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**

Выполнил: студент гр

Проверил:

Минск 2020

**Содержание**

1.Требование к программе3

2. Конструирование программы6

2.1 Разработка модульной структуры программы 6

2.2 Выбор способа организации данных 6

2.3 Разработка перечня пользовательских функций программы 7

3. Разработка алгоритмов работы программы 11

3.1 Алгоритм функции main()11

3.2 Алгоритм функцииredactionLoginPassword(users \*&users, int numberOfUsers)12

3.3 Алгоритм функции searchMonth(employees \*employee, int numberOfEmployee) 13

4. Описание работы программы14

4.1 Авторизация 14

4.2 Модуль администратора 15

4.3 Модуль пользователя22

4.4 Исключительные ситуации24

Приложение А. Листинг кода с комментариями24

**1. Требования к программе**

**Текст варианта задания: разработка программы продажи автобусных билетов**

Сведения о сотрудниках предприятия содержат: Ф.И.О. сотрудника; табельный номер; год; месяц; количество проработанных часов за месяц; почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере.  
Индивидуальное задание: рассчитать размер заработной платы каждого сотрудника за вычетом подоходного налога, который составляет 12% от суммы заработной платы. Определить объем выплат конкретному сотруднику за требуемый период времени (требуемый период времени вводится с клавиатуры).  
Общее для всех вариантов задание: реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя

**Исходные данные к курсовой работе:**

* Язык программирования С++
* Среда разработки Microsoft Visual Studio версии 2010 и выше
* 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
* Построение программного кода должно соответствовать правилам, определенным в документе «С++ Code Convention»
* Текст пояснительной записки оформляется в соответствии со стандартом предприятия СТП 01–2017

**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:

user;

password;

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

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

* в качестве администратора (в этом случае role = 2),
* в качестве пользователя (в этом случае role = 1).

Файл с пользователями всегда существуют, его можно изменять, но нельзя полностью удалить (в файле предусмотрен главный админ(логин-admin): его никто не может удалить и он сам себя так же не может удалить).

Вторым этапом работы программы является собственно работа с данными, которая становится доступной только после прохождения авторизации. Данные хранятся в отдельном файле и имеют вид таблицы с полями: Ф.И.О. сотрудника; год; месяц; количество дней, пропущенных по болезни; оплата за один день.

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

***Модуль администратора* включает следующие разделы:**

Управление учетными записями пользователей:

* Просмотр всех учетных записей;
* Активация аккаунтов;
* Добавление новой учетной записи;
* Редактирование учетной записи;
* Удаление учетной записи.

Работа с данными:

а) Режим редактирования:

* Просмотр всех данных;
* Добавление новых записей;
* Редактирование записей;
* Удаление записей;

б) Режим обработки данных:

* Поиск данных по Ф.И.О. сотрудника; год; месяц; почасовой тариф.;
* Cортировка по Ф.И.О. сотрудника; год; количество проработанных часов за месяц; почасовой тариф.;
* Индивидуальное задание.

***Модуль пользователя* включает раздел работы с данными со следующими функциональными возможностями:**

* Просмотр всех данных;
* Поиск данных по Ф.И.О. сотрудника; год; месяц; почасовой тариф.;
* Cортировка по Ф.И.О. сотрудника; год; количество проработанных часов за месяц; почасовой тариф.;
* Индивидуальное задание;

**Предусмотреть:**

Обработку исключительных ситуаций (ничего не найдено по результатам поиска; логин новой учетной записи уже существует; неправильно введен логин/пароль, повторное введение пароля при регистрации);

Возможность возврата назад (навигация);

Запрос на подтверждение удаления вида «Вы действительно хотите удалить запись?»;

Запрос на подтверждение редактирование вида «Вы действительно хотите редактировать запись?»;

Вывод сообщения об успешности создания /удаления записи.

**Требования к программной реализации проекта:**

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.

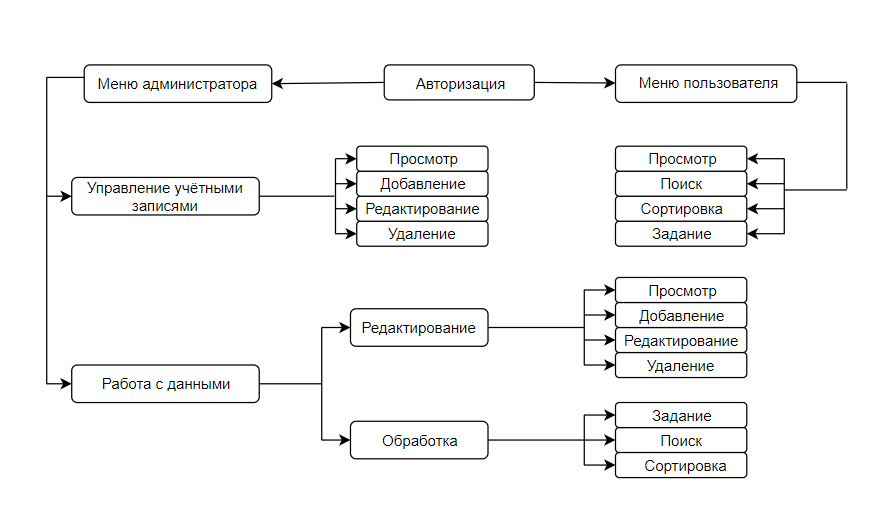
Код не должен содержать неименованных числовых констант (так называемых «магических» чисел), неименованных строковых констант (например, имен файлов и др.). Подобного рода информацию следует выносить в глобальные переменные с атрибутом const. Код необходимо комментировать (как минимум в части объявления 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.

1. **Конструирование программы**
   1. **Разработка модульной структуры программы**

**2.2 Выбор способа организации данных.**

В качестве выбора способа описания входных данных:

1. Структура users для учетных записей пользователя с полями user, password, code.

struct users //структура для хранения пользователей

{

string user;//логин пользователя

string password;//пароль пользователя

int code;//роль ползователя

};

2. Структура employees для информации о сотрудниках с полями: Ф.И.О. сотрудника; табельный номер; год; месяц; количество проработанных часов за месяц; почасовой тариф.

struct employees

{

string FIO; //ФИО

string personnelNumber;//табельный номер

int year; //год

int month;//месяц

int numberOfHours;// оплпта за час

double oneHourPayment;// кол-во часов

};

**2.3 Разработка перечня пользовательских функций программы.**

Таблица 2.1 – Описание пользовательских функция

|  |  |  |
| --- | --- | --- |
| Категория | Название функции | Описание |
| Меню | void menuAdmin(users \*&users, employees \*&employee,); | Меню администратора |
| void menuUsers(employees \*&employee,); | Меню пользователя |
| void menuWorkWithAccounts(users \*&users, int &numberOfUsers); | Меню для управления уч. записями(администратор) |
| void menuWorkWithDataEmployee(employees \*&employee, int &numberOfEmployee); | Меню для работы с данными(администратор) |
| void editDataAboutEmployee(employees \*&employee, int &numberOfEmployee); | Меню для редактирования данных (администратор) |
| void employeesProcessing(employees \*&employee, int &numberOfEmployee); | Меню для обработки данных( администратор) |
| void menuRedactionEmployee(employees \*&employee, int &numberOfEmployee); | Меню для редактирования данных о сотрудниках |
| void redactionLoginPassword(users \*&users, int numberOfUsers); | Меня для редактирования логина и пароля |
| void menuSorting(employees \*&employee, int &numberOfEmployee); | Меню для сортировки |
| void menuSearch(employees \*&employee, int &numberOfEmployee); | Меню для поиска |
| Работа с учетными записями | void viewLoginsPassword(users \*users, int numberOfUsers); | Просмотр уч. записей |
| int listUsers(users \*&users); | Считывание уч. записей из файла в массив |
| void deleteAccounts(users \*&users, int &numberOfUsers); | Удаление уч. записи |
| void addingNewUsers(users \*&users, int &numberOfUsers); | Добавление уч. Записи админом. |
| void redactionPassword(users \*&users, int numberOfUsers); | Редактирование пароля. |
| void redactionLogin(users \*&users, int numberOfUsers); | Редактирование логина. |
| void InputAllInformationAboutAccountsInFile (users \*users, int numberOfUsers); | Обновление информации в файле с информацией о пользователях |
| void addToFileOneAccount(string user, string password, int code); | Добавление одной записи в конец файла |
| void activationAccounts(users \*&users, int numberOfUsers); | Активация аккаунтов пользователей. |
| Автори-зация | string inputPasswordStars(); | Ввод пароля звездочками при регистрации и авторизации. |
| string inputLogin(users \*&users); | Ввод логина при регистрации и авторизации. |
| string inputPassword(); | Ввод пароля при регистрации и авторизации. |
| void registration(users \*&users); | Регистрация нового пользователя. |
| int searchLoginPassword(users \*&users, int numberOfUsers, int &number\_of\_attempts, string user, string password); | Поиск логина и пароля. Определение пользователя или администратора. |
| int entrance(users \*&users); | Авторизация(вход в систему). |
| int registrationAndAuthorization(users \*&users, int&press); | Выбор авторизация или регистрация. |
| Работа с данными | int listEmployeeData(employees \*&employee); | Считывание информации о сотрудниках из файла в массив |
| void addingNewEmployeeData(employees \*&employee, int &numberOfEmployee); | Добавление сотрудника |
| void ViewEmployee(employees \*employee, int numberOfEmployee); | Просмотр сотрудников |
| void ViewSearch(employees \*employee, int k, int i); | Просмотр сотрудников при поиске. |
| void deleteEmployeeData(employees \*&employee, int &numberOfEmployee); | Удаление сотрудников |
| void InputAllInformationAboutEmployeeDataInFile(employees \*employee, int numberOfEmployee); | Обновление информации в файле с информацией о сотрудниках |
| Сорти-ровка | void sortingFIO(employees \*employee, int numberOfEmployee); | Сортировка по ФИО. |
| void sortingAmountDays(employees \*employee, int numberOfEmployee); | Сортировка по количеству отработанных часов |
| void sortingOneDayPayment(employees \*employee, int numberOfEmployee); | Сортировка оплате за сач. |
| void sortingYear(employees \*employee, int numberOfEmployee); | Сортировка по году. |
| Поиск | void searchFIO(employees \*employee, int numberOfEmployee); | Поиск по ФИО. |
| void searchYear(employees \*employee, int numberOfEmployee); | Поиск по году. |
| void searchOneDayPayment(employees \*employee, int numberOfEmployee); | Поиск по оплате за час. |
| void searchMonth(employees \*employee, int numberOfEmployee); | Поиск по месяцу. |
| Редактиро-вание | void redactionFIO(employees \*&employee, int &numberOfEmployee); | Редактирование ФИО. |
| void redactionYear(employees \*&employee, int &numberOfEmployee); | Редактирование года. |
| void redactionAmountDays(employees \*&employee, int &numberOfEmployee); | Редактирование количества отработанных часов. |
| void redactionOneDayPayment(employees \*&employee, int &numberOfEmployee); | Редактирование оплаты за час. |
| void redactionMonth(employees \*&employee, int &numberOfEmployee); | Редактирование месяца. |
| void redactionPersonnelNumber(employees \*&employee, int &NumberOfEmployee) | Редактирование табельного номера |
| Проверка на ввод | int inputInt(); | Проверка на ввод целого числа |
| int inputMonth(); | Проверка на ввод месяца |
| int inputYear(); | Проверка на ввод года |
| double inputDouble(); | Проверка на ввод дробного числа. |
| int stringComparison(string s, string a); | Проверка на совпадение паролей. |
| string enterStringWithSmallLetter(string name); | Проверка на ввод слова с маленькой буквы. |
| bool loginTest(string user); | Проверка на ввод логина. |
| string passwordTest(); | Проверка на ввод пароля  при регистрации/доб. нового пользователя. |
| string inputFIO() | Проверяет ввод ФИО сотрудника (только латинские буквы и цифры) |

# 

# 3. Разработка алгоритмов работы программы

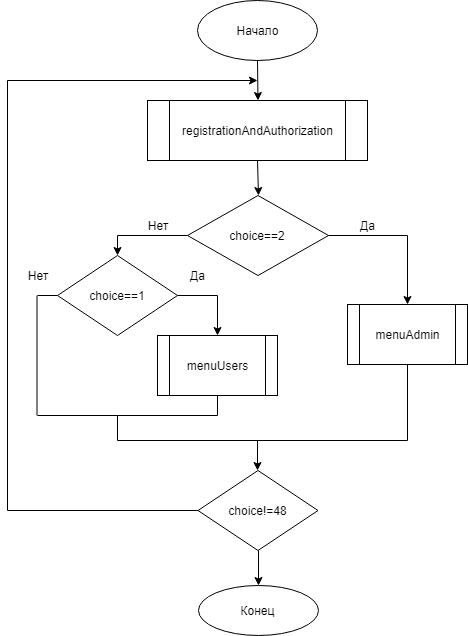
* 1. **Алгоритм функции *main()***

Рисунок 3.1 – Алгоритм функции *main*

**3.2 Алгоритм функции** *redactionLoginPassword(users \*&users, int numberOfUsers)*

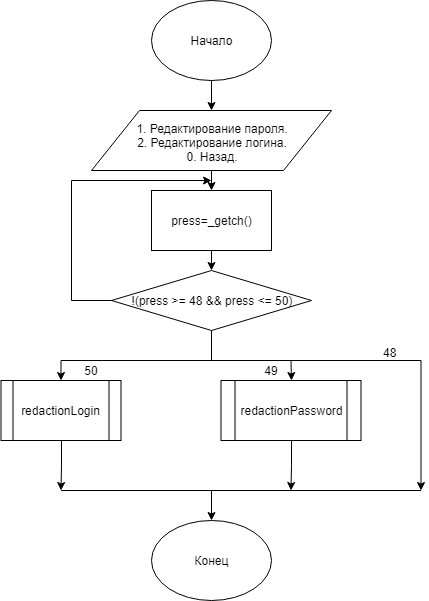
****

Рисунок 3.2 – Алгоритм функции *redactionLoginPassword*

**3.3 Алгоритм функции searchMonth(employees \*employee, int numberOfEmployee);**

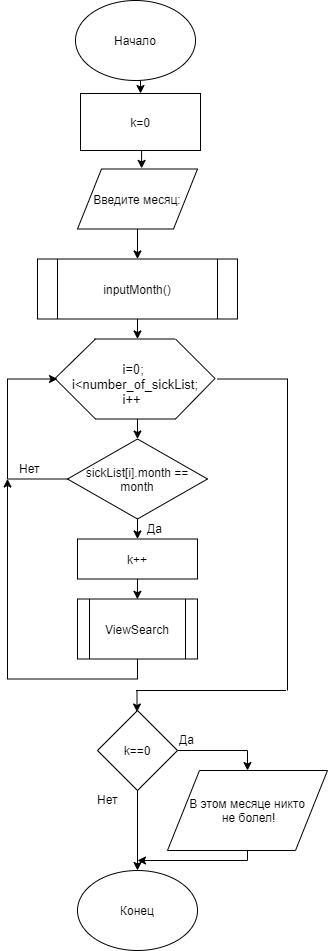
**

Рисунок 3.3 – Алгоритм функции *searchMonth*

**4. Описание работы программы**

**Главное меню**

При запуске программы открывается главное меню и пользователь может выбрать одно из действий: авторизоваться, зарегистрироваться, или выйти из программы (рисунок 4.1).

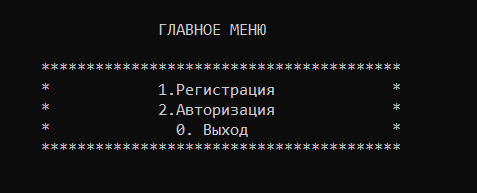


Рисунок 4.1 – Главное меню

**4.1 Авторизация**

Программа предусматривает 2 модуля: модуль администратора и модуль пользователя. Доступ к тому или иному модулю осуществляется посредством авторизации.

При выборе авторизации в главном меню пользователю предлагается ввести логин и пароль. При не нахождении совпадений пользователь продолжает вводить логин и пароль. Если пользователь вводит 3 раза правильный логин, но неправильный пароль, ему предлагается поменять пароль.

В зависимости от значения поля «Роль» соответствующей структуры предоставляется доступ либо к модулю пользователя, либо администратора (рисунок 4.2,4.3).



Рисунок 4.2 – Вход как администратор

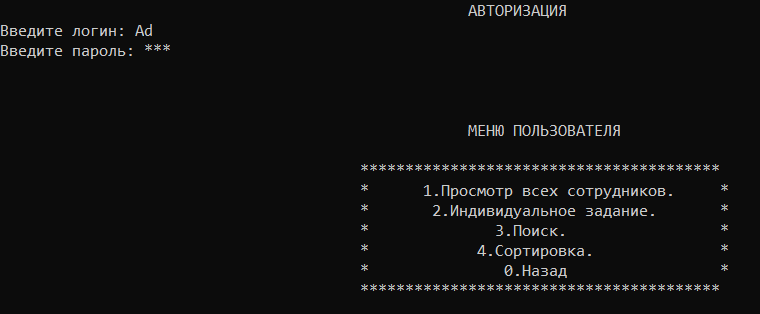


Рисунок 4.3 – Вход как пользователь

**4.2 Модуль администратора**

Модуль администратора предоставляет возможность работать с учетными записями и непосредственно с информацией о билетах. Предоставляется возможность перехода в главное меню (рисунок 4.4).

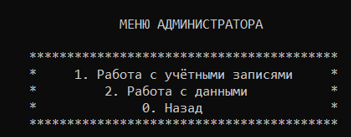


Рисунок 4.4 – Меню администратора

**Работа с учетными записями**

Управление учетными записями предоставляет возможность просмотра учетных записей, добавление учетной записи, редактирование учетной записи,

удаление учетной записи, активирование аккаунтов пользователей. Предоставляется возможность перехода в предыдущее меню. (рисунок 4.5).

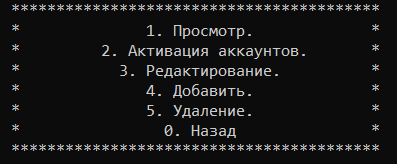


Рисунок 4.5 – Подмодуль «Учетные записи»

**Просмотр учетных записей**

Просмотр учетных записей выводит на экран все существующие учетные записи пользователей указывая приоритетность: 0 - не активированный аккаунт пользователя, 1 – пользователь, 2 – администратор, 3-не активированный аккаунт администратора, 4-заблокированный аккаунт. (рисунок 4.6)

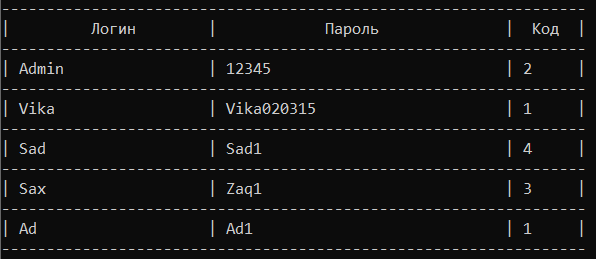


Рисунок 4.6 – Существующие учетные записи

**Добавление новой учетной записи**

Регистрация нового пользователя предоставляет возможность выбора приоритетности будущей учетной записи (рисунок 4.7)

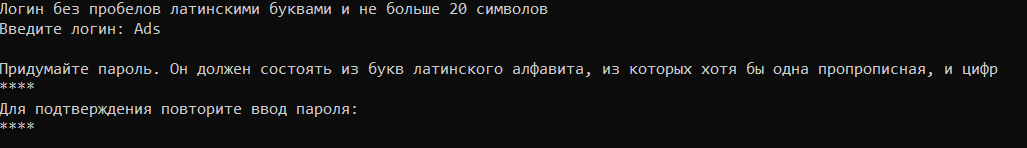


Рисунок 4.7 – Регистрация нового пользователя

**Удаление учетной записи**

Удаление учетной записи служит для удаления аккаунта пользователя, предлагая ввести логин пользователя, чью информацию вы хотите удалить. Реализовано подтверждение действия. При этом реализована отмена удаления главного админа (рисунок 4.8).

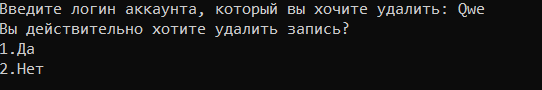


Рисунок 4.8 – Удаление учетных записей

**Редактирование учетной записи**

Редактирование учетной записи служит для изменения данных о пользователях (логин, пароль). Реализована отмена редактирования (рисунок 4.9, 4.9.1).

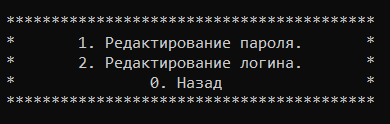


Рисунок 4.9 – Меню редактирования.

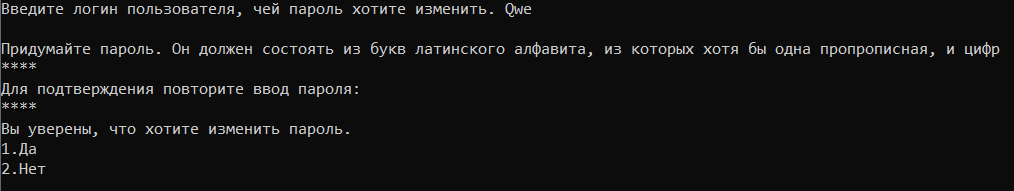


Рисунок 4.9.1 – Редактирование пароля.

**Активация учетной записи.**

Активация учетной записи служит для подтверждения или блокировки аккаунтов (рисунок 4.10).

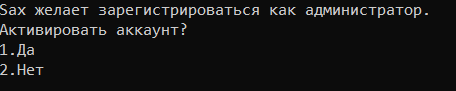


Рисунок 4.10 – Активация аккаунтов.

**Работа с данными**

Работа с данными предоставляет возможность редактирования записи и режим обработки данных. Редактирование записи служит для редактирования данных о сотрудниках, а режим обработки служит для работы с данными. Предоставляется возможность перехода в предыдущее меню (рисунок 4.11).

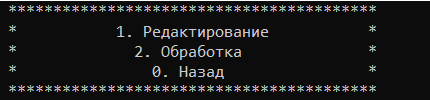


Рисунок 4.11 – Подмодуль «Работа с данными»

**Редактирование данных**

Редактирование данныхвключает в себя просмотр данных, добавление новой записи, редактирование записи, удаление записи. Предоставляется возможность перехода в предыдущее меню (рисунок 4.12).

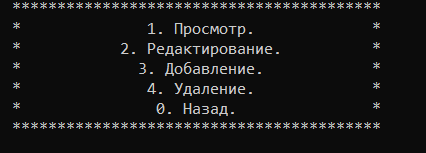


Рисунок 4.12 – Редактирование данных

**Просмотр данных**

Просмотр данных выводит на экран всю информацию о сотркдниках(рисунок 4.13).

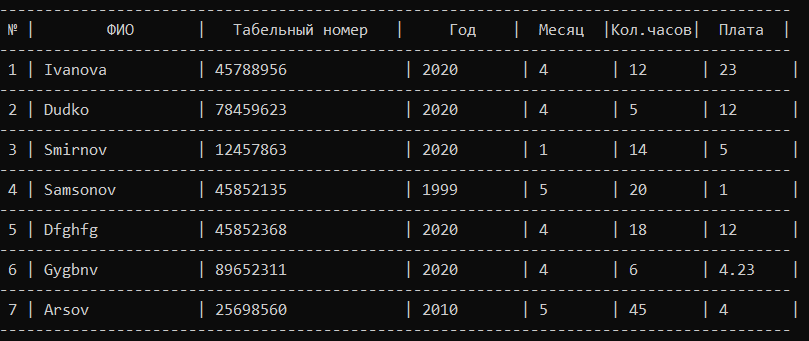


Рисунок 4.13 – Просмотр всех записей

**Добавление новой записи о сотрудниках**

Добавление новой записи о сотрудниках предлагает ввести данные для нового сотрудника (рисунок 4.14).

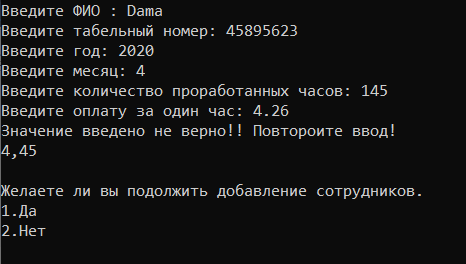


Рисунок 4.14 – Добавление записи о сотрудниках

**Удаление записи о сотрудниках**

Удаление записи о сотрудниках предлагает ввести ФИО сотрудника, информацию о котором вы хотите удалить. Реализовано подтверждение действия (рисунок 4.15).

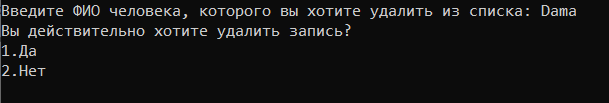


Рисунок 4.15 – Удаление записи о сотрудниках

**Редактирование записи о сотрудниках**

Редактирование записи о сотрудниках предлагает ввести ФИО человека, информацию о которой вы хотите редактировать. Реализована отмена редактирования (рисунок 4.16).

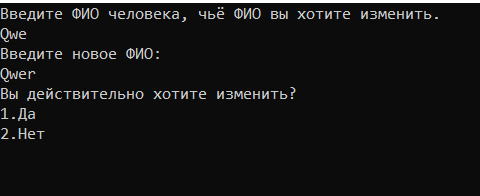


Рисунок 4.16 – Редактирование записи о сотруднике

**Режим обработки данных**

Режим обработки данных включает индивидуальное задание, поиск данных, сортировка данных. Предоставляется возможность перехода в предыдущее меню. (рисунок 4.17).

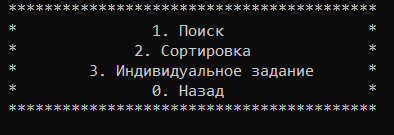


Рисунок 4.17 – Режим обработки данных

**Индивидуальное задание**

Индивидуальное задание: рассчитать размер заработной платы каждого сотрудника за вычетом подоходного налога, который составляет 12% от суммы заработной платы. Определить объем выплат конкретному сотруднику за требуемый период времени (требуемый период времени вводится с клавиатуры). (рисунок 4.18).

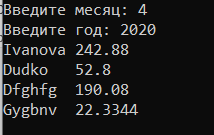


Рисунок 4.18 – Индивидуальное задание

**Поиск данных**

Реализован поиск данных по ФИО, по году, по месяцу, по оплате за один час. (рисунок 4.19).

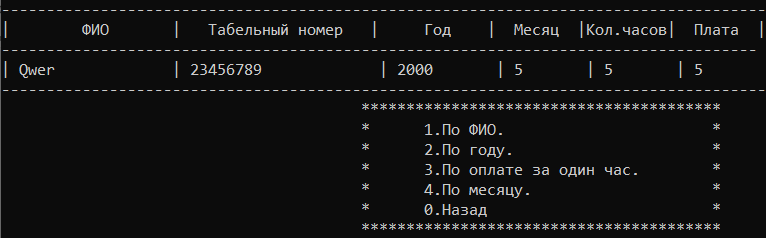


Рисунок 4.19 – Поиск данных

**Сортировка**

Сортировкиреализованы по ФИО, по году, по количеству отработанных часов, по оплате за час. Реализована возможность отмены сортировки (рисунок 4.20).

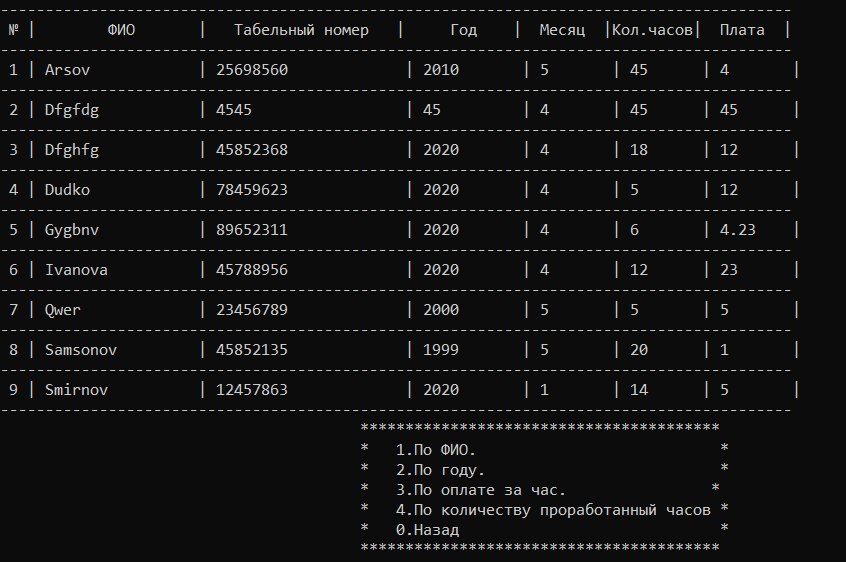


Рисунок 4.20 – Сортировка данных

**4.3 Модуль пользователя**

Модуль пользователя предполагает просмотр всех сотрудников, индивидуальное задание, поиск данных по всем характеристикам, сортировку данных по всем характеристикам, выход в главное меню.

Работа каждого подмодуля приведена выше (рисунок 4.21).

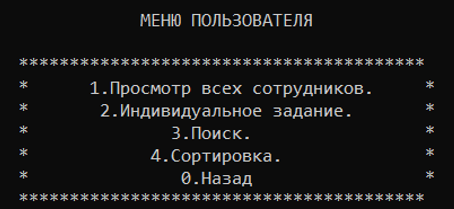


Рисунок 4.21 – Меню пользователя

**Регистрация**

Модуль регистрации дает возможность пользователю самому создать свой аккаунт из главного меню для дальнейшего пользования программы и модуля пользователя включительно. Пользователь сам выбирает как ему зарегистрироваться, но его регистрацию подтверждает текущий администратор. После регистрации пользователь снова оказывается в главном меню (рисунок 4.22.1, 4.22.2).

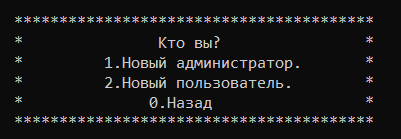


Рисунок 4.22.1- Выбор для регистрации

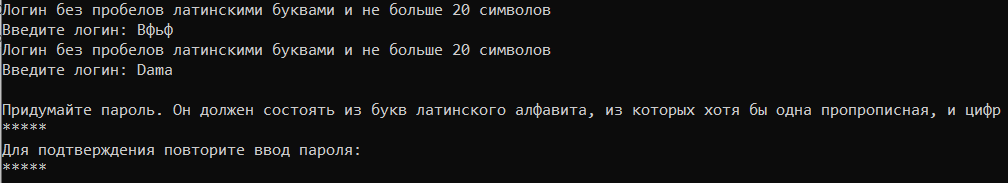


Рисунок 4.22.2 – Регистрация

**4.4 Исключительные ситуации**

Программа подразумевает обработку следующих исключительных ситуаций:

1. Неверно введен логин или пароль.
2. Неверный ввод цифр.
3. Логин уже существует.
4. Удаление главного админа.
5. Подтверждение удаления и редактирования.
6. Повторный ввод пароля.
7. Ввод ФИО с маленькой буквы.
8. Проверка на ввод ФИО.
9. Проверка на ввод месяца и года.

**Приложение А (обязательное).**

**Листинг кода с комментариями.**

**main.cpp**

#include<iostream>

#include"Authorization.h"

#include"Users.h"

#include"Admin.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

users \*user;

employees \*employee;

int choice;

do {

registrationAndAuthorization(user, choice);// регистрация и авторизация

if (choice == 2) menuAdmin(user, employee); //меню админа

else if (choice == 1) menuUsers(employee); //меню пользователя

} while (choice != 48);

}

**Authorization.h**

#pragma once

#include"Admin.h"

using namespace std;

string inputPasswordStars();//ввод пароля звёздачками

string passwordTest();//проверка на правильный ввод пароля

bool loginTest(string login);//проверка на ввод логина

string inputLogin(users \*&user);//ввод логина

string inputPassword();//ввод пароля с подтверждением

void registration(users \*&user);//Регистрация нового пользователя

int stringComparison(string s, string a);//сравнение двух паролей

int searchLoginPassword(users \*&user, int numberOfUsers, int &number\_of\_attempts, string login, string password);// Поиск логина и пароля.Определение пользователя или администратора.

int entrance(users \*&user);// Авторизация

int registrationAndAuthorization(users \*&user, int&press);//Выбор авторизация или регистрация.

**Authorization.cpp**

#include<iostream>

#include<string>

#include<conio.h>

#include<fstream>

#include<Windows.h>

#include<iomanip>

#include"Admin.h"

#include"File.h"

using namespace std;

string inputPasswordStars() //Ввод пароля звёздочками

{

string s;

int i = 0;

int ch = 0;

while (1) {

ch = \_getch();

if (ch == 83 || ch == 72 || ch == 80 || ch == 75 || ch == 77 || ch == 32) continue;

if ((ch >= 'a'&& ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <= '9')) { //возможные символы для ввода

s += (char)ch; i++;

cout << "\*";

continue;

}

if (ch == 13) break;

if (ch == 8) { //удаление символа

if (i > 0) {

cout << "\b \b";

i--;

s.erase(i, 1); //удаление 1 символа c i позиции

}

}

}

return s;

}

string passwordTest() //Проверка пароля на правильность ввода

{

string s;

int l, j = 0, k = 0;

bool flag = false;

do {

s = inputPasswordStars();

l = s.length();

for (int i = 0; i < l; i++) {

if (s[i] >= 'a'&& s[i] <= 'z') continue;

if (s[i] >= 'A' && s[i] <= 'Z') {

j++; continue;

}

if (s[i] >= '0' && s[i] <= '9') {

k++;

}

}

if (k == 0 || j == 0) {

cout << "\nНеверный ввод пароля! Повторите ввод!!\n";

flag = false;

}

else flag = true;

} while (flag == false);

return s;

}

bool loginTest(string login) // проверка на прильность ввода логина

{

int l;

l = login.length();

if (l > 20) {

system("CLS");

cout << "Логин слишком длинный.\n";

return false;

}

else

{

for (int i = 0; i < l; i++) {

if (login[i] == ' ' || (!(login[i] >= 'a'&& login[i] <= 'z')

&& !(login[i] >= 'A' && login[i] <= 'Z')

&& !(login[i] >= '0' && login[i] <= '9'))) return false;

}

return true;

};

}

string inputLogin(users \*&user) //Ввод пароля с проверкой на уникальность

{

int numberOfUsers;

string login;

bool flag;

numberOfUsers = listUsers(user); //занесение в массив данных о пользователей

do {

flag = false;

while (flag == false) {

cout << "Логин без пробелов латинскими буквами и не больше 20 символов\n";

cout << "Введите логин: ";

cin >> login;

flag = loginTest(login);

}

for (int i = 0; i < numberOfUsers; i++) { //сравнение с существующими

if (login == user[i].login) {

system("CLS");

cout << "Такой логин уже существует.\n";

flag = false;

break;

}

}

} while (flag == false);

return login;

}

string inputPassword() // Ввод пароля с повторным его вводом

{

string password, password1;

bool flag;

do {

cout << "\nПридумайте пароль. Он должен состоять из букв латинского алфавита, из которых хотя бы одна пропрописная, и цифр\n";

password = passwordTest();

cout << "\nДля подтверждения повторите ввод пароля: \n"; //повторный ввод пароля для подтверждения

password1 = inputPasswordStars();

if (password != password1) {

system("CLS");

cout << "Пароли не совпадают. Повторите ввод паролей.\n";

flag = false;

}

else flag = true;

} while (flag == false);

return password;

}

void registration(users \*&user) // Регистрация пльзователя или администратора

{

string login, password, password1;

int code, press;

bool flag = true;

system("CLS");

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

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* Кто вы? \*\n"

<< "\t\t\t\t\t\* 1.Новый администратор. \*\n"

<< "\t\t\t\t\t\* 2.Новый пользователь. \*\n"

<< "\t\t\t\t\t\* 0.Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (press != 49 && press != 50 && press != 48);

system("CLS");

if (press == 49)

code = 3;

else if (press == 50)

code = 0;

else return;

login = inputLogin(user);

password = inputPassword();

system("CLS");

/\*ofstream fadd(FILE\_ACCOUNT, ios::app);

fadd << setw(20) << left << login << setw(30) << password << setw(5) << left << code << endl;

fadd.close();\*/

addToFileOneAccount(login, password, code); //запись в файл нового пользователя

cout << "Спасибо за регистрацию. Вскором времени администратор подтвердит вашу регистрацию.";

Sleep(3000);

return;

}

int stringComparison(string s, string a) //Сравнение двух строк

{

int l, l1;

l = s.length();

l1 = a.length();

if (l != l1) return 0;

else

{

for (int i = 0; i < l; i++) {

if (s[i] == a[i])

continue;

else return 0;

}

return 1;

}

}

int searchLoginPassword(users \*&user, int numberOfUsers, int &number\_of\_attempts, string login, string password)//сравнивание логина и пароля

{

int d;

for (int i = 0; i < numberOfUsers; i++) {

if (user[i].login == login) //сначала ищем логин

{

number\_of\_attempts++;

d = stringComparison(user[i].password, password); // сравниваем пароли

if (d == 0) {

system("CLS");

cout << "Логин или пароль введены неверно!\nПовторите ввод.\n";

return 3;

}

else {

if (user[i].code == 4) {

system("CLS");

cout << "Ваш аккаунт заблокировали! Попробуйте зарегистрироваться ещё раз.";

Sleep(2000);

return 4;

}

if (user[i].code == 2) return 2; //код администратора

if (user[i].code == 1) return 1; //код пользователя

if (user[i].code == 0 && user[i].code == 3) //код неактивированных пользовател

{

system("CLS");

cout << "Аккаунт не активирован";

Sleep(2000);

return 0;

}

}

}

}

number\_of\_attempts = 0;

if (number\_of\_attempts == 0) {

system("CLS");

cout << "Логин введен неверно!\nПовторите ввод.\n";

return 3;

}

}

int entrance(users \*&user) //авторизация

{

string password, login;

int d = 0, press, numberOfAttempts = 0, numberOfUsers;

bool flag;

numberOfUsers = listUsers(user);

cout << "\t\t\t\t\t\t АВТОРИЗАЦИЯ\n";

do {

cout << "Введите логин: ";

cin >> login;

cout << "Введите пароль: ";

password = inputPasswordStars();

flag = true;

d = searchLoginPassword(user, numberOfUsers, numberOfAttempts, login, password);

if (d == 1) return 1;

if (d == 2) return 2;

if (d == 3) flag = false;

if (numberOfAttempts == 3) {

system("CLS");

cout << "\nЖелаете ли вы сменить пароль.\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

system("CLS");

if (press == 49) redactionPassword(user, numberOfUsers);

else numberOfAttempts = 0;

}

} while (flag == false);

}

int registrationAndAuthorization(users \*&login, int &press) //меня регистрации и авторизации

{

int choice;

do {

cout << "\n\n\n\n\t\t\t\t\t\t ГЛАВНОЕ МЕНЮ";

cout << "\n\n";

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1.Регистрация \*\n"

<< "\t\t\t\t\t\* 2.Авторизация \*\n"

<< "\t\t\t\t\t\* 0. Выход \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 50));

system("CLS");

switch (press)

{

case 48: return press; break;

case 49:

registration(login); //регистрация

system("CLS");

break;

case 50:

press = entrance(login); // авторизация

//system("CLS");

return press;

}

} while (press != 48);

}

**Admin.h**

#pragma once

using namespace std;

struct users

{

string login;//логин

string password;//пароль

int code;//роль

};

struct employees

{

string FIO; //ФИО

string personnelNumber;//табельный номер

int year; //год

int month;//месяц

int numberOfHours;// оплпта за час

double oneHourPayment;// кол-во часов

};

void menuAdmin(users \*&user, employees \*&employee);//Меню администратора

void menuWorkWithDataEmployee(employees \*&employee, int &NumberOfEmployee);//Меню для работы с данными(администратор)

void menuWorkWithAccounts(users \*&user, int &numberOfUsers);//Меню для управления уч. записями(администратор)

void employeesProcessing(employees \*&employee, int &NumberOfEmployee);//Меню для обработки данных(администратор)

void editDataAboutEmployee(employees \*&employee, int &NumberOfEmployee);//Меню для редактирования данных (администратор)

void redactionLoginPassword(users \*&user, int numberOfUsers);//Меня для редактирования логина и пароля

void redactionLogin(users \*&user, int numberOfUsers);//редактирование логина

void redactionPassword(users \*&user, int numberOfUsers);//редактирование пароля

void deleteAccounts(users \*&user, int &numberOfUsers);//удаление аккаунтов

void viewLoginsPassword(users \*user, int numberOfUsers);//просмотр логинов и паролей

void addingNewUsers(users \*&user, int &numberOfUsers);//добавление нового пользователя

void activationAccounts(users \*&user, int numberOfUsers);//активация новах аккаунтов

void addingNewEmployeeData(employees \*&employee, int &NumberOfEmployee);//добавление нового сотудника

void deleteEmployeeData(employees \*&employee, int &NumberOfEmployee);//удаление сотрудника

void redactionFIO(employees \*&employee, int &NumberOfEmployee);//редактирование ФИО

void redactionYear(employees \*&employee, int &NumberOfEmployee);//редактирование года

void redactionNumberOfHours(employees \*&employee, int &NumberOfEmployee);//редактирование количества часов

void redactionOneHourPayment(employees \*&employee, int &NumberOfEmployee);//редактирование оплаты

void redactionMonth(employees \*&employee, int &NumberOfEmployee);//редактирование месяца

void redactionPersonnelNumber(employees \*&employee, int &NumberOfEmployee);//редактирование месяца

void individualTask(employees \*&employee, int &NumberOfEmployee);//индивидуальное задание

void menuRedactionEmployee(employees \*&employee, int &NumberOfEmployee);//Меню для редактирования данных о сотруднике

int inputInt();//Ввод целого числа

double inputDouble();//ввод дробного числа

string inputFIO();//проверка на ввод названия

int inputYear();//проверка на ввод года

int inputMonth();//проверка на ввод месяца

**Admin.cpp**

#include<iostream>

#include<string>

#include<conio.h>

#include<fstream>

#include<Windows.h>

#include<iomanip>

#include"Admin.h"

#include"File.h"

#include"Users.h"

#include"Authorization.h"

using namespace std;

string inputFIO()//проверка на ввод ФИО

{

string name;

int l;

bool flag;

do {

flag = true;

cin >> name;

l = name.length();

for (int i = 0; i < l; i++) {

if (!(name[i] >= 'a'&& name[i] <= 'z') && !(name[i] >= 'A'&& name[i] <= 'Z'))

{

cout << "Повторите ввод!! Только латинские буквы\n";

flag = false;

break;

}

}

} while (flag == false);

return name;

}

int inputYear() //ввод года

{

int year;

while (1) {

year = inputInt();

if (year == 0 || year > 2020)

{

cout << "Введите корректно год!\n";

}

else return year;

}

}

int inputMonth()

{

int month;

while (1) {

month = inputInt();

if (month == 0 || month > 12)

{

cout << "Введите корректно месяц!\n";

}

else return month;

}

}

int inputInt() { //Ввод целого числа

string s;

int i, a = 0, l;

bool flag = true;

while (flag)

{

cin >> s;

l = s.length();

for (i = 0; i < l; i++)

{

if ((s[i] >= (char)48) && (s[i] <= (char)57) || (s[i] == (char)45))

{

if ((i > 0) && (s[i] == 45))

{

flag = true;

cout << "Значение введено не верно!! Повтороите ввод!" << endl;

break;

}

else flag = false;

}

else

{

flag = true;

cout << "Значение введено не верно!! Повтороите ввод!" << endl;

break;

}

}

}

return stof(s);

}

double inputDouble() //ввод дробного числа

{

int i, number = 0, l;

bool flag = true;

string s;

while (flag)

{

cin >> s;

l = s.length();

for (i = 0; i < l; i++)

{

if ((s[i] >= (char)48) && (s[i] <= (char)57) || (s[i] == ','))

{

if (s[i] == ',')

{

number++;

if (number > 1)

{

flag = true;

cout << "Значение введено не верно!! Повтороите ввод!" << endl;

number = 0;

break;

}

else flag = false;

}

else flag = false;

}

else

{

flag = true;

cout << "Значение введено не верно!! Повтороите ввод!" << endl;

break;

}

}

}

return stof(s);

}

void addingNewEmployeeData(employees \*&employee, int &NumberOfEmployee)//дабавление сотрудника

{

string name, personnelNumber;

int numberOfHours, year, press, month;

double oneHourPayment;

ofstream fadd(FILE\_LIST, ios::app);

do {

cout << "Введите ФИО : ";

name = inputFIO();

name = enterStringWithSmallLetter(name);//проверка на ввод с маленькой буквы

cout << "Введите табельный номер: ";

cin >> personnelNumber;

cout << "Введите год: ";

year = inputYear();

cout << "Введите месяц: ";

month = inputMonth();

cout << "Введите количество проработанных часов: ";

numberOfHours = inputInt();

cout << "Введите оплату за один час: ";

oneHourPayment = inputDouble();

fadd << setw(25) << left << name //запись в файл

<< setw(20) << left << personnelNumber

<< setw(7) << left << year

<< setw(5) << left << month

<< setw(7) << left << numberOfHours

<< setw(5) << left << oneHourPayment << endl;

cout << "\nЖелаете ли вы подолжить добавление сотрудников.\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

} while (press != 50);

fadd.close();

NumberOfEmployee = listEmployeeData(employee);

}

void deleteEmployeeData(employees \*&employee, int &NumberOfEmployee)//удаление сотрудника

{

int s = -1, press;

string FIO;

cout << "Введите ФИО человека, которого вы хотите удалить из списка: ";

cin >> FIO;

FIO = enterStringWithSmallLetter(FIO); //проверка на ввод с маленькой буквы

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].FIO == FIO) {

s = i;

break;

}

}

if (s == -1)

cout << "Данные этого человека не найдены.\n";

else {

cout << "Вы действительно хотите удалить запись?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49)

{

for (int i = s; i < NumberOfEmployee - 1; i++)

employee[i] = employee[i + 1];

NumberOfEmployee--;

cout << "УДАЛЕНИЕ ПРОШЛО УСПЕШНО!!\n";

return;

}

else return;

}

}

void redactionFIO(employees \*&employee, int &NumberOfEmployee)//редактирование ФИО

{

string FIO, newFIO;

int press;

cout << "Введите ФИО человека, чьё ФИО вы хотите изменить.\n";

cin >> FIO;

FIO = enterStringWithSmallLetter(FIO);//проверка на ввод с маленькой буквы

//system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].FIO == FIO)

{

cout << "Введите новое ФИО: \n";

cin >> newFIO;

newFIO = enterStringWithSmallLetter(newFIO);//проверка на ввод с маленькой буквы

cout << "Вы действительно хотите изменить?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49)

{

employee[i].FIO = newFIO;

system("CLS");

cout << "РЕДАКТИРОВАНИЕ УСПЕШНО ВЫПОЛНЕНО.";

Sleep(2000); system("CLS");

return;

}

else return;

}

}

cout << "Извените! Данного человека нет в этом списке\n";

return;

}

void redactionYear(employees \*&employee, int &NumberOfEmployee)//редактирование года

{

string FIO;

int press;

int year;

cout << "Введите ФИО человека, чей год хотите изменить.\n";

cin >> FIO;

FIO = enterStringWithSmallLetter(FIO);//проверка на ввод с маленькой буквы

system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].FIO == FIO)

{

cout << "Введите новый год: \n";

year = inputYear();

cout << "Вы действительно хотите изменить?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49)

{

employee[i].year = year;

system("CLS");

cout << "РЕДАКТИРОВАНИЕ УСПЕШНО ВЫПОЛНЕНО.";

Sleep(2000); system("CLS");

return;

}

else return;

}

}

cout << "Извените! Данного человека нет в этом списке\n";

return;

}

void redactionNumberOfHours(employees \*&employee, int &NumberOfEmployee)//редактирование кол-ва часов

{

string FIO;

int press, numberOfHours;

cout << "Введите ФИО человека, чьё количество проработанных часов вы хотите изменить.\n";

cin >> FIO;

FIO = enterStringWithSmallLetter(FIO);

system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].FIO == FIO)

{

cout << "Введите новое количество пророботанных часов: \n";

numberOfHours = inputInt();

cout << "Вы действительно хотите изменить?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49)

{

employee[i].numberOfHours = numberOfHours;

system("CLS");

cout << "РЕДАКТИРОВАНИЕ УСПЕШНО ВЫПОЛНЕНО.";

Sleep(2000); system("CLS");

return;

}

else return;

}

}

cout << "Извените! Данного человека нет в этом списке\n";

return;

}

void redactionOneHourPayment(employees \*&employee, int &NumberOfEmployee) //редактирование платы за час

{

string FIO;

int press;

double oneHourPayment;

cout << "Введите ФИО человека, чью платы за час хотите изменить.\n";

cin >> FIO;

FIO = enterStringWithSmallLetter(FIO);

system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].FIO == FIO)

{

cout << "Введите новую плату: \n";

oneHourPayment = inputDouble();

cout << "Вы действительно хотите изменить?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49)

{

employee[i].oneHourPayment = oneHourPayment;

system("CLS");

cout << "РЕДАКТИРОВАНИЕ УСПЕШНО ВЫПОЛНЕНО.";

Sleep(2000); system("CLS");

return;

}

else return;

}

}

cout << "Извените! Данного человека нет в этом списке\n";

return;

}

void redactionMonth(employees \*&employee, int &NumberOfEmployee)//редактирование месяца

{

string FIO;

int press, month;

cout << "Введите ФИО человека, чей месяц хотите изменить.\n";

cin >> FIO;

FIO = enterStringWithSmallLetter(FIO);//проверка на ввод с маленькой буквы

system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].FIO == FIO)

{

cout << "Введите новый месяц: \n";

month=inputMonth();

cout << "Вы действительно хотите изменить?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49)

{

employee[i].month = month;

system("CLS");

cout << "РЕДАКТИРОВАНИЕ УСПЕШНО ВЫПОЛНЕНО.";

Sleep(2000); system("CLS");

return;

}

else return;

}

}

cout << "Извените! Данного человека нет в этом списке\n";

return;

}

void redactionPersonnelNumber(employees \*&employee, int &NumberOfEmployee) {

string FIO;

int press, personnelNumber;

cout << "Введите ФИО человека, чей табельный номер хотите изменить.\n";

cin >> FIO;

FIO = enterStringWithSmallLetter(FIO);//проверка на ввод с маленькой буквы

system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].FIO == FIO)

{

cout << "Введите новый табельный номер: \n";

cin >> personnelNumber;

cout << "Вы действительно хотите изменить?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49)

{

employee[i].personnelNumber = personnelNumber;

system("CLS");

cout << "РЕДАКТИРОВАНИЕ УСПЕШНО ВЫПОЛНЕНО.";

Sleep(2000); system("CLS");

return;

}

else return;

}

}

cout << "Извените! Данного человека нет в этом списке\n";

return;

}

void individualTask(employees \*&employee, int &NumberOfEmployee)//индивидуальное занятие

{

int month, year;

double pay, sum=0;

cout << "Введите месяц: ";

month=inputMonth();

cout << "Введите год: ";

year=inputYear();

for (int i = 0; i < NumberOfEmployee; i++)

{

if ((employee[i].month == month) && (employee[i].year == year))

{

sum++;

if (employee[i].numberOfHours > 144) {

pay = (employee[i].oneHourPayment \* 144 + (employee[i].numberOfHours - 144) \* 2 \* employee[i].oneHourPayment)\* 0.88;

cout << employee[i].FIO << "\t" << pay << endl;

}

else {

pay = employee[i].oneHourPayment\*employee[i].numberOfHours\* 0.88;

cout << employee[i].FIO << "\t" << pay << endl;

}

}

pay = 0;

}

if (sum == 0)

{

cout << "Никто не работал в это месяце!\n";

Sleep(2000); system("CLS");

return;

}

}

void menuRedactionEmployee(employees \*&employee, int &NumberOfEmployee)//меню редактироавания сотрудников

{

int press;

do {

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1.ФИО. . \*\n"

<< "\t\t\t\t\t\* 2.Год. \*\n"

<< "\t\t\t\t\t\* 3.Оплата за час. \*\n"

<< "\t\t\t\t\t\* 4.Количество проработанных часов \*\n"

<< "\t\t\t\t\t\* 5.Месяц. \*\n"

<< "\t\t\t\t\t\* 0.Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 53));

system("CLS");

switch (press)

{

case 49:

redactionFIO(employee, NumberOfEmployee); break;//по ФИО

case 50:

redactionYear(employee, NumberOfEmployee); break;// по году

case 51:

redactionOneHourPayment(employee, NumberOfEmployee); break;//по плате

case 52:

redactionNumberOfHours(employee, NumberOfEmployee); break; //по кол-ву часов

case 53:

redactionMonth(employee, NumberOfEmployee); break;//по месяцу

case 48: system("CLS"); return;

}

} while (press != 53);

}

void employeesProcessing(employees \*&employee, int &NumberOfEmployee) //обработка данных о сотрудниках

{

int press;

do {

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1. Поиск \*\n"

<< "\t\t\t\t\t\* 2. Сортировка \*\n"

<< "\t\t\t\t\t\* 3. Индивидуальное задание \*\n"

<< "\t\t\t\t\t\* 0. Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 51));

system("CLS");

switch (press)

{

case 48: return;

case 49:

menuSearch(employee, NumberOfEmployee); break;//меню поиска

case 50:

menuSorting(employee, NumberOfEmployee); break;//меню сортировки

case 51:

individualTask(employee, NumberOfEmployee); break;//индивид. задание

}

} while (press != 48);

}

void editDataAboutEmployee(employees \*&employee, int &NumberOfEmployee) //редактирование данных о сотрудниках

{

int press;

do {

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1. Просмотр. \*\n"

<< "\t\t\t\t\t\* 2. Редактирование. \*\n"

<< "\t\t\t\t\t\* 3. Добавление. \*\n"

<< "\t\t\t\t\t\* 4. Удаление. \*\n"

<< "\t\t\t\t\t\* 0. Назад. \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 52));

system("CLS");

switch (press)

{

case 48: return;

case 49:

ViewEmployee(employee, NumberOfEmployee); break;//просмотр

case 50:

menuRedactionEmployee(employee, NumberOfEmployee);//меня редактирования сотрудниках

break;

case 51:

addingNewEmployeeData(employee, NumberOfEmployee); break;//дабавление сотрудниках

case 52:

deleteEmployeeData(employee, NumberOfEmployee); break;//удаление сотрудников

}

if (press != 49 && press != 48) {

InputAllInformationAboutEmployeeDataInFile(employee, NumberOfEmployee);//перезапись в файл

}

} while (press != 48);

}

void activationAccounts(users \*&user, int numberOfUsers)//активация аккаунтов

{

int press, t = 0;

for (int i = 0; i < numberOfUsers; i++) {

if (user[i].code == 0) {

t++;

cout << user[i].login << " желает зарегистрироваться\n"

<< "Активировать аккаунт?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49) user[i].code = 1;

else user[i].code = 4;

system("CLS");

}

if (user[i].code == 3) {

t++;

cout << user[i].login << " желает зарегистрироваться как администратор.\n"

<< "Активировать аккаунт?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49) user[i].code = 2;

else user[i].code = 4;

system("CLS");

}

}

if (!t) cout << "Никто не зарегистрировался\n";

return;

}

void redactionLogin(users \*&user, int numberOfUsers)//редактирование логин

{

string login, login1;

int press, s = 0;

bool flag = true;

cout << "Введите логин, который вы хотите изменить: ";

cin >> login;

for (int i = 0; i < numberOfUsers; i++) {

if (user[i].login == login) {

s++;

do {

cout << "На какой логин вы хотите заменить: ";

login1 = inputFIO();

for (int i = 0; i < numberOfUsers; i++) {

if (login1 == user[i].login) {

cout << "Такой логин уже существует.\n";

flag = false;

break;

}

else flag = true;

}

} while (flag == false);

cout << "Вы уверены, что хотите изменить логин.\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49) {

user[i].login = login1;

system("CLS");

cout << "РЕДАКТИРОВАНИЕ УСПЕШНО ВЫПОЛНЕНО.";

Sleep(2000); system("CLS");

return;

}

else return;

}

}

if (!s) {

cout << "Логин не найден.\n";

return;

}

}

void redactionPassword(users \*&user, int numberOfUsers) //Замена старого пароля на новый

{

int press, s = 0;

string login, password1;

cout << "Введите логин пользователя, чей пароль хотите изменить.";

cin >> login;

for (int i = 0; i < numberOfUsers; i++) {

if (user[i].login == login) {

s++;

password1 = inputPassword();

cout << "\nВы уверены, что хотите изменить пароль.\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49) {

user[i].password = password1;

system("CLS");

InputAllInformationAboutAccountsInFile(user, numberOfUsers);

cout << "РЕДАКТИРОВАНИЕ УСПЕШНО ВЫПОЛНЕНО.";

Sleep(2000); system("CLS");

return;

}

else return;

}

}

if (!s) {

cout << "Логин не найден.\n";

return;

}

}

void addingNewUsers(users \*&user, int &numberOfUsers)//добавление нового пользователя(администратором)

{

string login, password;

int kod = 1, press;

ofstream fadd(FILE\_ACCOUNT, ios::app);

do {

system("CLS");

login = inputLogin(user);

password = inputPassword();

fadd << setw(20) << left << login << setw(30) << password << setw(5) << left << kod << endl;

cout << "\nЖелаете ли вы продолжить добавление аккаунтов\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

} while (press != 50);

fadd.close();

cout << "ДОБАВЛЕНИЕ ВЫПОЛНЕНО УСПЕШНО!!";

Sleep(2000);

numberOfUsers = listUsers(user);

system("CLS");

}

void viewLoginsPassword(users \*user, int numberOfUsers)//вывод логинов и паролей

{

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

<< "| Логин | Пароль | Код |\n"

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

for (int i = 0; i < numberOfUsers; i++)

{

cout << "| " << setw(20) << left << user[i].login << " | " << setw(30) << left << user[i].password << " | "

<< setw(5) << left << user[i].code << " | " << endl;

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

}

}

void redactionLoginPassword(users \*&user, int numberOfUsers)//редактирование логина и пароля

{

int press;

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

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1. Редактирование пароля. \*\n"

<< "\t\t\t\t\t\* 2. Редактирование логина. \*\n"

<< "\t\t\t\t\t\* 0. Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 50));

system("CLS");

switch (press)

{

case 50: redactionLogin(user, numberOfUsers); break;

case 49: redactionPassword(user, numberOfUsers); break;

case 48: break;

}

}

void deleteAccounts(users \*&user, int &numberOfUsers)//удаление аккаунтов пользователей

{

int s = -1, press;

string login;

cout << "Введите логин аккаунта, который вы хочите удалить: ";

cin >> login;

for (int i = 0; i < numberOfUsers; i++) {

if (user[i].login == login) {

if (user[i].code == 2) s = -2;

else s = i;

break;

}

}

//system("CLS");

if (s == -2)

{

cout << "Это аккаунт администратора. Вы не можите его удалить!!\n";

return;

}

if (s == -1)

cout << "Данный аккаунт не найден.\n";

else {

cout << "Вы действительно хотите удалить запись?\n1.Да\n2.Нет\n";

do {

press = \_getch();

} while (press != 49 && press != 50);

if (press == 49)

{

for (int i = s; i < numberOfUsers - 1; i++)

user[i] = user[i + 1];

numberOfUsers--;

cout << "УДАЛЕНИЕ АККАУНТА ВЫПОЛНЕНО УСПЕШНО!\n";

viewLoginsPassword(user, numberOfUsers);

return;

}

else return;

}

}

void menuWorkWithDataEmployee(employees \*&employee, int &NumberOfEmployee)//Меню для работы с данными(администратор)

{

int press;

do {

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1. Редактирование \*\n"

<< "\t\t\t\t\t\* 2. Обработка \*\n"

<< "\t\t\t\t\t\* 0. Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 50));

system("CLS");

switch (press)

{

case 48: return;

case 50:

employeesProcessing(employee, NumberOfEmployee); break;

case 49:

editDataAboutEmployee(employee, NumberOfEmployee); break;

}

} while (press != 48);

}

void menuWorkWithAccounts(users \*&user, int &numberOfUsers)//Меню для управления уч. записями(администратор)

{

int press;

do {

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1. Просмотр. \*\n"

<< "\t\t\t\t\t\* 2. Активация аккаунтов. \*\n"

<< "\t\t\t\t\t\* 3. Редактирование. \*\n"

<< "\t\t\t\t\t\* 4. Добавить. \*\n"

<< "\t\t\t\t\t\* 5. Удаление. \*\n"

<< "\t\t\t\t\t\* 0. Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 53));

system("CLS");

switch (press)

{

case 49:

viewLoginsPassword(user, numberOfUsers); break;

case 50:

activationAccounts(user, numberOfUsers); break;

case 51:

redactionLoginPassword(user, numberOfUsers); break;

case 52:

addingNewUsers(user, numberOfUsers); break;

case 53:

deleteAccounts(user, numberOfUsers); break;

case 48: return;

}

if (press != 48 && press != 49)

{

InputAllInformationAboutAccountsInFile(user, numberOfUsers);

}

} while (press != 48);

}

void menuAdmin(users \*&user, employees \*&employee)//Меню администратора

{

int press, numberOfUsers = listUsers(user);

int NumberOfEmployee = listEmployeeData(employee);

do {

cout << "\n\n\n\n\t\t\t\t\t\t МЕНЮ АДМИНИСТРАТОРА";

cout << "\n\n";

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1. Работа с учётными записями \*\n"

<< "\t\t\t\t\t\* 2. Работа с данными \*\n"

<< "\t\t\t\t\t\* 0. Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 50));

system("CLS");

switch (press)

{

case 49:

menuWorkWithAccounts(user, numberOfUsers);

break;

case 50:

menuWorkWithDataEmployee(employee, NumberOfEmployee);

break;

case 48: return;

}

} while (press != 48);

}

**Users.h**

#pragma once

#include"Admin.h"

using namespace std;

void sortingFIO(employees \*employee, int NumberOfEmployee);//сортировка по ФИО

void sortingNumberOfHours(employees \*employee, int NumberOfEmployee);//сорт. по кол-ву проработанных часов

void ViewSearch(employees \*, int k, int);//просмотр при поиске

void sortingOneHourPayment(employees \*employee, int NumberOfEmployee);//сортировка по оплате

void sortingYear(employees \*employee, int NumberOfEmployee);//сортировка по году

void ViewEmployee(employees \*employee, int NumberOfEmployee);//просмотр сотрудников

void searchFIO(employees \*employee, int NumberOfEmployee);//поиск по ФИО

void searchYear(employees \*employee, int NumberOfEmployee);//поиск по году

void searchOneHourPayment(employees \*employee, int NumberOfEmployee);//поиск по оплате

void searchMonth(employees \*employee, int NumberOfEmployee);//поиск по месяцу

void menuSearch(employees \*employee, int NumberOfEmployee);//меню поиска

void menuSorting(employees \*employee, int NumberOfEmployee);//меню сортировки

void menuUsers(employees \*&employee);//меню пользователя

string enterStringWithSmallLetter(string FIO); //ввод строки с маленькой буквы

**Users.cpp**

#include<iostream>

#include<conio.h>

#include<iomanip>

#include<string>

#include"Admin.h"

#include"File.h"

using namespace std;

void ViewEmployee(employees \*employee, int NumberOfEmployee)//просмотр сотрудников

{

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

<< " № | ФИО | Табельный номер | Год | Месяц |Кол.часов| Плата |\n"

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

for (int i = 0; i < NumberOfEmployee; i++) {

cout << " " << i + 1 << " | " << setw(16) << left << employee[i].FIO << " | "

<< setw(20) << left << employee[i].personnelNumber << " | "

<< setw(10) << left << employee[i].year << " | "

<< setw(7) << left << employee[i].month << " | "

<< setw(7) << left << employee[i].numberOfHours << " | "

<< setw(7) << left << employee[i].oneHourPayment << " |" << endl;

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

}

}

void ViewSearch(employees \*employee, int k, int i)//просмотр при поиске

{

if (k == 1) {

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

<< "| ФИО | Табельный номер | Год | Месяц |Кол.часов| Плата |\n"

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

cout << "| " << setw(16) << left << employee[i].FIO << " | "

<< setw(20) << left << employee[i].personnelNumber << " | "

<< setw(10) << left << employee[i].year << " | "

<< setw(7) << left << employee[i].month << " | "

<< setw(7) << left << employee[i].numberOfHours << " | "

<< setw(7) << left << employee[i].oneHourPayment << " |" << endl;

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

}

else {

cout << "| " << setw(16) << left << employee[i].FIO << " | "

<< setw(10) << left << employee[i].year << " | "

<< setw(7) << left << employee[i].month << " | "

<< setw(7) << left << employee[i].numberOfHours << " | "

<< setw(7) << left << employee[i].oneHourPayment << " |" << endl;

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

}

}

string enterStringWithSmallLetter(string FIO)//ввод строки с маленькой буквы

{

int a;

if (FIO[0] >= (char)97 && FIO[0] <= 122) {

a = (int)FIO[0] - 32;

FIO[0] = (char)a;

}

return FIO;

}

void searchFIO(employees \*employee, int NumberOfEmployee)//поиск по ФИО

{

string FIO;

int k = 0;

cout << "Введите ФИО человека: ";

cin >> FIO;

FIO = enterStringWithSmallLetter(FIO);

system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].FIO == FIO)

{

k++;

ViewSearch(employee, k, i);

return;

}

}

cout << "Ничего не найдено!\n";

return;

}

void searchYear(employees \*employee, int NumberOfEmployee)//поиск по году

{

int year;

int k = 0;

cout << "Введите год, за который вы хотите найти информацию: \n";

year = inputYear();

system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].year == year)

{

k++;

ViewSearch(employee, k, i);

}

}

if (k == 0) {

cout << "Ничего не найдено!\n";

return;

}

else return;

}

void searchOneHourPayment(employees \*employee, int NumberOfEmployee)//поиск по проработанным часам

{

int oneHourPayment;

int k = 0;

cout << "Введите количество проработанных часов: \n";

oneHourPayment = inputInt();

system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].oneHourPayment <= oneHourPayment)

{

k++;

ViewSearch(employee, k, i);

}

}

if (k == 0) {

cout << "Извините! Все люди проработали больше чем "<< oneHourPayment <<" часа \n";

return;

}

else return;

}

void searchMonth(employees \*employee, int NumberOfEmployee)//поиск по месяцу

{

int month;

int k = 0;

cout << "Введите месяц: ";

month=inputMonth();

//system("CLS");

for (int i = 0; i < NumberOfEmployee; i++) {

if (employee[i].month == month)

{

k++;

ViewSearch(employee, k, i);

}

}

if (k == 0) {

cout << "В этом месяце никто не болел!\n";

}

}

void sortingFIO(employees \*employee, int NumberOfEmployee)//сортировка по ФИО

{

employees s;

for (int i = 0; i < NumberOfEmployee - 1; i++)

for (int j = i; j < NumberOfEmployee; j++)

if (employee[i].FIO > employee[j].FIO)

{

s = employee[i];

employee[i] = employee[j];

employee[j] = s;

}

ViewEmployee(employee, NumberOfEmployee);

}

void sortingYear(employees \*employee, int NumberOfEmployee)//сортировка по году

{

employees s;

for (int i = 0; i < NumberOfEmployee - 1; i++)

for (int j = i; j < NumberOfEmployee; j++)

if (employee[i].year > employee[j].year)

{

s = employee[i];

employee[i] = employee[j];

employee[j] = s;

}

ViewEmployee(employee, NumberOfEmployee);

}

void sortingOneHourPayment(employees \*employee, int NumberOfEmployee)//сортировка пооплате за за час

{

employees s;

for (int i = 0; i < NumberOfEmployee - 1; i++)

for (int j = i; j < NumberOfEmployee; j++)

if (employee[i].oneHourPayment > employee[j].oneHourPayment)

{

s = employee[i];

employee[i] = employee[j];

employee[j] = s;

}

ViewEmployee(employee, NumberOfEmployee);

}

void sortingNumberOfHours(employees \*employee, int NumberOfEmployee)//сорт. по кол-ву часов

{

employees s;

for (int i = 0; i < NumberOfEmployee - 1; i++)

for (int j = i; j < NumberOfEmployee; j++)

if (employee[i].numberOfHours > employee[j].numberOfHours)

{

s = employee[i];

employee[i] = employee[j];

employee[j] = s;

}

ViewEmployee(employee, NumberOfEmployee);

}

void menuSearch(employees \*employee, int NumberOfEmployee)//меня поиска

{

int press;

do {

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1.По ФИО. \*\n"

<< "\t\t\t\t\t\* 2.По году. \*\n"

<< "\t\t\t\t\t\* 3.По оплате за один час. \*\n"

<< "\t\t\t\t\t\* 4.По месяцу. \*\n"

<< "\t\t\t\t\t\* 0.Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 52));

system("CLS");

switch (press)

{

case 49:

searchFIO(employee, NumberOfEmployee); break;

case 50:

searchYear(employee, NumberOfEmployee); break;

case 51:

searchOneHourPayment(employee, NumberOfEmployee); break;

case 52:

searchMonth(employee, NumberOfEmployee); break;

case 48: system("CLS"); return;

}

} while (press != 48);

}

void menuSorting(employees \*employee, int NumberOfEmployee)//меню сортировки

{

int press;

do {

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1.По ФИО. \*\n"

<< "\t\t\t\t\t\* 2.По году. \*\n"

<< "\t\t\t\t\t\* 3.По оплате за час. \*\n"

<< "\t\t\t\t\t\* 4.По количеству проработанный часов \*\n"

<< "\t\t\t\t\t\* 0.Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 52));

system("CLS");

switch (press)

{

case 49:sortingFIO(employee, NumberOfEmployee); break;

case 50:

sortingYear(employee, NumberOfEmployee); break;

case 51:

sortingOneHourPayment(employee, NumberOfEmployee); break;

case 52:

sortingNumberOfHours(employee, NumberOfEmployee);

break;

case 48: system("CLS"); return;

}

} while (press != 48);

}

void menuUsers(employees \*&employee)//меню пользователя

{

int press, NumberOfEmployee;

cout << "\n\n\n\n\t\t\t\t\t\t МЕНЮ ПОЛЬЗОВАТЕЛЯ\n\n";

do {

cout << "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

<< "\t\t\t\t\t\* 1.Просмотр всех сотрудников. \*\n"

<< "\t\t\t\t\t\* 2.Индивидуальное задание. \*\n"

<< "\t\t\t\t\t\* 3.Поиск. \*\n"

<< "\t\t\t\t\t\* 4.Сортировка. \*\n"

<< "\t\t\t\t\t\* 0.Назад \*\n"

<< "\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

do {

press = \_getch();

} while (!(press >= 48 && press <= 53));

NumberOfEmployee = listEmployeeData(employee);

system("CLS");

switch (press)

{

case 49:

ViewEmployee(employee, NumberOfEmployee); break;

case 50:

individualTask(employee, NumberOfEmployee); break;

case 51:

menuSearch(employee, NumberOfEmployee); break;

case 52:

menuSorting(employee, NumberOfEmployee); break;

case 48:system("CLS"); return;

}

} while (press != 54);

}

**File.h**

#pragma once

#include"Admin.h"

const string FILE\_ACCOUNT = "user.txt";

const string FILE\_LIST = "employee.txt";

using namespace std;

int listEmployeeData(employees \*&employee);//вывод информации из файла в массив с Сотрудниками

int listUsers(users \*&user);//Чтение информации из файла с логином и паролем

void addToFileOneAccount(string login, string password, int code);//добавление в конец файла одной записи

void InputAllInformationAboutAccountsInFile(users \*user, int numberOfUsers);//перезапись файла с логинами и паролями

void InputAllInformationAboutEmployeeDataInFile(employees \*employee, int NumberOfEmployee);//перезапись файла с сотрудниками

**File.cpp**

#include<iostream>

#include<string>

#include<fstream>

#include<iomanip>

#include"Admin.h"

#include"File.h"

using namespace std;

int listEmployeeData(employees \*&employee)//вывод информации из файла в массив с сотрудниками

{

ifstream fin;

fin.open(FILE\_LIST, ofstream::in);

char str[100];

int kol = 0;

for (; fin.getline(str, 100); kol++);

fin.close();

fin.open(FILE\_LIST, ifstream::in);

employee = new employees[kol];

for (int i = 0; i < kol; i++)

{

fin >> employee[i].FIO

>> employee[i].personnelNumber

>> employee[i].year

>> employee[i].month

>> employee[i].numberOfHours

>> employee[i].oneHourPayment;

}

fin.close();

return kol;

}

int listUsers(users \*&user) //Чтение информации из файла с логином и паролем

{

ifstream fin;

fin.open(FILE\_ACCOUNT, ofstream::in);

char str[90];

int kol = 0;

for (; fin.getline(str, 90); kol++);

fin.close();

fin.open(FILE\_ACCOUNT, ifstream::in);

user = new users[kol];

for (int i = 0; i < kol; i++)

{

fin >> user[i].login;

fin >> user[i].password;

fin >> user[i].code;

}

fin.close();

return kol;

}

void addToFileOneAccount(string login, string password, int code)//добавление в конец файла одной записи

{

ofstream fadd(FILE\_ACCOUNT, ios::app);

fadd << setw(20) << left << login << setw(30) << password << setw(5) << left << code << endl;

fadd.close();

}

void InputAllInformationAboutAccountsInFile(users \*user, int numberOfUsers)//перезапись файла с логинами и паролями

{

ofstream fadd(FILE\_ACCOUNT);

for (int i = 0; i < numberOfUsers; i++) {

fadd << setw(20) << left << user[i].login << setw(30) << user[i].password << setw(5) << left << user[i].code << endl;

}

fadd.close();

}

void InputAllInformationAboutEmployeeDataInFile(employees \*employee, int NumberOfEmployee)//перезапись файла с бсотрудниками

{

ofstream fout(FILE\_LIST);

for (int i = 0; i < NumberOfEmployee; i++) {

fout << setw(25) << left << employee[i].FIO

<< setw(20) << left << employee[i].personnelNumber

<< setw(7) << left << employee[i].year

<< setw(5) << left << employee[i].month

<< setw(7) << left << employee[i].numberOfHours

<< setw(5) << left << employee[i].oneHourPayment << endl;

}

fout.close();

}