Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа № 5

По дисциплине «Основы алгоритмизации и программирования»

На тему «**Объединения, перечисления, битовые поля**»

Выполнил:

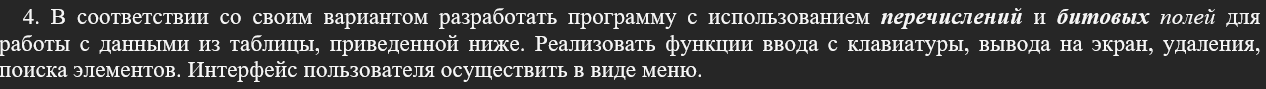
Федорович Вадим

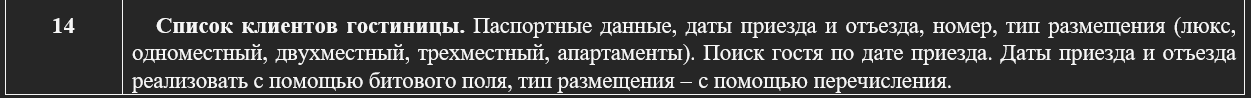
Студент 1 курса 8 группы

Преподаватель: асс. Андронова М.В.

Минск, 2024

Основная часть:





#include<iostream>

#include<Windows.h>

#include<stdio.h>

#include <string>

#include<fstream>

#define size 8

using namespace std;

enum Type\_of\_allocation

{

Luxury = 1, Single, Double, triple\_room, apartment

}allocation;

struct Clients {

string data; // заменил name на data

string allocation;

int number;

unsigned day\_Arrival : 5;// Для дня 5 битов

unsigned month\_Arrival : 4;// Для месяца 4 бита

unsigned year\_Arrival : 11;// Для года 11 битов

unsigned day\_departure : 5;

unsigned month\_departure : 4;

unsigned year\_departure : 11;

};

Clients list\_of\_clients[size];

Clients bad; // Для удаления

int num;

int choice;

int now\_size = 0;

void enter();

void out();

void found();

void delet();

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

cout << "Введите: " << endl;

cout << "1 - Ввод данных" << endl;

cout << "2 - Вывод на экран информации о поездах" << endl;

cout << "3 - Поиск гостя по дате приезда." << endl;

cout << "4 - Удаление записей" << endl;

cout << "5 - Выход" << endl;

cin >> choice;

do {

switch (choice) {

case 1: enter(); break;

case 2: out(); break;

case 3: found(); break;

case 4: delet(); break;

}

} while (choice != 5);

}

void enter() {

int answer;

int datа;

do {

if (now\_size >= size) { // проверка на место

cout << "Ошибка: массив поездов заполнен." << endl;

return;

}

cout << "Введите паспортные данные: ";

cin.ignore(); // Чистка буфера

getline(cin, list\_of\_clients[now\_size].data);

cout << "Введите номер: ";

cin >> list\_of\_clients[now\_size].number;

do {

cout << "Введите тип размещения (1 - люкс, 2 - одноместный, 3 - двухместный, 4 - трехместный, 5 - апартаменты)";

cin >> num;

if (num == (int)Luxury) list\_of\_clients[now\_size].allocation = "Luxury"; // Проверка последовательности и присвоение

else if (num == (int)Single) list\_of\_clients[now\_size].allocation = "Single";

else if (num == (int)Double) list\_of\_clients[now\_size].allocation = "Double";

else if (num == (int)triple\_room) list\_of\_clients[now\_size].allocation = "Triple room";

else if (num == (int)apartment) list\_of\_clients[now\_size].allocation = "Apartment";

} while (num != 1 && num != 2 && num != 3 && num != 4 && num != 5);

cout << "Введите дату приеда: ";

cin >> datа;

list\_of\_clients[now\_size].day\_Arrival = (datа / 1000000); // Запись для битового поля

list\_of\_clients[now\_size].month\_Arrival = ((datа / 10000) % 100);

list\_of\_clients[now\_size].year\_Arrival = (datа % 10000);

cout << "Введите дату отъезда: ";

cin >> datа;

list\_of\_clients[now\_size].day\_departure = (datа / 1000000);// Запись для битового поля

list\_of\_clients[now\_size].month\_departure = ((datа / 10000) % 100);

list\_of\_clients[now\_size].year\_departure = (datа % 10000);

now\_size++;

cout << "Продолжить ввод? (1 - да, 0 - нет): ";

cin >> answer;

} while (answer == 1);

cout << "Что дальше?" << endl;

cin >> choice;

}

void out() { // Вывод

cout << "Вывод информации..." << endl;

for (int i = 0; i < now\_size; i++) {

cout << "Паспортные данные: " << list\_of\_clients[i].data << endl;

cout << "Тип размещения: " << list\_of\_clients[i].allocation << endl;

cout << "Номер: " << list\_of\_clients[i].number << endl;

cout << "Дата заезда: " << list\_of\_clients[i].day\_Arrival << "." << list\_of\_clients[i].month\_Arrival << "." << list\_of\_clients[i].year\_Arrival << endl;

cout << "Дата выезда: " << list\_of\_clients[i].day\_departure << "." << list\_of\_clients[i].month\_departure << "." << list\_of\_clients[i].year\_departure << endl;

}

cout << "Что дальше?" << endl;

cin >> choice;

}

void found() { // Поиск

int buffer, buffer\_day, buffer\_month, buffer\_year;

cout << "Введите день отъезда: ";

cin >> buffer;

buffer\_day = buffer / 1000000;// Деление на дни и тд

buffer\_month = ((buffer / 10000) % 100);

buffer\_year = (buffer % 10000);

for (int i = 0; i < now\_size; i++) {

if (buffer\_day == list\_of\_clients[i].day\_Arrival && buffer\_month == list\_of\_clients[i].month\_Arrival && buffer\_year == list\_of\_clients[i].year\_Arrival) {

cout << "Паспортные данные: " << list\_of\_clients[i].data << endl;

cout << "Тип размещения: " << list\_of\_clients[i].allocation << endl;

cout << "Номер: " << list\_of\_clients[i].number << endl;

cout << "Дата заезда: " << list\_of\_clients[i].day\_Arrival << "." << list\_of\_clients[i].month\_Arrival << "." << list\_of\_clients[i].year\_Arrival << endl;

cout << "Дата выезда: " << list\_of\_clients[i].day\_departure << "." << list\_of\_clients[i].month\_departure << "." << list\_of\_clients[i].year\_departure << endl;

}

}

cout << "Что дальше?" << endl;

cin >> choice;

}

void delet(){ // Удаление строки структуры

int answer;

int select;

int number;

do{

cout << "1 - Удалить введенную строку" << endl;

cout << "2 - Удалить все" << endl;

cin >> select;

if (select == 1) {

cout << "Какую строку хотите удалить?" << endl;

cin >> number;

list\_of\_clients[number] = bad;

}

if (select == 2) {

for (int i = 0; i < now\_size; i++) {

list\_of\_clients[i] = bad;

}

}

cout << "Продолжить ввод? (1 - да, 0 - нет): ";

cin >> answer;

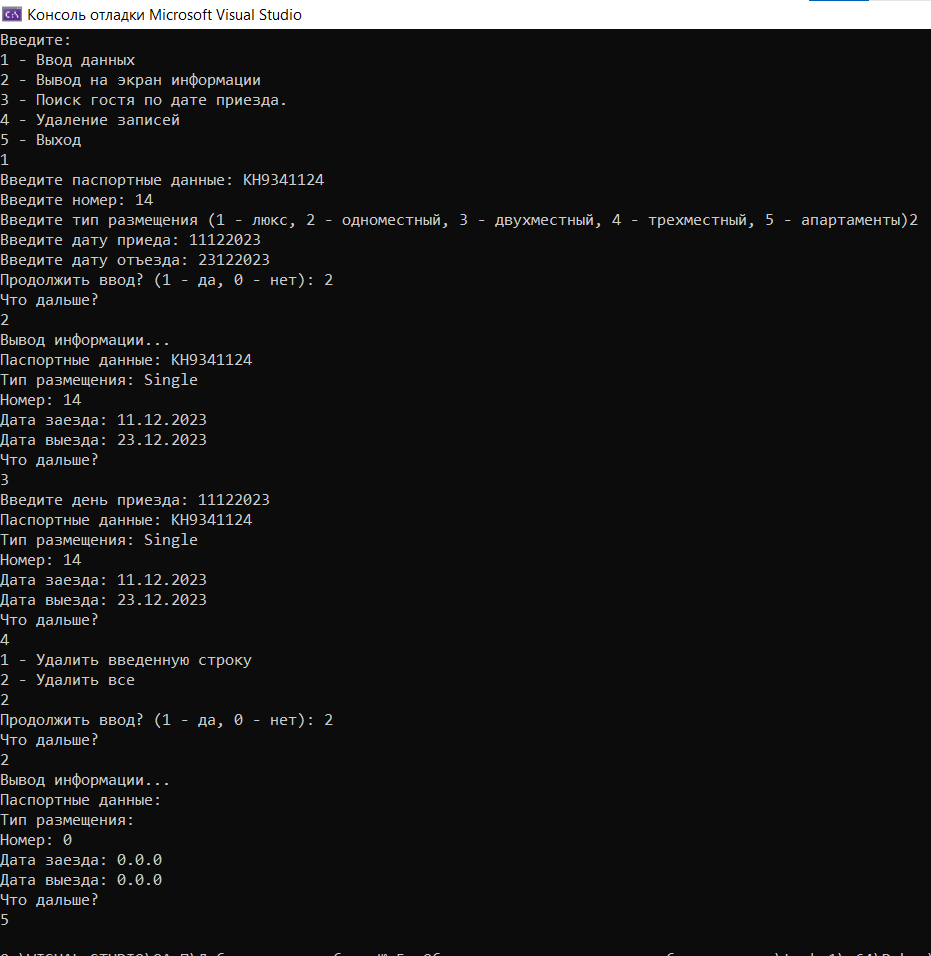
} while (answer == 1);

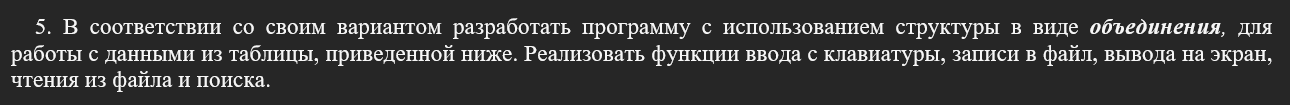
cout << "Что дальше?" << endl;

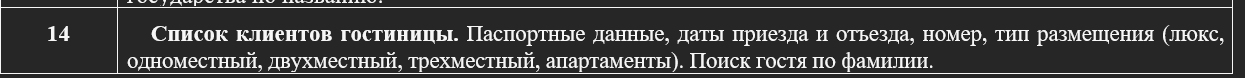
cin >> choice;

}

//unsigned day\_Arrival : 5; Т.к. 31 = 5 битов







#include<iostream>

#include<Windows.h>

#include<stdio.h>

#include<string>

#include<fstream>

#define size 8

using namespace std;

union person {

char last\_name[30];

char data[30];

char allocation[30];

int number;

char time\_arrival[15];

char time\_departure[15];

};

struct Clients {

person lastname;

person pasport\_data;

person place;

person room;

person time\_arr;

person time\_dep;

};

Clients list\_of\_clients[size];

int num;

int choice;

int now\_size = 0;

void enter();

void out();

void found();

void file\_write();

void file\_read();

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

cout << "Введите: " << endl;

cout << "1 - Ввод данных" << endl;

cout << "2 - Вывод на экран информации о поездах" << endl;

cout << "3 - Поиск гостя по Фамилии." << endl;

cout << "4 - Запись в файл" << endl;

cout << "5 - Чтение из файла" << endl;

cout << "6 - Выход" << endl;

cin >> choice;

do {

switch (choice) {

case 1: enter(); break;

case 2: out(); break;

case 3: found(); break;

case 4: file\_write(); break;

case 5: file\_read(); break;

}

} while (choice != 6);

}

void enter() { // Ввод

int answer;

int datа;

do {

if (now\_size >= size) { // проверка на место

cout << "Ошибка: массив поездов заполнен." << endl;

return;

}

cin.ignore(); // Чистка буфера

cout << "Введите фамилию клиента: ";

gets\_s(list\_of\_clients[now\_size].lastname.last\_name, 50);

cout << "Введите паспортные данные: ";

gets\_s(list\_of\_clients[now\_size].pasport\_data.data, 50);

cout << "Введите тип размещения (Люкс, Одноместный, Двухместный, Трехместный, Апартаменты): ";

gets\_s(list\_of\_clients[now\_size].place.allocation, 50);

cout << "Введите номер: ";

cin >> list\_of\_clients[now\_size].room.number;

cin.ignore(); // Чистка буфера

cout << "Введите дату приеда: ";

gets\_s(list\_of\_clients[now\_size].time\_arr.time\_arrival, 15);

cout << "Введите дату отъезда: ";

gets\_s(list\_of\_clients[now\_size].time\_dep.time\_departure, 15);

now\_size++;

cout << "Продолжить ввод? (1 - да, 0 - нет): ";

cin >> answer;

} while (answer == 1);

cout << "Что дальше?" << endl;

cin >> choice;

}

void out() {

cout << "Вывод информации..." << endl;

for (int i = 0; i < now\_size; i++) {

cout << "Фамилия: " << list\_of\_clients[i].lastname.last\_name << endl;

cout << "Паспортные данные: " << list\_of\_clients[i].pasport\_data.data << endl;

cout << "Тип размещения: " << list\_of\_clients[i].place.allocation << endl;

cout << "Номер: " << list\_of\_clients[i].room.number << endl;

cout << "Дата заезда: " << list\_of\_clients[i].time\_arr.time\_arrival << endl;

cout << "Дата выезда: " << list\_of\_clients[i].time\_dep.time\_departure << endl;

}

cout << "Что дальше?" << endl;

cin >> choice;

}

void found() {

char buffer[50];

cout << "Введите нужную фамилию: ";

cin.ignore(); // Очистка

gets\_s(buffer, 50);

for (int i = 0; i < now\_size; i++) {

if (strcmp(buffer,list\_of\_clients[i].lastname.last\_name) == 0){ // Сравнение строк

cout << "Фамилия: " << list\_of\_clients[i].lastname.last\_name << endl;

cout << "Паспортные данные: " << list\_of\_clients[i].pasport\_data.data << endl;

cout << "Тип размещения: " << list\_of\_clients[i].place.allocation << endl;

cout << "Номер: " << list\_of\_clients[i].room.number << endl;

cout << "Дата заезда: " << list\_of\_clients[i].time\_arr.time\_arrival << endl;

cout << "Дата выезда: " << list\_of\_clients[i].time\_dep.time\_departure << endl;

}

}

cout << "Что дальше?" << endl;

cin >> choice;

}

void file\_write() { // Запись в файл

ofstream write("task2.txt");

for (int i = 0; i < now\_size; i++) {

write << "Фамилия: " << list\_of\_clients[i].lastname.last\_name << endl;

write << "Паспортные данные: " << list\_of\_clients[i].pasport\_data.data << endl;

write << "Тип размещения: " << list\_of\_clients[i].place.allocation << endl;

write << "Номер: " << list\_of\_clients[i].room.number << endl;

write << "Дата заезда: " << list\_of\_clients[i].time\_arr.time\_arrival << endl;

write << "Дата выезда: " << list\_of\_clients[i].time\_dep.time\_departure << endl;

}

write.close();

cout<< "Данные записанны " << endl;

cout << "Что дальше?" << endl;

cin >> choice;

}

void file\_read() {

char buf[250];

ifstream file\_read("task2.txt");

cout << "Содержимое файла: " << endl;

while (file\_read.getline(buf, 250)) { // читаем каждую строку

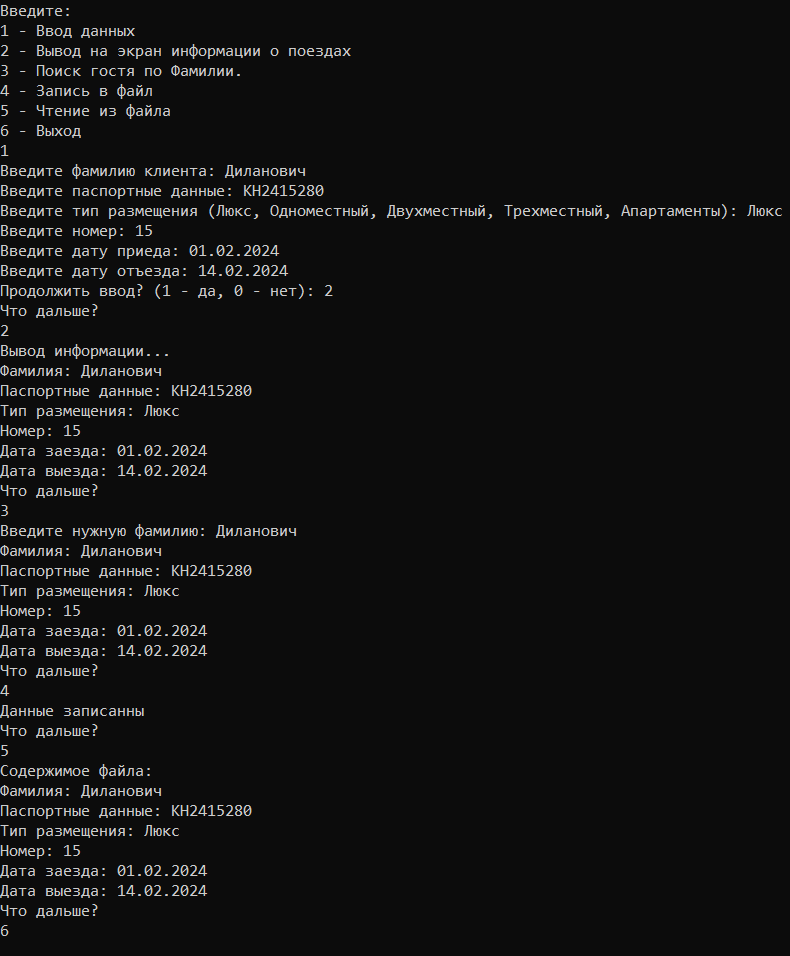
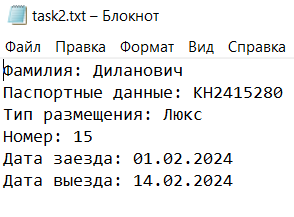
cout << buf << endl;

}

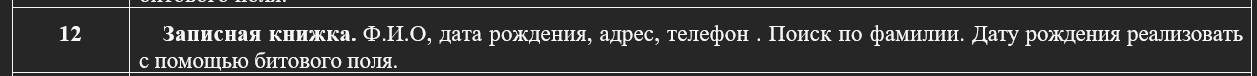
cout << "Что дальше?" << endl;

cin >> choice;

}

Дополнительные задания:



#include<iostream>

#include<Windows.h>

#include<stdio.h>

#include <string>

#include<fstream>

#define size 8

using namespace std;

struct peoples\_book {

string full\_name;

unsigned day\_birthday : 5;// Для дня 5 битов

unsigned month\_birthday : 4;// Для месяца 4 бита

unsigned year\_birthday : 11;// Для года 11 битов

string living\_place;

char personal\_number[15];

};

peoples\_book list\_of\_peoples[size];

peoples\_book bad; // Для удаления

int num;

int choice;

int now\_size = 0;

void enter();

void out();

void found();

void delet();

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

cout << "Введите: " << endl;

cout << "1 - Ввод данных" << endl;

cout << "2 - Вывод на экран информации" << endl;

cout << "3 - Поиск гостя по фамилии" << endl;

cout << "4 - Удаление записей" << endl;

cout << "5 - Выход" << endl;

cin >> choice;

do {

switch (choice) {

case 1: enter(); break;

case 2: out(); break;

case 3: found(); break;

case 4: delet(); break;

}

} while (choice != 5);

}

void enter() {

int answer;

int datа;

do {

if (now\_size >= size) { // проверка на место

cout << "Ошибка: массив поездов заполнен." << endl;

return;

}

cout << "Введите ФИО: ";

cin.ignore(); // Чистка буфера

getline(cin, list\_of\_peoples[now\_size].full\_name);

cout << "Введите дату рожления: ";

cin >> datа;

list\_of\_peoples[now\_size].day\_birthday = (datа / 1000000); // Запись для битового поля

list\_of\_peoples[now\_size].month\_birthday = ((datа / 10000) % 100);

list\_of\_peoples[now\_size].year\_birthday = (datа % 10000);

cout << "Введите адрес: ";

cin.ignore(); // Чистка буфера

getline(cin, list\_of\_peoples[now\_size].living\_place);

cout << "Введите телефон: ";

gets\_s(list\_of\_peoples[now\_size].personal\_number, 15);

now\_size++;

cout << "Продолжить ввод? (1 - да, 0 - нет): ";

cin >> answer;

} while (answer == 1);

cout << "Что дальше?" << endl;

cin >> choice;

}

void out() { // Вывод

cout << "Вывод информации..." << endl;

for (int i = 0; i < now\_size; i++) {

cout << "ФИО: " << list\_of\_peoples[i].full\_name << endl;

cout << "Дата рождения: " << list\_of\_peoples[i].day\_birthday << "." << list\_of\_peoples[i].month\_birthday << "." << list\_of\_peoples[i].year\_birthday << endl;

cout << "Адрес: " << list\_of\_peoples[i].living\_place << endl;

cout << "Номер: " << list\_of\_peoples[i].personal\_number << endl;

}

cout << "Что дальше?" << endl;

cin >> choice;

}

void found() { // Поиск

string buffer;

cout << "Введите фамилию: ";

cin.ignore();

getline(cin, buffer);

for (int i = 0; i < now\_size; i++) {

if ( buffer == list\_of\_peoples[i].full\_name) {

cout << "ФИО: " << list\_of\_peoples[i].full\_name << endl;

cout << "Дата рождения: " << list\_of\_peoples[i].day\_birthday << "." << list\_of\_peoples[i].month\_birthday << "." << list\_of\_peoples[i].year\_birthday << endl;

cout << "Адрес: " << list\_of\_peoples[i].living\_place << endl;

cout << "Номер: " << list\_of\_peoples[i].personal\_number << endl;

}

}

cout << "Что дальше?" << endl;

cin >> choice;

}

void delet() { // Удаление строки структуры

int answer;

int select;

int number;

do {

cout << "1 - Удалить введенную строку" << endl;

cout << "2 - Удалить все" << endl;

cin >> select;

if (select == 1) {

cout << "Какую строку хотите удалить?" << endl;

cin >> number;

list\_of\_peoples[number] = bad;

}

if (select == 2) {

for (int i = 0; i < now\_size; i++) {

list\_of\_peoples[i] = bad;

}

}

cout << "Продолжить ввод? (1 - да, 0 - нет): ";

cin >> answer;

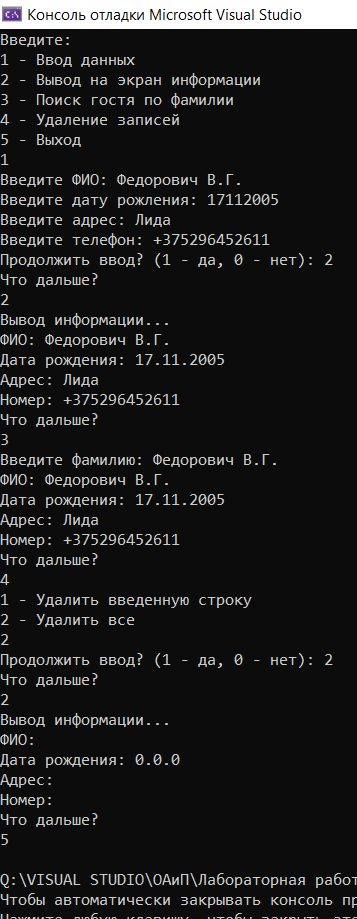
} while (answer == 1);

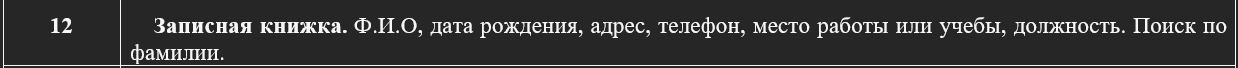
cout << "Что дальше?" << endl;

cin >> choice;

}

//unsigned day\_birthday : 5; Т.к. 31 = 5 битов





//Записная книжка. Ф.И.О, дата рождения, адрес, телефон, место работы или учебы, должность. Поиск по фамилии.

#include<iostream>

#include<Windows.h>

#include<stdio.h>

#include<string>

#include<fstream>

#define size 8

using namespace std;

union person {

char full\_name[50];

char birth\_time[50];

char location[50];

char number[50];

char work\_place[50];

};

struct Clients {

person name;

person birth;

person place;

person phone;

person spot;

};

Clients list\_of\_clients[size];

int num;

int choice;

int now\_size = 0;

void enter();

void out();

void found();

void file\_write();

void file\_read();

int main() {

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

cout << "Введите: " << endl;

cout << "1 - Ввод данных" << endl;

cout << "2 - Вывод на экран информации" << endl;

cout << "3 - Поиск по Фамилии." << endl;

cout << "4 - Запись в файл" << endl;

cout << "5 - Чтение из файла" << endl;

cout << "6 - Выход" << endl;

cin >> choice;

do {

switch (choice) {

case 1: enter(); break;

case 2: out(); break;

case 3: found(); break;

case 4: file\_write(); break;

case 5: file\_read(); break;

}

} while (choice != 6);

}

void enter() { // Ввод

int answer;

int datа;

do {

if (now\_size >= size) { // проверка на место

cout << "Ошибка: массив поездов заполнен." << endl;

return;

}

cin.ignore(); // Чистка буфера

cout << "Введите ФИО: ";

gets\_s(list\_of\_clients[now\_size].name.full\_name, 50);

cout << "Введите дату рождения: ";

gets\_s(list\_of\_clients[now\_size].birth.birth\_time, 50);

cout << "Введите адрес ";

gets\_s(list\_of\_clients[now\_size].place.location, 50);

cout << "Введите номер: ";

gets\_s(list\_of\_clients[now\_size].phone.number, 50);

cout << "Введите место работы или учебы: ";

gets\_s(list\_of\_clients[now\_size].spot.work\_place, 50);

now\_size++;

cout << "Продолжить ввод? (1 - да, 0 - нет): ";

cin >> answer;

} while (answer == 1);

cout << "Что дальше?" << endl;

cin >> choice;

}

void out() {

cout << "Вывод информации..." << endl;

for (int i = 0; i < now\_size; i++) {

cout << "ФИО: " << list\_of\_clients[i].name.full\_name << endl;

cout << "Дата рождения: " << list\_of\_clients[i].birth.birth\_time << endl;

cout << "Тип размещения: " << list\_of\_clients[i].place.location << endl;

cout << "Номер: " << list\_of\_clients[i].phone.number << endl;

cout << "Место работы или учебы: " << list\_of\_clients[i].spot.work\_place << endl;

}

cout << "Что дальше?" << endl;

cin >> choice;

}

void found() {

char buffer[50];

cout << "Введите нужную фамилию: ";

cin.ignore(); // Очистка

gets\_s(buffer, 50);

for (int i = 0; i < now\_size; i++) {

if (strcmp(buffer, list\_of\_clients[i].name.full\_name) == 0) { // Сравнение строк

cout << "ФИО: " << list\_of\_clients[i].name.full\_name << endl;

cout << "Дата рождения: " << list\_of\_clients[i].birth.birth\_time << endl;

cout << "Тип размещения: " << list\_of\_clients[i].place.location << endl;

cout << "Номер: " << list\_of\_clients[i].phone.number << endl;

cout << "Место работы или учебы: " << list\_of\_clients[i].spot.work\_place << endl;

}

}

cout << "Что дальше?" << endl;

cin >> choice;

}

void file\_write() { // Запись в файл

ofstream write("task2.txt");

for (int i = 0; i < now\_size; i++) {

write << "ФИО: " << list\_of\_clients[i].name.full\_name << endl;

write << "Дата рождения: " << list\_of\_clients[i].birth.birth\_time << endl;

write << "Тип размещения: " << list\_of\_clients[i].place.location << endl;

write << "Номер: " << list\_of\_clients[i].phone.number << endl;

write << "Место работы или учебы: " << list\_of\_clients[i].spot.work\_place << endl;

}

write.close();

cout << "Данные записанны " << endl;

cout << "Что дальше?" << endl;

cin >> choice;

}

void file\_read() {

char buf[250];

ifstream file\_read("task2.txt");

cout << "Содержимое файла: " << endl;

while (file\_read.getline(buf, 250)) { // читаем каждую строку

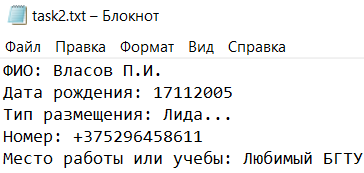
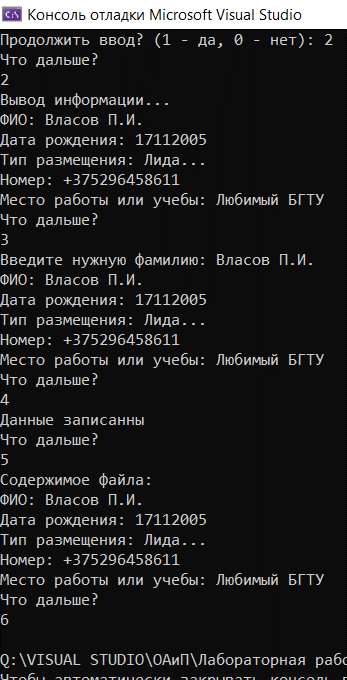
cout << buf << endl;

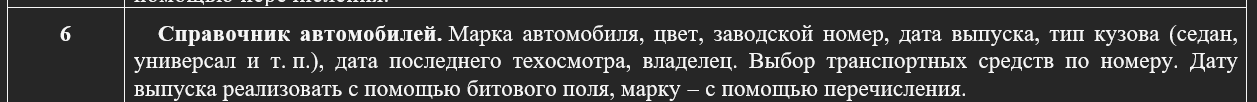
}

cout << "Что дальше?" << endl;

cin >> choice;

}





#include <string>

#define MAX\_CARS 10

using namespace std;

// Объявление функций

void inputCars(); // Ввод информации о машинах

void outputCars(); // Вывод информации о машинах

void checkCarsByNumber(); // Поиск машины по номеру

void deleteCars(); // Удаление машины

int currentCars = 0; // Текущее количество машин в списке

#define cinIn(type, target) { type value; cin >> value; target = value; } // Макрос для удобного ввода значения из потока ввода

enum brand

{

    Mercedes = 1, BMW, Porshe, Lotus, Koenigsegg, Lada, Lamborgini, McLaren, Ferrari, Ford, Wolkswagen, Opel, Nissan

};

struct Date {

    int day : 5;   // 5 бит для представления дня (значения от 1 до 31)

    int month : 4; // 4 бита для представления месяца (значения от 1 до 12)

    int year : 12; // 12 бит для представления года (значения от 0 до 4095)

};

struct Car

{

    brand brands; // Марка автомобиля

    string color; // Цвет машины

    int number;   // Номер автомобиля

    string bodyType; // Тип кузова

    string owner; // Имя владельца

    Date date; // Дата выпуска

    Date lastInspection; // Дата последнего технического осмотра

};

Car listOfCars[MAX\_CARS]; // Массив структур Car для хранения информации о машинах

int var6n1()

{

    int choice;

    do

    {

        cout << "1 - Ввод элементов структуры с клавиатуры" << endl;

        cout << "2 - Вывод элементов структуры в консольное окно" << endl;

        cout << "3 - Выбор транспортных средств по номеру" << endl;

        cout << "4 - Удаление машины из списка" << endl;

        cout << "0 - Выход из программы" << endl;

        cin >> choice; // Считываем выбор пользователя

        switch (choice)

        {

        case 1:

            inputCars();

            break;

        case 2:

            outputCars();

            break;

        case 3:

            checkCarsByNumber();

            break;

        case 4:

            deleteCars();

            break;

        case 0:

            exit(0);

            break;

        default:

            cout << "Некорректный ввод!";

            break;

        }

    } while (choice != 0);

    return 0;

}

// Функция для ввода информации о машинах

void inputCars()

{

    int numCars;

    cout << "Введите количество машин: ";

    cin >> numCars; // Считываем количество поездов

    cin.ignore(); // Игнорируем символ новой строки после ввода числа

    for (int i = 0; i < numCars; i++)

    {

        if (currentCars < MAX\_CARS)

        {

            cout << endl << "Цвет машины: ";

            getline(cin, listOfCars[currentCars].color);

            cout << "Номер автомобиля: ";

            cin >> listOfCars[currentCars].number;

            cout << "Введите марку автомобиля(1 - Mercedes, 2 - BMW, 3 - Porsche и т.д.): ";

            int BrandValue;

            cin >> BrandValue;

            listOfCars[currentCars].brands = static\_cast<brand>(BrandValue);

            cout << "Дата выпуска: ";

            cinIn(int, listOfCars[currentCars].date.day);

            cinIn(int, listOfCars[currentCars].date.month);

            cinIn(int, listOfCars[currentCars].date.year);

            cout << "Дата последнего тех.осмотра: ";

            cinIn(int, listOfCars[currentCars].lastInspection.day);

            cinIn(int, listOfCars[currentCars].lastInspection.month);

            cinIn(int, listOfCars[currentCars].lastInspection.year);

            cin.ignore(); // Игнорируем символ новой строки после ввода времени

            cout << "Введите имя владельца: ";

            getline(cin,listOfCars[currentCars].owner);

            cout << "Тип кузова: ";

            getline(cin,listOfCars[currentCars].bodyType);

            cout << endl;

            currentCars++;

        }

    }

}

//функция для вывода

void outputCars()

{

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

    {

        cout << "\nЦвет машины: " << listOfCars[i].color << endl;

        cout << "Номер автомобиля: " << listOfCars[i].number << endl;

        cout << "Марка автомобиля: " << listOfCars[i].brands << endl;

        cout << "Дата выпуска: " << listOfCars[i].date.day << "." << listOfCars[i].date.month << "."  << listOfCars[i].date.year << endl;

        cout << "Дата последнего тех.осмотра: " << listOfCars[i].lastInspection.day << "." << listOfCars[i].lastInspection.month << "."  << listOfCars[i].lastInspection.year << endl;

        cout << "Имя владельца: " << listOfCars[i].owner << endl;

        cout << "Тип кузова: " << listOfCars[i].bodyType << endl;

    }

}

void checkCarsByNumber()

{

    int number;

    cout << "Введите номер машины: ";

    cin >> number;

    bool findCarWithThatNumber = false;

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

    {

        if (number == listOfCars[i].number)

        {

            findCarWithThatNumber = true;

            cout << "\nНайденный автомобиль:";

            cout << "\nЦвет машины: " << listOfCars[i].color << endl;

            cout << "Номер автомобиля: " << listOfCars[i].number << endl;

            cout << "Марка автомобиля: " << listOfCars[i].brands << endl;

            cout << "Дата выпуска: " << listOfCars[i].date.day << "." << listOfCars[i].date.month << "." << listOfCars[i].date.year << endl;

            cout << "Дата последнего тех.осмотра: " << listOfCars[i].lastInspection.day << "." << listOfCars[i].lastInspection.month << "." << listOfCars[i].lastInspection.year << endl;

            cout << "Имя владельца: " << listOfCars[i].owner << endl;

            cout << "Тип кузова: " << listOfCars[i].bodyType << endl;

        }

    }

    if (!findCarWithThatNumber)

    {

        cout << "\nНе найдено машин с таким номером.";

    }

}

void deleteCars()

{

    int number;

    cout << "Введите номер машины для удаления: ";

    cin >> number;

    bool findCarWithThatNumber = false;

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

    {

        if (number == listOfCars[i].number)

        {

            findCarWithThatNumber = true;

            for (int j = i; j < currentCars - 1; j++)

            {

                listOfCars[j] = listOfCars[j + 1];

            }

            currentCars--;

            cout << "\nАвтомобиль удален.\n";

            break;

        }

    }

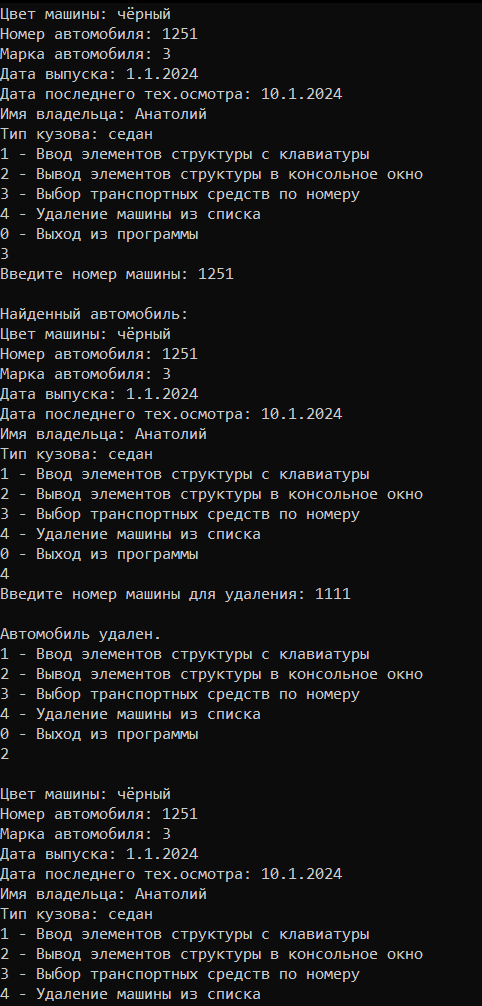
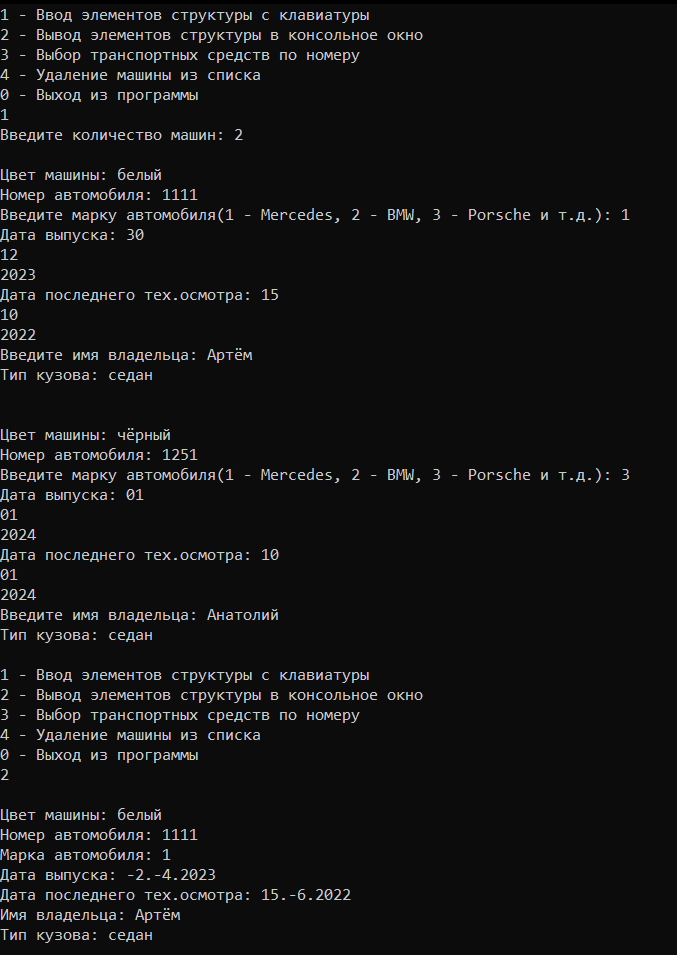
    if (!findCarWithThatNumber)

    {

        cout << "\nНе найдено машин с таким номером.\n";

    }

}



**2)**

// Структура для хранения полного имени

struct fullName

{

    string name;

    string surname;

    string partonymic;

};

// Структура для хранения информации о сотруднике

struct personnelDepartment

{

    fullName FIO;        // ФИО

    string education;    // Образование

    string speciality;   // Специальность

    string jobTitle;     // Должность

    int Salary;          // Оклад (зарплата)

    Date inviteDate;     // Дата приглашения на работу

};

// Функция для ввода данных о сотрудниках с клавиатуры и записи в файл

void input(int size)

{

    FILE\* f;

    personnelDepartment buf;

    if (fopen\_s(&f, "base.txt", "ab+") == 0) // Открываем файл в режиме добавления/чтения в двоичном режиме

    {

        for (int p = 0; p < size; p++)

        {

            cout << "ФИО: ";

            getline(cin, buf.FIO.name);

            getline(cin, buf.FIO.surname);

            getline(cin, buf.FIO.partonymic);

            cout << "Образование: ";

            getline(cin, buf.education);

            cout << "Специальность: ";

            getline(cin, buf.speciality);

            cout << "Должность: ";

            getline(cin, buf.jobTitle);

            cout << "Зарплата: ";

            cin >> buf.Salary;

            cout << "Дата вступления:";

            cinIn(int, buf.inviteDate.day);  // Здесь предполагается, что у вас есть функция cinIn() для ввода целых чисел

            cinIn(int, buf.inviteDate.month);

            cinIn(int, buf.inviteDate.year);

            fwrite(&buf, sizeof(buf), 1, f); // Записываем структуру в файл

        }

        fclose(f);

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

// Функция для вывода данных о сотрудниках из файла

void output()

{

    FILE\* f;

    personnelDepartment buf;

    if (fopen\_s(&f, "base.txt", "rb") == 0) // Открываем файл в режиме чтения в двоичном режиме

    {

        cout << "\nФИО\t\tОбразование\tСпециальность\tДолжность\tЗарплата\tДата вступления\n";

        fread(&buf, sizeof(buf), 1, f); // Читаем структуру из файла

        while (!feof(f))

        {

            cout << buf.FIO.name << ' ' << buf.FIO.surname << ' ' << buf.FIO.partonymic << "  " << buf.education << "  " << buf.speciality << "  " << buf.jobTitle << "  " << buf.Salary << "  " << buf.inviteDate.day << '.' << buf.inviteDate.month << '.' << buf.inviteDate.year << endl;

            fread(&buf, sizeof(buf), 1, f); // Читаем следующую структуру из файла

        }

        cout << endl;

        fclose(f);

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

// Функция для поиска сотрудников по должности

void findByJobTitle(string Title)

{

    FILE\* f;

    personnelDepartment buf;

    bool found = false;

    if (fopen\_s(&f, "base.txt", "rb") == 0) // Открываем файл в режиме чтения в двоичном режиме

    {

        while (fread(&buf, sizeof(buf), 1, f) == 1)

        {

            if (buf.jobTitle == Title)

            {

                cout << "\nФИО\t\tОБразование\tСпециальность\tДолжность\tЗарплата\tДата вступления\n";

                cout << buf.FIO.name << ' ' << buf.FIO.surname << ' ' << buf.FIO.partonymic << "  " << buf.education << "  " << buf.speciality << "  " << buf.jobTitle << "  " << buf.Salary << "  " << buf.inviteDate.day << '.' << buf.inviteDate.month << '.' << buf.inviteDate.year << endl;

                found = true;

            }

        }

        fclose(f);

        if (!found)

            cout << "Не найдено\n";

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

int var6n2()

{

    int choice, number;

    string Title;

    do

    {

        cout << "\n1. Ввод данных с клавиатуры и запись в файл\n";

        cout << "2. Вывод данных из файла\n";

        cout << "3. Поиск по должности\n";

        cout << "0. Выход из программы\n\n";

        cout << "Введите номер операции: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Введите количество работников: ";

            cin >> number;

            cin.ignore(); // Игнорируем символ новой строки после ввода числа

            input(number);

            break;

        case 2:

            output();

            break;

        case 3:

            cout << "Введите должность: ";

            cin.ignore(); // Игнорируем символ новой строки перед вводом строки

            getline(cin, Title);

            findByJobTitle(Title);

            break;

        case 0:

            exit(0);

            break;

        default:

            cout << "Некорректный ввод. Пожалуйста, выберите допустимую операцию." << endl;

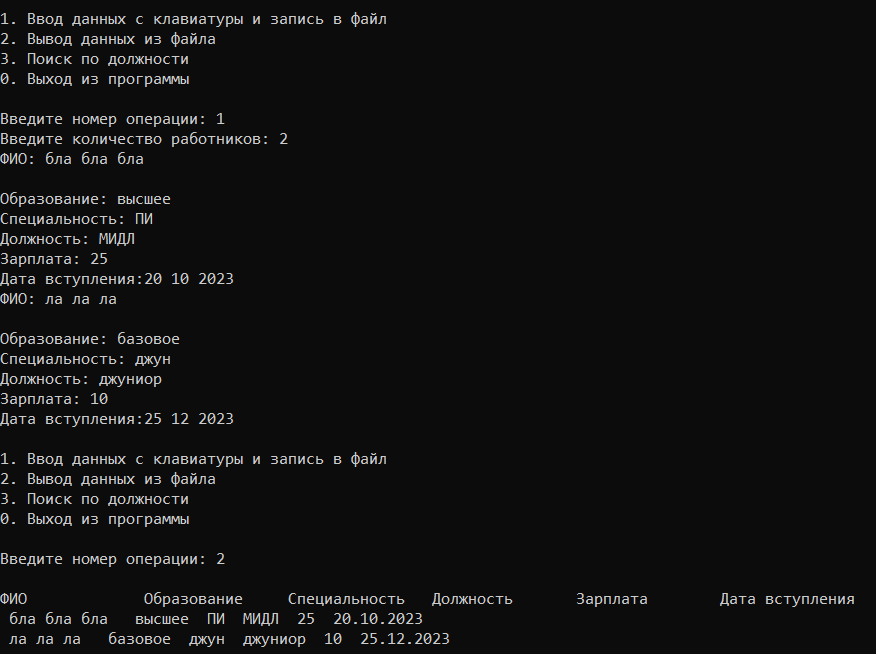
            break;

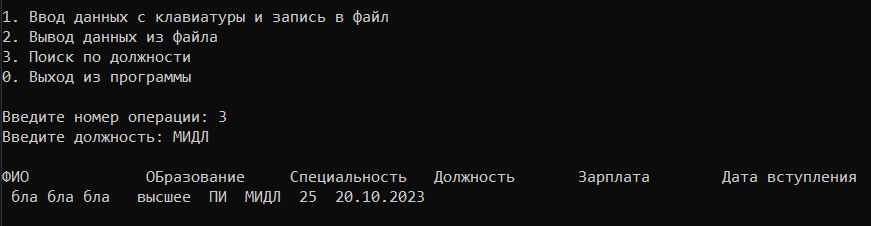
        }

    } while (choice != 0);

    return 0;

}

****



// Структура для хранения полного имени

struct fullName

{

    string name;

    string surname;

    string partonymic;

};

// Структура для хранения информации о сотруднике

struct personnelDepartment

{

    fullName FIO;        // ФИО

    string education;    // Образование

    string speciality;   // Специальность

    string jobTitle;     // Должность

    int Salary;          // Оклад (зарплата)

    Date inviteDate;     // Дата приглашения на работу

};

// Функция для ввода данных о сотрудниках с клавиатуры и записи в файл

void input(int size)

{

    FILE\* f;

    personnelDepartment buf;

    if (fopen\_s(&f, "base.txt", "ab+") == 0) // Открываем файл в режиме добавления/чтения в двоичном режиме

    {

        for (int p = 0; p < size; p++)

        {

            cout << "ФИО: ";

            getline(cin, buf.FIO.name);

            getline(cin, buf.FIO.surname);

            getline(cin, buf.FIO.partonymic);

            cout << "Образование: ";

            getline(cin, buf.education);

            cout << "Специальность: ";

            getline(cin, buf.speciality);

            cout << "Должность: ";

            getline(cin, buf.jobTitle);

            cout << "Зарплата: ";

            cin >> buf.Salary;

            cout << "Дата вступления:";

            cinIn(int, buf.inviteDate.day);  // Здесь предполагается, что у вас есть функция cinIn() для ввода целых чисел

            cinIn(int, buf.inviteDate.month);

            cinIn(int, buf.inviteDate.year);

            fwrite(&buf, sizeof(buf), 1, f); // Записываем структуру в файл

        }

        fclose(f);

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

// Функция для вывода данных о сотрудниках из файла

void output()

{

    FILE\* f;

    personnelDepartment buf;

    if (fopen\_s(&f, "base.txt", "rb") == 0) // Открываем файл в режиме чтения в двоичном режиме

    {

        cout << "\nФИО\t\tОбразование\tСпециальность\tДолжность\tЗарплата\tДата вступления\n";

        fread(&buf, sizeof(buf), 1, f); // Читаем структуру из файла

        while (!feof(f))

        {

            cout << buf.FIO.name << ' ' << buf.FIO.surname << ' ' << buf.FIO.partonymic << "  " << buf.education << "  " << buf.speciality << "  " << buf.jobTitle << "  " << buf.Salary << "  " << buf.inviteDate.day << '.' << buf.inviteDate.month << '.' << buf.inviteDate.year << endl;

            fread(&buf, sizeof(buf), 1, f); // Читаем следующую структуру из файла

        }

        cout << endl;

        fclose(f);

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

// Функция для поиска сотрудников по должности

void findByJobTitle(string Title)

{

    FILE\* f;

    personnelDepartment buf;

    bool found = false;

    if (fopen\_s(&f, "base.txt", "rb") == 0) // Открываем файл в режиме чтения в двоичном режиме

    {

        while (fread(&buf, sizeof(buf), 1, f) == 1)

        {

            if (buf.jobTitle == Title)

            {

                cout << "\nФИО\t\tОБразование\tСпециальность\tДолжность\tЗарплата\tДата вступления\n";

                cout << buf.FIO.name << ' ' << buf.FIO.surname << ' ' << buf.FIO.partonymic << "  " << buf.education << "  " << buf.speciality << "  " << buf.jobTitle << "  " << buf.Salary << "  " << buf.inviteDate.day << '.' << buf.inviteDate.month << '.' << buf.inviteDate.year << endl;

                found = true;

            }

        }

        fclose(f);

        if (!found)

            cout << "Не найдено\n";

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

int var6n2()

{

    int choice, number;

    string Title;

    do

    {

        cout << "\n1. Ввод данных с клавиатуры и запись в файл\n";

        cout << "2. Вывод данных из файла\n";

        cout << "3. Поиск по должности\n";

        cout << "0. Выход из программы\n\n";

        cout << "Введите номер операции: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Введите количество работников: ";

            cin >> number;

            cin.ignore(); // Игнорируем символ новой строки после ввода числа

            input(number);

            break;

        case 2:

            output();

            break;

        case 3:

            cout << "Введите должность: ";

            cin.ignore(); // Игнорируем символ новой строки перед вводом строки

            getline(cin, Title);

            findByJobTitle(Title);

            break;

        case 0:

            exit(0);

            break;

        default:

            cout << "Некорректный ввод. Пожалуйста, выберите допустимую операцию." << endl;

            break;

        }

    } while (choice != 0);

    return 0;

}

// Структура для хранения полного имени

struct fullName

{

    string name;

    string surname;

    string partonymic;

};

// Структура для хранения информации о сотруднике

struct personnelDepartment

{

    fullName FIO;        // ФИО

    string education;    // Образование

    string speciality;   // Специальность

    string jobTitle;     // Должность

    int Salary;          // Оклад (зарплата)

    Date inviteDate;     // Дата приглашения на работу

};

// Функция для ввода данных о сотрудниках с клавиатуры и записи в файл

void input(int size)

{

    FILE\* f;

    personnelDepartment buf;

    if (fopen\_s(&f, "base.txt", "ab+") == 0) // Открываем файл в режиме добавления/чтения в двоичном режиме

    {

        for (int p = 0; p < size; p++)

        {

            cout << "ФИО: ";

            getline(cin, buf.FIO.name);

            getline(cin, buf.FIO.surname);

            getline(cin, buf.FIO.partonymic);

            cout << "Образование: ";

            getline(cin, buf.education);

            cout << "Специальность: ";

            getline(cin, buf.speciality);

            cout << "Должность: ";

            getline(cin, buf.jobTitle);

            cout << "Зарплата: ";

            cin >> buf.Salary;

            cout << "Дата вступления:";

            cinIn(int, buf.inviteDate.day);  // Здесь предполагается, что у вас есть функция cinIn() для ввода целых чисел

            cinIn(int, buf.inviteDate.month);

            cinIn(int, buf.inviteDate.year);

            fwrite(&buf, sizeof(buf), 1, f); // Записываем структуру в файл

        }

        fclose(f);

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

// Функция для вывода данных о сотрудниках из файла

void output()

{

    FILE\* f;

    personnelDepartment buf;

    if (fopen\_s(&f, "base.txt", "rb") == 0) // Открываем файл в режиме чтения в двоичном режиме

    {

        cout << "\nФИО\t\tОбразование\tСпециальность\tДолжность\tЗарплата\tДата вступления\n";

        fread(&buf, sizeof(buf), 1, f); // Читаем структуру из файла

        while (!feof(f))

        {

            cout << buf.FIO.name << ' ' << buf.FIO.surname << ' ' << buf.FIO.partonymic << "  " << buf.education << "  " << buf.speciality << "  " << buf.jobTitle << "  " << buf.Salary << "  " << buf.inviteDate.day << '.' << buf.inviteDate.month << '.' << buf.inviteDate.year << endl;

            fread(&buf, sizeof(buf), 1, f); // Читаем следующую структуру из файла

        }

        cout << endl;

        fclose(f);

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

// Функция для поиска сотрудников по должности

void findByJobTitle(string Title)

{

    FILE\* f;

    personnelDepartment buf;

    bool found = false;

    if (fopen\_s(&f, "base.txt", "rb") == 0) // Открываем файл в режиме чтения в двоичном режиме

    {

        while (fread(&buf, sizeof(buf), 1, f) == 1)

        {

            if (buf.jobTitle == Title)

            {

                cout << "\nФИО\t\tОБразование\tСпециальность\tДолжность\tЗарплата\tДата вступления\n";

                cout << buf.FIO.name << ' ' << buf.FIO.surname << ' ' << buf.FIO.partonymic << "  " << buf.education << "  " << buf.speciality << "  " << buf.jobTitle << "  " << buf.Salary << "  " << buf.inviteDate.day << '.' << buf.inviteDate.month << '.' << buf.inviteDate.year << endl;

                found = true;

            }

        }

        fclose(f);

        if (!found)

            cout << "Не найдено\n";

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

int var6n2()

{

    int choice, number;

    string Title;

    do

    {

        cout << "\n1. Ввод данных с клавиатуры и запись в файл\n";

        cout << "2. Вывод данных из файла\n";

        cout << "3. Поиск по должности\n";

        cout << "0. Выход из программы\n\n";

        cout << "Введите номер операции: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Введите количество работников: ";

            cin >> number;

            cin.ignore(); // Игнорируем символ новой строки после ввода числа

            input(number);

            break;

        case 2:

            output();

            break;

        case 3:

            cout << "Введите должность: ";

            cin.ignore(); // Игнорируем символ новой строки перед вводом строки

            getline(cin, Title);

            findByJobTitle(Title);

            break;

        case 0:

            exit(0);

            break;

        default:

            cout << "Некорректный ввод. Пожалуйста, выберите допустимую операцию." << endl;

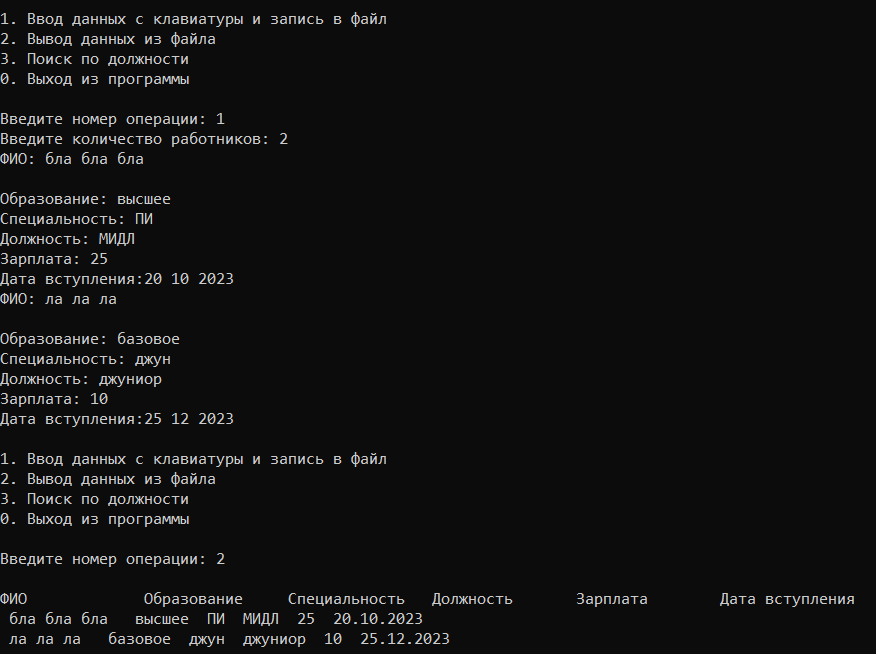
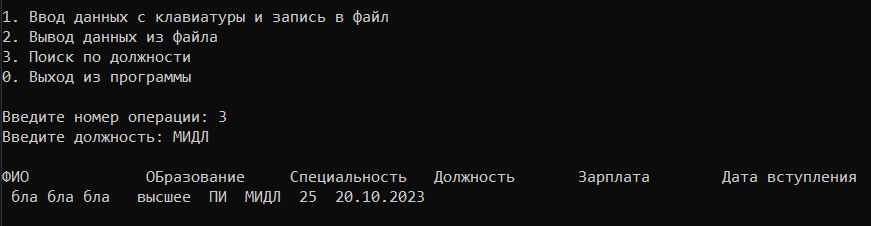
            break;

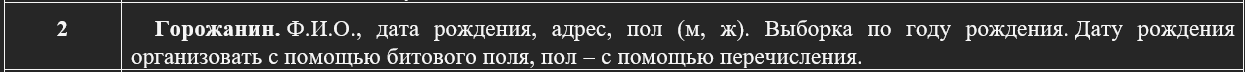
        }

    } while (choice != 0);

    return 0;

}

****



#define MAX\_CITIZENS 10

int current\_citizens = 0;

enum Gender

{

    male = 1, female

};

struct Citizens

{

    fullName FIO;   // фио

    Date BirthdayDate;   // дата рождения

    string address;  // адресс

    Gender gender;   // пол

};

    Citizens listOfCitizens[MAX\_CITIZENS];

void input();

void output();

void findByBirthdayDate();

void deleteCitizen();

int var2n1()

{

    int choice;

    do

    {

        cout << "\n1. Ввод данных с клавиатуры\n";

        cout << "2. Вывод данных\n";

        cout << "3. Поиск по дате рождения\n";

        cout << "4. удаление по фамилии\n";

        cout << "0. Выход из программы\n\n";

        cout << "Введите номер операции: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            input();

            break;

        case 2:

            output();

            break;

        case 3:

            findByBirthdayDate();

            break;

        case 4:

            deleteCitizen();

            break;

        case 0:

            exit(0);

            break;

        default:

            cout << "Некорректный ввод. Пожалуйста, выберите допустимую операцию." << endl;

            break;

        }

    } while (choice != 0);

    return 0;

}

void input()

{

    int citizensCount;

    cout << "Введите кол-во персонала: ";

    cin >> citizensCount;

    cin.ignore();

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

    {

        if (current\_citizens < MAX\_CITIZENS)

        {

            cout << "ФИО: ";

            cin >> listOfCitizens[current\_citizens].FIO.surname >> listOfCitizens[current\_citizens].FIO.name >> listOfCitizens[current\_citizens].FIO.partonymic;

            cout << "Дата рождения: ";

            cinIn(int, listOfCitizens[current\_citizens].BirthdayDate.day);

            cinIn(int, listOfCitizens[current\_citizens].BirthdayDate.month);

            cinIn(int, listOfCitizens[current\_citizens].BirthdayDate.year);

            cout << "Адресс: ";

            cin >> listOfCitizens[current\_citizens].address;

            cout << "Выберите пол(1-male, 2- female): ";

            int gender;

            cin >> gender;

            listOfCitizens[current\_citizens].gender = static\_cast<Gender>(gender);

            current\_citizens++;

        }

    }

}

void output()

{

    for (int i = 0; i < current\_citizens; i++)

    {

        cout << "\nФИО:\t" << listOfCitizens[i].FIO.surname << ' ' << listOfCitizens[i].FIO.name << ' ' << listOfCitizens[i].FIO.partonymic << endl;

        cout << "Дата рождения:\t" << listOfCitizens[i].BirthdayDate.day << "." << listOfCitizens[i].BirthdayDate.month << "." << listOfCitizens[i].BirthdayDate.year << endl;

        cout << "Адресс:\t" << listOfCitizens[i].address << endl;

        cout << "Пол:\t" << listOfCitizens[i].gender << endl;

    }

}

void findByBirthdayDate()

{

    int day, month, year;

    cout << "Введите дату рождения: ";

    cin >> day >> month >> year;

    bool found = false;

    for (int i = 0; i < current\_citizens; i++)

    {

        if (day == listOfCitizens[i].BirthdayDate.day && month == listOfCitizens[i].BirthdayDate.month && year == listOfCitizens[i].BirthdayDate.year)

        {

            cout << "\nФИО:\t" << listOfCitizens[i].FIO.surname << ' ' << listOfCitizens[i].FIO.name << ' ' << listOfCitizens[i].FIO.partonymic << endl;

            cout << "Дата рождения:\t" << listOfCitizens[i].BirthdayDate.day << "." << listOfCitizens[i].BirthdayDate.month << "." << listOfCitizens[i].BirthdayDate.year << endl;

            cout << "Адресс:\t" << listOfCitizens[i].address << endl;

            cout << "Пол:\t" << listOfCitizens[i].gender << endl;

            found = true;

        }

    }

    if (!found)

    {

        cout << "Не найдено\n";

    }

}

void deleteCitizen()

{

    string Surname;

    cout << "Введите фамилию для удаления: ";

    cin >> Surname;

    bool findBySurname = false;

    for (int i = 0; i < current\_citizens; i++)

    {

        if (Surname == listOfCitizens[i].FIO.surname)

        {

            findBySurname = true;

            for (int j = i; j < current\_citizens - 1; j++)

            {

                listOfCitizens[j] = listOfCitizens[j + 1];

            }

            current\_citizens--;

            cout << "\nЧеловек удален.\n";

            break;

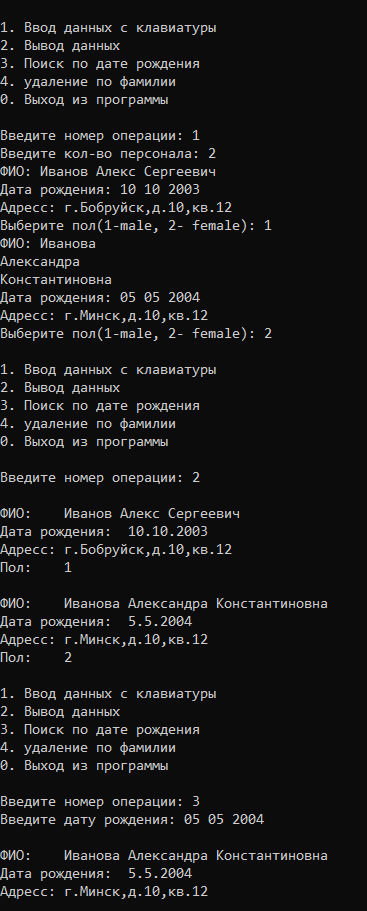
        }

    }

    if (!findBySurname)

    {

        cout << "\nНе найдено \n";

****

2)

struct Time

{

    unsigned int hours : 4;    // 1 - 12

    unsigned int minutes : 6;  // 1 - 60

};

#define MAX\_TRAINS  10 // максимальное кол-во

int current\_trains = 0;

struct trains

{

    string numberTrain;        // номер поезда

    string destination;     // пункт назначения

    Date dayTrain;           // дни следования

    Time timeArrive;        // время прибытия

    Time timeDeparture;     // время отправления

};

trains ListOfTrains[MAX\_TRAINS];//объявляем структуру

//прототипы ф-ций

void input();

void output();

void findByDestination();

int var2n2()

{

    int choice;

    do

    {

        cout << "\n1. Ввод данных с клавиатуры и запись в файл\n";

        cout << "2. Вывод данных из файла\n";

        cout << "3. Поиск по пункту назначения\n";

        cout << "0. Выход из программы\n\n";

        cout << "Введите номер операции: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            input();

            break;

        case 2:

            output();

            break;

        case 3:

            findByDestination();

            break;

        case 0:

            exit(0);

            break;

        default:

            cout << "Некорректный ввод. Пожалуйста, выберите допустимую операцию." << endl;

            break;

        }

    } while (choice != 0);

    return 0;

}

//ф-ция ввода

void input()

{

    FILE\* f;

    int TrainsCount;

    cout << "Введите кол-во поездов: ";

    cin >> TrainsCount;

    cin.ignore();

    if (fopen\_s(&f, "var2n2.txt", "ab+") == 0) // Открываем файл в режиме добавления/чтения в двоичном режиме

    {

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

        {

            if (current\_trains < MAX\_TRAINS)

            {

                cout << "Номер поезда: ";

                cin >>ListOfTrains[current\_trains].numberTrain;

                cout << "Пункт назначения: ";

                cin >> ListOfTrains[current\_trains].destination;

                cout << "Дни следования: ";

                cinIn(int, ListOfTrains[current\_trains].dayTrain.day);

                cinIn(int, ListOfTrains[current\_trains].dayTrain.month);

                cinIn(int, ListOfTrains[current\_trains].dayTrain.year);

                cout << "Время прибытия: ";

                cinIn(int, ListOfTrains[current\_trains].timeArrive.hours);

                cinIn(int, ListOfTrains[current\_trains].timeArrive.minutes);

                cout << "Время отправления: ";

                cinIn(int, ListOfTrains[current\_trains].timeDeparture.hours);

                cinIn(int, ListOfTrains[current\_trains].timeDeparture.minutes);

                std::fwrite(&ListOfTrains, sizeof(ListOfTrains), 1, f); //запись в файл

                current\_trains++;

            }

        }

        if (std::fwrite(&ListOfTrains[current\_trains], sizeof(ListOfTrains[current\_trains]), 1, f) != 1)

        {

            cout << "Ошибка записи в файл" << endl;

        }

        fclose(f); // закрытие файла

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

//ф-ция вывода

void output()

{

    FILE\* f;

    if (fopen\_s(&f, "var2n2.txt", "rb") == 0) //открытие файла для чтения

    {

        fread(&ListOfTrains, sizeof(ListOfTrains), 1, f); // читаем содержимое из файла

        for (int i = 0; i < current\_trains; i++)

        {

            cout << endl;

            cout << "Номер поезда: " << ListOfTrains[i].numberTrain <<endl;

            cout << "Пункт назначения: " << ListOfTrains[i].destination << endl;

            cout << "Дни следования:\t" << ListOfTrains[i].dayTrain.day << "." << ListOfTrains[i].dayTrain.month << "." << ListOfTrains[i].dayTrain.year << endl;

            cout << "Время прибытия: " << ListOfTrains[i].timeArrive.hours << ":" << ListOfTrains[i].timeArrive.minutes << endl;

            cout << "Время отправление: " << ListOfTrains[i].timeDeparture.hours << ":" << ListOfTrains[i].timeDeparture.minutes << endl;

            cout << endl;

            fread(&ListOfTrains, sizeof(ListOfTrains), 1, f);

        }

        fclose(f);

    }

    else

    {

        cout << "Ошибка открытия файла";

        return;

    }

}

//ф-ция поиска

void findByDestination()

{

    FILE\* f;

    string Destination;

    cout << "Введите пункт назначения: ";

    cin >> Destination;

    bool found = false;

    if (fopen\_s(&f, "var2n2.txt", "rb") == 0)//открываем файл

    {

        for (int i = 0; i < current\_trains; i++)

        {

            if (Destination == ListOfTrains[i].destination)//если нашли, то выводим инфу

            {

                cout << "Номер поезда: " << ListOfTrains[current\_trains].numberTrain;

                cout << "Пункт назначения: " << ListOfTrains[current\_trains].destination;

                cout << "Дни следования:\t" << ListOfTrains[i].dayTrain.day << "." << ListOfTrains[i].dayTrain.month << "." << ListOfTrains[i].dayTrain.year << endl;

                cout << "Время прибытия: " << ListOfTrains[i].timeArrive.hours << "." << ListOfTrains[i].timeArrive.minutes << endl;

                cout << "Время отправление: " << ListOfTrains[i].timeDeparture.hours << "." << ListOfTrains[i].timeDeparture.minutes << endl;

                cout << endl;

                found = true;

            }

        }

    }

    if (!found)

    {

        cout << "Не найдено\n";

    }

}

