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 планирования бюджета проекта**

Выполнил: студент группы 914303, Дворник М.С.

Проверил: Львов П.С.

Минск 2020

СОДЕРЖАНИЕ

[1 ТРЕБОВАНИЯ К ПРОГРАММЕ 5](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906304)

[1.1 Исходные данные к курсовой работе 5](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906305)

[1.2 Функциональные требования к курсовой работе 5](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906306)

[1.3 Требования к программной реализации курсовой работы 6](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906307)

[2 КОНСТРУИРОВАНИЕ ПРОГРАММЫ 8](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906308)

[2.1 Разработка структуры программы 8](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906309)

[2.2 Разработка способа организации данных 9](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906310)

[2.3 Разработка перечня пользовательских функций программы 10](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906311)

[3 РАЗРАБОТКА АЛГОРИТМОВ РАБОТЫ ПРОГРАММЫ 16](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906312)

[3.1 Алгоритм функции int main() 16](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906313)

[3.2 Алгоритм функции void workWithAccountsAdmin](file:///C:\\Users\\Shegol\\Desktop\\Zapiska%20(1).docx" \l "_Toc513906314)

[(vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable)](file:///C:\\Users\\Shegol\\Desktop\\Zapiska%20(1).docx" \l "_Toc513906314)

[17](file:///C:\\Users\\Shegol\\Desktop\\Zapiska%20(1).docx" \l "_Toc513906314)

[3.3 Алгоритм функции void verifiedAccounts (vector <Autorization>&) 18](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906315)

[4 ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ 19](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906316)

[ПРИЛОЖЕНИЕ А: ЛИСТИНГ КОДА С КОММЕНТАРИЯМИ 25](file:///C:\Users\Shegol\Desktop\Zapiska%20(1).docx#_Toc513906317)

**1 ТРЕБОВАНИЯ К ПРОГРАММЕ**

В компании по разработке программного обеспечения для планирования

работ по проекту необходимо обработать информацию следующего вида:

наименование проекта, вид работ (работа над требованиями, разработка

архитектуры, реализация, тестирование), Ф.И.О. сотрудника, предполагаемое

количество часов, стоимость одного часа.

Индивидуальное задание: для каждого проекта вывести его итоговую

стоимость, перечень видов работ с указанием общего количества задействованных

специалистов и стоимости данного этапа.

Общее для всех вариантов задание: реализовать авторизацию для входа в

систему, функционал администратора и функционал пользователя (см. более

подробно в функциональных требованиях к курсовой работе в ОКП Курсовое

проектирование. Пособие).

* 1. **Исходные данные к курсовой работе**

Тема курсовой работы: разработка программы учёта сведений о пациентах медицинского центра.

Язык программирования С++.

Среда разработки Microsoft: Visual Studio 2019.

Вид приложения – консольное.

Парадигма программирования – процедурная.

Способ организации данных – структуры (struct).

Способ хранения данных – файлы.

Каждая логически завершенная задача программы должна быть реализована в виде функции.

* 1. **Функциональные требования к курсовой работе**

Первым этапом работы программы является авторизация – предоставление прав доступа. В рамках данного этапа необходимо считать данные из файла с учетными записями пользователей следующего вида:

* login;
* password;
* role (данное поле служит для разделения в правах администраторов и пользователей).

После ввода пользователем своих персональных данных (логина и пароля) и сверки со считанной из файла информацией необходимо предусмотреть возможность входа в качестве администратора (в этом role = 1) или в качестве пользователя (в этом случае role = 0).

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

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

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

1. Управление учетными записями пользователей:

* просмотр всех учетных записей;
* добавление новой учетной записи;
* редактирование учетной записи;
* удаление учетной записи.

1. Работа с файлом данных:

* создание файла;
* открытие существующего файла;
* удаление файла.

1. Работа с данными:
   * + просмотр всех данных;
     + добавление новой записи;
     + удаление записи;
     + редактирование записи;

* выполнение индивидуального задания;
* поиск данных (как минимум три вида);
* сортировка (как минимум три вида).

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

* + просмотр всех данных;
  + выполнение индивидуального задания;
  + поиск данных (как минимум три вида);
  + сортировка (как минимум три вида).

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

Предусмотреть:

* обработку исключительных ситуаций (например, файл с данными не создан, ничего не найдено по результатам поиска, введенные данные не соответствуют формату поля, номер удаляемой записи выходит за пределы вектора, логин новой учетной записи уже существует и т.д.);
* возможность возврата назад (навигация);
* запрос на подтверждение удаления (изменения) вида «Вы действительно хотите удалить файл?»;
* вывод сообщения об успешности создания файла/удаления записи/ и т.д.

**1.3 Требования к программной реализации курсовой работы**

Все переменные и константы должны иметь [осмысленные](http://ci-plus-plus-snachala.ru/?page_id=228) имена в рамках тематики варианта к курсовой работе.

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).

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.

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

Код не должен дублироваться – для этого существуют функции!

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.

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

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**

**2.1 Разработка структуры программы**

Разработана структура программы, представленная на рисунке 2.1.

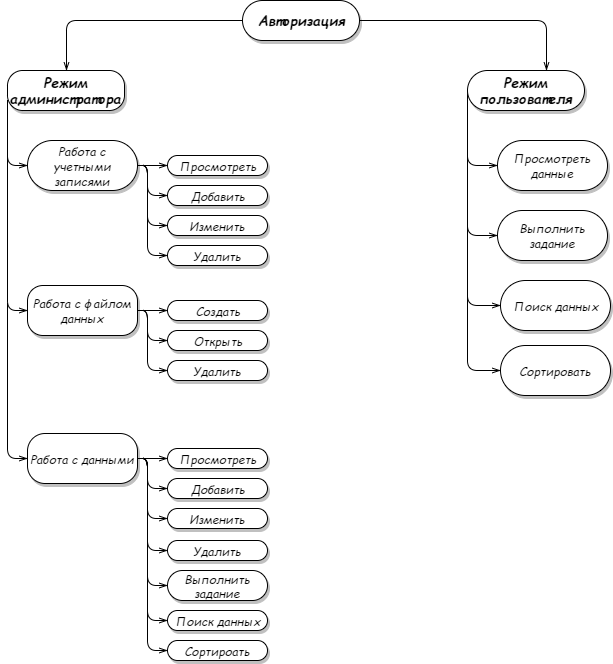


Рисунок 2.1 – Структура программы

* 1. **Разработка способа организации данных**

Для описания входных данных использовались файлы. Каждый файл отвечал за отдельную функцию. В программе задекларированы следующие файлы:

#pragma once //Функция мэйн;

#include "startInfo.h" //Работа с данными;

#include "autorisation.h" //Авторизация;

#include "adminMenu.h" //Меню админа;

#include "workWithAccountsAdmin.h" //Работа аккаунтом админа;

#include "workWithFilesAdmin.h" //Работа с файлами;

#include "workWithDataAdminMenu.h" //Работа с данными админа;

#include "redactingModeAdmin.h" //Редактирование админа;

#include "processingModeAdmin.h" //Обработка данных админа;

#include "searchMenu.h"" //Поиск в меню;

#include "sortMenu.h" //Сортировка меню;

#include "userMenu.h" //Меню юзера;

#include "hashFunction.h" //Нэш функция;

* 1. **Разработка перечня пользовательских функций программы**

Таблица 1 – Перечень функций программы

|  |  |  |  |
| --- | --- | --- | --- |
| Тип возвращаемого значения | Название  функции | Передаваемые значения | Описание  функции |
| void | adminMenu | vector <autorisation>& accounts\_data, vector<product\_info>& data, vector <encrypt\_element> hashtable | Работа с меню |
| void | makeAutorisation | vector <product\_info>& data, vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable | Авторизация |
| string | inputStarsPassword | - | Ввод пароля |
| bool | isYouNotvarifiedUser | vector <autorisation> account\_data, string login, string password, vector <encrypt\_element> hashtable | Проверка заргестрированности пользователя |
| void | addFirstAdmin | string path, string first\_admin\_login, string first\_admin\_passowrd, vector <encrypt\_element> hashtable | Добавление главного админа |
| bool | isFileExist | string path | Проверка на наличие файла |
| void | copyInVector | (vector<product\_info>& data, string file\_data, vector <autorisation>& accounts\_data, string file\_accounts) | Копирование в вектор |
| void | mainMenu | (vector <autorisation>& accounts\_data, vector <product\_info>& data, vector <encrypt\_element>& hashtable, string first\_admin\_login, string first\_admin\_passowrd, string file\_accounts, string file\_data) | Главное меню |
| void | fromFileToVectorACCOUNTS | (vector <autorisation>& accounts\_data, string file\_accounts) | Перевод из файла в вектор( аккаунтов) |
| void | fromFileToVectorDATA | (vector<product\_info>& data, string file\_data) | Перевод из файла в вектор(данных) |
| void | pushDataInFile(vector <product\_info> data, string path\_data, vector <autorisation> accounts\_data, string path\_accounts) | (vector <product\_info> data, string path\_data, vector <autorisation> accounts\_data, string path\_accounts) | Добавление данных в файлы |
| void | fromVectorToFileACCOUNTS | (vector <autorisation> accounts\_data, string path\_accounts) | Перевод из вектора в файл(аккаунта) |
| void | fromVectorToFileDATA | (vector <product\_info> data, string path\_data)  { | Перевод из вектора в файл(данные) |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Тип возвращаемого значения | Название  Функции | Передаваемые значения | Описание  функции |
| string | generateSalt | - | Создание соли |
| string | string blendWithSalt | (string passowrd, string salt) | Смешивание с солью |
| void | fillHashTable | (vector <encrypt\_element>& hashtable) | Заполнение хэш-таблицы |
| char | findInTable | (char x, vector <encrypt\_element> hashtable) | Нахождение в таблице и сопоставление |
| string | encryptCurrentPass | (string passowrd, vector <encrypt\_element> hashtable, string salt) | Зашифровать пароль |
| void | workWithAccountsAdmin | (vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable) | Работа с аккаунтом админа |
| void | verifiedAccounts | (vector <autorisation>& accounts\_data) | Зарегистрированные аккаунты |
| string | inputLogin | - | Ввод логина |
| bool | isAllSymbolsCorrect | (string str) | Проверка на верность введенных символов |
| bool | isTheSameLoginInVector | (vector <autorisation> accounts\_data, string login, int current\_index) | Проверка на векторность |
| void | viewAccounts | (vector <autorisation> accounts\_data) | Просмотри аккаунтов |
| void | addNewAccount | (vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable) | Добавление аккаунта |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Тип возвращаемого значения | Название  Функции | Передаваемые значения | Описание  функции |
| autorisation | inputInfoAboutNewAccount | (vector <encrypt\_element> hashtable, vector <autorisation> accounts\_data) | Ввод информации о новом аккаунте |
| void | addAdministrator(vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable) | (vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable) | Добавление администратора |
| void | addUser | (vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable) | Дабавление пользователя |
| void | redactAccount | (vector <autorisation>& accounts\_data, vector <encrypt\_element> hashtable) | Редактировать аккаунты |
| void | deleteAccount | (vector <autorisation>& account\_data, bool& block\_delete, char\* warning\_block\_delete) | Удаление аккаунта |
| void | workWithDataAdmin | (vector<product\_info>& data) | Обработка информации админа |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Тип возвращаемого значения | Название  функции | Передаваемые значения | Описание  функции |
| void | redactingModeAdmin | (vector<product\_info>& data) | Редактирование информации админа |
| void | addProjectData | (vector<product\_info>& data) | Добавление информации о проекте |
| string | inputNameOfArticle | - | Ввод информации |
| string | inputWithUpRegister | - | Повторный ввод |
| string | reBuildStringDesign(string stroka) | (string stroka) | Ввод информации |
| bool | IsFirstCharInUpperRegisterAndNoSpace(char symbol) | (char symbol) | Проверка |
| void | viewData(vector<product\_info> data) | (vector<product\_info> data) | Просмотри информации |
| void | redactData | (vector<product\_info>& data) | Редактирование информации |
| int | protectRedactingNumber | (int vector\_size) | Защита |
| void | deleteData | (vector<product\_info>& data) | Удаление данных |
| void | processingModeAdmin | (vector<product\_info> data) | Обработка данных |
| void | individualTask | (vector<product\_info> data) | Индивидуальное задание |
| void | searchMenu | (vector<product\_info> data) | Меню поиска |
| void | searchForProjectName | (vector<product\_info> data) | Поиск по названию проекта |
| void | SearchForTypeOfWork | (vector<product\_info> data) | Поиск по типу работы |
| void | searchForFIO | (vector<product\_info> data) | Поиск по ФИО |
| void | searchForHours | (vector<product\_info> data) | Поиск по часам работы |
| void | searchForCost | (vector<product\_info> data) | Поиск по цене |
| void | searchForNumberPair | (vector<product\_info> data) | Поиск по количеству пар |
| void | sortMenu | (vector<product\_info> data) | Меню сортировки |
| void | sortForProjectName | (vector<product\_info> data) | Сортировка по имени проект |
| void | sortForTypeOfWork | (vector<product\_info> data) | Сортировка по типу работы |
| void | sortForFIO | (vector<product\_info> data) | Сортировка по ФИО |
| void | sortForHours | (vector<product\_info> data) | Сортировать по часам |
| void | sortForCost | (vector<product\_info> data) | Сортировать по цене |

**3 РАЗРАБОТКА АЛГОРИТМОВ РАБОТЫ ПРОГРАММЫ**

**3.1 Алгоритм функции int main()**

Разработан алгоритм функции int main(), представленный на рисунке 3.1.

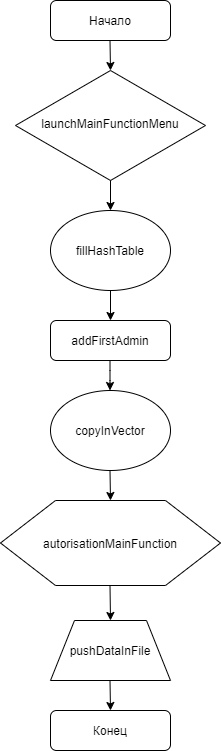


Рисунок 3.1 – Алгоритм функции int main()

**3.2 Алгоритм функции void workWithAccountsAdmin**

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

Разработан алгоритм функции void workWithAccountsAdmin

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

представленный на рисунке 3.2.

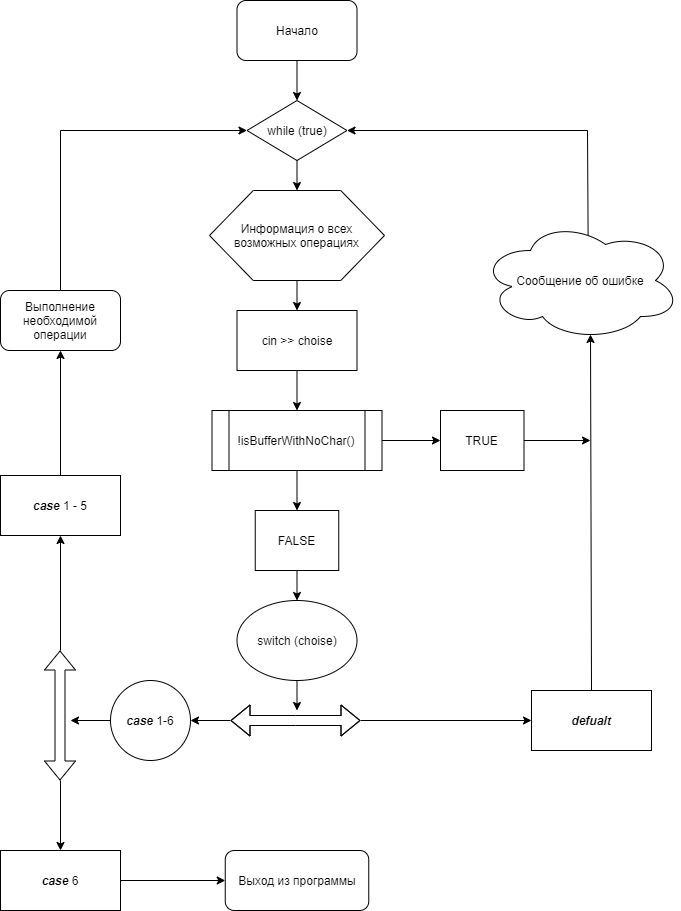


Рисунок 3.2 - Алгоритм функции void workWithAccountsAdmin

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

**3.3 Алгоритм функции void fromFileToVectorACCOUNTS**

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

Разработан алгоритм функции void fromFileToVectorACCOUNTS(vector <autorisation>& accounts\_data, string file\_accounts), представленный на рисунке 3.3.

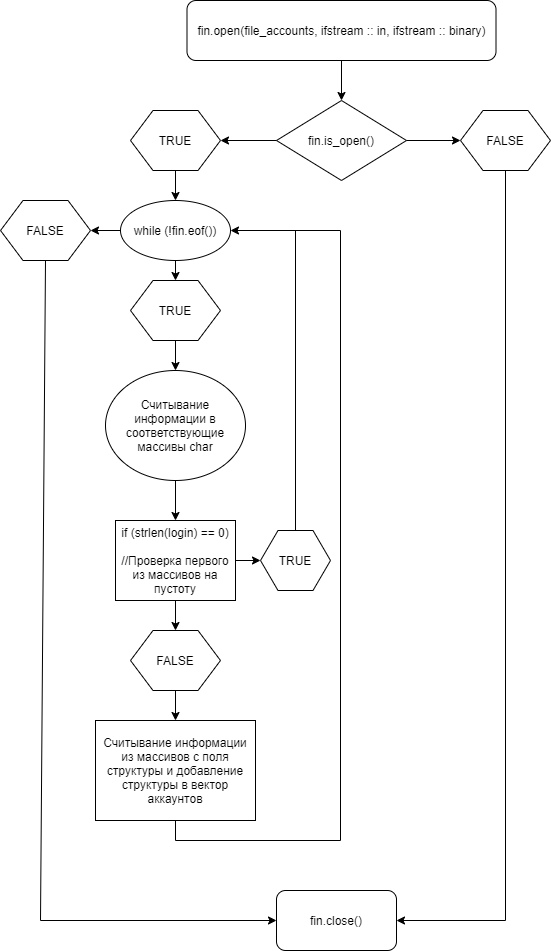


Рисунок 3.3 – Алгоритм функции void fromFileToVectorACCOUNTS

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

1. **ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ**

При запуске программы нужно выбрать учетную запись. При авторизации программа запрашивает пароль и логин, в случае неправильного ввода одного из полей снова предлагается ввести данные пользователя. Если учетных записей не существует, программа считает вошедшего пользователя администратором и добавляет соответствующую учетную запись (рисунок 4.1).

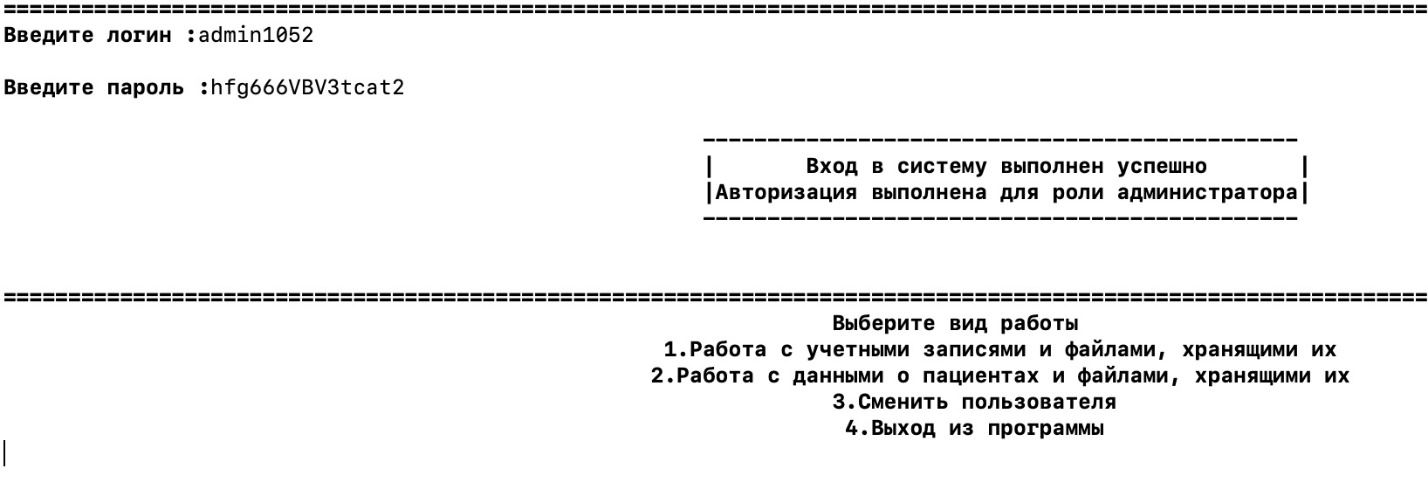


Рисунок 4.1 – Вход администратора в программу

Меню администратора включает 4 пункта: Работа с учетными записями и файлами, хранящими их, работа с данными о пациентах и файлами, хранящими их, смена пользователя и выход из программы.

Итак, меню работы с учетными записями пользователей (рисунок 4.2). В подпункте просмотр учётных записей выводятся все учетные записи. В подпункте добавление учётной записи последовательно вводятся данные полей новой учетной записи. В подпункте редактирование учётных записей запрашивается номер учетной записи, которую мы хотим изменить, а затем выбор изменяемого поля. В подпункте удаление учётных записей запрашивается номер учетной записи, которую мы хотим удалить. В случае некорректного ввода номера учетной записи в подпунктах редактировать и удалить (отрицательное число или число, большее количества учетных записей) появляется сообщение об ошибке и предлагается повторить ввод. При успешном добавлении, удалении и редактировании выводится соответствующее сообщение. Подпункт, возврат в меню, возвращает к меню администратора. В подпункте выход из программы осуществляется выход и завершение программы, с сохранением всех изменений.



Рисунок 4.2 - Меню работы с учетными записями пользователей и файлами, хранящими их

Меню работы с данными и файлами, хранящими их включает в себя множество возможностей, которые представлены на рисунке 4.3.



Рисунок 4.3 - Меню работы с данными и файлами, хранящими их

При переходе в первый подпункт: работа с файлами (рисунок 4.4).



Рисунок 4.4 – Работа с файлами



Рисунок 4.6 – Обработка информации

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг кода с комментариями

//25. Разработка программы планирования бюджета проекта

//В компании по разработке программного обеспечения для планирования работ по проекту необходимо обработать информацию следующего вида :

//наименование проекта,

//вид работ (работа над требованиями, разработка архитектуры, реализация, тестирование),

//ФИО сотрудника,

//предполагаемое количество часов,

//стоимость одного часа.

//

//Индивидуальное задание : для каждого проекта вывести:

//его итоговую стоимость,

//перечень видов работ с указанием общего количества задействованных специалистов и стоимости данного этапа.

//

//Общее для всех вариантов задание : реализовать:

//авторизацию для входа в систему,

//функционал администратора и функционал пользователя(см.более подробно в функциональных требованиях к курсовой работе, подраздел 1.2).

#pragma once

#include "startInfo.h"

#include "autorisation.h"

#include "adminMenu.h"

#include "workWithAccountsAdmin.h"

#include "workWithFilesAdmin.h"

#include "workWithDataAdminMenu.h"

#include "redactingModeAdmin.h"

#include "processingModeAdmin.h"

#include "searchMenu.h"

#include "sortMenu.h"

#include "userMenu.h"

#include "hashFunction.h"

int main()

{

setlocale(0, "rus");

srand(time(0));

string first\_admin\_login = "Maks";

string first\_admin\_passowrd = "1111";

string file\_accounts = "accounts.txt";

string file\_data = "data.txt";

vector<autorisation> accounts\_data;

vector<project\_data> data;

vector <encrypt\_element> hashtable;

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

}

#include "adminMenu.h"

void adminMenu(vector <autorisation>& accounts\_data, vector<project\_data>& 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;

}

}

}

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

//{

// int kod;

// bool fail = 0;

// 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;

// do

// {

// cout << "Ввод: ";

// 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:

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

// continue;

// }

// if (kod == 3)

// {

// break;

// }

// else if (kod == 1 || kod == 2)

// {

// adminMenu(accounts\_data, data, hashtable);

// break;

// }

// } while (true);

//}

#include "autorisation.h"

void autorisationMainFunction(vector <project\_data>& 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:

// search from the list of not varified users

stopLoginningNotVarifiedUsers(vvod, accounts\_data, registration\_data, hashtable, unhashed\_pass);

if (vvod != -1)

{

if (vvod == 1)

{

break;

}

else

{

continue;

}

}

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 stopLoginningNotVarifiedUsers(int &vvod, vector <autorisation> accounts\_data, autorisation registration\_data, vector <encrypt\_element> hashtable, string unhashed\_pass)

{

if (isYouNotvarifiedUser(accounts\_data, registration\_data.login, registration\_data.passowrd, hashtable, unhashed\_pass))

{

system("cls");

do

{

cout << "Данная учетная запись находится на стадии верификации.\n\n";

cout << "Пожалуйста, дождитесь, когда администратор просмотрит ваш аккаунт.\n\n";

cout << "Вы можете выйти из программы или же продолжить работу под другой учетной записью.\n\n";

cout << "Ваш выбор:\n\n";

cout << "[1] - Выйти из программы;\n";

cout << "[2] - Выполнить вход в другую учетную запись;\n\n";

cout << "Ввод: ";

cin >> vvod;

if (!isBufferWithNoChar())

{

continue;

}

else if (vvod == 1)

{

system("cls");

cout << "Программа завершена.\n\n";

break;

}

else if (vvod == 2)

{

system("cls");

break;

}

else

{

choiseError();

continue;

}

} while (true);

}

else

{

vvod = -1; // this user isn't in the list of not verified users

}

}

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

{

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

{

case\_equal = 0;

if (registration\_data.login == accounts\_data.at(i).login && accounts\_data.at(i).access == 1)

{

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 << "Есть два пути, ведущие к такой ситуации:\n\n";

cout << "[1] - Данные введены некорректно;\n";

cout << "[2] - Если вы уже пытались создать данную учетную запись, но запрос был отклонен администратором или, если такой учетной записи вовсе не существовало;\n\n";

cout << "В случае №1, пожалуйста, внимательно проверьте введённые данные, возможно вы допустили ошибку.\n";

cout << "Если же у вас ситуация под №2, создайте новую учетную запись.\n\n";

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

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

cout << "[2] - Отсутствие учётной записи;\n";

cout << "[3] - Не хочу ни с чем разбираться, просто выйду из программы;\n\n";

cout << "Ввод: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

else if (choise == 1)

{

errorDuringInputData();

break;

}

else if (choise == 2)

{

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

break;

}

else if (choise == 3)

{

break;

}

else

{

choiseError();

continue;

}

} while (true);

}

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

{

system("cls");

do

{

cout << "Сейчас перед вами стоит выбор: " << endl << endl;

cout << "[1] - Создать новую учётную запись, где логином и паролем будут являться введенные вами сейчас данные;" << endl;

cout << "[2] - Попробовать ввести данные ещё раз;" << endl << endl;

cout << "Введенная вами информация:\n\n";

cout << "Логин: " << registration\_data.login << endl;

cout << "Пароль: " << unhashed\_pass << endl << endl;

cout << "При выборе пункта 1, она станет вашим ключом к учетной записи.\n\n";

cout << "Ввод: ";

cin >> create\_account;

if (!isBufferWithNoChar())

{

continue;

}

else if (create\_account == 1)

{

pushAccountToVarifiedList(registration\_data, unhashed\_pass, hashtable, accounts\_data, continue\_work);

break;

}

else if (create\_account == 2)

{

system("cls");

errorDuringInputData();

break;

}

else

{

choiseError();

continue;

}

} while (true);

}

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

{

system("cls");

registration\_data.access = 0;

registration\_data.salt = generateSalt();

registration\_data.passowrd = encryptCurrentPass(unhashed\_pass, hashtable, registration\_data.salt);

accounts\_data.push\_back(registration\_data);

do

{

cout << "Отлично ! Ваш запрос отправлен администратору.\n\n";

cout << "Пожалуйста, ожидайте подтвержения вашей учётной записи.\n\n";

cout << "Хотите продолжить работу в качестве другой учётной записи ?\n\n";

cout << "[1] - Да, хочу продолжить работу в программе;\n";

cout << "[2] - Нет, выйду из программы и буду ждать подтвержедия этой учётной записи.\n\n";

cout << "Ввод: ";

cin >> continue\_work;

if (!isBufferWithNoChar())

{

continue;

}

else if (continue\_work == 1)

{

system("cls");

break;

}

else if (continue\_work == 2)

{

system("cls");

cout << "Программа успешно завершена. Пожалуйста, попробуйте выполнить вход в данную учётную запись позже.\n\n";

//cout << "Не волнуйтесь, мы помним о вас. Среднее время подтверждения учётной записи - 1 день." << endl << endl; // Количество дней зависит от IT компании

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;

}

bool isYouNotvarifiedUser(vector <autorisation> account\_data, string login, string password, vector <encrypt\_element> hashtable, string unhashed\_pass)

{

bool case\_not\_varified = 0;

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

{

if (account\_data.at(i).access == 0 && account\_data.at(i).login == login)

{

password = encryptCurrentPass(unhashed\_pass, hashtable, account\_data.at(i).salt);

if (password == account\_data.at(i).passowrd)

{

case\_not\_varified = 1;

break;

}

}

}

return case\_not\_varified;

}

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");

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

}

#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;

}

#include "processingModeAdmin.h"

void processingModeAdmin(vector<project\_data> 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 проекте;" << endl;

cout << "[3] - Сортировка информации о проектах;" << 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;

}

}

}

//functions from the admin processing mode:

void individualTask(vector<project\_data> data)

{

cout << "Здесь вы можете просмотреть ключевую статистику о каждом проекте." << endl << endl;

vector <project\_data> one\_project;

string str;

double stage\_cost, result\_cost;

int num\_workers;

do

{

str = data.begin()->project\_name;

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

{

if (data.at(i).project\_name == str)

{

one\_project.push\_back(data.at(i));

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

i--;

}

}

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

cout << "Название проекта: " << str << endl << endl;

result\_cost = 0;

for (int j = 0; j < one\_project.size(); j++)

{

result\_cost += one\_project.at(j).hours \* one\_project.at(j).cost;

}

cout << "Итоговая стоимость проекта: " << result\_cost << endl << endl;

num\_workers = 0;

stage\_cost = 0;

cout << "Информация о видах работ, реализуемых на проекте:\n\n";

while (one\_project.empty() != 1)

{

str = one\_project.begin()->type\_of\_work;

for (int m = 0; m < one\_project.size(); m++)

{

if (one\_project.at(m).type\_of\_work == str)

{

num\_workers++;

stage\_cost += one\_project.at(m).hours \* one\_project.at(m).cost;

one\_project.erase(one\_project.begin() + m);

m--;

}

}

cout << "Вид работ: " << str << endl;

cout << "\nОбщее количество задействованых специалистов: " << num\_workers << endl;

cout << "\nСтоимость данного этапа: " << stage\_cost << endl << endl;

num\_workers = 0;

stage\_cost = 0;

}

} while (data.empty() != 1);

}

#include "redactingModeAdmin.h"

void redactingModeAdmin(vector<project\_data>& 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 проекте;" << 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");

addProjectData(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;

}

string inputTypeOfWork()

{

string type\_of\_work;

int kod;

do

{

cout << "\nПожалуйста, выберите вид работ, реализуемых на проектах:\n\n";

cout << "[1] - <Работа\_над\_требованиями>\n";

cout << "[2] - <Разработка\_архитектуры>\n";

cout << "[3] - <Реализация>\n";

cout << "[4] - <Тестирование>\n\n";

cout << "Ваш выбор: ";

cin >> kod;

if (!isBufferWithNoChar())

{

continue;

}

switch (kod)

{

case 1:

type\_of\_work = "<Работа\_над\_требованиями>";

break;

case 2:

type\_of\_work = "<Разработка\_архитектуры>";

break;

case 3:

type\_of\_work = "<Реализация>";

break;

case 4:

type\_of\_work = "<Тестирование>";

break;

default:

system("cls");

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

system("pause");

system("cls");

}

if (kod == 1 || kod == 2 || kod == 3 || kod == 4)

{

break;

}

} while (true);

return type\_of\_work;

}

void addProjectData(vector<project\_data>& data)

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

project\_data zap;

do

{

cout << "Введите информацию о проекте:\n\n";

cout << "Введите наименование проекта: ";

zap.project\_name = inputString();

zap.type\_of\_work = inputTypeOfWork();

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

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

cout << "Ввод: ";

zap.fio = inputString();

cout << "\nПредполагаемое количество часов работы сотрудника: ";

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

cin >> zap.hours;

printf("\nСтоимость одного часа ($): ");

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

cin >> zap.cost;

data.push\_back(zap);

if (!isWorkContinue())

{

break;

}

} while (true);

SetConsoleOutputCP(866);

SetConsoleCP(866);

}

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;

}

}

void viewData(vector<project\_data> data)

{

cout << "\n-------------------------------------------Информация о всех проектах-------------------------------------------\n\n\n";

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

printf("%5s%3s%17s%2s%31s%2s%22s%2s%14s%2s%11s\n", "№","|", "Проект", "|", "Вид работ", "|", "ФИО сотрудника", "|", "Кол. ч.", "|", "1 ч.($)");

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

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

{

cout << setw(5) << i + 1 << setw(3) <<"|"<< setw(17) << data.at(i).project\_name << setw(2) <<"|"<< setw(31) << data.at(i).type\_of\_work << setw(2) << "|" << setw(22) << data.at(i).fio << setw(2) << "|" << setw(13) << fixed << setprecision(2) << data.at(i).hours << setw(3) << "|" << setw(10) << fixed << setprecision(2) << data.at(i).cost << endl;

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

}

cout << endl << endl;

}

void redactData(vector<project\_data>& data)

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

int number, index, choise;

bool one\_more\_time;

do

{

viewData(data);

number = protectRedactingNumber(data.size());

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

cout << "\nВыберите:\n\n[1] - Наименование проекта;\n[2] - Вид работ;\n[3] - Фамилия и инициалы сотрудника;\n[4] - Предполагаемое количество часов;\n[5] - Стоимость одного часа;\n\n";

one\_more\_time = 0;

cout << "Ввод: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

cout << "\nНаименование проекта: ";

data.at(index).project\_name = inputString();

break;

case 2:

cout << "\nВид работ:\n";

data.at(index).type\_of\_work = inputTypeOfWork();

break;

case 3:

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

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

break;

case 4:

cout << "\nПредполагаемое количество часов: ";

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

cin >> data.at(index).hours;

break;

case 5:

cout << "\nСтоимость одного часа: ";

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

cin >> data.at(index).cost;

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);

}

int protectRedactingNumber(int vector\_size)

{

bool check\_value = 0;

int number;

do // Реализуем защиту от некорректно введённого номера студента. Эта защита для полученного целочисленного значения номера студента

{

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

cin >> number;

if (!isBufferWithNoChar())

{

continue;

}

check\_value = 1;

if (number < 1 || number > vector\_size)

{

check\_value = 0;

system("cls");

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

system("pause");

system("cls");

}

} while (check\_value == 0);

return number;

}

void deleteData(vector<project\_data>& 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;

}

#include "searchMenu.h"

void searchMenu(vector<project\_data> 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 наименованию проекта;\n";

cout << "[2] - Поиск по виду работ;\n";

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

cout << "[4] - Поиск по количеству рабочих часов;\n";

cout << "[5] - Поиск по стоимости одного часа работы сотрудника;\n";

cout << "[6] - Назад;\n\n";

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

system("cls");

searchForProjectName(data);

system("cls");

break;

case 2:

system("cls");

searchForTypeOfWork(data);

system("cls");

break;

case 3:

system("cls");

searchForFIO(data);

system("cls");

break;

case 4:

system("cls");

searchForWorkingHours(data);

system("cls");

break;

case 5:

system("cls");

searchForWorkingCost(data);

system("cls");

break;

case 6:

system("cls");

break;

default:

choiseError();

}

if (choise == 6)

{

break;

}

}

SetConsoleOutputCP(866);

SetConsoleCP(866);

}

void searchForProjectName(vector<project\_data> data)

{

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

copy\_data = data;

string your\_project;

do

{

cout << "Введите наименование проекта: ";

your\_project = inputWithUpRegister();

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

{

if (data.at(i).project\_name != your\_project)

{

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";

break;

}

else

{

viewData(data);

}

data.clear();

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

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForTypeOfWork(vector<project\_data> data)

{

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

copy\_data = data;

string type\_of\_work;

do

{

cout << "Информация для поиска:\n\n";

type\_of\_work = inputTypeOfWork();

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

{

if (data.at(i).type\_of\_work != type\_of\_work)

{

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";

break;

}

else

{

viewData(data);

}

data.clear();

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

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForFIO(vector<project\_data> data)

{

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

copy\_data = data;

string fio;

do

{

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

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

cout << "Ввод: ";

fio = inputWithUpRegister();

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 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";

break;

}

else

{

viewData(data);

}

data.clear();

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

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForWorkingHours(vector<project\_data> data)

{

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

copy\_data = data;

double hours;

do

{

cout << "Введите количество часов работы сотрудника: ";

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

cin >> hours;

if (!isBufferWithNoChar())

{

continue;

}

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

{

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

{

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";

break;

}

else

{

viewData(data);

}

data.clear();

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

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

void searchForWorkingCost(vector<project\_data> data)

{

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

copy\_data = data;

double cost;

do

{

cout << "Введите стоимость одного часа работы сотрудника ($): ";

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

cin >> cost;

if (!isBufferWithNoChar())

{

continue;

}

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

{

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

{

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";

break;

}

else

{

viewData(data);

}

data.clear();

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

data = copy\_data;

if (!isWorkContinue())

{

break;

}

} while (true);

}

#include "sortMenu.h"

void sortMenu(vector<project\_data>data)

{

system("cls");

int choise;

while (true)

{

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

cout << "[1] - Сортировка по наименованию проекта;\n";

cout << "[2] - Сортировка по виду работ;\n";

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

cout << "[4] - Сортировка по количеству рабочих часов;\n";

cout << "[5] - Сортировка по стоимости одного часа работы сотрудника;\n";

cout << "[6] - Назад;\n\n";

cout << "Ваш выбор: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

switch (choise)

{

case 1:

system("cls");

sortForProjectName(data);

system("pause");

system("cls");

break;

case 2:

system("cls");

sortForTypeOfWork(data);

system("pause");

system("cls");

break;

case 3:

system("cls");

sortForFIO(data);

system("pause");

system("cls");

break;

case 4:

system("cls");

sortForHours(data);

system("pause");

system("cls");

break;

case 5:

system("cls");

sortForCost(data);

system("pause");

system("cls");

break;

case 6:

system("cls");

break;

default:

choiseError();

}

if (choise == 6)

{

break;

}

}

}

void sortForProjectName(vector<project\_data> data)

{

sort(data.begin(), data.end(), [](project\_data el\_1, project\_data el\_2)

{

return el\_1.project\_name < el\_2.project\_name;

});

viewData(data);

}

void sortForTypeOfWork(vector<project\_data> data)

{

sort(data.begin(), data.end(), [](project\_data el\_1, project\_data el\_2)

{

return el\_1.type\_of\_work < el\_2.type\_of\_work;

});

viewData(data);

}

void sortForFIO(vector<project\_data> data)

{

sort(data.begin(), data.end(), [](project\_data el\_1, project\_data el\_2)

{

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

});

viewData(data);

}

void sortForHours(vector<project\_data> data)

{

sort(data.begin(), data.end(), [](project\_data el\_1, project\_data el\_2)

{

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

});

viewData(data);

}

void sortForCost(vector<project\_data> data)

{

sort(data.begin(), data.end(), [](project\_data el\_1, project\_data el\_2)

{

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

});

viewData(data);

}

#include "userMenu.h"

void userMenu(vector<project\_data> 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 проекте;\n";

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

}

}

}

#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] - Подтверждение / Блокировка новых пользовательских учётных записей;" << endl;

cout << "[6] - 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");

verifiedAccounts(accounts\_data);

break;

case 6:

system("cls");

break;

default:

choiseError();

}

if (choise == 6)

{

break;

}

}

}

void verifiedAccounts(vector <autorisation>& accounts\_data)

{

int kol = 0, choise;

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

{

if (accounts\_data.at(i).access == 0)

{

kol++;

do

{

cout << kol<<"-ый "<<"пользователь, ожидающий верификацию:\n\n";

cout << "Логин: " << accounts\_data.at(i).login<<endl;

cout << "Пароль: " << accounts\_data.at(i).passowrd<<endl<<endl;

if (isTheSameLoginInVector(accounts\_data, accounts\_data.at(i).login, i))

{

system("cls");

cout << "Данный аккаунт будет автоматически заблокирован системой, так как учетная запись с таким логином уже существует.\n\n";

system("pause");

system("cls");

break;

}

cout << "Ваш выбор:\n\n";

cout << "[1] - Подтвердить учетную запись;\n";

cout << "[2] - Отклонить учетную запись;\n\n";

cout << "Ввод: ";

cin >> choise;

if (!isBufferWithNoChar())

{

continue;

}

if (choise == 1)

{

accounts\_data.at(i).access = 1;

cout << "\nУчетная запись успешно подтверждена.\n\n";

system("pause");

system("cls");

break;

}

else if (choise == 2)

{

accounts\_data.erase(accounts\_data.begin() + i);

cout << "\nУчетная запись успешно отклонена.\n\n";

system("pause");

system("cls");

break;

}

else

{

choiseError();

continue;

}

} while (true);

}

}

if (kol == 0)

{

cout << "Действие невозможно ! Учетные записи, ожидающие верификацию отсутствуют.\n\n";

system("pause");

system("cls");

}

}

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).access == 1)

{

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);

box.access = 1;

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);

number = protectRedactingNumber(accounts\_data.size());

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);

}

#include "workWithDataAdminMenu.h"

void workWithDataAdmin(vector<project\_data>& 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;

}

}

}

#include "workWithFilesAdmin.h"

void launchMainFunctionMenu(vector <autorisation> &accounts\_data, vector <project\_data> &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<project\_data>& 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.access = access;

box.salt = salt;

accounts\_data.push\_back(box);

}

}

fin.close();

}

void fromFileToVectorDATA(vector<project\_data>& 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, len\_project\_name, len\_tow;

char \*project\_name, \*tow, \*fio;

double hours, cost;

project\_data box;

string str;

getline(fin, str);

if (str.empty())

{

return;

}

else

{

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

}

while (!fin.eof())

{

fin >> len\_project\_name;

project\_name = new char[len\_project\_name];

fin >> project\_name>> len\_tow;

tow = new char[len\_tow];

fin >> tow>>len\_fio;

fio = new char[len\_fio];

fin >> fio >> hours>> cost;

if (strlen(project\_name) == 0)

{

continue;

}

box.project\_name = project\_name;

box.type\_of\_work = tow;

box.fio = fio;

box.hours = hours;

box.cost = cost;

data.push\_back(box);

}

}

fin.close();

}

}

void pushDataInFile(vector <project\_data> 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).access << " "<< accounts\_data.at(i).salt.length()<<" "<< accounts\_data.at(i).salt<<" ";

}

fout.close();

}

void fromVectorToFileDATA(vector <project\_data> 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).project\_name.length() << " " << data.at(i).project\_name << " " << data.at(i).type\_of\_work.length() << " " << data.at(i).type\_of\_work << " " << data.at(i).fio.length()

<< " " << data.at(i).fio << " " << data.at(i).hours << " " << data.at(i).cost << " ";

}

fout.close();

}