// Вариант 11

// Разработать информационно справочную систему для работы с базой данных. Входные данные:

// База данных состоит из записей. (код блюда, наименование, ингредиенты, время приготовления, номер диетического стола, калории, в отдельном файле справочная информация о диетических столах)

// Обеспечить:

// - введение данных в оперативную память в двунаправленный список:

// а) из текстового файла;

// б) с клавиатуры;

// - просмотр содержимого базы данных;

// - редактирование базы данных:

// а) удаление записи;

// б) добавление записи;

// - выведение данных:

// а) в текстовый файл;

// б) на экран;

// - поиск данных по коду блюда;

// - сортировка данных по калорийности;

// - получение справочной информации о диетическом столе;

// fout.open("test.txt", ofstream::app); дописывает данные в конец файла без полного изменения ДОБАВИТЬ В ИЗМЕНЕНИЕ ДБ;

// 1. Скрыть пароль при вводе в консоли

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <Windows.h>

#include <string>

#include <vector>

#include <algorithm>

#include <cctype>

using namespace std;

struct Dish {

int code;

string name;

string ingredients;

int time;

int numberOfTable;

int calories;

};

struct Table {

int code;

string name;

};

struct Worker {

string secondName, name, phoneNumber, \_login, \_password;

int id, \_accessLevel;

};

int createDefaultDishes() {

ofstream fout;

fout.open("DishesInfo.dat");

if (!fout.is\_open()) {

printf("Default dishes initialization is failed.\n");

return EXIT\_FAILURE;

}

fout << "0 name ingredients 0 0 0\n1 name ingredients 0 0 0\n2 name ingredients 0 0 0\n";

fout.close();

printf("Default dishes settings upload is successful.\n");

return EXIT\_SUCCESS;

}

vector<Dish> ShowDishesDatabase(vector<Dish>& Dishes) {

printf("Список блюд:\nКод Название Ингрeдиенты Время приготовления Номер стола Калории\n");

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

printf("%d\t%s\t%s\t%d\t%d\t%d\t", Dishes[i].code, Dishes[i].name.c\_str(), Dishes[i].ingredients.c\_str(), Dishes[i].time, Dishes[i].numberOfTable, Dishes[i].calories);

}

printf("---------------------------------------\n");

return(Dishes);

}

vector<Dish> ChangeDishesDatabase(vector<Dish>& Dishes) {

int func = 0;;

printf("Введите номер желаемой функции 1 - удалить запись, 2 - добавить запись\n");

scanf("%d", &func);

switch (func) {

case 1: {

int searchId = 0;

ShowDishesDatabase(Dishes);

printf("Выберите code заказа который хотите удалить\n");

scanf("%d", &searchId);

int counter = 0;

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

if (searchId == Dishes[i].code) {

counter += 1;

printf("Вы действительно хотите удалить данное блюдо? 1 - Да. 0 - Нет\n");

int confirm = 0;

scanf("%d", &confirm);

if (confirm == 1) {

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

printf("Вы успешно удалили элемент.\nВ данный момент база данных заказов выглядит так:\n");

ShowDishesDatabase(Dishes);

} else {

printf("Вы отклонили удаление элемента\n");

}

break;

}

}

if (counter == 0) printf("Заказ с таким Id не был найден\n");

break;

}

case 2: {

Dish temp;

int lastId;

lastId = Dishes[Dishes.size() - 1].code;

printf("Введите данные блюда в порядке Имя Ингридиенты Время\_приготовления Номер\_стола Калории\n");

cin >> temp.name;

cin >> temp.ingredients;

cin >> temp.time;

cin >> temp.numberOfTable;

cin >> temp.calories;

temp.code = lastId + 1;

Dishes.push\_back(temp);

printf("Вы успешно добавили заказ в базу данных. Id добавленного вами заказа - %d\n", Dishes[Dishes.size() - 1].code);

printf("---------------------------------------\n");

ShowDishesDatabase(Dishes);

break;

}

default: "Вы ввели неверное значение, перенаправляю вас в главное меню."; return Dishes;

}

return Dishes;

}

vector <Dish> ShowSerialDish(vector <Dish>& Dishes) {

int searchId = 0;

printf("Введите code искомого блюда\n");

scanf("%d", &searchId);

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

if (searchId == Dishes[i].code) {

printf("Искомый вами заказ:\nКод продукта - %d, название - %s, ингрeдиенты - %s, время приготовления - %d, номер стола - %d, калории - %d\n",

Dishes[i].code, Dishes[i].name.c\_str(), Dishes[i].ingredients.c\_str(), Dishes[i].time, Dishes[i].numberOfTable, Dishes[i].calories);

return(Dishes);

}

}

printf("Искомый вами заказ не найден");

return Dishes;

}

vector <Dish> SortDishesDataBaseByCalory(vector <Dish>& Dishes) {

int listLength = Dishes.size();

while (listLength--) {

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

if (Dishes[i].calories > Dishes[i + 1].calories) {

swap(Dishes[i], Dishes[i + 1]);

}

}

}

return Dishes;

}

int CreateDefaultTables() {

ofstream fout;

fout.open("TablesInfo.dat");

if (!fout.is\_open()) {

printf("Default Tables initialization failed.\n");

return EXIT\_FAILURE;

}

fout << "0 name\n1 name\n2 name\n";

fout.close();

printf("Default Tables settings upload success.\n");

return EXIT\_SUCCESS;

}

vector <Table> ShowTablesDatabase(vector <Table>& Tables) {

printf("Список столов:\nCode Name\n");

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

printf("%d\t%s\n", Tables[i].code, Tables[i].name.c\_str());

}

printf("---------------------------------------\n");

return(Tables);

}

vector <Table> addTable(vector <Table>& Tables) {

ofstream fout;

Table temp;

int code;

string name;

printf("Введите данные в указанном порядке: Название\n");

temp.code = Tables.size();

cin >> temp.name;

Tables.push\_back(temp);

fout.open("TablesInfo.dat", ofstream::app);

code = temp.code; fout << code << " ";

name = temp.name; fout << name << " " << '\n';

printf("Вы успешно добавили стол %s\n", temp.name.c\_str());

return Tables;

}

vector <Table> removeTable(vector <Table>& Tables) {

int searchId = 0;

int counter = 0;

ShowTablesDatabase(Tables);

printf("Введите code стола которого хотите удалить из системы\n");

scanf("%d", &searchId);

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

if (searchId == Tables[i].code) {

int confirm = 0;

counter += 1;

printf("Вы действительно хотите удалить стол %d из системы? 1 - Да. 0 - Нет\n", Tables[i].code);

scanf("%d", &confirm);

if (confirm == 1) {

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

printf("Вы успешно удалили элемент. Теперь база данных выглядит так:\n");

ShowTablesDatabase(Tables);

}

else {

printf("Вы отклонили удаление элемента\n");

}

break;

}

}

if (counter == 0) printf("работник с указанным Id не найден");

return Tables;

}

int CreateDefaultWorkers() {

ofstream fout;

fout.open("UsersData.dat");

if (!fout.is\_open()) {

printf("Default Workers initialization failed.\n");

return EXIT\_FAILURE;

}

fout << "0 Ivanov Ivan 81231231212 ivivan vanek 0\n1 Vladov Vlad 89567562354 vlvlad vladosik 1\n2 Lana Lana 81251251212 admin admin 2\n";

fout.close();

printf("Default Workers settings upload success.\n");

return EXIT\_SUCCESS;

}

vector <Worker> ShowWorkersDataBase(vector <Worker>& Workers) {

printf("Список работников:\nId SecondName Name Numberphone\n");

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

printf("%d\t%s\t%s\t%s\n", Workers[i].id, Workers[i].secondName.c\_str(), Workers[i].name.c\_str(), Workers[i].phoneNumber.c\_str());

}

printf("---------------------------------------\n");

return Workers;

}

vector <Worker> AddWorker(vector <Worker>& Workers) {

ofstream fout;

Worker temp;

int Int;

string str;

printf("Введите данные в указанном порядке: Фамилия, Имя, Номер Телефона, Логин, Пароль, Уровень доступа\n");

temp.id = Workers.size();

cin >> temp.secondName;

cin >> temp.name;

cin >> temp.phoneNumber;

cin >> temp.\_login;

cin >> temp.\_password;

cin >> temp.\_accessLevel;

Workers.push\_back(temp);

fout.open("UsersData.dat", ofstream::app);

Int = temp.id; fout << Int << " ";

str = temp.secondName; fout << str << " ";

str = temp.secondName; fout << str << " ";

str = temp.phoneNumber; fout << str << " ";

str = temp.\_login; fout << str << " ";

Int = temp.\_accessLevel; fout << Int << "\n";

printf("Вы успешно добавили работника %s %s\n", temp.secondName.c\_str(), temp.name.c\_str());

return Workers;

}

vector <Worker> RemoveWorker(vector <Worker>& Workers) {

int searchId = 0;

int counter = 0;

ShowWorkersDataBase(Workers);

printf("Введите Id работника которого хотите удалить из системы\n");

scanf("%d", &searchId);

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

if (searchId == Workers[i].id) {

int confirm = 0;

counter += 1;

printf("Вы действительно хотите удалить работника %d из системы? 1 - Да. 0 - Нет\n", Workers[i].id);

scanf("%d", &confirm);

if (confirm == 1) {

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

printf("Вы успешно удалили элемент. Теперь база данных выглядит так:\n");

ShowWorkersDataBase(Workers);

} else {

printf("Вы отклонили удаление элемента\n");

}

break;

}

}

if (counter == 0) printf("работник с указанным Id не найден");

return Workers;

}

vector <Worker> GetWorkerInfo(vector <Worker>& Workers) {

int searchId = 0;

int counter = 0;

printf("Введите Id искомого работника\n");

scanf("%d", &searchId);

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

if (searchId == Workers[i].id) {

printf("Искомый вами работник:\nId - %d, Фамилия - %s, Имя - %s, Номер телефона - %s, Уровень доступа - %d\n", Workers[i].id, Workers[i].secondName.c\_str(), Workers[i].name.c\_str(), Workers[i].phoneNumber.c\_str(), Workers[i].\_accessLevel);

counter += 1;

break;

}

}

if (counter == 0) printf("Работник с указанным Id не найден. Он уволен либо не присутствует в базе данных.");

return(Workers);

}

int AccessLevel0Tools(vector <Dish>& Dishes, vector <Worker>& Workers, vector<Table>& Tables) {

int function = -1;

while (function != 0) {

printf("Список функций:\n 1 - Показать список блюд, 2 - Отсортировать список блюд. 3 - Изменить список блюд, 4 - Показать данные блюда по code, 0 - Выход из системы\n");

scanf("%d", &function);

switch (function) {

case 1: ShowDishesDatabase(Dishes); break;

case 2: SortDishesDataBaseByCalory(Dishes); break;

case 3: ChangeDishesDatabase(Dishes); break;

case 4: ShowSerialDish(Dishes); break;

case 0: break;

}

}

printf("Вы успешно вышли из системы\n");

return EXIT\_SUCCESS;

}

int AccessLevel1Tools(vector <Dish>& Dishes, vector <Worker>& Workers, vector<Table>& Tables) {

int function = -1;

while (function != 0) {

printf("Список функций:\n 1 - Показать список заказов, 2 - Отсортировать список заказов. 3 - Изменить список заказов, 4 - Показать данные заказа по Id, 5 - Взамодействия с работниками, 0 - Выход из системы\n");

scanf("%d", &function);

switch (function) {

case 1: ShowDishesDatabase(Dishes); break;

case 2: SortDishesDataBaseByCalory(Dishes); break;

case 3: ChangeDishesDatabase(Dishes); break;

case 4: ShowSerialDish(Dishes); break;

case 5: {

int workChoose = 0;

printf("Список функций с работниками:\n1 - Получить список работников, 2 - Получить информацию о работнике по Id, 0 - Выход\n");

scanf("%d", &workChoose);

switch (workChoose) {

case 1: ShowWorkersDataBase(Workers); break;

case 2: GetWorkerInfo(Workers); break;

case 0: break;

}

}

}

}

printf("Вы успешно вышли из системы\n");

return EXIT\_SUCCESS;

}

int AccessLevel2Tools(vector <Dish>& Dishes, vector <Worker>& Workers, vector<Table>& Tables) {

int Function = -1;

while (Function != 0) {

printf("Список функций:\n 1 - Показать список заказов, 2 - Отсортировать список заказов. 3 - Изменить список заказов, 4 - Показать данные заказа по Id, 5 - Взамодействия с работниками, 0 - Выход из системы\n");

scanf("%d", &Function);

switch (Function) {

case 1: ShowDishesDatabase(Dishes); break;

case 2: SortDishesDataBaseByCalory(Dishes); break;

case 3: ChangeDishesDatabase(Dishes); break;

case 4: ShowSerialDish(Dishes); break;

case 5: {

int WorkChoose = 0;

printf("Список функций с работниками:\n1 - Получить список работников, 2 - Получить информацию о работнике по Id, 3 - Добавить работника, 4 - Удалить работника, 0 - Выход\n");

scanf("%d", &WorkChoose);

switch (WorkChoose) {

case 1: ShowWorkersDataBase(Workers); break;

case 2: GetWorkerInfo(Workers); break;

case 3: AddWorker(Workers); break;

case 4: RemoveWorker(Workers); break;

case 0: break;

}

}

}

}

printf("Вы успешно вышли из системы\n");

return EXIT\_SUCCESS;

}

//Работа с файлами

vector <Worker> WInit(vector <Worker>& Workers) {

string str;

ifstream fin;

Worker temp;

fin.open("UsersData.dat");

if (fin.is\_open()) {

while (!fin.eof()) {

str = "";

fin >> str; if (str != "") temp.id = stoi(str.c\_str());

fin >> str; if (str != "") temp.secondName = str.c\_str();

fin >> str; if (str != "") temp.name = str.c\_str();

fin >> str; if (str != "") temp.phoneNumber = str.c\_str();

fin >> str; if (str != "") temp.\_login = str.c\_str();

fin >> str; if (str != "") temp.\_password = str.c\_str();

fin >> str; if (str != "") temp.\_accessLevel = stoi(str.c\_str());

if (str != "") Workers.push\_back(temp);

}

fin.close();

printf("Workers initialization complete.\n");

} else {

fin.close();

printf("Workers initialization failed. Loading Default settings.\n");

CreateDefaultWorkers();

WInit(Workers);

}

return Workers;

}

vector<Table> TInit(vector <Table>& Tables) {

string str;

ifstream fin;

Table temp;

fin.open("TablesInfo.dat");

if (fin.is\_open()) {

while (!fin.eof()) {

str = "";

fin >> str; if (str != "") temp.code = stoi(str.c\_str());

fin >> str; if (str != "") temp.name = str.c\_str();

if (str != "") Tables.push\_back(temp);

}

fin.close();

printf("Tables Initialization Complete.\n");

} else {

fin.close();

printf("Tables initialization failed. Loading Default settings.\n");

CreateDefaultTables();

TInit(Tables);

}

return Tables;

}

vector <Dish> DInit(vector <Dish>& Dishes) {

string str;

ifstream fin;

Dish temp;

fin.open("DishesInfo.dat");

if (fin.is\_open()) {

while (!fin.eof()) {

str = "";

fin >> str; if (str != "") temp.code = stoi(str);

fin >> str; if (str != "") temp.name = str.c\_str();

fin >> str; if (str != "") temp.ingredients = str.c\_str();

fin >> str; if (str != "") temp.time = stoi(str);

fin >> str; if (str != "") temp.numberOfTable = stoi(str);

fin >> str; if (str != "") temp.calories = stoi(str);

if (str != "") Dishes.push\_back(temp);

}

fin.close();

printf("Dishes initialization complete.\n");

} else {

fin.close();

printf("Dishes initialization failed. Loading default settings.\n");

createDefaultDishes();

DInit(Dishes);

}

return Dishes;

}

int Auth(vector <Worker>& Workers) {

string password = "";

string login = "";

printf("Введите логин\n");

cin >> login;

printf("Введите пароль\n");

cin >> password;

transform(login.begin(), login.end(), login.begin(), tolower);

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

if (Workers[i].\_login == login) {

if (Workers[i].\_password == password) {

printf("success\n");

return(Workers[i].\_accessLevel);

} else {

break;

}

}

}

printf("Вы ввели неверные данные, попробуйте снова\n");

return -1;

}

int main() {

setlocale(LC\_ALL, "Russian");

vector<Dish> Dishes;

vector<Table> Tables;

vector<Worker> Workers;

WInit(Workers);

TInit(Tables);

DInit(Dishes);

int accessLevel = -1;

while (accessLevel < 0) {

accessLevel = Auth(Workers);

}

switch (accessLevel) {

case 0: AccessLevel0Tools(Dishes, Workers, Tables); break;

case 1: AccessLevel1Tools(Dishes, Workers, Tables); break;

case 2: AccessLevel2Tools(Dishes, Workers, Tables); break;

}

return 0;

}