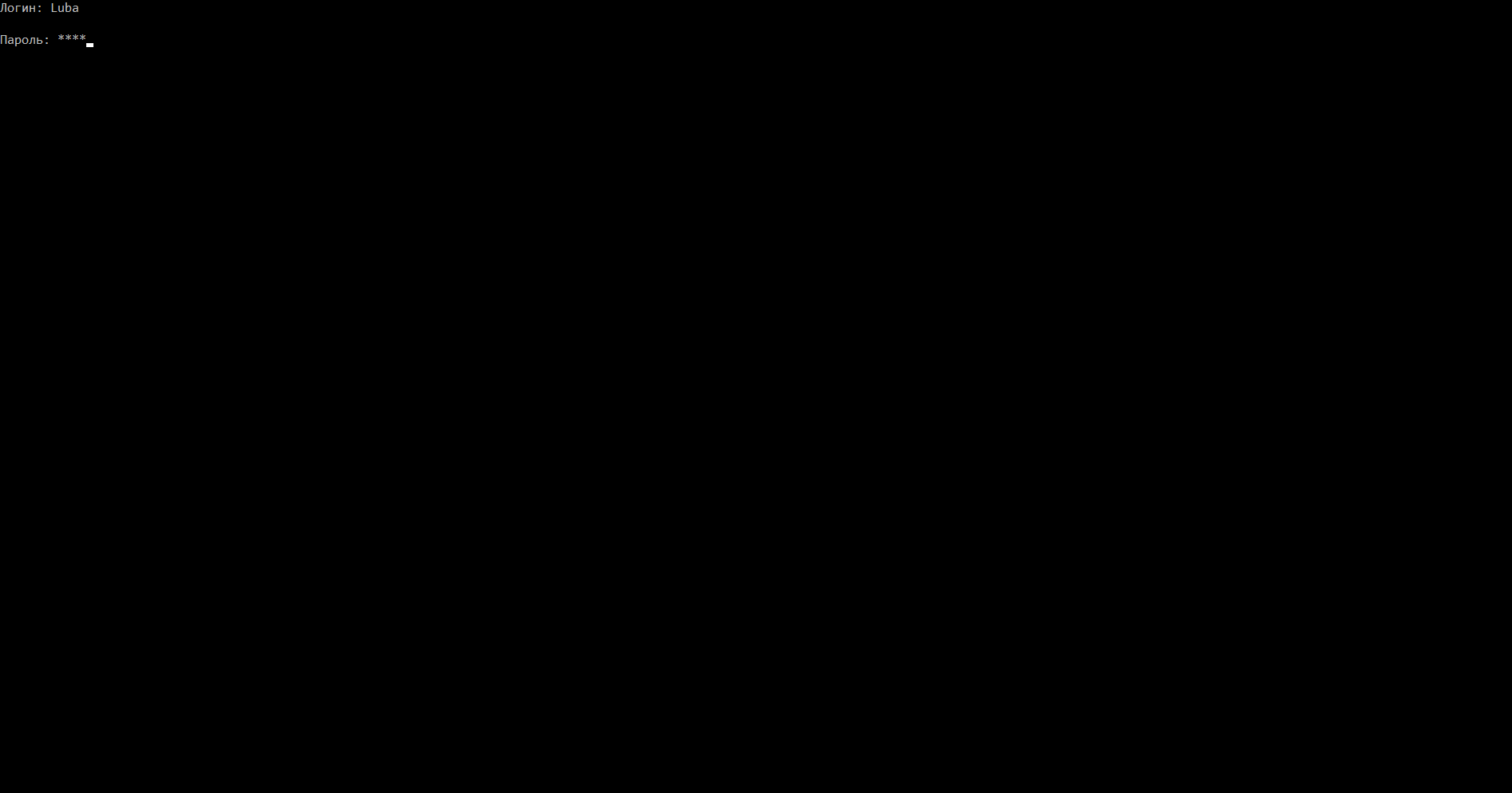
**3.3 Алгоритм функции generateSalt.**

****

**4. Описание работы программы**

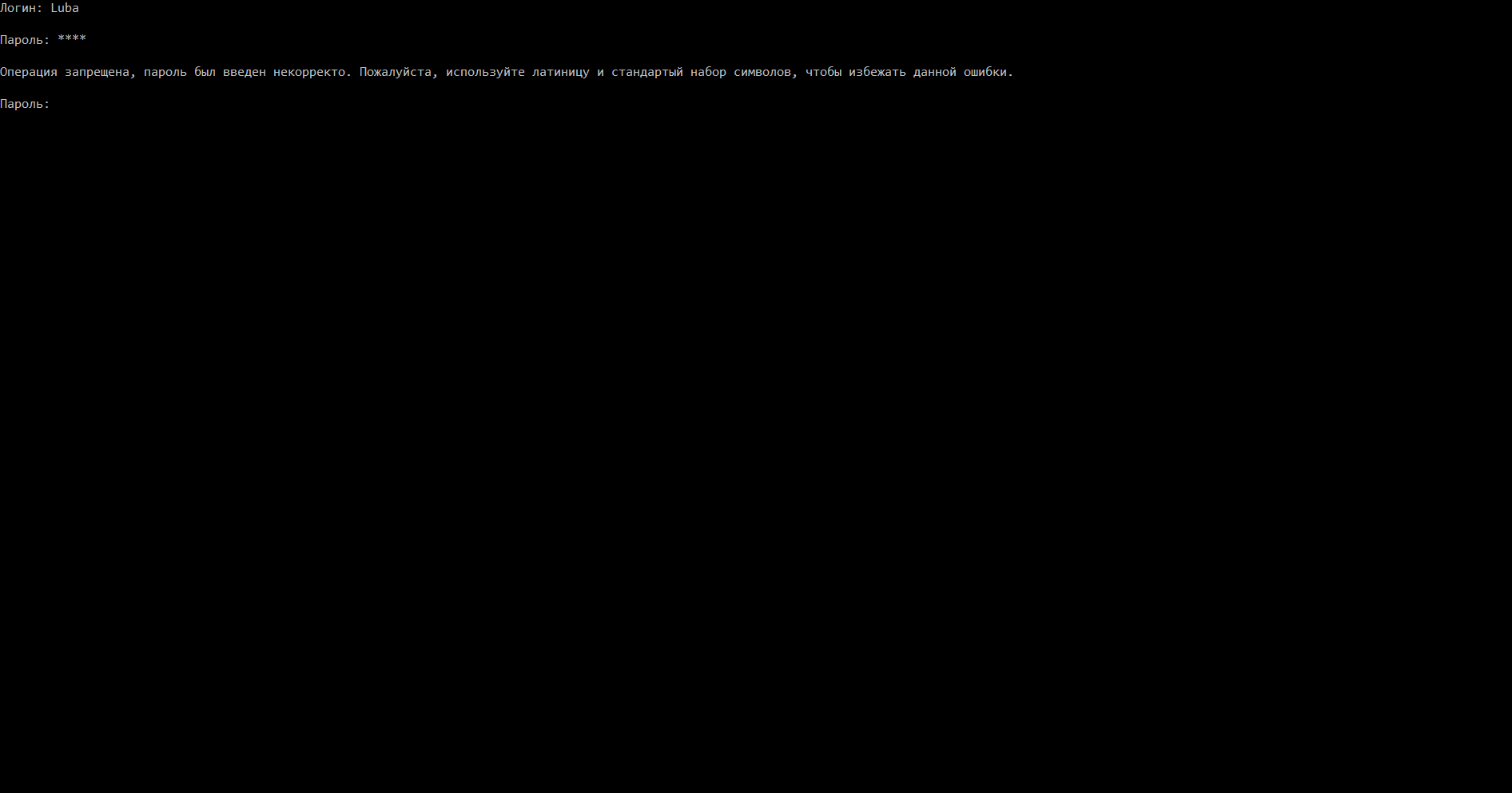
**4.1 Авторизация**

При запуске программы происходит процесс авторизации *(Рис 1.1).*

**

*Рис 1.1—авторизация*

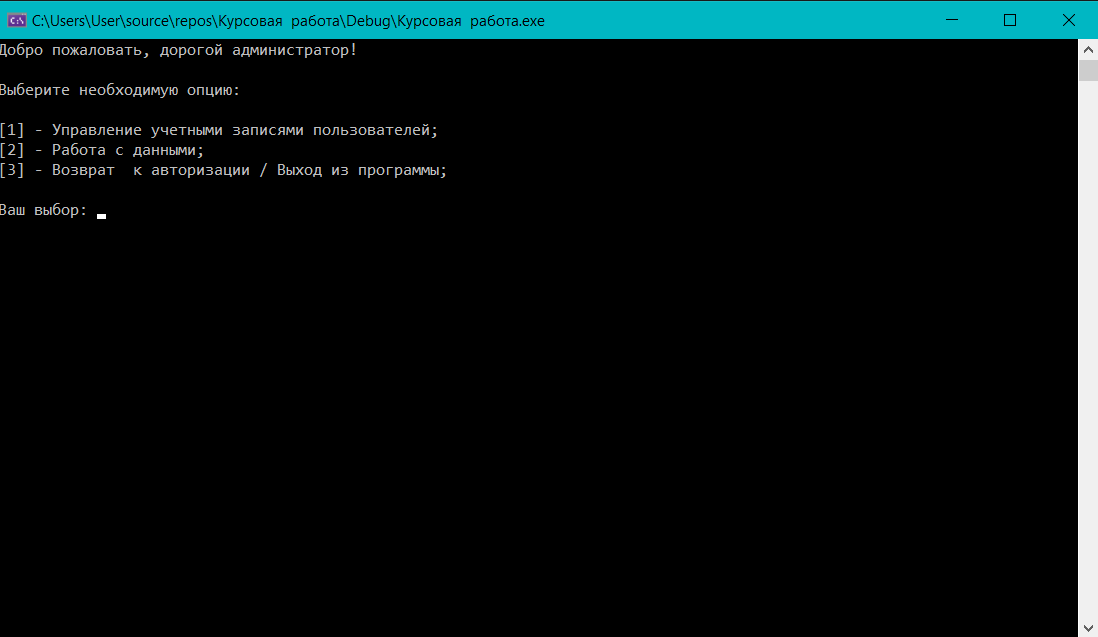
Проверка введенных данных на соответствие проводиться после введения логина и пароля. Логин чувствителен к регистру. В случае несовпадения данных выводится соответствующее сообщение *(Рис 1.2).*

**

*Рис 1.2—сообщение об ошибке*

**4.2 Модуль администратора**

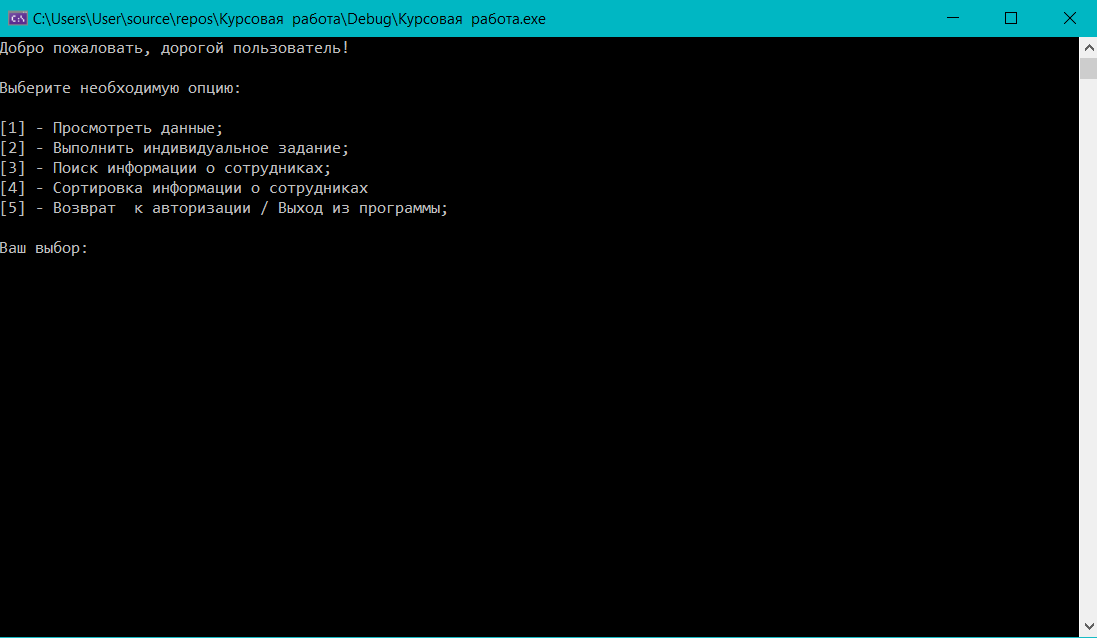
Модуль администратора состоит из следующих подмодулей: управление учетными записями пользователей (просмотр, добавление, редактирование, удаление учетных записей) и работа с данными (режим редактирования и режим обработки) *(Рис 1.3).*



*Рис 1.3—Модуль администратора*

**4.3 Модуль пользователя**

Модуль пользователя состоит из следующих подмодулей: просмотреть данные, выполнить индивидуальное задание, поиск информации о сотрудниках, сортировка информации о сотрудниках *(Рис 1.4).*



*Рис 1.4—Модуль пользователя*

**4.3 Исключительные ситуации**

В курсовой работе учтено достаточно много случаев, где пользователь может ввести некорректные данные. Также учтены и системные случаи. Список некоторых случаев:

— Запрещен ввод нечисловых данных, где необходимы числа;

— ввод несуществующей даты (нижний порог 1920 г., верхний – 2020);

— проверка соответствия после ввода (например, логин и пароль от аккаунта);

— проверка наличия файла данных;

— защита аккаунта от взлома (маскирование и шифрование пароля);

— если база данных пуста, то пользователь не может функционировать с ней;

— пользователь не может использовать логин, числящийся в базе данных.

**Приложение**

**main.cpp**

// Информация о сотрудниках предприятия содержит :

//ФИО сотрудника,

//дату рождения,

//название отдела,

//должность,

//дату начала работы.

//Индивидуальное задание :

//вывести список сотрудников пенсионного возраста.Вывести список сотрудников в порядке убывания стажа.

#include "startInfo.h"

#include "autorisation.h"

#include "adminMenu.h"

#include "workWithAccountsAdmin.h"

#include "workWithFilesAdmin.h"

#include "workWithDataAdmin.h"

#include "searchMenu.h"

#include "sortMenu.h"

#include "userMenu.h"

#include "hashFunction.h"

#include "Windows.h"

int main()

{

setlocale(0, "rus");

srand(time(0));

string first\_admin\_login = "Luba";

string first\_admin\_passowrd = "1111";

string file\_accounts = "accounts.txt";

string file\_data = "data.txt";

vector<autorisation> accounts\_data;

vector<company\_information> data;

vector <encrypt\_element> hashtable;

launchMainFunctionMenu(accounts\_data, data, hashtable, first\_admin\_login, first\_admin\_passowrd, file\_accounts, file\_data);

}

**Autorisation.cpp**

#include "autorisation.h"

void autorisationMainFunction(vector <company\_information>& data, vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable)

{

int choise, vvod, kod, create\_account, continue\_work;

bool case\_equal;

autorisation registration\_data;

while (true)

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

registration\_data.login = inputLogin();

registration\_data.passowrd = inputStarsPassword();

SetConsoleOutputCP(866);

SetConsoleCP(866);

string unhashed\_pass = registration\_data.passowrd;

// поиск аккаунта с введёнными данными и запуск меню в случае успеха:

entranceInProgram(case\_equal, registration\_data, hashtable, accounts\_data, data);

// Если аккаунт был найден, то case\_equal == 1 и // повторный вызов меню авторизации после работы

if (case\_equal == 1) // если пользователь найден и был вызов меню

{

behaviorAfterWorkInProgram(kod);

if (kod == 1)

{

continue;

}

else

{

break;

}

}

if (case\_equal == 0) // если такого пользователя нет, то либо ошибка ввода данных, либо нет такой учётной записи

{

if (isAutorisationCycleBreakOrContinue(registration\_data, unhashed\_pass, accounts\_data, hashtable, create\_account, continue\_work, choise))

{

break;

}

else

{

continue;

}

}

}

}

void entranceInProgram(bool& case\_equal, autorisation& registration\_data, vector <encrypt\_element> hashtable, vector <autorisation>& accounts\_data, vector <company\_information>& data)

{

for (int i = 0; i < accounts\_data.size(); i++)

{

case\_equal = 0;

if (registration\_data.login == accounts\_data.at(i).login )

{

case\_equal = 1;

}

if (case\_equal == 1)

{

registration\_data.passowrd = encryptCurrentPass(registration\_data.passowrd, hashtable, accounts\_data.at(i).salt);

if (registration\_data.passowrd == accounts\_data.at(i).passowrd)

{

if (accounts\_data.at(i).role == 1)

{

adminMenu(accounts\_data, data, hashtable);

break;

}

else

{

userMenu(data);

break;

}

}

else

{

case\_equal = 0;

continue;

}

}

else

{

continue;

}

}

}

void behaviorAfterWorkInProgram(int& kod)

{

// Повторный ввод данных и запуск авторизации по желанию пользователя либо завершение программы

do

{

cout << "Выполнить повторную авторизацию, или же выйти из программы ?\n\n";

cout << "[1] - Повторная авторизация;\n";

cout << "[2] - Выход из программы;\n\n";

cout << "Ввод: ";

cin >> kod;

if (!isBufferWithNoChar())

{

continue;

}

else if (kod == 1)

{

system("cls");

break;

}

else if (kod == 2)

{

system("cls");

cout << "Программа завершена.\n\n";

break;

}

else

{

choiseError();

continue;

}

} while (true);

}

bool isBufferWithNoChar()

{

bool no\_char = 1;

if (cin.fail())

{

no\_char = 0;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

system("cls");

cout << "Ошибка ввода ! Данное поле поддерживает только целочисленные значения. Пожалуйста, попробуйте еще раз.\n\n";

system("pause");

system("cls");

}

return no\_char;

}

bool isAutorisationCycleBreakOrContinue(autorisation& registration\_data, string unhashed\_pass, vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable, int& create\_account, int& continue\_work, int& choise)

{

bool case\_break\_continue = 0; // break == 1, continue == 0;

accountNotExist(registration\_data, unhashed\_pass, accounts\_data, hashtable, create\_account, continue\_work, choise);

if (choise == 1) // Ошибка ввода данных. Выполняется повторный ввод.

{

case\_break\_continue = 0;

}

else if (choise == 2) // Нет учётной записи. Нужно создать, но только по желанию пользователя.

{

if (create\_account == 2)

{

case\_break\_continue = 0;

}

else

{

if (continue\_work == 1)

{

case\_break\_continue = 0;

}

else

{

case\_break\_continue = 1;

}

}

}

else if (choise == 3)

{

system("cls");

cout << "Программа завершена.\n\n";

case\_break\_continue = 1;

}

return case\_break\_continue;

}

void accountNotExist(autorisation& registration\_data, string unhashed\_pass, vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable, int& create\_account, int& continue\_work, int& choise)

{

registration\_data.role = 0;

do

{

cout << "\n\nДанной учётной записи не существует.\n\n";

cout << "В чём заключается ваша проблема ?\n\n";

cout << "[1] - Неправильно введённые данные;\n";

cout << "[2] - Хочу выйти из программы;\n\n";

cout << "Ввод: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

else if (choise == 1)

{

errorDuringInputData();

break;

}

else if (choise == 2)

{

break;

}

else

{

choiseError();

continue;

}

} while (true);

}

void errorDuringInputData()

{

system("cls");

cout << "Выполните повторный ввод данных." << endl;

cout << "!!! БУДЬТЕ ВНИМАТЕЛЬНЫ !!!" << endl << endl;

}

string inputStarsPassword()

{

string password;

vector <int> kods;

int kod\_char = 0;

cin.ignore(cin.rdbuf()->in\_avail());

cout << "\nПароль: ";

while (true)

{

kod\_char = \_getwch(); // считывание кода нажатия клавиши

kods.push\_back(kod\_char);

if (kod\_char == 13) // Код Enter

{

break;

}

else

if (kod\_char == 8)

{

if (password.empty())

{

continue;

}

cout << (char)8; //<< ' ' << (char)8;

cout << ' ';

cout << (char)8;

if (!password.empty())

{

password.erase(password.length() - 1); //удаляет 1 элемент

}

}

else

{

cout << "\*";

password += (char)kod\_char;

}

}

for (int i = 0; i < kods.size(); i++)

{

if (kods.at(i) >= 33 && kods.at(i) <= 127)

{

continue;

}

else if (kods.at(i) == 13 && i == kods.size() - 1 && i != 0)

{

continue;

}

else

{

cout << "\n\nОперация запрещена, пароль был введен некорректо. Пожалуйста, используйте латиницу и стандартый набор символов, чтобы избежать данной ошибки.\n";

password = inputStarsPassword();

break;

}

}

return password;

}

void addFirstAdmin(string path, string first\_admin\_login, string first\_admin\_passowrd, vector <encrypt\_element> hashtable)

{

if (!isFileExist(path))

{

ofstream accounts\_file;

accounts\_file.open(path, ofstream::out | ofstream::binary);

if (accounts\_file.is\_open())

{

string salt = generateSalt();

first\_admin\_passowrd = encryptCurrentPass(first\_admin\_passowrd, hashtable, salt);

accounts\_file << first\_admin\_login.length() << " " << first\_admin\_login << " " << first\_admin\_passowrd.length() << " "

<< first\_admin\_passowrd << " " << 1 << " " << 1 << " " << salt.length() << " " << salt << " ";

accounts\_file.close();

}

else

{

cout << "Error in creating file accounts.\n\n";

exit(1);

}

}

}

bool isFileExist(string path)

{

ifstream fin;

autorisation box;

bool file\_exist = 0;

fin.open(path, ifstream::in | ifstream::binary);

if (fin.is\_open())

{

file\_exist = 1;

}

fin.close();

return file\_exist;

}

void choiseError()

{

system("cls");

cout << "Ошибка ввода ! Пожалуйста, выберите один из предложенных вариантов.\n\n";

system("pause");

system("cls");

}

**adminMenu.cpp**

#include "adminMenu.h"

void adminMenu(vector <autorisation>& accounts\_data, vector<company\_information>& data, vector <encrypt\_element> hashtable)

{

int kod;

while (true)

{

system("cls");

cout << "Добро пожаловать, дорогой администратор!" << endl << endl;

cout << "Выберите необходимую опцию:" << endl << endl;

cout << "[1] - Управление учетными записями пользователей;" << endl;

cout << "[2] - Работа с данными;" << endl;

cout << "[3] - Возврат к авторизации / Выход из программы;" << endl << endl;

cout << "Ваш выбор: ";

cin >> kod;

if (!isBufferWithNoChar())

{

continue;

}

switch (kod)

{

case 1:

system("cls");

workWithAccountsAdmin(accounts\_data, hashtable);

system("cls");

break;

case 2:

system("cls");

workWithDataAdmin(data);

system("cls");

break;

case 3:

system("cls");

break;

default:

choiseError();

}

if (kod == 3)

{

break;

}

}

}

**workWithAccountsAdmin.cpp**

#include "workWithAccountsAdmin.h"

void workWithAccountsAdmin(vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable)

{

int choise;

bool block\_delete = 0;

char warning\_block\_delete[] = "Дальнейшее удаление невозможно. В списке аккаунтов осталась только одна запись, являющаяся единственным администратором системы и не подлежащая удалению.\n\n";

while (true)

{

cout << "Вы находитесь в режиме работы с учетными записями пользователей" << endl << endl;

cout << "Выберите необходимую опцию:" << endl << endl;

cout << "[1] - Просмотр всех учетных записей;" << endl;

cout << "[2] - Добавление новой учетной записи;" << endl;

cout << "[3] - Редактирование учетной записи; " << endl;

cout << "[4] - Удаление учетной записи;" << endl;

cout << "[5] - Назад;" << endl << endl;

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

system("cls");

viewAccounts(accounts\_data);

system("pause");

system("cls");

break;

case 2:

system("cls");

addNewAccount(accounts\_data, hashtable);

break;

case 3:

system("cls");

redactAccount(accounts\_data, hashtable);

system("cls");

break;

case 4:

system("cls");

if (block\_delete == 0)

{

deleteAccount(accounts\_data, block\_delete, warning\_block\_delete);

}

else

{

cout << warning\_block\_delete;

system("pause");

}

system("cls");

break;

case 5:

system("cls");

break;

default:

choiseError();

}

if (choise == 5)

{

break;

}

}

}

string inputLogin()

{

string login;

cout << "Логин: ";

cin.ignore(cin.rdbuf()->in\_avail());

getline(cin, login);

if (!isAllSymbolsCorrect(login))

{

cout << "\nОперация запрещена, логин был введен некорректо. Пожалуйста, используйте латиницу и стандартый набор символов, чтобы избежать данной ошибки.\n\n";

login = inputLogin();

}

return login;

}

bool isAllSymbolsCorrect(string str)

{

bool case\_correct = 1;

if (str.empty())

{

case\_correct = 0;

}

else

{

for (int i = 0; i < str.length(); i++)

{

if (str.at(i) >= 33 && str.at(i) <= 127)

{

continue;

}

else

{

case\_correct = 0;

break;

}

}

}

return case\_correct;

}

bool isTheSameLoginInVector(vector <autorisation> accounts\_data, string login, int current\_index)

{

bool case\_same = 0;

for (int i = 0; i < accounts\_data.size(); i++)

{

if (accounts\_data.at(i).login == login && i != current\_index)

{

case\_same = 1;

break;

}

}

return case\_same;

}

void viewAccounts(vector <autorisation> accounts\_data)

{

if (accounts\_data.empty())

{

cout << "Операция невозможна, информация для вывода отсутствует. Пожалуйста, введите данные для дальнейшей корректной работы программы.\n\n";

return;

}

cout << "----------------------------------------------------------------------------------------------------------------\n";

cout << setw(10) << "№ номер" << setw(3) << "|" << setw(30) << "Логин" << setw(2) << "|" << setw(30) << "Пароль" << setw(2) << "|" << setw(33) << "Роль в системе" << endl;

cout << "----------------------------------------------------------------------------------------------------------------\n";

for (int i = 0; i < accounts\_data.size(); i++)

{

if (accounts\_data.at(i).role == 1)

{

cout << setw(7) << i + 1 << setw(6) << "|" << setw(30) << accounts\_data.at(i).login << setw(2) << "|" << setw(30) << accounts\_data.at(i).passowrd << setw(2) << "|" << setw(33) << "Админ" << endl;

}

else

{

cout << setw(7) << i + 1 << setw(6) << "|" << setw(30) << accounts\_data.at(i).login << setw(2) << "|" << setw(30) << accounts\_data.at(i).passowrd << setw(2) << "|" << setw(33) << "Пользователь" << endl;

}

cout << "----------------------------------------------------------------------------------------------------------------\n";

}

cout << endl;

}

void addNewAccount(vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable)

{

int kod;

while (true)

{

cout << "Пожалуйста, выберите необходимую опцию:" << endl << endl;

cout << "[1] - Добавить администратора;" << endl;

cout << "[2] - Добавить пользователя;" << endl;

cout << "[3] - Назад;" << endl << endl;

cout << "Ваш выбор: ";

cin >> kod;

if (!isBufferWithNoChar())

{

continue;

}

switch (kod)

{

case 1:

system("cls");

addAdministrator(accounts\_data, hashtable);

system("cls");

break;

case 2:

system("cls");

addUser(accounts\_data, hashtable);

system("cls");

break;

case 3:

system("cls");

break;

default:

choiseError();

}

if (kod == 3)

{

break;

}

}

}

autorisation inputInfoAboutNewAccount(vector <encrypt\_element> hashtable, vector <autorisation> accounts\_data)

{

autorisation box;

int choise;

do

{

cout << "Пожалуйста, введите информацию о новой учетной записи:\n\n";

box.login = inputUniqueLogin(accounts\_data);

box.passowrd = inputStarsPassword();

box.salt = generateSalt();

box.passowrd = encryptCurrentPass(box.passowrd, hashtable, box.salt);

cout << "\n\nПеред тем как создать новый аккаунт проверьте введенные данные. Возможно вы допустили ошибку или желайте изменить введенную информацию.\n\n";

cout << "Создать новую учетную запись ?\n\n";

cout << "[1] - Да, все данные введены корректно;\n";

cout << "[2] - Нет, хочу внести изменения в логин / пароль;\n\n";

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

if (choise == 2)

{

system("cls");

}

else if (choise == 1)

{

break;

}

else

{

choiseError();

}

}

while (choise == 2);

return box;

}

string inputUniqueLogin(vector <autorisation> accounts\_data)

{

string login = inputLogin();

for (int i = 0; i < accounts\_data.size(); i++)

{

if (accounts\_data.at(i).login == login)

{

cout << "\nОперация невозможна, такой логин уже существует. Пожалуйста, повторите ввод.\n\n";

login = inputUniqueLogin(accounts\_data);

break;

}

}

return login;

}

void addAdministrator(vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable)

{

autorisation box;

do

{

box = inputInfoAboutNewAccount(hashtable, accounts\_data);

box.role = 1;

accounts\_data.push\_back(box);

if (!isWorkContinue())

{

break;

}

} while (true);

}

void addUser(vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable)

{

autorisation box;

do

{

box = inputInfoAboutNewAccount(hashtable, accounts\_data);

box.role = 0;

accounts\_data.push\_back(box);

if (!isWorkContinue())

{

break;

}

} while (true);

}

void redactAccount(vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable)

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int number, choise, index;

bool one\_more\_time;

do

{

viewAccounts(accounts\_data);

cout << "Введите номер записи, которую хотите отредактировать: ";

cin >> number;

if (!isBufferWithNoChar())

{

continue;

}

else if (number < 1 || number > accounts\_data.size())

{

system("cls");

cout << "Ошибка ! Введите корректный номер записи.\n\n";

system("pause");

system("cls");

continue;

}

index = number - 1; // Считаем индекс текущего элемента. Помните, что нумерация элементов массива начинается с "0"

one\_more\_time = 0;

cout << "\nВыберите:\n\n[1] - Логин;\n[2] - Пароль;\n\n";

cout << "Ввод: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

one\_more\_time = 0;

cout << "\nВведите новый логин: ";

cin.ignore(cin.rdbuf()->in\_avail());

getline(cin, accounts\_data.at(index).login);

cout << endl;

break;

case 2:

one\_more\_time = 0;

cout << "\nВведите новый пароль:\n";

accounts\_data.at(index).passowrd = inputStarsPassword();

accounts\_data.at(index).passowrd = encryptCurrentPass(accounts\_data.at(index).passowrd, hashtable, accounts\_data.at(index).salt);

cout << endl;

break;

default:

one\_more\_time = 1;

choiseError();

}

} while (one\_more\_time == 1);

cout << endl << endl;

viewAccounts(accounts\_data);

if (isWorkContinue())

{

redactAccount(accounts\_data, hashtable);

}

SetConsoleOutputCP(866);

SetConsoleCP(866);

}

void deleteAccount(vector <autorisation>& account\_data, bool& block\_delete, char\* warning\_block\_delete)

{

int number\_zap, index;

bool find\_more\_admin;

block\_delete = 0;

do

{

viewAccounts(account\_data);

cout << "\nВыберите номер записи, данные которой хотите удалить:\n";

cout << "\nВвод: ";

cin >> number\_zap;

if (!isBufferWithNoChar())

{

continue;

}

else if (number\_zap < 0 || number\_zap > account\_data.size())

{

system("cls");

cout << "Ошибка удаления - такой записи не существует.\n\n";

system("pause");

system("cls");

continue;

}

find\_more\_admin = 0;

index = number\_zap - 1;

if (account\_data.at(index).role == 1)

{

for (int i = 0; i < account\_data.size(); i++)

{

if (account\_data.at(i).role == 1 && account\_data.at(i).login != account\_data.at(index).login && account\_data.at(i).passowrd != account\_data.at(index).passowrd)

{

find\_more\_admin = 1;

}

}

if (find\_more\_admin == 0 && account\_data.size() > 1)

{

system("cls");

cout << "Невозможно удалить данную учетную запись, так как она является аккаунтом единственного администратора системы.\n\n";

cout << "Вы можете продолжить удаление оставшихся пользователей или же закончить операцию удаления.\n";

}

else if (find\_more\_admin == 0 && account\_data.size() == 1)

{

block\_delete = 1;

system("cls");

cout << warning\_block\_delete;

system("pause");

break;

}

}

if (account\_data.at(index).role == 0 || (account\_data.at(index).role == 1 && find\_more\_admin == 1))

{

account\_data.erase(account\_data.begin() + index);

cout << "\nПосле удаления информация о проектах имеет вид:\n\n";

viewAccounts(account\_data);

}

if (!isWorkContinue())

{

cout << endl;

break;

}

} while (true);

}

**workWithFilesAdmin.cpp**

#include "workWithFilesAdmin.h"

void launchMainFunctionMenu(vector <autorisation>& accounts\_data, vector <company\_information>& data, vector <encrypt\_element>& hashtable, string first\_admin\_login, string first\_admin\_passowrd, string file\_accounts, string file\_data)

{

fillHashTable(hashtable);

addFirstAdmin(file\_accounts, first\_admin\_login, first\_admin\_passowrd, hashtable);

copyInVector(data, file\_data, accounts\_data, file\_accounts);

autorisationMainFunction(data, accounts\_data, hashtable);

pushDataInFile(data, file\_data, accounts\_data, file\_accounts);

}

void copyInVector(vector<company\_information>& data, string file\_data, vector <autorisation>& accounts\_data, string file\_accounts)

{

fromFileToVectorACCOUNTS(accounts\_data, file\_accounts);

fromFileToVectorDATA(data, file\_data);

}

void fromFileToVectorACCOUNTS(vector <autorisation>& accounts\_data, string file\_accounts)

{

ifstream fin;

fin.open(file\_accounts, ifstream::in, ifstream::binary);

if (fin.is\_open())

{

int len\_login, len\_passowrd, len\_salt;

bool role, access;

autorisation box;

while (!fin.eof())

{

fin >> len\_login;

char\* login = new char[len\_login];

fin >> login >> len\_passowrd;

char\* password = new char[len\_passowrd];

fin >> password >> role>> access >> len\_salt;

char\* salt = new char[len\_salt];

fin >> salt;

if (strlen(login) == 0)

{

continue;

}

box.login = login;

box.passowrd = password;

box.role = role;

box.salt = salt;

accounts\_data.push\_back(box);

}

}

fin.close();

}

void fromFileToVectorDATA(vector<company\_information>& data, string file\_data)

{

if (isFileExist(file\_data))

{

ifstream fin;

fin.open(file\_data, ifstream::in, ifstream::binary);

if (fin.is\_open())

{

int len\_fio, birthday\_day, birthday\_mounth, birthday\_year, len\_department, len\_occupation, begin\_day, begin\_mounth, begin\_year;

char\* department, \* occupation, \* fio, sex;

company\_information box;

string str;

getline(fin, str);

if (str.empty())

{

return;

}

else

{

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

}

while (!fin.eof())

{

fin >> len\_fio;

fio = new char[len\_fio];

fin >> fio >> sex >> birthday\_day >> birthday\_mounth >> birthday\_year >> len\_department;

department = new char[len\_department];

fin >> department >> len\_occupation;

occupation = new char[len\_occupation];

fin >> occupation >> begin\_day >> begin\_mounth >> begin\_year;

if (strlen(fio) == 0)

{

continue;

}

box.fio = fio;

box.sex = sex;

box.birthday.day = birthday\_day;

box.birthday.mounth = birthday\_mounth;

box.birthday.year = birthday\_year;

box.department = department;

box.occupation = occupation;

box.begin\_work\_date.day = begin\_day;

box.begin\_work\_date.mounth = begin\_mounth;

box.begin\_work\_date.year = begin\_year;

data.push\_back(box);

}

}

fin.close();

}

}

void pushDataInFile(vector <company\_information> data, string path\_data, vector <autorisation> accounts\_data, string path\_accounts)

{

fromVectorToFileACCOUNTS(accounts\_data, path\_accounts);

fromVectorToFileDATA(data, path\_data);

}

void fromVectorToFileACCOUNTS(vector <autorisation> accounts\_data, string path\_accounts)

{

ofstream fout;

fout.open(path\_accounts, ofstream::out | ofstream::binary);

for (int i = 0; i < accounts\_data.size(); i++)

{

fout << accounts\_data.at(i).login.length() << " " << accounts\_data.at(i).login << " " << accounts\_data.at(i).passowrd.length() << " " << accounts\_data.at(i).passowrd

<< " " << accounts\_data.at(i).role << " " << accounts\_data.at(i).salt.length() << " " << accounts\_data.at(i).salt << " ";

}

fout.close();

}

void fromVectorToFileDATA(vector <company\_information> data, string path\_data)

{

ofstream fout;

fout.open(path\_data, ofstream::out | ofstream::binary);

for (int i = 0; i < data.size(); i++)

{

fout << data.at(i).fio.length() << " " << data.at(i).fio << " " << data.at(i).sex << " " << data.at(i).birthday.day << " " << data.at(i).birthday.mounth << " " << data.at(i).birthday.year

<< " " << data.at(i).department.length() << " " << data.at(i).department << " " << data.at(i).occupation.length() << " " << data.at(i).occupation

<< " " << data.at(i).begin\_work\_date.day << " " << data.at(i).begin\_work\_date.mounth << " " << data.at(i).begin\_work\_date.year << " ";

}

fout.close();

}

**workWithDataAdmin.cpp**

#include "workWithDataAdmin.h"

void workWithDataAdmin(vector<company\_information>& data)

{

int choise;

while (true)

{

cout << "Вы находитесь в режиме работы с данными" << endl << endl;

cout << "Выберите необходимую опцию:" << endl << endl;

cout << "[1] - Режим редактирования;" << endl;

cout << "[2] - Режим обработки;" << endl;

cout << "[3] - Назад; " << endl << endl;

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

system("cls");

redactingModeAdmin(data);

system("cls");

break;

case 2:

system("cls");

processingModeAdmin(data);

system("cls");

break;

case 3:

system("cls");

break;

default:

choiseError();

}

if (choise == 3)

{

break;

}

}

}

void redactingModeAdmin(vector<company\_information>& data)

{

system("cls");

int choise;

while (true)

{

cout << "Вы находитесь в режиме редактирования данных" << endl << endl;

cout << "Выберите необходимую опцию:" << endl << endl;

cout << "[1] - Добавить информацию о сотрудниках предприятия;" << endl;

cout << "[2] - Просмотреть данные;" << endl;

cout << "[3] - Редактировать данные;" << endl;

cout << "[4] - Удалить данные;" << endl;

cout << "[5] - Назад; " << endl << endl;

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

if (data.empty() && (choise == 2 || choise == 3 || choise == 4))

{

system("cls");

cout << "Невозможно выполнить операцию, так как отсутствует информация для обработки. Пожалуйста, введите данные в программу для дальнейшей корректной работы.\n\n";

system("pause");

system("cls");

continue;

}

switch (choise)

{

case 1:

system("cls");

addEmployeesData(data);

system("cls");

break;

case 2:

system("cls");

viewData(data);

system("pause");

system("cls");

break;

case 3:

system("cls");

redactData(data);

system("cls");

break;

case 4:

system("cls");

deleteData(data);

system("cls");

break;

case 5:

system("cls");

break;

default:

choiseError();

}

if (choise == 5)

{

break;

}

}

}

string inputWithUpRegister()

{

string str;

cin.ignore(cin.rdbuf()->in\_avail());

getline(cin, str);

while (IsFirstCharInUpperRegisterAndNotSpace(str[0]) == 0)

{

cout << "\nПовторный ввод: ";

cin.ignore(cin.rdbuf()->in\_avail());

getline(cin, str);

}

return str;

}

void addEmployeesData(vector<company\_information>& data) //Employee

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

company\_information zap;

int sum\_date\_1, sum\_date\_2, old\_mounth\_1, old\_mounth\_2;

do

{

cout << "Введите информацию о сотруднике\n\n";

cout << "Введите фамилию и инициалы сотрудника:\n\n";

cout << "Пример: Иванов И. И. , Федоров И. И. , Смирнов А. А.\n";

cout << "\nВвод: ";

zap.fio = inputString();

cout << "\nВыберите пол сотрудника. Эта информация крайне важна при подсчете пенсионного возраста.\n";

zap.sex = inputSex();

cout << "\nВведите дату рождения сотрудника:\n\n";

zap.birthday = inputDate();

cout << "\nВведите название отдела, где работает сотрудник: ";

zap.department = inputString();

cout << "\nВведите должность сотрудинка: ";

zap.occupation = inputString();

do

{

cout << "\nВведите дату начала работы сотрудника:\n\n";

zap.begin\_work\_date = inputDate();

for (int m = 0; m < NUMBER\_MOUNTH; m++)

{

if (m == zap.birthday.mounth - 1)

{

old\_mounth\_1 = zap.birthday.mounth;

zap.birthday.mounth = NUMBER\_DAYS\_IN\_MONTH[m];

}

if (m == zap.begin\_work\_date.mounth - 1)

{

old\_mounth\_2 = zap.begin\_work\_date.mounth;

zap.begin\_work\_date.mounth = NUMBER\_DAYS\_IN\_MONTH[m];

}

}

sum\_date\_1 = zap.birthday.day + zap.birthday.mounth + zap.birthday.year \* NUMBER\_DAYS\_IN\_YEAR;

sum\_date\_2 = zap.begin\_work\_date.day + zap.begin\_work\_date.mounth + zap.begin\_work\_date.year \* NUMBER\_DAYS\_IN\_YEAR;

if (sum\_date\_1 >= sum\_date\_2)

{

cout << "\nОшибка ! Вы не могли утроиться на работу в указанную дату, так как к этому моменту вы либо только родились, либо вас еще нет на свете.\n\n";

cout << "Пожалуйста, введите корректные данные.\n";

zap.birthday.mounth = old\_mounth\_1;

continue;

}

else

{

zap.birthday.mounth = old\_mounth\_1;

zap.begin\_work\_date.mounth = old\_mounth\_2;

break;

}

} while (true);

data.push\_back(zap);

if (!isWorkContinue())

{

break;

}

} while (true);

SetConsoleOutputCP(866);

SetConsoleCP(866);

}

char inputSex()

{

int choise;

char sex;

cout << "\nВаш пол:\n\n";

cout << "[1] - Мужчина;\n";

cout << "[2] - Женщина;\n\n";

do

{

cout << "Ввод: ";

cin >> choise;

if (cin.fail())

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

cout << "\nОшибка ввода ! Данное поле поддерживает только целочисленные значения. Пожалуйста, попробуйте еще раз.\n\n";

continue;

}

else if (choise == 1)

{

sex = 'М';

break;

}

else if (choise == 2)

{

sex = 'Ж';

break;

}

else

{

cout << "\nОшибка ввода ! Пожалуйста, выберите один из предложенных вариантов.\n\n";

continue;

}

} while (true);

return sex;

}

date inputDate()

{

int choise;

date box;

do

{

cout << "День: ";

cin >> box.day;

if (cin.fail())

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

cout << "\nОшибка ввода ! Данное поле поддерживает только целочисленные значения. Пожалуйста, попробуйте еще раз.\n\n";

continue;

}

else if (box.day <= MIN\_DAYS || box.day > MAX\_DAYS)

{

cout << "\nОшибка ! Пожалуйста, введите корректнре число.\n\n";

continue;

}

else

{

break;

}

} while (true);

cout << "\nВыберите месяц:\n\n";

cout << "[1] - Январь;\t[4] - Апрель;\t[7] - Июль;\t[10] - Октябрь;\n";

cout << "[2] - Февраль;\t[5] - Май;\t[8] - Август;\t[11] - Ноябрь;\n";

cout << "[3] - Март;\t[6] - Июнь;\t[9] - Сентябрь;\t[12] - Декабрь;\n\n";

do

{

cout << "Ввод: ";

cin >> choise;

if (cin.fail())

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

cout << "\nОшибка ввода ! Данное поле поддерживает только целочисленные значения. Пожалуйста, попробуйте еще раз.\n\n";

continue;

}

switch (choise)

{

case 1:

box.mounth = 1;

break;

case 2:

box.mounth = 2;

break;

case 3:

box.mounth = 3;

break;

case 4:

box.mounth = 4;

break;

case 5:

box.mounth = 5;

break;

case 6:

box.mounth = 6;

break;

case 7:

box.mounth = 7;

break;

case 8:

box.mounth = 8;

break;

case 9:

box.mounth = 9;

break;

case 10:

box.mounth = 10;

break;

case 11:

box.mounth = 11;

break;

case 12:

box.mounth = 12;

break;

default:

cout << "\nОшибка ! Пожалуйста, введите корректное чмсло.\n\n";

continue;

}

break;

} while (true);

do

{

cout << "\nГод: ";

cin >> box.year;

if (cin.fail())

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

cout << "\nОшибка ввода ! Данное поле поддерживает только целочисленные значения. Пожалуйста, попробуйте еще раз.\n";

continue;

}

else if (box.year<MIN\_YEARS || box.year>MAX\_YEARS)

{

cout << "\nОшибка ! Пожалуйста, введите корректное число.\n";

continue;

}

else

{

break;

}

} while (true);

return box;

}

string inputString()

{

string str;

str = inputWithUpRegister();

str = reBuildStringDesign(str);

return str;

}

string reBuildStringDesign(string fio)

{

fio.insert(fio.begin(), 1, '<');

for (int i = 0; i < fio.length(); i++)

{

if (fio.at(i) == ' ')

{

fio.erase(fio.begin() + i);

fio.insert(fio.begin() + i, 1, '\_');

}

}

fio.insert(fio.end(), 1, '>');

return fio;

}

bool IsFirstCharInUpperRegisterAndNotSpace(char symbol)

{

if ((symbol >= 'A' && symbol <= 'Z') || (symbol >= 'А' && symbol <= 'Я'))

{

return 1;

}

else if ((symbol >= 'a' && symbol <= 'z') || (symbol >= 'а' && symbol <= 'я'))

{

cout << "\nОшибка ввода ! Данный пункт не может иметь имён с маленькой буквы. Пожалуйста, повторите попытку, используя заглавную букву.\n";

return 0;

}

else if (symbol == ' ')

{

cout << "\nОшибка ввода ! Кажется, вы ввели лишний пробел в самом начале. Пожалуйста, будьте внимательны.\n";

return 0;

}

}

string buildStringDate(date current\_date)

{

string str\_date;

string day = to\_string(current\_date.day);

string mounth = to\_string(current\_date.mounth);

string year = to\_string(current\_date.year);

if (day.length() == 1)

{

day.insert(day.begin(), 1, '0');

}

if (mounth.length() == 1)

{

mounth.insert(mounth.begin(), 1, '0');

}

return str\_date = day + "." + mounth + "." + year;

}

void viewData(vector<company\_information> data)

{

string birthday, begin\_work;

cout << "\n---------------------------------------Информация о сотрудниках предприятия--------------------------------------\n\n\n";

cout << "----------------------------------------------------------------------------------------------------------------\n";

printf("%3s%3s%22s%2s%5s%3s%15s%2s%20s%2s%17s%2s%14s\n", "№", "|", "ФИО сотрудника", "|", "Пол", "|", "Дата рождения", "|", "Название отдела", "|", "Должность", "|", "Дата нач раб");

cout << "----------------------------------------------------------------------------------------------------------------\n";

for (int i = 0; i < data.size(); i++)

{

birthday = buildStringDate(data.at(i).birthday);

begin\_work = buildStringDate(data.at(i).begin\_work\_date);

cout << setw(3) << i + 1 << setw(3) << "|" << setw(22) << data.at(i).fio << setw(2) << "|" << setw(4) << data.at(i).sex << setw(4) << "|" << setw(15) << birthday << setw(2) << "|" << setw(20) << data.at(i).department << setw(2) << "|" << setw(17) << data.at(i).occupation << setw(2) << "|" << setw(13) << begin\_work << endl;

cout << "----------------------------------------------------------------------------------------------------------------\n";

}

cout << endl << endl;

}

void redactData(vector<company\_information>& data)

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int number, index, choise;

bool one\_more\_time;

do

{

viewData(data);

cout << "Введите номер записи, которую хотите отредактировать: ";

cin >> number;

if (!isBufferWithNoChar())

{

continue;

}

else if (number < 1 || number > data.size())

{

system("cls");

cout << "Ошибка ! Введите корректный номер записи.\n\n";

system("pause");

system("cls");

continue;

}

index = number - 1; // Считаем индекс текущего элемента. Помните, что нумерация элементов массива начинается с "0"

cout << "\nВыберите:\n\n[1] - Фамилия и инициалы сотрудника;\n[2] - Пол сотрудника;\n[3] - Дата рождения сотрудника;\n[4] - Название отдела сотрудника;\n[5] - Должность сотрудника;\n[6] - Дата начала работы;\n\n";

one\_more\_time = 0;

cout << "Ввод: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

cout << "\nФамилия и инициалы сотрудника:\n\n";

cout << "Пример: Иванов И. И. , Федоров И. И. , Смирнов А. А.\n";

cout << "\nВвод: ";

data.at(index).fio = inputString();

break;

case 2:

cout << "\nВыберите корректный пол сотрудника:\n";

data.at(index).sex = inputSex();

break;

case 3:

cout << "\nДата рождения сотрудника:\n\n";

data.at(index).birthday = inputDate();

break;

case 4:

cout << "\nНазвание отдела сотрудника: ";

data.at(index).department = inputString();

break;

case 5:

cout << "\nДолжность сотрудника: ";

data.at(index).occupation = inputString();

break;

case 6:

cout << "\nДата начала работы:\n\n";

data.at(index).begin\_work\_date = inputDate();

break;

default:

one\_more\_time = 1;

choiseError();

}

} while (one\_more\_time == 1);

cout << endl << endl;

viewData(data);

if (isWorkContinue())

{

redactData(data);

}

SetConsoleOutputCP(866);

SetConsoleCP(866);

}

void deleteData(vector<company\_information>& data)

{

int number\_zap, index;

do

{

viewData(data);

cout << "\nВыберите номер записи, данные которой хотите удалить:\n";

cout << "\nВвод: ";

cin >> number\_zap;

if (!isBufferWithNoChar())

{

continue;

}

index = number\_zap - 1;

if (number\_zap < 0 || number\_zap > data.size())

{

system("cls");

cout << "Ошибка удаления - такой записи не существует.\n\n";

system("pause");

system("cls");

continue;

}

else

{

data.erase(data.begin() + index);

}

if (data.empty())

{

system("cls");

cout << "Дальнейшее удаление невозможно, информация для вывода отсутствует. Пожалуйста, введите данные для корректной работы программы.\n\n";

system("pause");

system("cls");

break;

}

cout << "\nПосле удаления информация о проектах имеет вид:\n\n";

viewData(data);

if (!isWorkContinue())

{

break;

}

} while (true);

}

bool isWorkContinue()

{

bool work\_continue = 0;

int choise;

cout << "\nХотите продолжить ?\n\n";

cout << "[1] - Да;\n[2] - Нет;\n\n";

do

{

cout << "Ввод: ";

cin >> choise;

if (cin.fail())

{

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

cout << "\nОшибка ввода ! Данное поле поддерживает только целочисленные значения. Пожалуйста, попробуйте еще раз.\n\n";

continue;

}

if (choise == 1)

{

system("cls");

work\_continue = 1;

break;

}

else if (choise == 2)

{

work\_continue = 0;

break;

}

else

{

cout << "\nОшибка ввода ! Пожалуйста введите корректные данные для дальнейшей работы программы.\n\n";

}

} while (true);

return work\_continue;

}

void processingModeAdmin(vector<company\_information> data)

{

system("cls");

int choise;

while (true)

{

cout << "Вы находитесь в режиме обработки данных" << endl << endl;

cout << "Выберите необходимую опцию:" << endl << endl;

cout << "[1] - Выполнить индивидуальное задание;" << endl;

cout << "[2] - Поиск информации о сотрудниках;" << endl;

cout << "[3] - Сортировка информации о сотрудниках;" << endl;

cout << "[4] - Назад; " << endl << endl;

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

if (data.empty() && (choise == 1 || choise == 2 || choise == 3))

{

system("cls");

cout << "Невозможно выполнить операцию, так как отсутствует информация для обработки. Пожалуйста, введите данные в программу для дальнейшей корректной работы.\n\n";

system("pause");

system("cls");

continue;

}

switch (choise)

{

case 1:

system("cls");

individualTask(data);

system("pause");

system("cls");

break;

case 2:

system("cls");

searchMenu(data);

system("cls");

break;

case 3:

system("cls");

sortMenu(data);

system("cls");

break;

case 4:

system("cls");

break;

default:

choiseError();

}

if (choise == 4)

{

break;

}

}

}

void individualTask(vector<company\_information> data)

{

//вывести список сотрудников пенсионного возраста. Вывести список сотрудников в порядке убывания стажа.

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

showListOfSeniors(data);

cout << "----------------------------------------------------------------------------------------------------------------\n";

showListOfWorkersForExperiances(data);

SetConsoleOutputCP(866);

SetConsoleCP(866);

}

void showListOfSeniors(vector<company\_information> data)

{

char men = 'М', woman = 'Ж';

vector<company\_information> seniors;

int number\_seniors = 0;

for (int i = 0; i < data.size(); i++)

{

if (data.at(i).sex == men)

{

if (SENIOR\_MAN\_AGE <= CURRENT\_YEAR - data.at(i).birthday.year)

{

seniors.push\_back(data.at(i));

number\_seniors++;

}

}

else

{

if (SENIOR\_WOMAN\_AGE <= CURRENT\_YEAR - data.at(i).birthday.year)

{

seniors.push\_back(data.at(i));

number\_seniors++;

}

}

}

if (number\_seniors == 0)

{

cout << "Увы, но среди сотрудников нет ни одного человека, достигшего пенсионного возраста.\n\n";

}

else

{

cout << "Список сотрудников пенсионного возраста:\n\n";

viewData(seniors);

}

}

void showListOfWorkersForExperiances(vector<company\_information> data)

{

cout << "\nСписок сотрудников в порядке убывания стажа:\n";

int day, mounth, year, current\_sum, experiance\_1, experiance\_2, old\_mounth\_1, old\_mounth\_2;

struct tm newtime;

time\_t long\_time;

errno\_t err;

time(&long\_time);

// Convert to local time.

err = \_localtime64\_s(&newtime, &long\_time);

day = newtime.tm\_mday; // 1 -31

mounth = newtime.tm\_mon; // 0 - 11

year = newtime.tm\_year + 1900; // since 1900

for (int i = 0; i < NUMBER\_MOUNTH; i++)

{

if (i == mounth)

{

mounth = NUMBER\_DAYS\_IN\_MONTH[i];

break;

}

}

current\_sum = day + mounth + year \* NUMBER\_DAYS\_IN\_YEAR;

company\_information box;

for (int i = 0; i < data.size() - 1; i++)

{

for (int j = i + 1; j < data.size(); j++)

{

for (int m = 0; m < NUMBER\_MOUNTH; m++)

{

if (m == data.at(i).begin\_work\_date.mounth - 1)

{

old\_mounth\_1 = data.at(i).begin\_work\_date.mounth;

data.at(i).begin\_work\_date.mounth = NUMBER\_DAYS\_IN\_MONTH[m];

}

if (m == data.at(j).begin\_work\_date.mounth - 1)

{

old\_mounth\_2 = data.at(j).begin\_work\_date.mounth;

data.at(j).begin\_work\_date.mounth = NUMBER\_DAYS\_IN\_MONTH[m];

}

}

experiance\_1 = current\_sum - (data.at(i).begin\_work\_date.day + data.at(i).begin\_work\_date.mounth + data.at(i).begin\_work\_date.year \* NUMBER\_DAYS\_IN\_YEAR);

experiance\_2 = current\_sum - (data.at(j).begin\_work\_date.day + data.at(j).begin\_work\_date.mounth + data.at(j).begin\_work\_date.year \* NUMBER\_DAYS\_IN\_YEAR);

if (experiance\_1 < experiance\_2)

{

data.at(i).begin\_work\_date.mounth = old\_mounth\_1;

data.at(j).begin\_work\_date.mounth = old\_mounth\_2;

box = data.at(i);

data.at(i) = data.at(j);

data.at(j) = box;

}

else

{

data.at(i).begin\_work\_date.mounth = old\_mounth\_1;

data.at(j).begin\_work\_date.mounth = old\_mounth\_2;

}

}

}

viewData(data);

}

**searchMenu.cpp**

#include "searchMenu.h"

void searchMenu(vector<company\_information> data)

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

system("cls");

int choise;

while (true)

{

cout << "Вы находитесь в меню поиска. Пожалуйста, выберите необходимую вам операцию.\n\n";

cout << "[1] - Поиск по ФИО сотрудника;\n";

cout << "[2] - Поиск по полу сотрудника;\n";

cout << "[3] - Поиск по дате рождения сотрудника;\n";

cout << "[4] - Поиск по названию отдела сотрудника;\n";

cout << "[5] - Поиск по должности сотрудника;\n";

cout << "[6] - Поиск по дате начала работы сотрудника;\n";

cout << "[7] - Назад;\n\n";

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

system("cls");

searchForFIO(data);

system("cls");

break;

case 2:

system("cls");

searchForSex(data);

system("cls");

break;

case 3:

system("cls");

searchForBirthdayDate(data);

system("cls");

break;

case 4:

system("cls");

searchForDepartment(data);

system("cls");

break;

case 5:

system("cls");

searchForOccupation(data);

system("cls");

break;

case 6:

system("cls");

searchForBeginWorkDate(data);

system("cls");

break;

case 7:

system("cls");

break;

default:

choiseError();

}

if (choise == 7)

{

break;

}

}

SetConsoleOutputCP(866);

SetConsoleCP(866);

}

void searchForFIO(vector<company\_information> data)

{

vector<company\_information> copy\_data(data.size());

copy\_data = data;

string fio;

do

{

cout << "Информация для поиска:\n\n";

cout << "Введите фамилию и инициалы сотрудника.\n\n";

cout << "Пример: Сидоров М. С. , Иванов А. А. , Петоров Г. В.\n\n";

cout << "Ввод: ";

fio = inputString();

for (int i = 0; i < data.size(); i++)

{

if (data.at(i).fio != fio)

{

data.erase(data.begin() + i);

i--;

}

}

if (data.empty() == 1)

{

cout << "\nУвы, но среди данных отсутствует проект с таким названием.\n";

cout << "Пожалуйста, проверьте введенную вами информацию на наличие ошибки. Возможно, именно это мешает корректной работе поиска.\n\n";

system("pause");

break;

}

else

{

viewData(data);

}

data.clear();

data.resize(copy\_data.size());

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForSex(vector<company\_information> data)

{

vector<company\_information> copy\_data(data.size());

copy\_data = data;

char sex;

do

{

cout << "Информация для поиска:\n\n";

cout << "Выберите корректный пол сотрудника:\n";

sex = inputSex();

for (int i = 0; i < data.size(); i++)

{

if (data.at(i).sex != sex)

{

data.erase(data.begin() + i);

i--;

}

}

if (data.empty() == 1)

{

cout << "\nУвы, но среди данных отсутствует проект с таким названием.\n";

cout << "Пожалуйста, проверьте введенную вами информацию на наличие ошибки. Возможно, именно это мешает корректной работе поиска.\n\n";

system("pause");

break;

}

else

{

viewData(data);

}

data.clear();

data.resize(copy\_data.size());

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForBirthdayDate(vector<company\_information> data)

{

vector<company\_information> copy\_data(data.size());

copy\_data = data;

date birthday;

do

{

cout << "Информация для поиска:\n\n";

cout << "Введите дату рождения сотрудника:\n\n";

birthday = inputDate();

for (int i = 0; i < data.size(); i++)

{

if (data.at(i).birthday.day != birthday.day || data.at(i).birthday.mounth != birthday.mounth || data.at(i).birthday.year != birthday.year)

{

data.erase(data.begin() + i);

i--;

}

}

if (data.empty() == 1)

{

cout << "\nУвы, но данные о выбранном вами типе работ отсутствуют.\n";

cout << "Пожалуйста, проверьте введенную вами информацию на наличие ошибки. Возможно, именно это мешает корректной работе поиска.\n\n";

system("pause");

break;

}

else

{

viewData(data);

}

data.clear();

data.resize(copy\_data.size());

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForDepartment(vector<company\_information> data)

{

vector<company\_information> copy\_data(data.size());

copy\_data = data;

string department;

do

{

cout << "Информация для поиска:\n\n";

cout << "Введите название отдела, где работает сотрудник: ";

department = inputString();

for (int i = 0; i < data.size(); i++)

{

if (data.at(i).department != department)

{

data.erase(data.begin() + i);

i--;

}

}

if (data.empty() == 1)

{

cout << "\nУвы, но данные о выбранном вами сотруднике отсутствуют.\n";

cout << "Пожалуйста, проверьте введенную вами информацию на наличие ошибки. Возможно, именно это мешает корректной работе поиска.\n\n";

system("pause");

break;

}

else

{

viewData(data);

}

data.clear();

data.resize(copy\_data.size());

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForOccupation(vector<company\_information> data)

{

vector<company\_information> copy\_data(data.size());

copy\_data = data;

string occupation;

do

{

cout << "Информация для поиска:\n\n";

cout << "Введите должность сотрудника: ";

occupation = inputString();

for (int i = 0; i < data.size(); i++)

{

if (data.at(i).occupation != occupation)

{

data.erase(data.begin() + i);

i--;

}

}

if (data.empty() == 1)

{

cout << "\nУвы, но среди данных отсутствует информация о сотрудниках, с таким количеством часов работы.\n";

cout << "Пожалуйста, проверьте введенную вами информацию на наличие ошибки. Возможно, именно это мешает корректной работе поиска.\n\n";

system("pause");

break;

}

else

{

viewData(data);

}

data.clear();

data.resize(copy\_data.size());

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForBeginWorkDate(vector<company\_information> data)

{

vector<company\_information> copy\_data(data.size());

copy\_data = data;

date begin\_work;

do

{

cout << "Информация для поиска:\n\n";

cout << "Введите дату начала работы сотрудника:\n\n";

begin\_work = inputDate();

for (int i = 0; i < data.size(); i++)

{

if (data.at(i).begin\_work\_date.day != begin\_work.day || data.at(i).begin\_work\_date.mounth != begin\_work.mounth || data.at(i).begin\_work\_date.year != begin\_work.year)

{

data.erase(data.begin() + i);

i--;

}

}

if (data.empty() == 1)

{

cout << "\nУвы, но среди данных отсутствует информация о сотрудниках, с такой стоимостью одного часа работы.\n";

cout << "Пожалуйста, проверьте введенную вами информацию на наличие ошибки. Возможно, именно это мешает корректной работе поиска.\n\n";

system("pause");

break;

}

else

{

viewData(data);

}

data.clear();

data.resize(copy\_data.size());

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

**sortMenu.cpp**

#include "sortMenu.h"

void sortMenu(vector<company\_information>data)

{

system("cls");

int choise;

while (true)

{

cout << "Вы находитесь в меню сортировки. Пожалуйста, выберите необходимую вам операцию.\n\n";

cout << "[1] - Сортировка по ФИО сотрудника;\n";

cout << "[1] - Сортировка по полу сотрудника;\n";

cout << "[3] - Сортировка по названию отдела сотрудника;\n";

cout << "[4] - Сортировка по должности сотрудника;\n";

cout << "[5] - Назад;\n\n";

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

system("cls");

sortForFIO(data);

system("pause");

system("cls");

break;

case 2:

system("cls");

sortForSex(data);

system("pause");

system("cls");

break;

case 3:

system("cls");

sortForDepartment(data);

system("pause");

system("cls");

break;

case 4:

system("cls");

sortForOccupation(data);

system("pause");

system("cls");

break;

case 5:

system("cls");

break;

default:

choiseError();

}

if (choise == 5)

{

break;

}

}

}

void sortForFIO(vector<company\_information> data)

{

sort(data.begin(), data.end(), [](company\_information el\_1, company\_information el\_2)

{

return el\_1.fio < el\_2.fio;

});

viewData(data);

}

void sortForSex(vector<company\_information> data)

{

sort(data.begin(), data.end(), [](company\_information el\_1, company\_information el\_2)

{

return el\_1.sex < el\_2.sex;

});

viewData(data);

}

void sortForDepartment(vector<company\_information> data)

{

sort(data.begin(), data.end(), [](company\_information el\_1, company\_information el\_2)

{

return el\_1.department < el\_2.department;

});

viewData(data);

}

void sortForOccupation(vector<company\_information> data)

{

sort(data.begin(), data.end(), [](company\_information el\_1, company\_information el\_2)

{

return el\_1.occupation < el\_2.occupation;

});

viewData(data);

}

**userMenu.cpp**

#include "userMenu.h"

void userMenu(vector<company\_information> data)

{

system("cls");

int kod;

while (true)

{

cout << "Добро пожаловать, дорогой пользователь!\n\n";

cout << "Выберите необходимую опцию:\n\n";

cout << "[1] - Просмотреть данные;\n";

cout << "[2] - Выполнить индивидуальное задание;\n";

cout << "[3] - Поиск информации о сотрудниках;\n";

cout << "[4] - Сортировка информации о сотрудниках\n";

cout << "[5] - Возврат к авторизации / Выход из программы;\n\n";

cout << "Ваш выбор: ";

cin >> kod;

if (!isBufferWithNoChar())

{

continue;

}

if (data.empty() && (kod == 1 || kod == 2 || kod == 3 || kod == 4))

{

system("cls");

cout << "Невозможно выполнить операцию, так как отсутствует информация для обработки. Пожалуйста, введите данные в программу для дальнейшей корректной работы.\n\n";

system("pause");

system("cls");

continue;

}

switch (kod)

{

case 1:

system("cls");

viewData(data);

system("pause");

system("cls");

break;

case 2:

system("cls");

individualTask(data);

system("pause");

system("cls");

break;

case 3:

system("cls");

searchMenu(data);

system("cls");

break;

case 4:

system("cls");

sortMenu(data);

system("cls");

break;

case 5:

system("cls");

break;

default:

choiseError();

}

if (kod == 5)

{

break;

}

}

}

**hashFunction.cpp**

#include "hashFunction.h"

// Сгенерировать соль

string generateSalt()

{

string salt;

int kod, length\_salt = 8;

char symbol;

for (int i = 0; i < length\_salt; i++)

{

kod = rand() % 95 + 33;

symbol = (char)kod;

salt += symbol;

}

return salt;

}

// Смешать с солью

string blendWithSalt(string passowrd, string salt)

{

string rez\_pass;

int m = 0, n = 0, i = 0;

while (i < passowrd.length())

{

rez\_pass += " ";

rez\_pass[i] = passowrd[m];

i++;

m++;

if (i % 2 == 0)

{

rez\_pass += " ";

rez\_pass[i] = salt[n];

i++;

n++;

}

}

return rez\_pass;

}

// Заполнение хеш-таблицы

void fillHashTable(vector <encrypt\_element>& hashtable)

{

encrypt\_element el;

char start\_pos\_in\_ascii = 33, end\_pos\_in\_ascii = 127;

for (int i = 0; i != end\_pos\_in\_ascii; i++)

{

el.info = start\_pos\_in\_ascii;

el.key = el.info \* el.info % 94 + 44;

start\_pos\_in\_ascii++;

hashtable.push\_back(el);

}

}

// Найти в таблице и сопоставить с символом

char findInTable(char x, vector <encrypt\_element> hashtable)

{

for (int i = 0; i < 127 - 33; i++)

{

if (x == hashtable[i].info)

x = hashtable[i].key;

else continue;

}

return x;

}

// Зашифровать текущий пароль

string encryptCurrentPass(string passowrd, vector <encrypt\_element> hashtable, string salt)

{

passowrd = blendWithSalt(passowrd, salt); // Смешивание пароля с "солью"

for (int i = 0; i < passowrd.length(); i++) // Шифрование каждого символа пароля в соответствии с хеш-таблицей

passowrd[i] = findInTable(passowrd[i], hashtable);

passowrd = blendWithSalt(passowrd, salt); // Смешивание пароля с "солью"

return passowrd;

}