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:

Студент 1 курса 8 группы

Лужецкий Владислав Константинович

Преподаватель: асс. Андронова М.В.

2024, Минск

**Основной вариант(6):**

**1)**

#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;

}

}

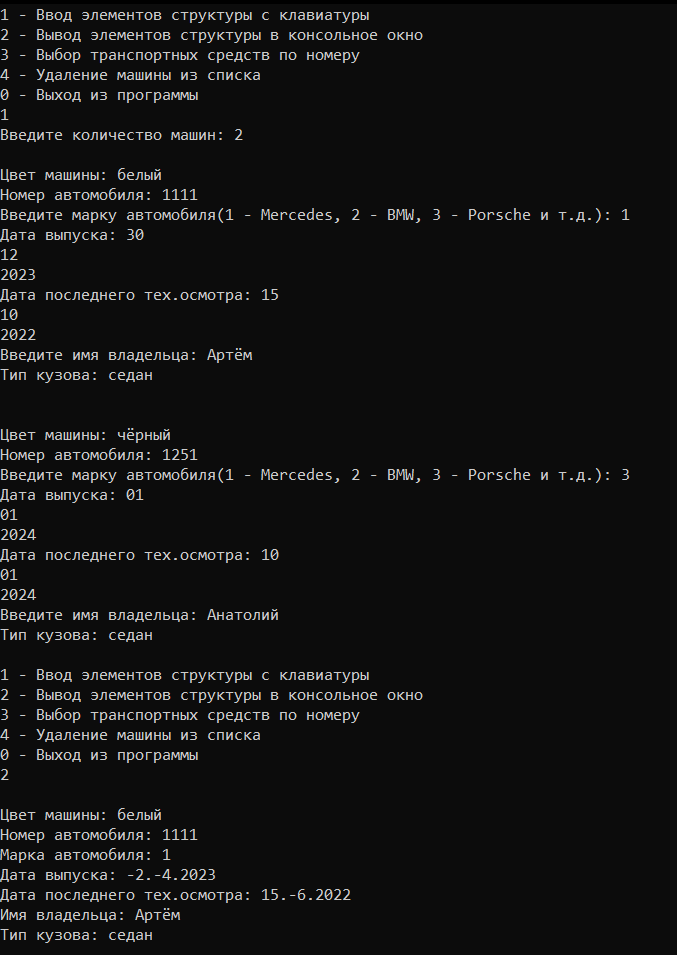
if (!findCarWithThatNumber)

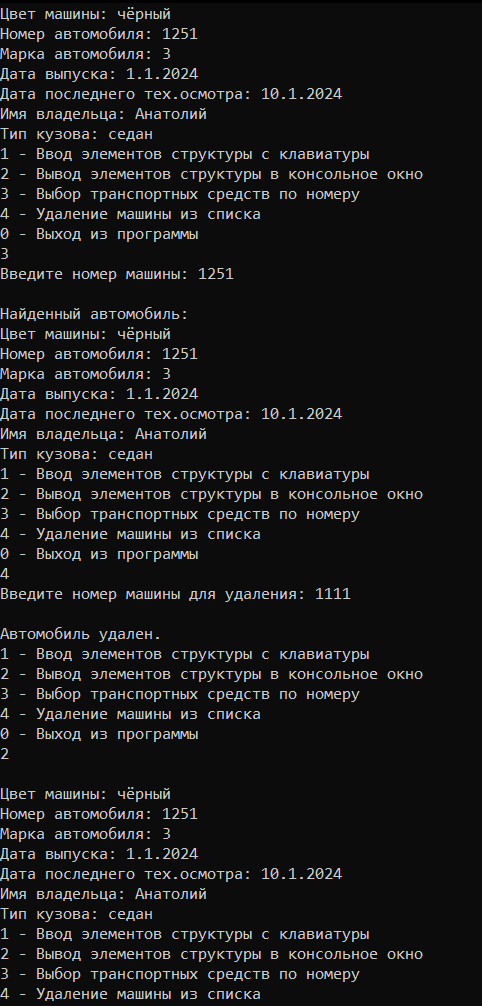
{

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;

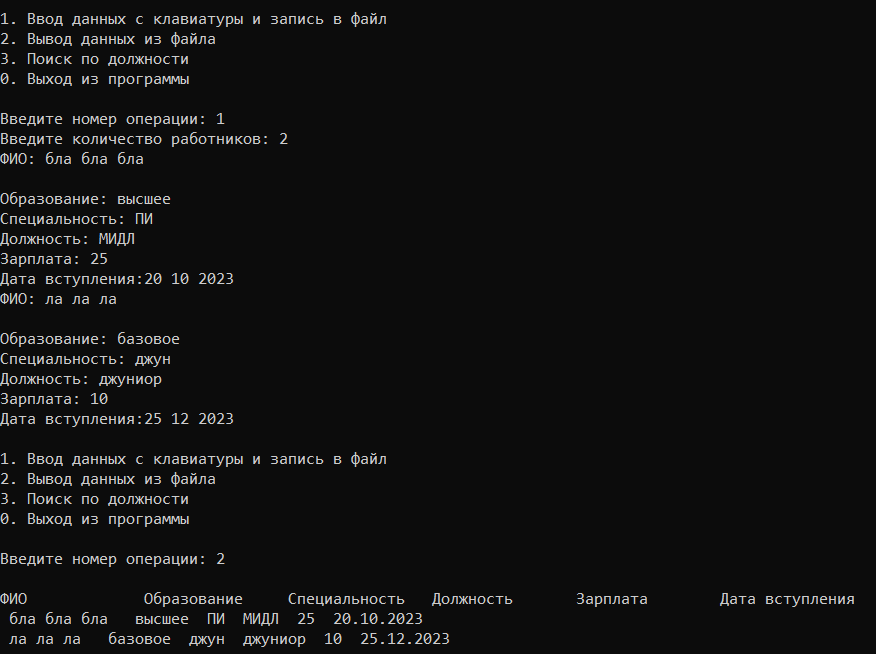
break;

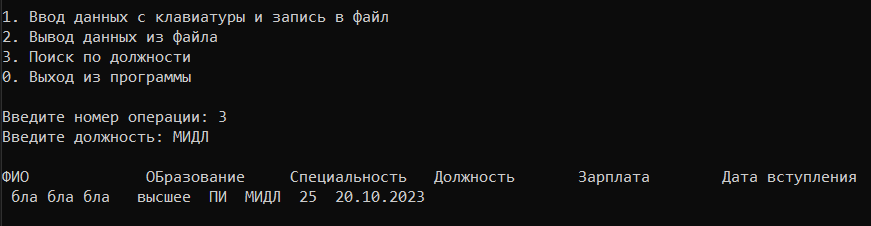
}

} while (choice != 0);

return 0;

}

****



**Допы:**

**// На самом деле допы будут отличаться лишь названиями переменных и структур. Смысл заданий одинаковый и функции +-тоже. Поэтому, возможно, в некоторых допах будет мало комментариев из-за отсутствия смысла писать одно и тоже несколько раз.**

**1)**

using namespace std;

#define MAX\_PERSONNEL 10

int current\_personnel = 0;

enum jobTitle

{

junior = 1, middle, senior

};

// Структура для хранения полного имени (ФИО)

struct fullName

{

string surname; // фамилия

string name; // имя

string partonymic; // отчество

};

// Структура для хранения даты

struct Date

{

int day;

int month;

int year;

};

// Структура для хранения информации о сотруднике

struct personnelDepartment

{

fullName FIO; // ФИО

string education; // образование

string speciality; // специальность

jobTitle Title; // должность

int Salary; // оклад (зарплата)

Date inviteDate; // дата приглашения на работу

};

personnelDepartment listOfPersonnel[MAX\_PERSONNEL];

// Функции объявления

void input();

void output();

void findByJobTitle();

void deletePersonnel();

int main()

{

int choice;

do

{

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

cout << "2. Вывод данных\n";

cout << "3. Поиск по должности\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:

findByJobTitle();

break;

case 4:

deletePersonnel();

break;

case 0:

exit(0);

break;

default:

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

break;

}

} while (choice != 0);

return 0;

}

// Функция для ввода данных о сотрудниках

void input()

{

int personnelCount;

cout << "Введите количество персонала: ";

cin >> personnelCount;

cin.ignore();

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

{

if (current\_personnel < MAX\_PERSONNEL)

{

cout << "ФИО: ";

cin >> listOfPersonnel[current\_personnel].FIO.surname >> listOfPersonnel[current\_personnel].FIO.name >> listOfPersonnel[current\_personnel].FIO.partonymic;

cout << "Образование: ";

cin >> listOfPersonnel[current\_personnel].education;

cout << "Специальность: ";

cin >> listOfPersonnel[current\_personnel].speciality;

cout << "Выберите должность (1-junior, 2-middle, 3-senior): ";

int Title;

cin >> Title;

listOfPersonnel[current\_personnel].Title = static\_cast<jobTitle>(Title);

cout << "Зарплата: ";

cin >> listOfPersonnel[current\_personnel].Salary;

cout << "Дата вступления: ";

cin >> listOfPersonnel[current\_personnel].inviteDate.day;

cin >> listOfPersonnel[current\_personnel].inviteDate.month;

cin >> listOfPersonnel[current\_personnel].inviteDate.year;

current\_personnel++;

}

}

}

// Функция для вывода данных о сотрудниках

void output()

{

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

{

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

cout << "Образование:\t" << listOfPersonnel[i].education << endl;

cout << "Специальность:\t" << listOfPersonnel[i].speciality << endl;

cout << "Должность:\t" << listOfPersonnel[i].Title << endl;

cout << "Зарплата:\t" << listOfPersonnel[i].Salary << endl;

cout << "Дата вступления:\t" << listOfPersonnel[i].inviteDate.day

}

}

// Функция для поиска сотрудников по должности

void findByJobTitle()

{

int searchTitle;

cout << "Введите номер должности для поиска (1-junior, 2-middle, 3-senior): ";

cin >> searchTitle;

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

{

if (listOfPersonnel[i].Title == static\_cast<jobTitle>(searchTitle))

{

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

cout << "Образование:\t" << listOfPersonnel[i].education << endl;

cout << "Специальность:\t" << listOfPersonnel[i].speciality << endl;

cout << "Должность:\t" << listOfPersonnel[i].Title << endl;

cout << "Зарплата:\t" << listOfPersonnel[i].Salary << endl;

cout << "Дата вступления:\t" << listOfPersonnel[i].inviteDate.day << '/' << listOfPersonnel[i].inviteDate.month << '/' << listOfPersonnel[i].inviteDate.year << endl;

}

}

}

// Функция для удаления сотрудников по фамилии

void deletePersonnel()

{

string searchSurname;

cout << "Введите фамилию сотрудника для удаления: ";

cin >> searchSurname;

int deletedCount = 0;

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

{

if (listOfPersonnel[i].FIO.surname == searchSurname)

{

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

{

listOfPersonnel[j] = listOfPersonnel[j + 1];

}

deletedCount++;

}

}

current\_personnel -= deletedCount;

if (deletedCount == 0)

{

cout << "Сотрудник с фамилией " << searchSurname << " не найден." << endl;

}

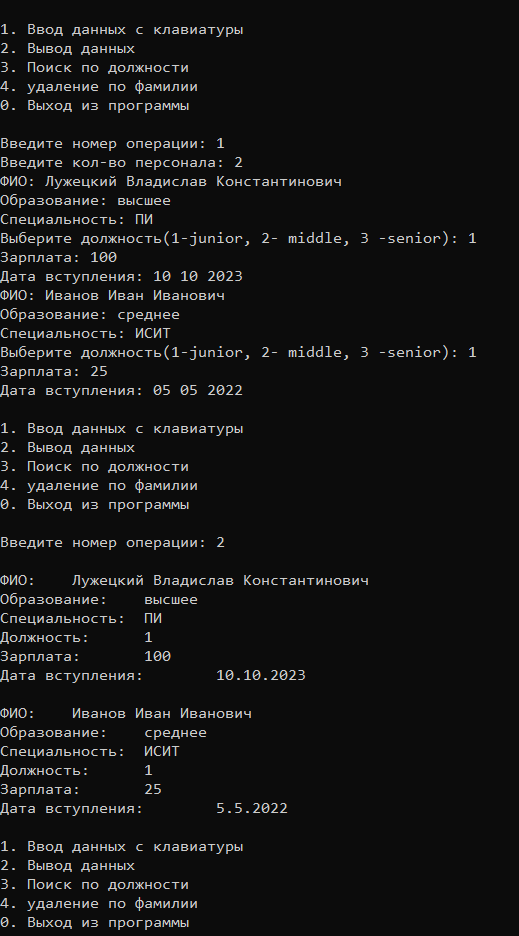
else

{

cout << "Удалено " << deletedCount << " сотрудников с фамилией " << searchSurname << "." << endl;

}

}



**2)**

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

int current\_teacher = 0;

struct personnelDepartment

{

fullName FIO; // фио

string examName; // Название экзамена

Date examDate; // дата экзамена

};

personnelDepartment listOfTeachers[MAX\_TEACHERS];//объявляем структуру

//прототипы ф-ций

void input();

void output();

void findBySurname();

int var1n2()

{

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 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:

findBySurname();

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 teachersCount;

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

cin >> teachersCount;

cin.ignore();

if (fopen\_s(&f, "var1n2.txt", "ab+") == 0) // Открываем файл в режиме добавления/чтения в двоичном режиме

{

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

{

if (current\_teacher < MAX\_TEACHERS)

{

cout << "ФИО: ";

cin >> listOfTeachers[current\_teacher].FIO.surname >> listOfTeachers[current\_teacher].FIO.name >> listOfTeachers[current\_teacher].FIO.partonymic;

cout << "Название экзамена: ";

cin >> listOfTeachers[current\_teacher].examName;

cout << "Дата экзамена: ";

cinIn(int, listOfTeachers[current\_teacher].examDate.day);

cinIn(int, listOfTeachers[current\_teacher].examDate.month);

cinIn(int, listOfTeachers[current\_teacher].examDate.year);

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

current\_teacher++;

}

}

if (std::fwrite(&listOfTeachers[current\_teacher], sizeof(listOfTeachers[current\_teacher]), 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, "var1n2.txt", "rb") == 0) //открытие файла для чтения

{

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

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

{

cout << endl;

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

cout << "Название экзамена:\t" << listOfTeachers[i].examName << endl;

cout << "Дата экзамена:\t" << listOfTeachers[i].examDate.day << "." << listOfTeachers[i].examDate.month << "." << listOfTeachers[i].examDate.year << endl;

cout << endl;

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

}

fclose(f);

}

else

{

cout << "Ошибка открытия файла";

return;

}

}

//ф-ция поиска

void findBySurname()

{

FILE\* f;

string Surname;

cout << "Введите фамилию: ";

cin >> Surname;

bool found = false;

if(fopen\_s(&f, "var1n2.txt", "rb") == 0)//открываем файл

{

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

{

if (Surname == listOfTeachers[i].FIO.surname)//если нашли, то выводим инфу

{

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

cout << "Название экзамена:\t" << listOfTeachers[i].examName << endl;

cout << "Дата экзамена:\t" << listOfTeachers[i].examDate.day << "." << listOfTeachers[i].examDate.month << "." << listOfTeachers[i].examDate.year << endl;

cout << endl;

found = true;

}

}

}

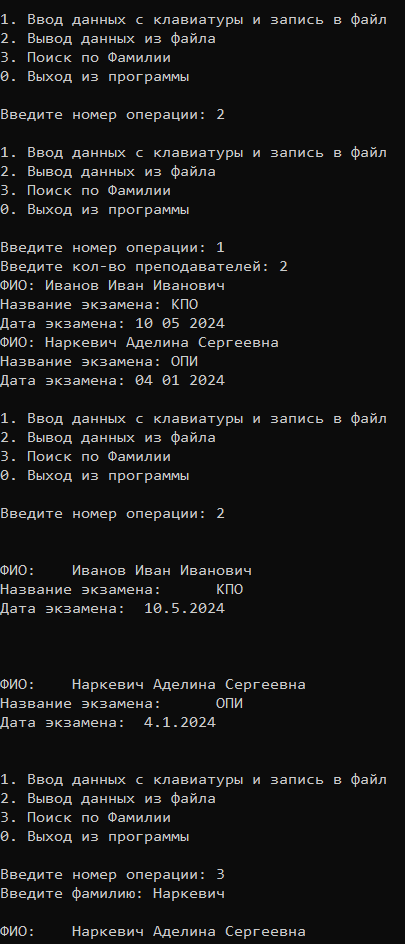
if (!found)

{

cout << "Не найдено\n";

}

}



**2 доп**

**1)**

#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;

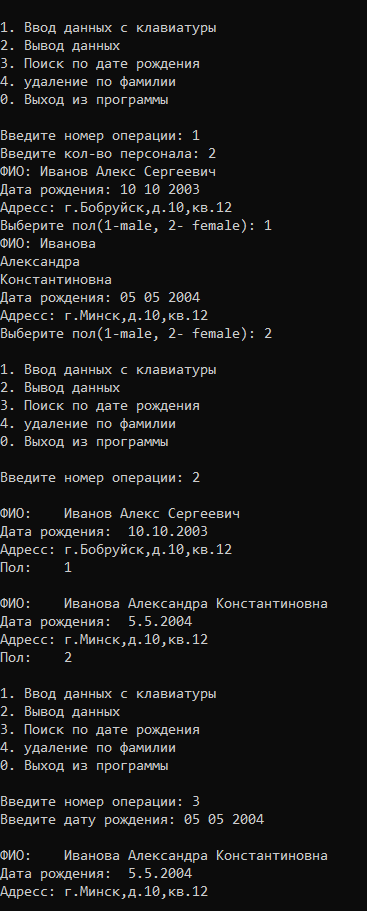
}

}

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;

}

}

}

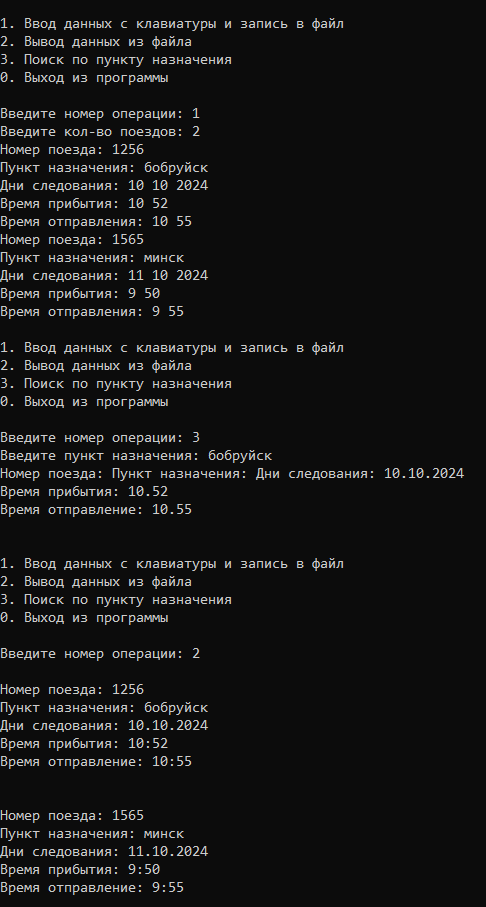
if (!found)

{

cout << "Не найдено\n";

}

}



**3 доп:**

**1)**

#define MAX\_STATES 10

int current\_states = 0;

enum formOfGovernment

{

республика = 1, монархия, диктатура, тоталитаризм, демократия

};

struct States

{

string name; // Название

string capital; // столица

int population; // численность

string square; // площадь

formOfGovernment government; // форма правления

};

States listOfStates[MAX\_STATES];

void input();

void output();

void findByCapital();

void deleteState();

int var10n1()

{

int choice;

do

{

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

cout << "2. Вывод данных\n";

cout << "3. Поиск по столице\n";

cout << "4. удаление по названию города\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:

findByCapital();

break;

case 4:

deleteState();

break;

case 0:

exit(0);

break;

default:

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

break;

}

} while (choice != 0);

return 0;

}

void input()

{

int statesCount;

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

cin >> statesCount;

cin.ignore();

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

{

if (current\_states < MAX\_STATES)

{

cout << "Название: ";

cin >> listOfStates[current\_states].name;

cout << "Столица: ";

cin >> listOfStates[current\_states].capital;

cout << "Население: ";

cin >> listOfStates[current\_states].population;

cout << "Площадь: ";

cin >> listOfStates[current\_states].square;

cout << "Выберите форму правления(республика - 1, монархия -2 , диктатура - 3, тоталитаризм -4 , демократия -5 ): ";

int Government;

cin >> Government;

listOfStates[current\_states].government = static\_cast<formOfGovernment>(Government);

current\_states++;

}

}

}

void output()

{

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

{

cout << "\nНазвание: " << listOfStates[i].name << endl;

cout << "Столица: " << listOfStates[i].capital << endl;

cout << "Население: " << listOfStates[i].population << endl;

cout << "Площадь: " << listOfStates[i].square << endl;

cout << "Форма правления: " << listOfStates[i].government;

cout << endl;

}

}

void findByCapital()

{

string capital;

cout << "Введите столицу: ";

cin >> capital;

bool found = false;

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

{

if (capital == listOfStates[i].capital)

{

cout << "\nНазвание: " << listOfStates[i].name << endl;

cout << "Столица: " << listOfStates[i].capital << endl;

cout << "Население: " << listOfStates[i].population << endl;

cout << "Площадь: " << listOfStates[i].square << endl;

cout << "Форма правления: " << listOfStates[i].government;

cout << endl;

found = true;

}

}

if (!found)

{

cout << "Не найдено\n";

}

}

void deleteState()

{

string Name;

cout << "Введите название для удаления: ";

cin >> Name;

bool findByState = false;

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

{

if (Name == listOfStates[i].name)

{

findByState = true;

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

{

listOfStates[j] = listOfStates[j + 1];

}

current\_states--;

cout << "\nгосударство удалено.\n";

break;

}

}

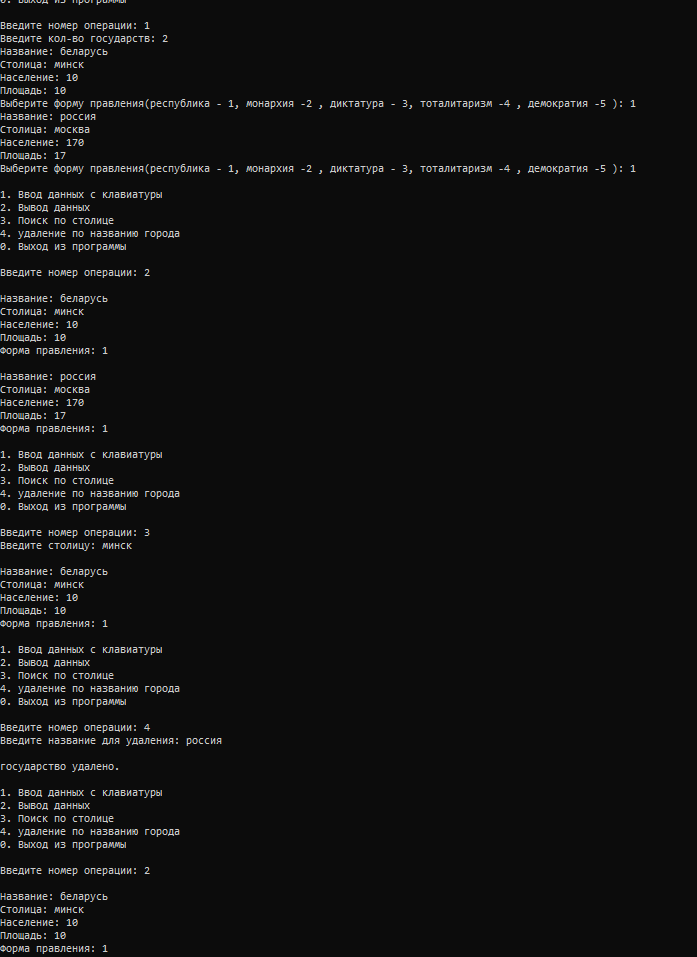
if (!findByState)

{

cout << "\nНе найдено \n";

}

}



2)

#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];

//прототипы ф-ций

void input();

void output();

void findByBirthdayDate();

int var10n2()

{

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 дню рождения\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 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 CitizensCount;

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

cin >> CitizensCount;

cin.ignore();

if (fopen\_s(&f, "var10n2.txt", "ab+") == 0) // Открываем файл в режиме добавления/чтения в двоичном режиме

{

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++;

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

}

}

if (std::fwrite(&listOfCitizens[current\_citizens], sizeof(listOfCitizens[current\_citizens]), 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, "var10n2.txt", "rb") == 0) //открытие файла для чтения

{

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

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;

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

}

fclose(f);

}

else

{

cout << "Ошибка открытия файла";

return;

}

}

//ф-ция поиска

void findByBirthdayDate()

{

FILE\* f;

int day, month, year;

cout << "Введите дату рождения: ";

cin >> day >> month >> year;

bool found = false;

if (fopen\_s(&f, "var10n2.txt", "rb") == 0)//открываем файл

{

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";

}

fclose(f);

}

