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

Статько Герман Вячеславович

Преподаватель: асс. Андронова М.В.

Минск, 2024

**В соответствии со своим вариантом разработать программу с использованием перечислений и битовых полей для работы с данными из таблицы, приведенной ниже. Реализовать функции ввода с клавиатуры, вывода на экран, удаления, поиска элементов. Интерфейс пользователя осуществить в виде меню.**

**Вариант 12**

****

#include <iostream>

#include <Windows.h>

#include <string>

using namespace std;

// Объявление функций

void newContact();

void list();

void deleteContact();

void search();

// Структура для хранения информации о контакте

struct Contact

{

char fam[20]; // Фамилия

char name[20]; // Имя

char otc[20]; // Отчество

unsigned birthDay : 5; // День рождения (максимум 31)

unsigned birthMonth : 4; // Месяц рождения (максимум 12)

unsigned birthYear : 12; // Год рождения

string address; // Адрес

string phoneNumber; // Номер телефона

};

const int MAX\_CONTACTS\_COUNT = 6;

int contacts\_count = 0;

Contact contacts\_list[MAX\_CONTACTS\_COUNT];

int main()

{

SetConsoleCP(1251); // Установка кодовой страницы для ввода

SetConsoleOutputCP(1251); // Установка кодовой страницы для вывода

int choice;

do

{

// Вывод меню

cout << "1 - добавить новый контакт" << endl

<< "2 - отобразить все добавленные контакты" << endl

<< "3 - удалить контакт" << endl

<< "4 - поиск по фамилии" << endl

<< "5 - выход" << endl

<< "Выберите пункт меню: ";

cin >> choice;

switch (choice)

{

case 1:

newContact();

break;

case 2:

list();

break;

case 3:

deleteContact();

break;

case 4:

search();

break;

case 5:

cout << "Выход из программы..." << endl;

system("pause");

break;

}

} while (choice != 5);

}

// Функция для добавления нового контакта

void newContact()

{

if (contacts\_count == MAX\_CONTACTS\_COUNT)

{

cout << "Вы ввели максимальное кол-во контактов" << endl;

return;

}

Contact newCont;

cout << "Введите фамилию: ";

cin >> newCont.fam;

cout << "Введите имя: ";

cin >> newCont.name;

cout << "Введите отчество: ";

cin >> newCont.otc;

int day;

cout << "Введите день рождения: ";

cin >> day;

newCont.birthDay = static\_cast<unsigned>(day);

cin.ignore();

int month;

cout << "Введите месяц рождения: ";

cin >> month;

newCont.birthMonth = static\_cast<unsigned>(month);

cin.ignore();

int year;

cout << "Введите год рождения: ";

cin >> year;

newCont.birthYear = static\_cast<unsigned>(year);

cin.ignore();

cout << "Введите адрес: ";

getline(cin, newCont.address);

cout << "Введите номер телефона: ";

cin.ignore();

getline(cin, newCont.phoneNumber);

contacts\_list[contacts\_count] = newCont;

contacts\_count++;

cout << "Данные успешно добавлены!" << endl;

}

// Функция для отображения всех добавленных контактов

void list()

{

for (int i = 0; i < contacts\_count; i++)

{

cout << "ФИО: " << contacts\_list[i].fam << " " << contacts\_list[i].name << " " << contacts\_list[i].otc << endl;

cout << "Дата рождения: " << contacts\_list[i].birthDay << "." << contacts\_list[i].birthMonth << "." << contacts\_list[i].birthYear << endl;

cout << "Адрес: " << contacts\_list[i].address << endl;

cout << "Телефон: " << contacts\_list[i].phoneNumber << endl;

}

}

// Функция для удаления контакта

void deleteContact()

{

int choice;

cout << "Введите номер контакта, который нужно удалить: ";

cin >> choice;

if (choice < 1 || choice > contacts\_count)

{

cout << "Некорректный номер контакта." << endl;

return;

}

for (int i = choice - 1; i < contacts\_count - 1; i++)

{

contacts\_list[i] = contacts\_list[i + 1];

}

contacts\_count--;

cout << "Контакт успешно удален!" << endl;

}

// Функция для поиска по фамилии

void search()

{

char inputFam[20];

bool found = false;

cout << "Введите фамилию для поиска: ";

cin >> inputFam;

for (int i = 0; i < contacts\_count; i++)

{

if (strcmp(contacts\_list[i].fam, inputFam) == 0)

{

found = true;

cout << "ФИО: " << contacts\_list[i].fam << " " << contacts\_list[i].name << " " << contacts\_list[i].otc << endl;

cout << "Дата рождения: " << contacts\_list[i].birthDay << "." << contacts\_list[i].birthMonth << "." << contacts\_list[i].birthYear << endl;

cout << "Адрес: " << contacts\_list[i].address << endl;

cout << "Телефон: " << contacts\_list[i].phoneNumber << endl;

break;

}

}

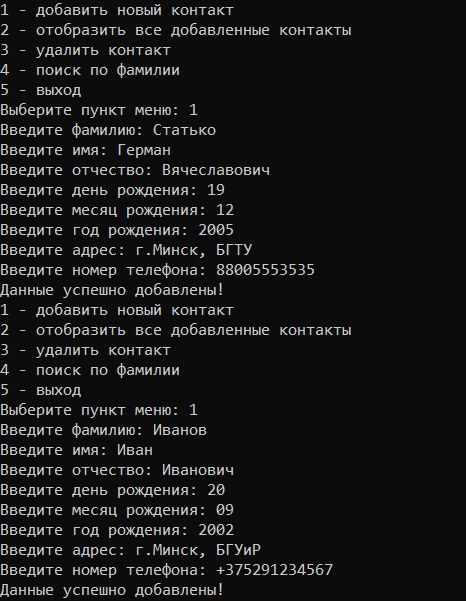
if (!found)

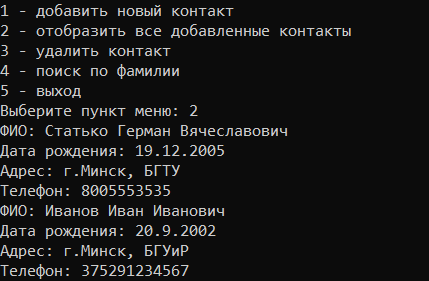
{

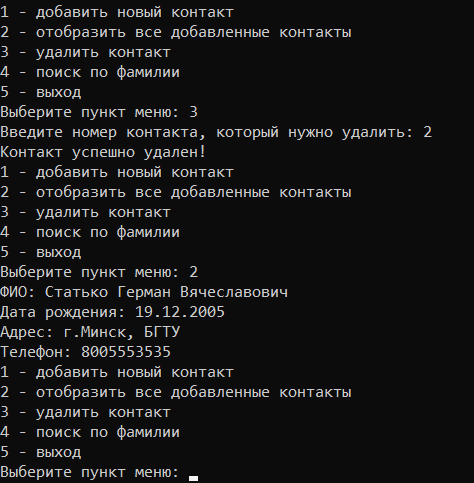
cout << "Контакт с такой фамилией не найден" << endl;

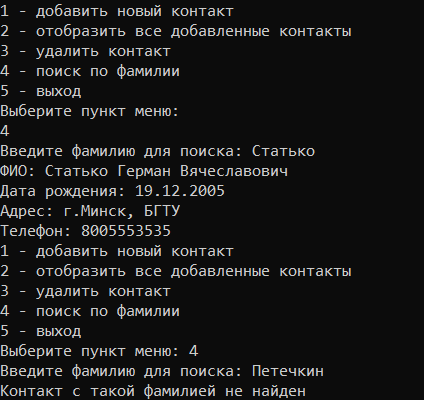
}

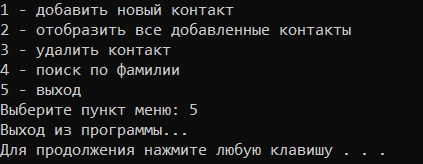
}

****

****

****

****

****

**В соответствии со своим вариантом разработать программу с использованием структуры в виде объединения, для работы с данными из таблицы, приведенной ниже. Реализовать функции ввода с клавиатуры, записи в файл, вывода на экран, чтения из файла и поиска.**

****

#include <iostream>

#include <Windows.h>

#include <string>

#include <fstream>

using namespace std;

// Объявление функций

void newContact();

void list();

void write();

void read();

void search();

// Структура для хранения данных контакта

union datas

{

char datach[60];

unsigned datanum;

};

struct Contact

{

datas fio;

datas birthDay;

datas birthMonth;

datas birthYear;

datas address;

datas phoneNumber;

datas place;

datas jobtitle;

};

// Константа для максимального количества контактов

const int MAX\_CONTACTS\_COUNT = 6;

int contacts\_count = 0;

Contact contacts\_list[MAX\_CONTACTS\_COUNT];

// Основная функция

int main()

{

// Установка кодировки для корректного отображения на кириллице

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

// Основной цикл программы

do

{

// Вывод меню

cout << "1 - добавить новый контакт" << endl

<< "2 - отобразить все добавленные контакты" << endl

<< "3 - записать контакты в файл" << endl

<< "4 - поиск по фамилии" << endl

<< "5 - прочитать данные из файла" << endl

<< "6 - выход из программы" << endl

<< "Выберите пункт меню: ";

// Считывание выбора пользователя

cin >> choice;

// Обработка выбора

switch (choice)

{

case 1:

newContact();

break;

case 2:

list();

break;

case 3:

write();

break;

case 4:

search();

break;

case 5:

read();

break;

case 6:

cout << "Выход из программы..." << endl;

system("pause");

break;

}

} while (choice != 6);

}

// Функция для добавления нового контакта

void newContact()

{

// Проверка на максимальное количество контактов

if (contacts\_count == MAX\_CONTACTS\_COUNT)

{

cout << "Вы ввели максимальное кол-во контактов" << endl;

return;

}

cin.ignore();

// Ввод данных о новом контакте

cout << "Введите ФИО: ";

gets\_s(contacts\_list[contacts\_count].fio.datach, 60);

cout << "Введите день рождения: ";

cin >> contacts\_list[contacts\_count].birthDay.datanum;

cin.ignore();

cout << "Введите месяц рождения: ";

cin >> contacts\_list[contacts\_count].birthMonth.datanum;

cin.ignore();

cout << "Введите год рождения: ";

cin >> contacts\_list[contacts\_count].birthYear.datanum;

cin.ignore();

cout << "Введите адрес: ";

gets\_s(contacts\_list[contacts\_count].address.datach, 60);

cout << "Введите номер телефона: ";

gets\_s(contacts\_list[contacts\_count].phoneNumber.datach, 60);

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

gets\_s(contacts\_list[contacts\_count].place.datach, 60);

cout << "Введите должность: ";

gets\_s(contacts\_list[contacts\_count].jobtitle.datach, 60);

contacts\_count++;

cout << "Данные успешно добавлены!" << endl;

}

// Функция для отображения всех добавленных контактов

void list()

{

for (int i = 0; i < contacts\_count; i++)

{

// Вывод информации о контакте

cout << "ФИО: " << contacts\_list[i].fio.datach << endl;

cout << "Дата рождения: " << contacts\_list[i].birthDay.datanum << "." << contacts\_list[i].birthMonth.datanum << "." << contacts\_list[i].birthYear.datanum << endl;

cout << "Адрес: " << contacts\_list[i].address.datach << endl;

cout << "Телефон: " << contacts\_list[i].phoneNumber.datach << endl;

cout << "Место работы или учебы: " << contacts\_list[i].place.datach << endl;

cout << "Должность: " << contacts\_list[i].jobtitle.datach << endl;

}

}

// Функция для записи контактов в файл

void write()

{

ofstream fin("file.txt");

if (!fin.is\_open())

{

cout << "Ошибка открытия файла" << endl;

return;

}

for (int i = 0; i < contacts\_count; i++)

{

// Запись данных в файл

fin << i + 1 << "-й контакт:" << endl;

fin << "ФИО: " << contacts\_list[i].fio.datach << endl;

fin << "Дата рождения: " << contacts\_list[i].birthDay.datanum << "." << contacts\_list[i].birthMonth.datanum << "." << contacts\_list[i].birthYear.datanum << endl;

fin << "Адрес: " << contacts\_list[i].address.datach << endl;

fin << "Номер телефона: " << contacts\_list[i].phoneNumber.datach << endl;

fin << "Место работы или учебы: " << contacts\_list[i].place.datach << endl;

fin << "Должность: " << contacts\_list[i].jobtitle.datach << endl << endl;

}

fin.close();

}

// Функция для чтения данных из файла

void read()

{

ifstream fout("file.txt");

string line;

while (getline(fout, line))

{

// Вывод данных из файла

cout << line << endl;

}

fout.close();

}

// Функция для поиска контакта по ФИО

void search()

{

char inputFam[60];

bool found = false;

cout << "Введите ФИО для поиска: ";

cin >> inputFio;

for (int i = 0; i < contacts\_count; i++)

{

// Поиск контакта по фамилии

if (strcmp(contacts\_list[i].fio.datach, inputFio) == 0)

{

found = true;

cout << "ФИО: " << contacts\_list[i].fio.datach << endl;

cout << "Дата рождения: " << contacts\_list[i].birthDay.datanum << "." << contacts\_list[i].birthMonth.datanum << "." << contacts\_list[i].birthYear.datanum << endl;

cout << "Адрес: " << contacts\_list[i].address.datanum << endl;

cout << "Телефон: " << contacts\_list[i].phoneNumber.datanum << endl;

break;

}

}

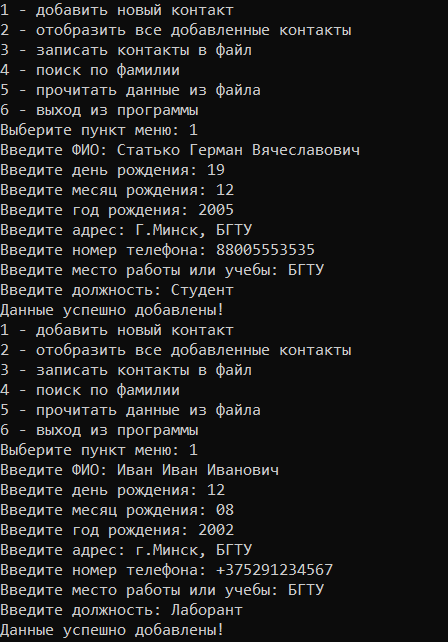
if (!found)

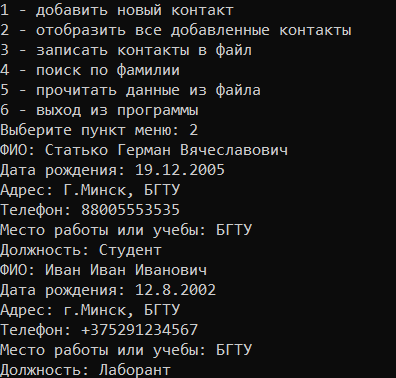
{

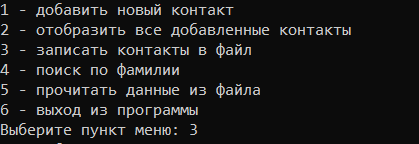
cout << "Контакт с такими ФИО на найден" << endl;

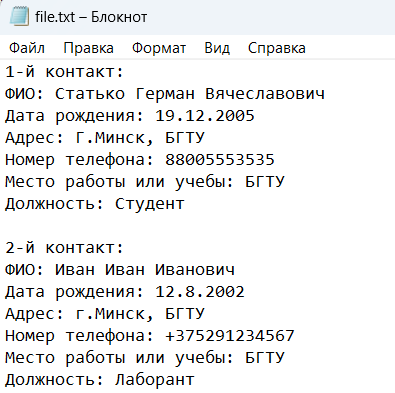
}

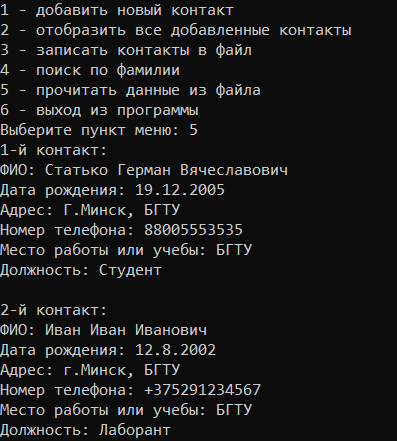
}

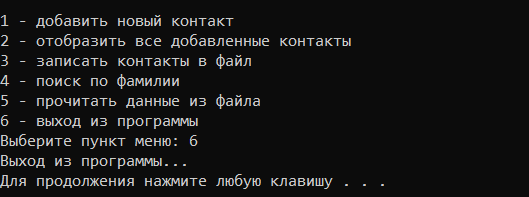
****

****

****

****

****

****

**Доп. Задачи**

**Вариант 3**

****

#include <iostream>

#include <string>

#include <Windows.h>

using namespace std;

void add();

void list();

void del();

void search();

// Константы и глобальные переменные

const int MAX\_STUD\_COUNT = 5;

const int MAX\_LESSONS\_COUNT = 5;

const int MAX\_MARKS\_COUNT = 5;

int stud\_count = 0;

// Перечисление для предметов

enum lessons

{

Алгебра = 1, Биология, Физика, Английский, Политология

};

// Структура для хранения данных об ученике

struct student

{

char fio[50];

unsigned studclass : 4;

char letter;

lessons lesson;

int marks[5];

double avg;

};

// Массив учеников

student stud\_list[MAX\_STUD\_COUNT];

// Главная функция

int main()

{

// Установка кодировки консоли для корректного вывода на кириллице

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

do

{

// Опции меню

cout << "1 - добавить ученика(макс. кол-во - 5)" << endl

<< "2 - отобразить всех учеников" << endl

<< "3 - удалить ученика" << endl

<< "4 - поиск по ФИО" << endl

<< "5 - выход" << endl

<< "Выберите пункт меню: ";

cin >> choice;

// Переключатель для меню

switch (choice)

{

case 1:

add();

break;

case 2:

list();

break;

case 3:

del();

break;

case 4:

search();

break;

case 5:

cout << "Выход из программы..." << endl;

system("pause");

break;

}

} while (choice != 5);

}

// Функция для добавления нового ученика

void add()

{

student newStud;

if (stud\_count == MAX\_STUD\_COUNT)

{

cout << "Вы ввели максимально допустимое кол-во учеников" << endl;

return;

}

cout << "Введите ФИО: ";

cin.ignore();

gets\_s(newStud.fio, 50);

cout << "Введите класс: ";

int cl;

cin >> cl;

newStud.studclass = static\_cast<unsigned>(cl);

cin.ignore();

cout << "Введите букву класса:";

cin >> newStud.letter;

cin.ignore();

int choice;

cout << "Введите предмет, который изучает ученик(1 - алгебра, 2 - биология, 3 - физика, 4 - английский, 5 - политология): ";

cin >> choice;

switch (choice)

{

case 1:

newStud.lesson = Алгебра;

break;

case 2:

newStud.lesson = Биология;

break;

case 3:

newStud.lesson = Физика;

break;

case 4:

newStud.lesson = Английский;

break;

case 5:

newStud.lesson = Политология;

break;

default:

cout << "Неверный выбор: " << endl;

return;

break;

}

cout << "Введите оценки ученика(кол-во - 5): ";

for (int i = 0; i < MAX\_MARKS\_COUNT; i++)

{

cin >> newStud.marks[i];

}

int sum = 0;

for (int i = 0; i < MAX\_MARKS\_COUNT; i++)

{

sum += newStud.marks[i];

}

sum = static\_cast<double>(sum);

newStud.avg = sum / static\_cast<double>(MAX\_MARKS\_COUNT);

stud\_list[stud\_count] = newStud;

stud\_count++;

cout << "Данные успешно записаны!" << endl << endl;

}

// Функция для вывода списка учеников

void list()

{

for (int i = 0; i < stud\_count; i++)

{

cout << "ФИО: " << stud\_list[i].fio << endl;

cout << "Класс: " << stud\_list[i].studclass << stud\_list[i].letter << endl;

cout << "Предмет, который изучает ученик: ";

if (stud\_list[i].lesson == Алгебра)

{

cout << "Алгебра" << endl;

}

else if (stud\_list[i].lesson == Биология)

{

cout << "Биология" << endl;

}

else if (stud\_list[i].lesson == Физика)

{

cout << "Физика" << endl;

}

else if (stud\_list[i].lesson == Английский)

{

cout << "Английский" << endl;

}

else if (stud\_list[i].lesson == Политология)

{

cout << "Политология" << endl;

}

cout << "Оценки ученика: ";

for (int k = 0; k < MAX\_MARKS\_COUNT; k++)

{

cout << stud\_list[i].marks[k] << " ";

}

cout << endl;

cout << "Средний балл ученика: " << stud\_list[i].avg << endl << endl;

}

}

// Функция для удаления ученика

void del()

{

int choice;

cout << "Введите номер ученика, которого нужно удалить: ";

cin >> choice;

if (choice < 1 || choice > stud\_count)

{

cout << "Некорректный номер ученика." << endl;

return;

}

for (int i = choice - 1; i < stud\_count - 1; i++)

{

stud\_list[i] = stud\_list[i + 1];

}

stud\_count--;

cout << "Данные успешно удалены! " << endl << endl;

}

// Функция для поиска ученика по ФИО

void search()

{

char buf[50];

bool found = false;

cout << "Введите ФИО ученика: ";

cin.ignore();

gets\_s(buf, 50);

for (int i = 0; i < stud\_count; i++)

{

if (strcmp(buf, stud\_list[i].fio) == 0)

{

found = true;

cout << "ФИО: " << stud\_list[i].fio << endl;

cout << "Класс: " << stud\_list[i].studclass << stud\_list[i].letter << endl;

cout << "Предмет, который изучает ученик: ";

if (stud\_list[i].lesson == Алгебра)

{

cout << "Алгебра" << endl;

}

else if (stud\_list[i].lesson == Биология)

{

cout << "Биология" << endl;

}

else if (stud\_list[i].lesson == Физика)

{

cout << "Физика" << endl;

}

else if (stud\_list[i].lesson == Английский)

{

cout << "Английский" << endl;

}

else if (stud\_list[i].lesson == Политология)

{

cout << "Политология" << endl;

}

cout << "Оценки ученика: ";

for (int k = 0; k < MAX\_MARKS\_COUNT; k++)

{

cout << stud\_list[i].marks[k] << " ";

}

cout << endl;

cout << "Средний балл ученика: " << stud\_list[i].avg << endl << endl;

break;

}

}

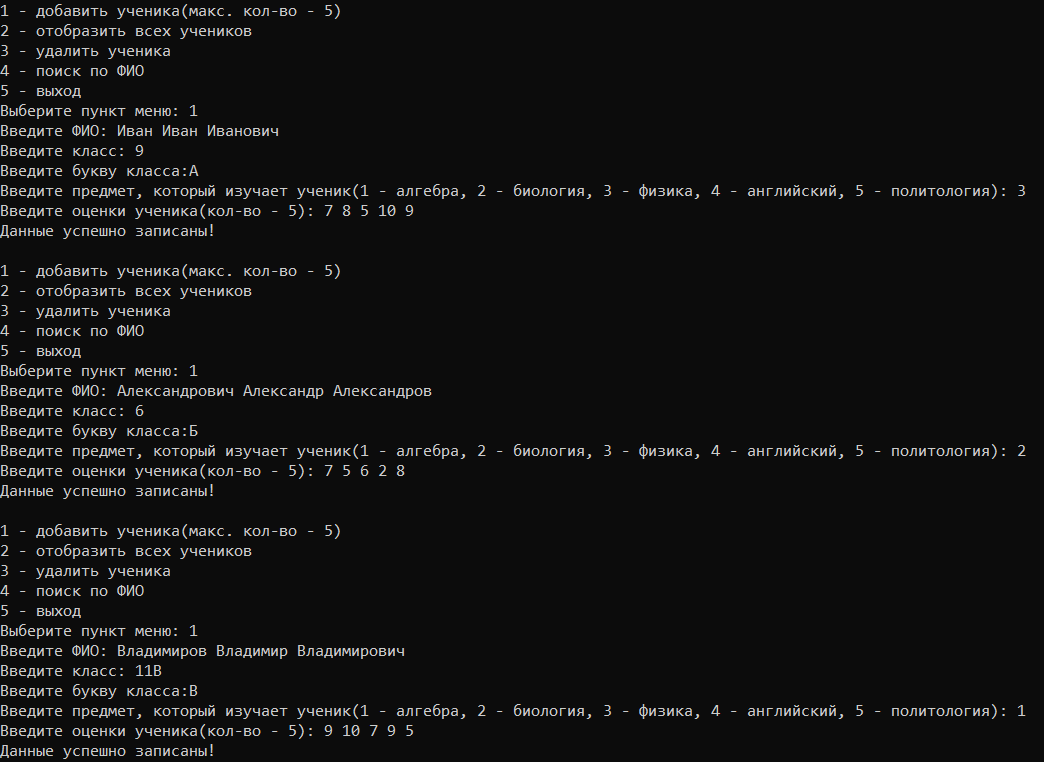
if (!found)

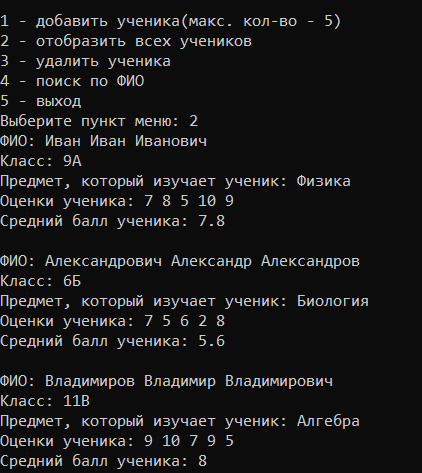
{

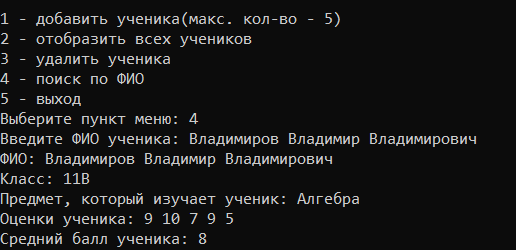
cout << "Ученик с такими ФИО не найден" << endl << endl;

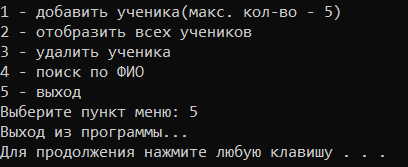
}

}









****

#include <iostream>

#include <Windows.h>

#include <fstream>

#include <string>

using namespace std;

// Объявление структуры для хранения данных клиентов

struct pawnshop

{

union buf

{

char ch[50];

double db;

int num;

};

buf fio; // ФИО

buf name; // Наименование товара

buf price; // Оценочная стоимость товара

buf sum; // Сумма, выданная под залог

buf date; // Дата сдачи

buf term; // Срок хранения

};

// Константа для максимального количества клиентов

const int MAX\_CLIENTS\_COUNT = 5;

// Массив для хранения клиентов

pawnshop clients\_list[MAX\_CLIENTS\_COUNT];

// Переменная для отслеживания количества клиентов

int clients\_count = 0;

// Прототипы функций

void add(); // Добавить клиента

void list(); // Вывести данные клиентов

void write(); // Записать данные в файл

void read(); // Прочитать данные из файла

void search(); // Поиск по ФИО

// Основная функция программы

int main()

{

// Установка кодировки консоли

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

// Главный цикл программы

do

{

// Вывод меню

cout << "1 - добавить клиента" << endl

<< "2 - вывести данные клиентов" << endl

<< "3 - записать данные в файл" << endl

<< "4 - прочитать данные из файла" << endl

<< "5 - поиск по ФИО" << endl

<< "6 - выход из программы" << endl

<< "Выберите пункт меню: ";

// Ввод выбора пользователя

cin >> choice;

// Обработка выбора пользователя

switch (choice)

{

case 1:

add();

break;

case 2:

list();

break;

case 3:

write();

break;

case 4:

read();

break;

case 5:

search();

break;

case 6:

cout << "Выход из программы..." << endl;

system("pause");

break;

}

}

} while (choice != 6);

}

// Функция добавления клиента

void add()

{

pawnshop newClient;

// Проверка на максимальное количество клиентов

if (clients\_count == MAX\_CLIENTS\_COUNT)

{

cout << "Вы ввели максимальное кол-во клиентов" << endl;

return;

}

// Ввод данных о клиенте

cout << "Введите ФИО клиента: ";

cin.ignore();

gets\_s(newClient.fio.ch, 50);

cout << "Введите наименование товара: ";

cin.ignore();

gets\_s(newClient.name.ch, 50);

cout << "Введите оценочную стоимость товара в $: ";

cin >> newClient.price.db;

cout << "Введите сумму, выданную под залог в $: ";

cin >> newClient.sum.db;

cout << "Введите дату сдачи в формате ДД.ММ.ГГГГ: ";

cin.ignore();

gets\_s(newClient.date.ch, 50);

cout << "Введите срок хранения в днях: ";

cin >> newClient.term.num;

// Добавление клиента в массив

clients\_list[clients\_count] = newClient;

// Увеличение счетчика клиентов

clients\_count++;

cout << "Данные успешно записаны!" << endl << endl;

}

// Функция вывода данных всех клиентов

void list()

{

for (int i = 0; i < clients\_count; i++)

{

cout << "ФИО: " << clients\_list[i].fio.ch << endl;

cout << "Наименование товара: " << clients\_list[i].name.ch << endl;

cout << "Оценочная стоимость товара: " << clients\_list[i].price.db << "$" << endl;

cout << "Сумма, выданная под залог: " << clients\_list[i].sum.db << "$" << endl;

cout << "Дата сдачи: " << clients\_list[i].date.ch << endl;

cout << "Срок хранения: " << clients\_list[i].term.num << " дней" << endl << endl;

}

}

// Функция записи данных всех клиентов в файл

void write()

{

ofstream fin("file.txt");

// Проверка открытия файла

if (!fin.is\_open())

{

cout << "Ошибка открытия файла" << endl << endl;

return;

}

// Запись данных в файл

for (int i = 0; i < clients\_count; i++)

{

fin << "ФИО: " << clients\_list[i].fio.ch << endl;

fin << "Наименование товара: " << clients\_list[i].name.ch << endl;

fin << "Оценочная стоимость товара: " << clients\_list[i].price.db << "$" << endl;

fin << "Сумма, выданная под залог: " << clients\_list[i].sum.db << "$" << endl;

fin << "Дата сдачи: " << clients\_list[i].date.ch << endl;

fin << "Срок хранения: " << clients\_list[i].term.num << " дней" << endl << endl;

}

// Закрытие файла

fin.close();

}

// Функция чтения данных из файла и вывода на экран

void read()

{

ifstream fout("file.txt");

string line;

// Проверка открытия файла

if (!fout.is\_open())

{

cout << "Ошибка открытия файла" << endl << endl;

return;

}

// Чтение данных из файла и вывод на экран

while (getline(fout, line))

{

cout << line << endl;

}

cout << endl;

// Закрытие файла

fout.close();

}

// Функция поиска клиента по ФИО

void search()

{

char buffer[50];

bool found = false;

// Ввод ФИО для поиска

cout << "Введите ФИО: ";

cin.ignore();

gets\_s(buffer, 50);

// Поиск клиента по введенному ФИО

for (int i = 0; i < clients\_count; i++)

{

if (strcmp(clients\_list[i].fio.ch, buffer) == 0)

{

found = true;

cout << "ФИО: " << clients\_list[i].fio.ch << endl;

cout << "Наименование товара: " << clients\_list[i].name.ch << endl;

cout << "Оценочная стоимость товара: " << clients\_list[i].price.db << "$" << endl;

cout << "Сумма, выданная под залог: " << clients\_list[i].sum.db << "$" << endl;

cout << "Дата сдачи: " << clients\_list[i].date.ch << endl;

cout << "Срок хранения: " << clients\_list[i].term.num << " дней" << endl << endl;

}

}

// Вывод сообщения, если клиент не найден

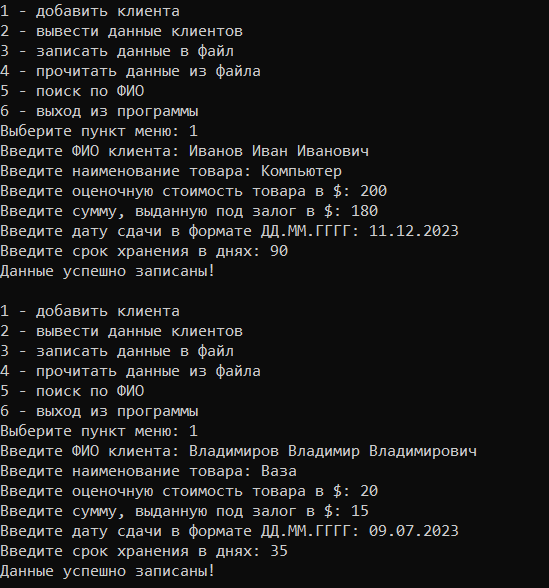
if (!found)

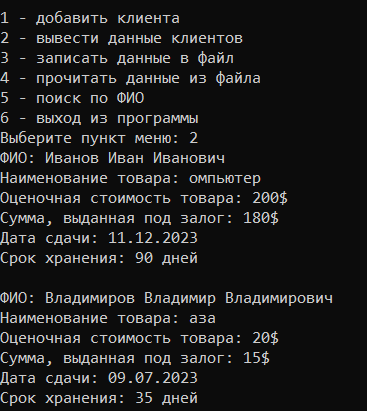
{

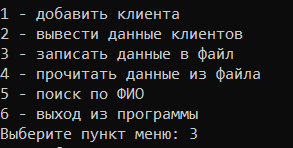
cout << "Клиент с такими ФИО не найден" << endl << endl;

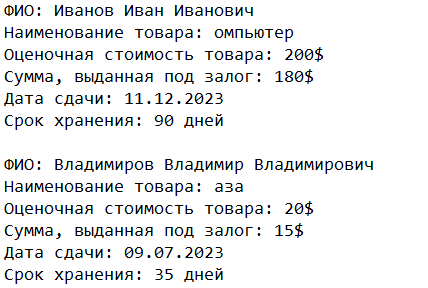
}

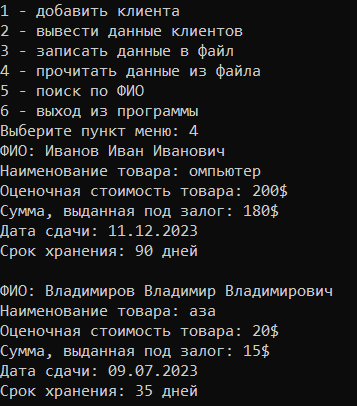
}

****

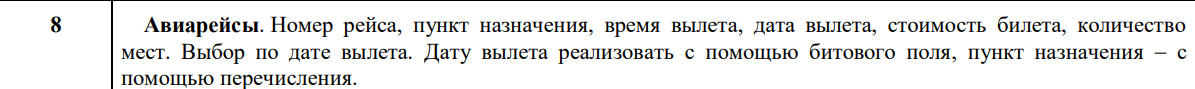
****

****

****

****

**Вариант 8**

****

#include <iostream>

#include <Windows.h>

using namespace std;

void add();

void list();

void del();

void search();

// Перечисление для указания пунктов назначения

enum form

{

Москва = 1, Минск, Монако

};

// Определение псевдонимов для типов данных

typedef unsigned mm;

typedef unsigned hh;

// Константа для максимального количества рейсов

const int MAX\_AIR\_COUNT = 5;

// Структура для хранения информации о рейсе

struct air

{

unsigned number;

form point;

unsigned day : 5;

unsigned month : 4;

unsigned year : 12;

struct timer

{

hh hours;

mm minutes;

} time;

double cost;

unsigned amount;

};

// Глобальные переменные

int air\_count = 0;

air air\_list[MAX\_AIR\_COUNT];

// Главная функция программы

int main()

{

// Установка кодовой страницы для корректного отображения русских символов

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

// Основной цикл программы

do

{

cout << "1 - Добавить новый рейс(макс. кол-во - 5)" << endl

<< "2 - Отобразить все рейсы" << endl

<< "3 - Удалить рейс из списка" << endl

<< "4 - Поиск информации о рейсе по дате" << endl

<< "5 - Выход" << endl

<< "Выберите вариант выполнения: ";

cin >> choice;

// Обработка выбора пользователя

switch (choice)

{

case 1:

add();

break;

case 2:

list();

break;

case 3:

del();

break;

case 4:

search();

break;

case 5:

cout << "Выход из программы..." << endl;

system("pause");

break;

}

} while (choice != 5);

}

// Функция для добавления нового рейса

void add()

{

if (air\_count == MAX\_AIR\_COUNT)

{

cout << "Вы ввели максимальное количество рейсов" << endl

<< endl;

return;

}

air newAir;

cout << "Введите номер рейса:";

cin >> newAir.number;

int choice;

cout << "Введите пункт назначения(1 - Москва, 2 - Минск, 3 - Монако): ";

cin >> choice;

switch (choice)

{

case 1:

newAir.point = Москва;

break;

case 2:

newAir.point = Минск;

break;

case 3:

newAir.point = Монако;

break;

}

int day;

cout << "Введите день отправления: ";

cin >> day;

newAir.day = static\_cast<unsigned>(day);

int month;

cout << "Введите месяц отправления: ";

cin >> month;

newAir.month = static\_cast<unsigned>(month);

int year;

cout << "Введите год отправления: ";

cin >> year;

newAir.year = static\_cast<unsigned>(year);

cout << "Введите время вылета(ЧЧ ММ): ";

cin >> newAir.time.hours;

cin >> newAir.time.minutes;

cout << "Введите стоимость билета в $: ";

cin >> newAir.cost;

cout << "Введите количество мест: ";

cin >> newAir.amount;

air\_list[air\_count] = newAir;

air\_count++;

cout << "Данные успешно записаны! " << endl

<< endl;

}

// Функция для отображения всех рейсов

void list()

{

for (int i = 0; i < air\_count; i++)

{

cout << "Номер рейса: " << air\_list[i].number << endl;

cout << "Пункт назначения: ";

if (air\_list[i].point == Москва)

{

cout << "Москва" << endl;

}

else if (air\_list[i].point == Минск)

{

cout << "Минск" << endl;

}

else if (air\_list[i].point == Монако)

{

cout << "Монако" << endl;

}

cout << "Дата вылета: " << air\_list[i].day << "." << air\_list[i].month << "." << air\_list[i].year << endl;

cout << "Время вылета: " << air\_list[i].time.hours << ":" << air\_list[i].time.minutes << endl;

cout << "Стоимость билета: " << air\_list[i].cost << "$" << endl;

cout << "Количество мест: " << air\_list[i].amount << endl

<< endl;

}

}

// Функция для удаления рейса из списка

void del()

{

int choice;

cout << "Введите номер рейса, который нужно удалить: ";

cin >> choice;

if (choice < 1 || choice > air\_count)

{

cout << "Некорректный номер рейса." << endl;

return;

}

for (int i = choice - 1; i < air\_count - 1; i++)

{

air\_list[i] = air\_list[i + 1];

}

air\_count--;

cout << "Данные успешно удалены! " << endl

<< endl;

}

// Функция для поиска информации о рейсе по дате

void search()

{

unsigned day;

unsigned month;

unsigned year;

bool found = false;

cout << "Введите дату для поиска в формате ДД ММ ГГГГ: ";

cin >> day;

cin >> month;

cin >> year;

for (int i = 0; i < air\_count; i++)

{

if (air\_list[i].day == day && air\_list[i].month == month && air\_list[i].year == year)

{

found = true;

cout << "Номер рейса: " << air\_list[i].number << endl;

cout << "Пункт назначения: ";

if (air\_list[i].point == Москва)

{

cout << "Москва" << endl;

}

else if (air\_list[i].point == Минск)

{

cout << "Минск" << endl;

}

else if (air\_list[i].point == Монако)

{

cout << "Монако" << endl;

}

cout << "Дата вылета: " << air\_list[i].day << "." << air\_list[i].month << "." << air\_list[i].year << endl;

cout << "Время вылета: " << air\_list[i].time.hours << ":" << air\_list[i].time.minutes << endl;

cout << "Стоимость билета: " << air\_list[i].cost << "$" << endl;

cout << "Количество мест: " << air\_list[i].amount << endl

<< endl;

break;

}

}

if (!found)

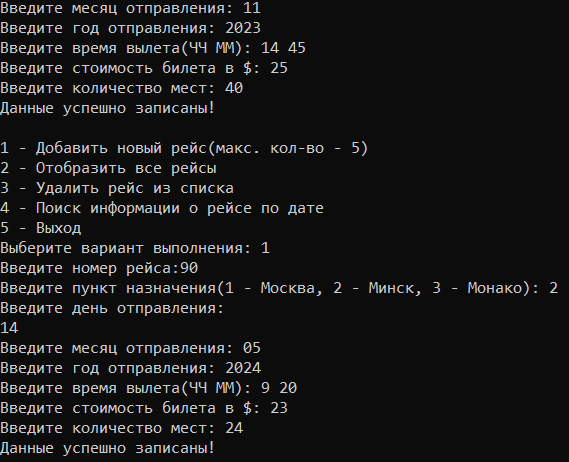
{

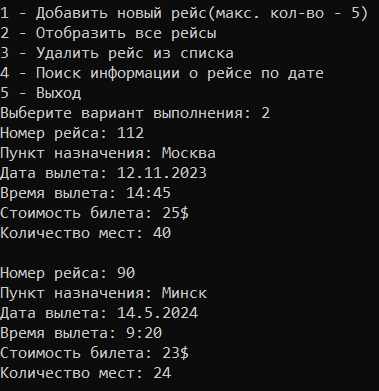
cout << "Рейс на эту дату не найден" << endl

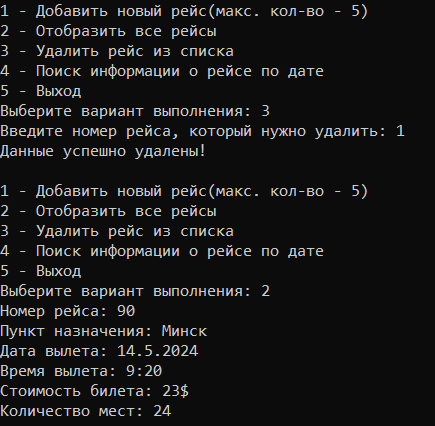
<< endl;

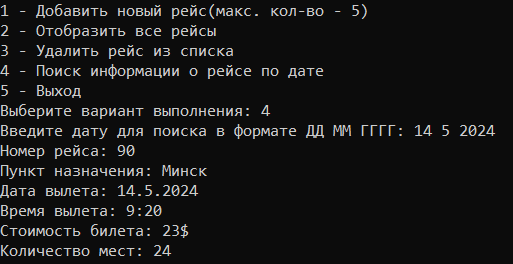
}

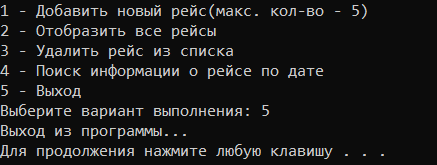
}

****

****

****

****

****

****

#include <iostream>

#include <Windows.h>

#include <fstream>

#include <string>

using namespace std;

void add();

void list();

void del();

void search();

void write();

void read();

// Определение псевдонимов для типов данных

typedef unsigned mm;

typedef unsigned hh;

// Константа для максимального количества рейсов

const int MAX\_AIR\_COUNT = 5;

// Структура для хранения информации о рейсе

struct air

{

union form

{

unsigned un;

char ch[50];

double db;

};

form number;

form point;

form day;

form month;

form year;

struct timer

{

hh hours;

mm minutes;

} time;

form cost;

form amount;

};

// Глобальные переменные

int air\_count = 0;

air air\_list[MAX\_AIR\_COUNT];

// Главная функция программы

int main()

{

// Установка кодовой страницы для корректного отображения русских символов

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

// Основной цикл программы

do

{

cout << "1 - Добавить новый рейс(макс. кол-во - 5)" << endl

<< "2 - Отобразить все рейсы" << endl

<< "3 - Удалить рейс из списка" << endl

<< "4 - Поиск информации о рейсе по пункту назначения" << endl

<< "5 - запись в файл" << endl

<< "6 - чтение из файла" << endl

<< "7 - Выход" << endl

<< "Выберите вариант выполнения: ";

cin >> choice;

// Обработка выбора пользователя

switch (choice)

{

case 1:

add();

break;

case 2:

list();

break;

case 3:

del();

break;

case 4:

search();

break;

case 5:

write();

break;

case 6:

read();

break;

case 7:

cout << "Выход из программы..." << endl;

system("pause");

break;

}

} while (choice != 7);

}

// Функция для добавления нового рейса

void add()

{

if (air\_count == MAX\_AIR\_COUNT)

{

cout << "Вы ввели максимальное количество рейсов" << endl

<< endl;

return;

}

air newAir;

cout << "Введите номер рейса:";

cin >> newAir.number.un;

int choice;

cout << "Введите пункт назначения(1 - Москва, 2 - Минск, 3 - Монако): ";

cin >> choice;

switch (choice)

{

case 1:

strcpy\_s(newAir.point.ch, "Москва");

break;

case 2:

strcpy\_s(newAir.point.ch, "Минск");

break;

case 3:

strcpy\_s(newAir.point.ch, "Монако");

break;

}

cout << "Введите день отправления: ";

cin >> newAir.day.un;

cout << "Введите месяц отправления: ";

cin >> newAir.month.un;

cout << "Введите год отправления: ";

cin >> newAir.year.un;

cout << "Введите время вылета(ЧЧ ММ): ";

cin >> newAir.time.hours;

cin >> newAir.time.minutes;

cout << "Введите стоимость билета в $: ";

cin >> newAir.cost.db;

cout << "Введите количество мест: ";

cin >> newAir.amount.un;

air\_list[air\_count] = newAir;

air\_count++;

cout << "Данные успешно записаны! " << endl

<< endl;

}

// Функция для отображения всех рейсов

void list()

{

for (int i = 0; i < air\_count; i++)

{

cout << "Номер рейса: " << air\_list[i].number.un << endl;

cout << "Пункт назначения: " << air\_list[i].point.ch << endl;

cout << "Дата вылета: " << air\_list[i].day.un << "." << air\_list[i].month.un << "." << air\_list[i].year.un << endl;

cout << "Время вылета: " << air\_list[i].time.hours << ":" << air\_list[i].time.minutes << endl;

cout << "Стоимость билета: " << air\_list[i].cost.db << "$" << endl;

cout << "Количество мест: " << air\_list[i].amount.un << endl << endl;

}

}

// Функция для удаления рейса из списка

void del()

{

int choice;

cout << "Введите номер рейса, который нужно удалить: ";

cin >> choice;

if (choice < 1 || choice > air\_count)

{

cout << "Некорректный номер рейса." << endl;

return;

}

for (int i = choice - 1; i < air\_count - 1; i++)

{

air\_list[i] = air\_list[i + 1];

}

air\_count--;

cout << "Данные успешно удалены! " << endl

<< endl;

}

// Функция для поиска информации о рейсе по месту назначения

void search()

{

char buf[50];

bool found = false;

cout << "Введите место назначения для поиска: ";

cin.ignore();

gets\_s(buf, 50);

for (int i = 0; i < air\_count; i++)

{

if (strcmp(buf, air\_list[air\_count].point.ch))

{

found = true;

cout << "Номер рейса: " << air\_list[i].number.un << endl;

cout << "Пункт назначения: " << air\_list[i].point.ch << endl;

cout << "Дата вылета: " << air\_list[i].day.un << "." << air\_list[i].month.un << "." << air\_list[i].year.un << endl;

cout << "Время вылета: " << air\_list[i].time.hours << ":" << air\_list[i].time.minutes << endl;

cout << "Стоимость билета: " << air\_list[i].cost.db << "$" << endl;

cout << "Количество мест: " << air\_list[i].amount.un << endl << endl;

break;

}

}

if (!found)

{

cout << "Рейс на это место не найден" << endl

<< endl;

}

}

// Функция для записи данные в файл

void write()

{

ofstream fout("file.txt");

if (!fout.is\_open())

{

cout << "Ошибка открытия файла" << endl << endl;

return;

}

for (int i = 0; i < air\_count; i++)

{

fout << "Номер рейса: " << air\_list[i].number.un << endl;

fout << "Пункт назначения: " << air\_list[i].point.ch << endl;

fout << "Дата вылета: " << air\_list[i].day.un << "." << air\_list[i].month.un << "." << air\_list[i].year.un << endl;

fout << "Время вылета: " << air\_list[i].time.hours << ":" << air\_list[i].time.minutes << endl;

fout << "Стоимость билета: " << air\_list[i].cost.db << "$" << endl;

fout << "Количество мест: " << air\_list[i].amount.un << endl << endl;

}

cout << "Данные успешно записаны! " << endl << endl;

fout.close();

}

//Функция для чтения данные из файла

void read()

{

ifstream fin("file.txt");

if (!fin.is\_open())

{

cout << "Ошибка открытия файла" << endl << endl;

return;

}

string buf;

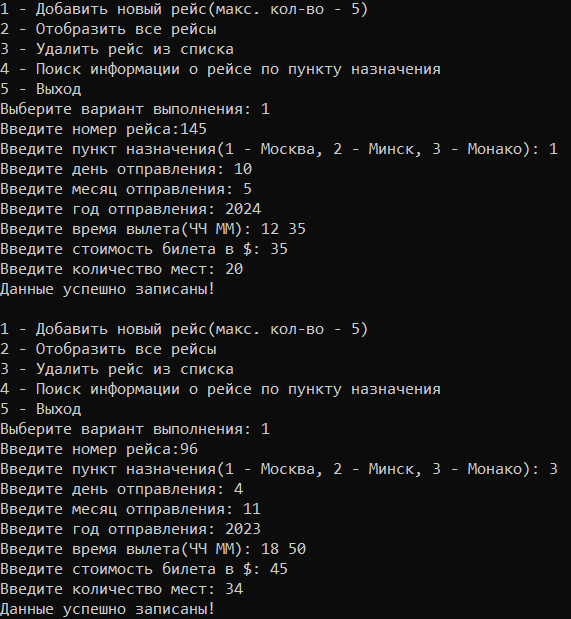
while (getline(fin, buf))

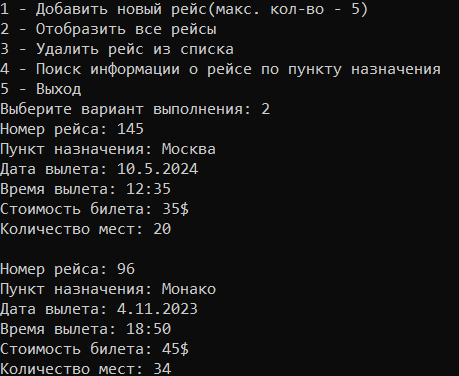
{

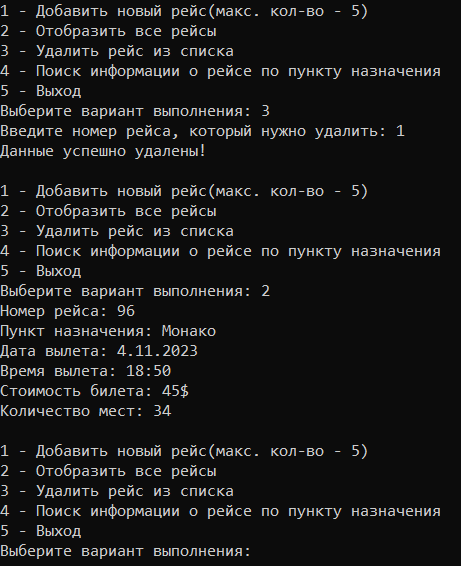
cout << buf << endl;

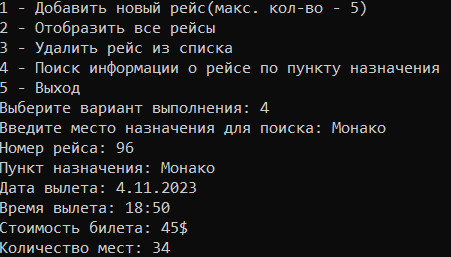
}

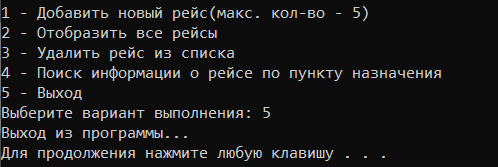
fin.close();

}****

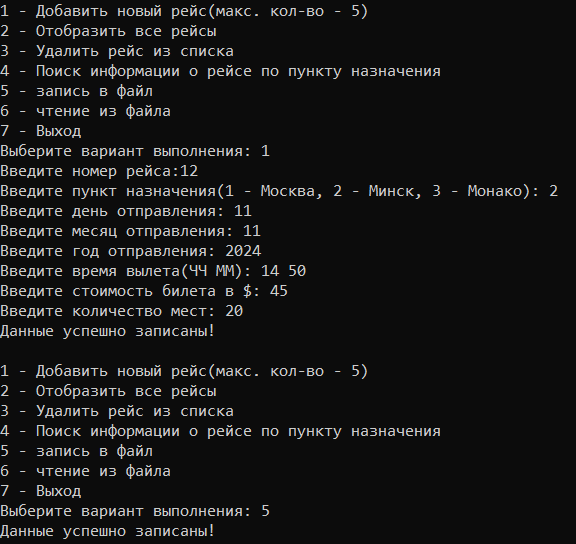
****

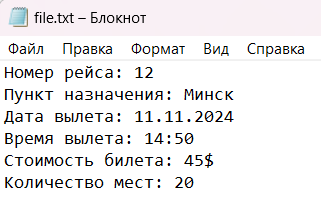
****

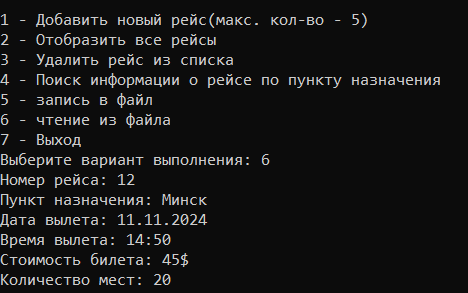
****

****

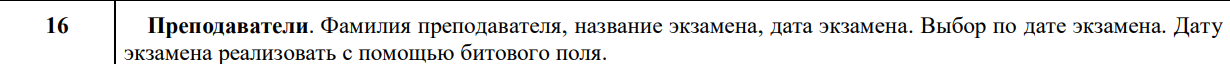
**Я забыл, что надо было добавить запись и чтение из файла и дописал программу, но предыдущие скрины из старой**

****

****

****

**Вариант 16**

****

#include <iostream>

#include <string>

#include <windows.h>

#define size 10 // Определение размера массива

using namespace std;

void input(); // Прототип функции ввода

void output(); // Прототип функции вывода

void del(); // Прототип функции удаления

void find(); // Прототип функции поиска

struct teacher // Описание структуры для хранения данных о преподавателе

{

string name;

string exam;

unsigned day : 5;

unsigned month : 4;

unsigned year : 12;

};

teacher teachers[size]; // Объявление массива структур

int choice; // Переменная для хранения выбора пользователя

int current\_size = 0; // Текущее количество записей о преподавателях

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

do

{

cout << "1 - Ввод элементов" << endl;

cout << "2 - Вывод элементов" << endl;

cout << "3 - Удаление заданного преподавателя из списка" << endl;

cout << "4 - Поиск информации " << endl;

cout << "5 - Выход из программы" << endl;

cin >> choice;

switch (choice)

{

case 1:

input();

break;

case 2:

output();

break;

case 3:

del();

break;

case 4:

find();

break;

}

} while (choice != 5);

return 0;

}

void input() // Функция ввода данных о преподавателях

{

int day, month, year;

int teach;

cout << "Введите количество преподавателей: "; cin >> teach;

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

{

if (current\_size < size) // Проверка наличия места в массиве

{

cout << "Информация о " << i + 1 << " преподавателе: " << endl;

cin.ignore();

cout << endl << "ФИО преподавателя: " << endl;

getline(cin, teachers[current\_size].name);

cout << "Название экзамена: " << endl;

getline(cin, teachers[current\_size].exam);

cout << "Введите дату в формате ДД ММ ГГГГ: " << endl;

cin >> day >> month >> year;

teachers[current\_size].day = day;

teachers[current\_size].month = month;

teachers[current\_size].year = year;

current\_size++;

}

}

}

void output() // Функция вывода информации о преподавателях

{

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

{

cout << "Информация о " << i + 1 << " преподавателе " << endl;

cout << endl << "ФИО преподавателя: " << teachers[i].name << endl;

cout << "Название экзамена: " << teachers[i].exam << endl;

cout << "Дата экзамена: " << teachers[i].day << " " << teachers[i].month << " " << teachers[i].year << endl;

}

}

void del() // Функция удаления записи о преподавателе

{

int d;

bool f = false;

int day, month, year;

cout << "Введите дату экзамена, который хотите удалить в формате ДД ММ ГГГГ" << endl;

cin >> day >> month >> year;

cin.ignore();

for (int i = 0; i < current\_size; i++) // Поиск записи о преподавателе

{

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

{

d = i;

f = true;

}

}

if (f) // Если запись найдена, удаляем её

{

for (int de1 = d; de1 < current\_size; de1++)

{

teachers[de1] = teachers[de1 + 1];

}

current\_size = current\_size - 1;

cout << "Информация удалена" << endl;

}

else

cout << "Преподаватель не найден!" << endl; // Иначе выводим сообщение о том, что запись не найдена

}

void find() // Функция поиска информации о преподавателе по дате экзамена

{

bool f = true;

int day, month, year;

cout << "Введите дату экзамена, чтобы найти преподавателя в формате ДД ММ ГГГГ" << endl;

cin >> day >> month >> year;

cin.ignore();

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

{

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

{

cout << "Информация о " << i + 1 << " преподавателе: " << endl;

cout << endl << "ФИО преподавателя: " << teachers[i].name << endl;

cout << "Название экзамена: " << teachers[i].exam << endl;

cout << "Дата экзамена: " << teachers[i].day << " " << teachers[i].month << " " << teachers[i].year << endl;

f = false;

}

}

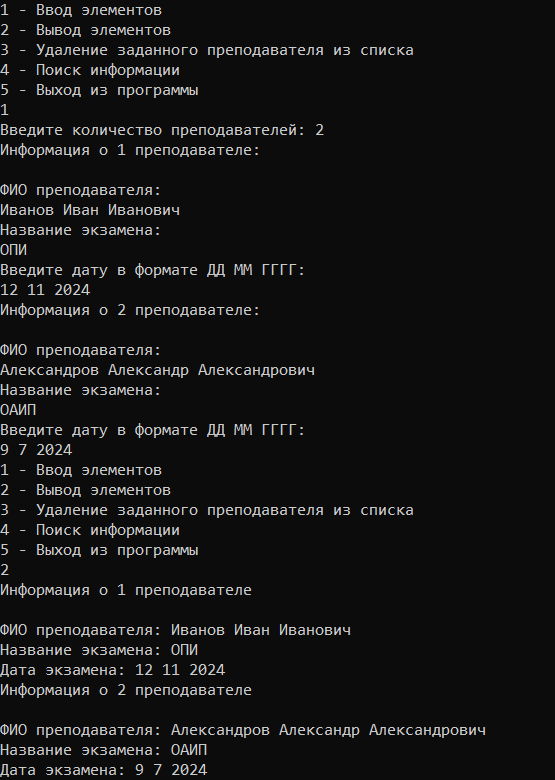
if (f)

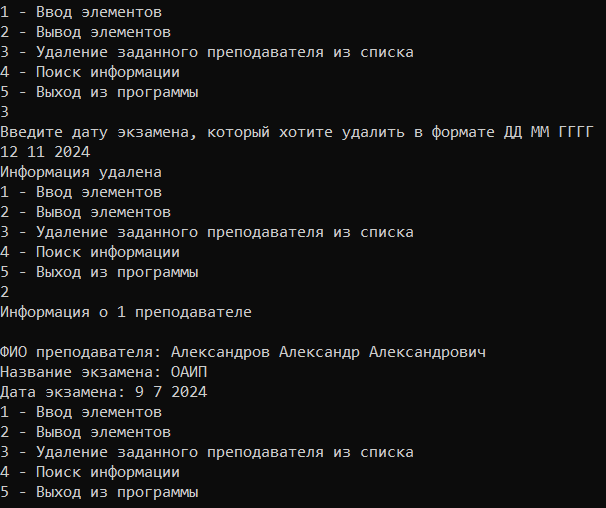
{

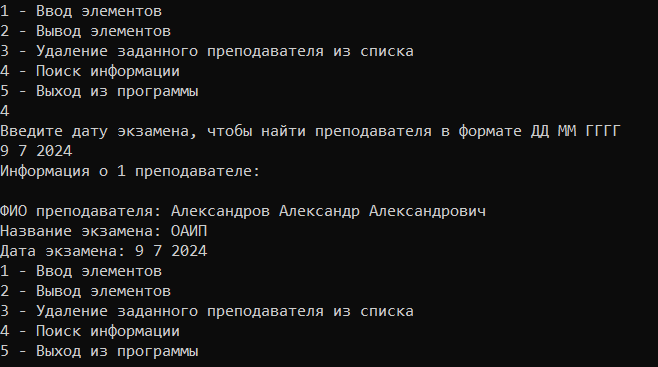
cout << "Преподаватель не найден!" << endl;

}

}

****

****

****

****

#include <iostream>

#include <fstream>

#include <string>

#include <windows.h>

#define size 10

using namespace std;

void input(); // Функция для ввода данных о студентах

void output(); // Функция для вывода данных о студентах

void inFile(); // Функция для записи данных в файл

void outFile(); // Функция для чтения данных из файла

void find(); // Функция для поиска студентов по среднему баллу

union buf // Объединение для хранения различных типов данных атрибутов студента

{

char fullname[50];

char day[50];

char spec[50];

int gr;

char fac[50];

float av;

};

struct student // Структура для представления студента

{

buf full\_name;

buf entrday;

buf speciality;

buf group;

buf facultyd;

buf average;

};

student students[size]; // Массив студентов

int choice; // Выбор пользователя

int current\_size; // Текущий размер массива студентов

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

do

{

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

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

cout << "3 - Запись информации в файл" << endl;

cout << "4 - Чтение информации из файла" << endl;

cout << "5 - Поиск информации" << endl;

cout << "6 - Выход из программы" << endl;

cin >> choice;

switch (choice)

{

case 1:

input();

break;

case 2:

output();

break;

case 3:

inFile();

break;

case 4:

outFile();

break;

case 5:

find();

break;

}

} while (choice != 6);

return 0;

}

void input() // Функция для ввода данных о студентах

{

int numb; // Количество студентов для ввода

int choice; // Выбор пользователя

cout << "Введите количество студентов: "; cin >> numb;

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

{

if (current\_size < size) // Проверка на заполненность массива

{

cout << "Информация о " << i + 1 << " студенте: " << endl;

cin.ignore();

cout << endl << "ФИО: " << endl;

gets\_s(students[current\_size].full\_name.fullname, 50);

cout << "Дата поступления: " << endl;

gets\_s(students[current\_size].entrday.day, 50);

cout << "Специальность: " << endl;

gets\_s(students[current\_size].speciality.spec, 50);

cout << "Группа: " << endl;

cin >> students[current\_size].group.gr;

cin.ignore();

cout << "Факультет: " << endl;

gets\_s(students[current\_size].facultyd.fac, 50);

cout << "Средний балл: " << endl;

cin >> students[current\_size].average.av;

current\_size++;

}

}

}

void output() // Функция для вывода данных о студентах

{

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

{

cout << "Информация о " << i + 1 << " студенте: " << endl;

cout << endl << "ФИО: " << students[i].full\_name.fullname << endl;

cout << "Дата поступления: " << students[i].entrday.day << endl;

cout << "Специальность: " << students[i].speciality.spec << endl;

cout << "Группа: " << students[i].group.gr << endl;

cout << "Факультет: " << students[i].facultyd.fac << endl;

cout << "Средний балл: " << students[i].average.av << endl;

}

}

void inFile() // Функция для записи данных о студентах в файл

{

ofstream file1("file.txt");

if (file1)

{

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

{

file1 << "Информация о " << i + 1 << " студенте: " << endl;

file1 << endl << "ФИО: " << students[i].full\_name.fullname << endl;

file1 << "Дата поступления: " << students[i].entrday.day << endl;

file1 << "Специальность: " << students[i].speciality.spec << endl;

file1 << "Группа: " << students[i].group.gr << endl;

file1 << "Факультет: " << students[i].facultyd.fac << endl;

file1 << "Средний балл: " << students[i].average.av << endl;

}

cout << "Информация записана в файл!" << endl;

file1.close();

}

else

cout << "Не удалось открыть файл!" << endl;

}

void outFile() // Функция для чтения данных о студентах из файла

{

ifstream file1("file.txt");

string line;

while (getline(file1, line))

{

cout << line << endl;

}

file1.close();

}

void find() // Функция для поиска студентов по среднему баллу

{

float aver; // Средний балл для поиска

bool f = true;

cout << "Введите средний балл: ";

cin >> aver;

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

{

if (aver == students[i].average.av)

{

cout << "Информация о " << i + 1 << " студенте: " << endl;

cout << endl << "ФИО: " << students[i].full\_name.fullname << endl;

cout << "Дата поступления: " << students[i].entrday.day << endl;

cout << "Специальность: " << students[i].speciality.spec << endl;

cout << "Группа: " << students[i].group.gr << endl;

cout << "Факультет: " << students[i].facultyd.fac << endl;

cout << "Средний балл: " << students[i].average.av << endl;

f = false;

}

}

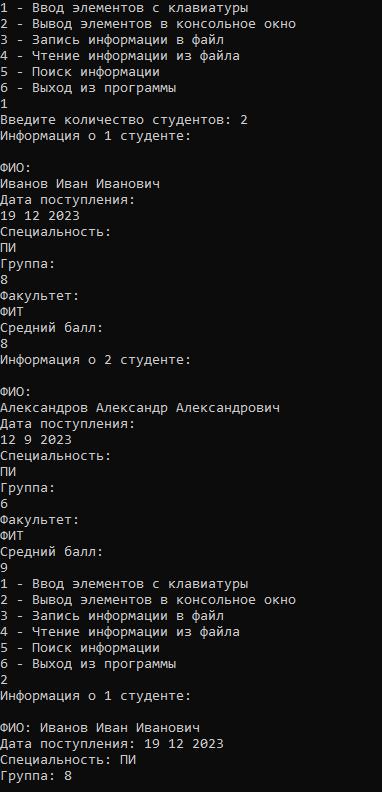
if (f)

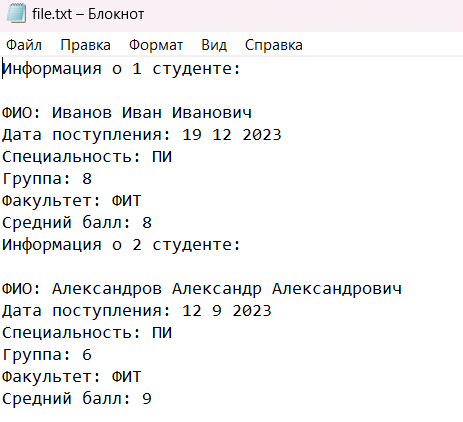
{

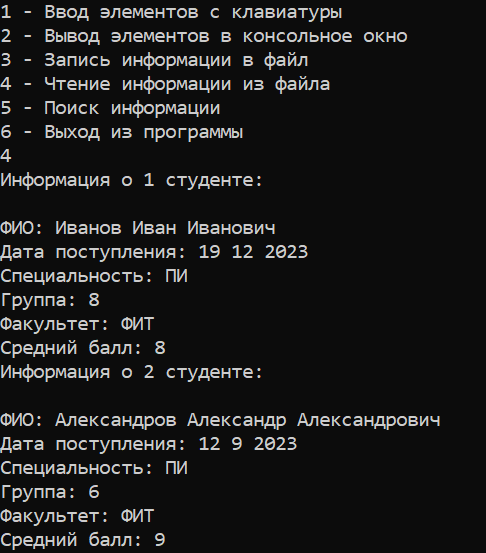
cout << "Информация не найдена!" << endl;

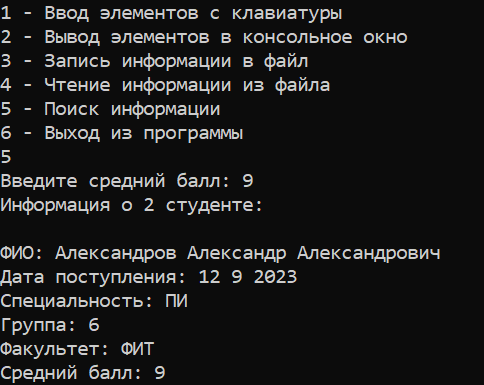
}

}

****

****

****

****