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 10 группы

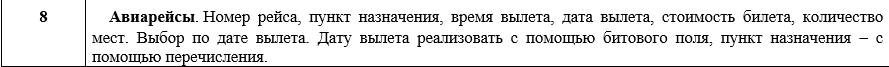
Мандрик Алексей Иванович

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 8**



**Код программы:**

#include <iostream>

#include <string>

using namespace std;

// Создание перечисления

enum DESTINATION {

Minsk = 1, Moscow, Kiev, Berlin, Paris, London, New\_York

};

// Структура для значений

struct Flights {

string destination; // Пункт назначения

int numFlight; // Номер

int priceTicket; // Стоимость

int numberPlace; // Количество мест

// Структура для времени (битовые поля)

struct TIME {

unsigned int hour : 5;

unsigned int minute : 6;

} time;

// Структура для даты (битовые поля)

struct DATE {

unsigned int day : 5;

unsigned int month : 4;

unsigned int year : 11;

} date;

// Функция пользоваеля для ввода данных

void input() {

cin.ignore(); // Удаление символа новой строки

cout << "Введите номер: ";

cin >> numFlight;

cout << "Выберите пункт назначения\n(1 - Минск, 2 - Москва, 3 - Киев, 4 - Берлин, 5 - Париж, 6 - Лондон, 7 - Нью-Йорк)\n";

unsigned int variable;

cin >> variable;

// Перечисления в кейсах

switch (variable)

{

case DESTINATION::Minsk:

destination = "Минск";

break;

case DESTINATION::Moscow:

destination = "Москва";

break;

case DESTINATION::Kiev:

destination = "Киев";

break;

case DESTINATION::Berlin:

destination = "Берлин";

break;

case DESTINATION::Paris:

destination = "Париж";

break;

case DESTINATION::London:

destination = "Лондон";

break;

case DESTINATION::New\_York:

destination = "Нью-Йорк";

break;

default:

cout << "Вы ввели неверное значение";

break;

}

cin.ignore(); // Удаление символа новой строки

cout << "Введите цену: ";

cin >> priceTicket;

cout << "Введите кол-во мест: ";

cin >> numberPlace;

unsigned int day, month, year, hour, minute;

// goto что ввели корректное время

repeatTime:cout << "Введите время чч:мм: ";

cin >> hour >> minute;

if (hour < 25 && minute < 61) {

time.hour = hour;

time.minute = minute;

}

else {

cout << "Вы неверно ввели время";

goto repeatTime;

}

// goto что ввели корректную дату

repeatDate:cout << "Введите дату дд.мм.гггг: ";

cin >> day >> month >> year;

if (day < 33 && month < 13 && year < 2049) {

date.day = day;

date.month = month;

date.year = year;

}

else {

cout << "Вы неверно ввели дату";

goto repeatDate;

};

}

// Функция пользователя для вывода элементов

void output() {

if (destination != "") {

cout << "---------Номер рейса: " << numFlight << "-------------" << endl;

cout << "Пункт назначения: " << destination << endl;

cout << "Цена билета: " << priceTicket << endl;

cout << "Кол-во мест" << numberPlace << endl;

cout << "Время: " << time.hour << ":" << time.minute << endl;

cout << "Дата: " << date.day << "." << date.month << "." << date.year << endl;

}

}

// Функция пользователя для удаления элементов (если удалять нулевой элемент, удалится первый из консоли, если первый, то удалится второй из консоли и т.д.)

void del()

{

numFlight = 0;

destination = "";

priceTicket = 0;

numberPlace = 0;

time.hour = 0;

time.minute = 0;

date.day = 0;

date.month = 0;

date.year = 0;

}

// Функция пользователя для поиска

bool searh(int day, int month, int year) {

// Если введённое совпадает с нужным, возврат true

if (date.day == day && date.month == month && date.year == year)

{

return true;

}

else {

return false;

}

}

};

int main()

{

setlocale(LC\_ALL, "Rus");

int variable, num;

Flights flights[10]; // Для структуры

// Меню

do

{

cout << "1 - Ввод новых рейсов\n"

<< "2 - Вывод имеющихся рейсов\n"

<< "3 - Удалить рейс\n"

<< "4 - Поиск рейса\n"

<< "0 - Выход\n";

cout << "Выберите действие: ";

cin >> variable;

switch (variable)

{

case 1:

{

cout << "Введите кол-во рейсов, которое хотите ввести: ";

cin >> num;

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

{

flights[i].input(); // Вызов функции пользователя input() для заполнения

}

break;

}

case 2:

{

for (int i = 0; i < 10; i++)

{

flights[i].output(); // Вызов функции пользователя output() для вывода

}

break;

}

case 3:

{

cout << "Введите списочный номер, который хотите удалить: ";

cin >> num;

for (int i = 0; i < 10; i++)

{

if (num == i)

flights[i].del(); // Вызов функции пользователя del() для удаления авиарейса

}

break;

}

case 4:

{

int day, month, year;

cout << "Введите дату дд.мм.гггг: ";

cin >> day >> month >> year;

if (day < 33 && month < 13 && year < 2049) {

day = day;

month = month;

year = year;

}

for (int i = 0; i < 10; i++)

{

flights[i].searh(day, month, year); // Вызов функции пользователя searh() для поиска

// Если true, то вывод информации по авиарейсам

if (flights[i].searh(day, month, year)) {

flights[i].output();

}

else {

cout << "В этот день нет перелётов!" << endl;

}

}

break;

}

// Если пользователь не умеет читать

default:

cout << "Вы ввели не верное значение\n";

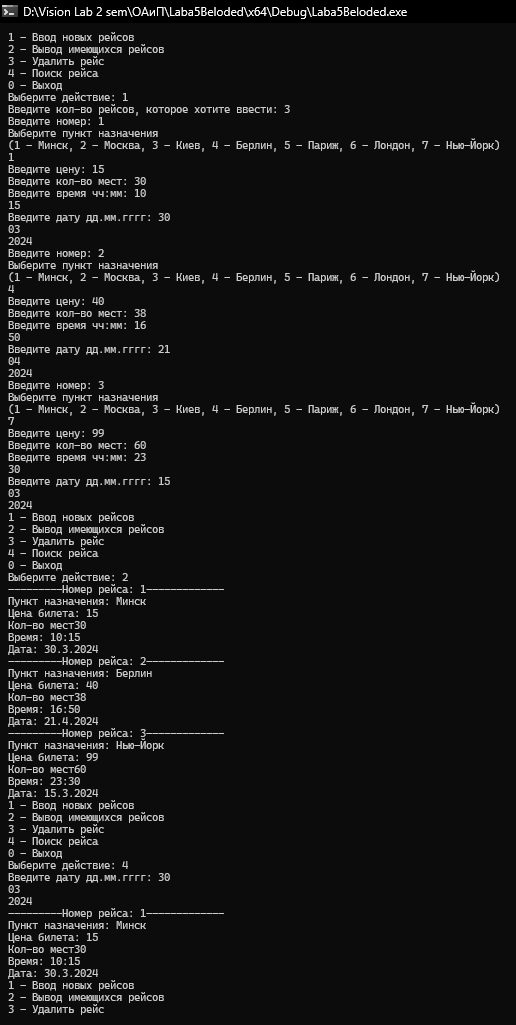
break;

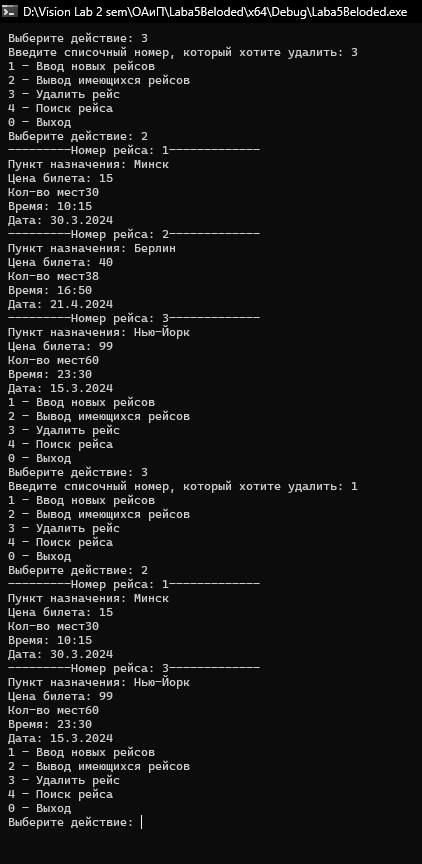
}

} while (variable != 0);

}

**Результат выполнения:**





**Задание 2**

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.

****

**Код программы:**

#include <iostream>

#include <fstream>

#include <cstring>

#include <Windows.h>

using namespace std;

const int MAX\_FLIGHTS = 100;

// Объявляем объединение для работы с датой

union Date {

struct {

int day;

int month;

int year;

};

char dateStr[11];

};

// Структура для хранения информации о рейсе

struct Flight {

int flightNumber; // Номер рейса

char destination[50]; // Пункт назначения

int departureTime; // Время вылета

Date departureDate; // Дата вылета

float ticketPrice; // Стоимость билета

int availableSeats; // Количество мест

};

// Функция для ввода информации о рейсе

void inputFlight(Flight& flight) {

cout << "Введите номер рейса: ";

cin >> flight.flightNumber;

cout << "Введите пункт назначения: ";

cin.ignore();

cin.getline(flight.destination, 50);

cout << "Введите время вылета (в формате 24 часа): ";

cin >> flight.departureTime;

cout << "Введите дату вылета (ДД/ММ/ГГГГ): ";

cin >> flight.departureDate.dateStr;

cout << "Введите стоимость билета: ";

cin >> flight.ticketPrice;

cout << "Введите количество мест: ";

cin >> flight.availableSeats;

}

// Функция для вывода информации о рейсе

void printFlight(const Flight& flight) {

cout << "Номер рейса: " << flight.flightNumber << endl;

cout << "Пункт назначения: " << flight.destination << endl;

cout << "Время вылета: " << flight.departureTime << endl;

cout << "Дата вылета: " << flight.departureDate.dateStr << endl;

cout << "Стоимость билета: " << flight.ticketPrice << endl;

cout << "Количество мест: " << flight.availableSeats << endl;

}

// Функция для записи информации о рейсах в файл

void writeToFile(const Flight flights[], int size) {

ofstream outFile("flights.txt");

if (outFile.is\_open()) {

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

outFile << flights[i].flightNumber << " "

<< flights[i].destination << " "

<< flights[i].departureTime << " "

<< flights[i].departureDate.dateStr << " "

<< flights[i].ticketPrice << " "

<< flights[i].availableSeats << endl;

}

outFile.close();

}

else {

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

}

}

// Функция для чтения информации о рейсах из файла

void readFromFile(Flight flights[], int& size) {

ifstream inFile("flights.txt");

if (inFile.is\_open()) {

size = 0;

while (!inFile.eof() && size < MAX\_FLIGHTS) {

inFile >> flights[size].flightNumber;

inFile.ignore();

inFile.getline(flights[size].destination, 50);

inFile >> flights[size].departureTime;

inFile >> flights[size].departureDate.dateStr;

inFile >> flights[size].ticketPrice;

inFile >> flights[size].availableSeats;

size++;

}

inFile.close();

}

else {

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

}

}

// Функция для поиска рейсов по пункту назначения

void searchByDestination(const Flight flights[], int size, const char\* destination) {

bool found = false;

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

if (strcmp(flights[i].destination, destination) == 0) {

found = true;

cout << "Рейс найден:" << endl;

printFlight(flights[i]);

cout << endl;

}

}

if (!found) {

cout << "Рейсы по пункту назначения " << destination << " не найдены." << endl;

}

}

int main() {

// Для русского алфавита

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Flight flights[MAX\_FLIGHTS];

int numFlights = 0;

char choice;

do {

cout << "Меню:\n"

<< "1. Добавить рейс\n"

<< "2. Вывести все рейсы\n"

<< "3. Поиск по пункту назначения\n"

<< "4. Записать в файл\n"

<< "0. Выйти\n"

<< "Введите ваш выбор: ";

cin >> choice;

switch (choice) {

case '1': // Ввод

if (numFlights < MAX\_FLIGHTS) {

inputFlight(flights[numFlights]); // Вызов функции пользователя

numFlights++;

}

else {

cout << "Вы ввели максимальное количество перелётов" << endl;

}

break;

case '2': // Вывод

cout << "Все рейсы:" << endl;

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

printFlight(flights[i]);// Вызов функции пользователя

cout << endl;

}

break;

case '3': // Поиск

char destination[50];

cout << "Введите пункт назначения: ";

cin.ignore();

cin.getline(destination, 50);

searchByDestination(flights, numFlights, destination);// Вызов функции пользователя

break;

case '4': // Запись

writeToFile(flights, numFlights);// Вызов функции пользователя

cout << "Данные записаны в файл." << endl;

break;

case '0': // Выход

cout << "Выход." << endl;

break;

default:

cout << "Вы ввели некорректное значение" << endl;

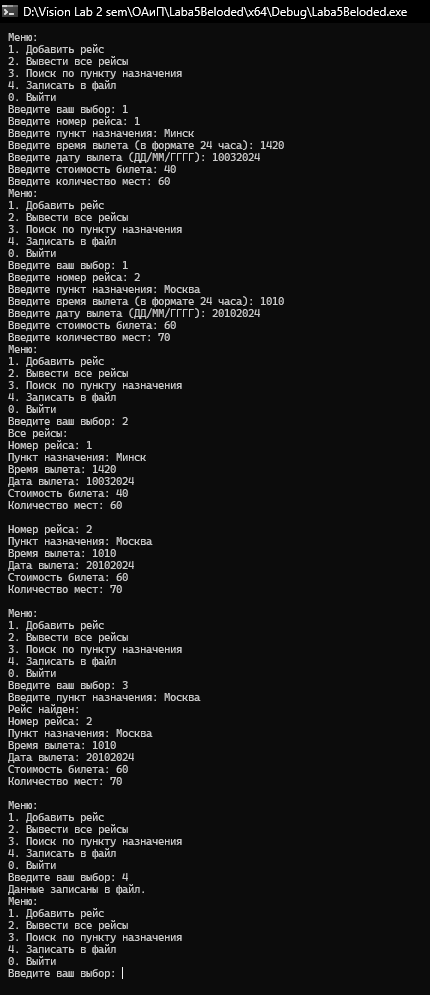
}

} while (choice != '0');

return 0;

}

**Результат выполнения:**

****

**Дополнительные задания:**

**Вариант 2**

**Задание 1**

****

**Код программы:**

#include <iostream>

#include <string>

#include <Windows.h>

#define SIZE 5 // Размер массива граждан

using namespace std;

enum SEX // Перечисление для пола

{

man = 1, woman

};

struct CITIZENS // Структура для хранения данных о гражданах

{

string FIO;

struct BIRTH\_DATE // Структура для хранения даты рождения

{

unsigned int day : 5; // День

unsigned int month : 4; // Месяц

unsigned int year : 11; // Год

} date = { 0,0,0 }; // Инициализация даты нулями

string address; // Адрес

string sex; // Пол

void input() // Функция ввода данных о гражданине

{

cin.ignore(); // Очистка буфера

cout << "Введите ФИО" << endl;

getline(cin, FIO);

cout << "Введите дату рождения (00 00 0000)" << endl;

unsigned int day = 0, month = 0, year = 0;

cin >> day >> month >> year; //

if (day < 32 && month < 13 && year < 2049) // Проверка валидности даты

{

date.day = day;

date.month = month;

date.year = year;

}

cin.ignore(); // Очистка буфера

cout << "Введите адрес" << endl;

getline(cin, address); // Ввод адреса

cin.ignore(); // Очистка буфера

cout << "Введите пол (1 - мужской, 2 - женский)" << endl;

int choice = 0;

cin >> choice; // Выбор пола

switch (choice)

{

case SEX::man:

{

sex = "Мужской";

break;

}

case SEX::woman:

{

sex = "Женский";

break;

}

default: // Если выбрано неверное значение

{

cout << "Неверный вариант" << endl;

break;

}

}

}

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

{

cout << "ФИО: " << FIO << endl;

cout << "Дата рождения:" << date.day << " " << date.month << " " << date.year << endl;

cout << "Адрес: " << address << endl;

cout << "Пол: " << sex << endl;

}

void del\_citizen() // Функция для удаления данных о гражданине

{

FIO = "";

date = { 0,0,0 };

address = "";

sex = "";

}

bool search(unsigned int year) // Функция для поиска гражданина по году рождения

{

bool flag = false;

if (this->date.year == year)

{

flag = true;

}

return flag;

}

};

int main()

{

// Для русского алфавита

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

CITIZENS citizen[SIZE]; // Создание массива граждан

int choice;

// Меню

do

{

cout << "Выберите вариант: " << endl;

cout << "1 - Добавить гражданина" << endl;

cout << "2 - Вывести информацию о гражданине" << endl;

cout << "3 - Удалить информацию о гражданине" << endl;

cout << "4 - Поиск гражданина по году рождения" << endl;

cout << "5 - Выход" << endl;

cin >> choice;

switch (choice)

{

case 1:

{

cout << "Введите номер пустой записи" << endl;

int number;

cin >> number;

if (number <= SIZE)

{

citizen[number - 1].input(); // Вызов функции для ввода данных

}

else

{

cout << "Ошибка!" << endl;

}

break;

}

case 2:

{

cout << "Введите номер записи" << endl;

int number;

cin >> number;

if (number <= SIZE)

{

citizen[number - 1].output(); // Вызов функции для вывода данных

}

else

{

cout << "Ошибка!" << endl;

}

break;

}

case 3: // Удаление информации о гражданине

{

cout << "Введите номер записи" << endl;

int number;

cin >> number;

if (number <= SIZE)

{

citizen[number - 1].del\_citizen(); // Вызов функции для удаления данных

}

else

{

cout << "Ошибка!" << endl;

}

break;

}

case 4: // Поиск гражданина по году рождения

{

cout << "Введите год рождения: ";

int day1, month1, year1;

cin >> day1 >> month1 >> year1;

bool flag = false;

cin.ignore();

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

{

if (citizen[i].date.day == day1 && citizen[i].date.month == month1 && citizen[i].date.year == year1)

{

citizen[i].output(); // Вызов функции для вывода данных

flag = true;

}

}

if (!flag)

{

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

}

break;

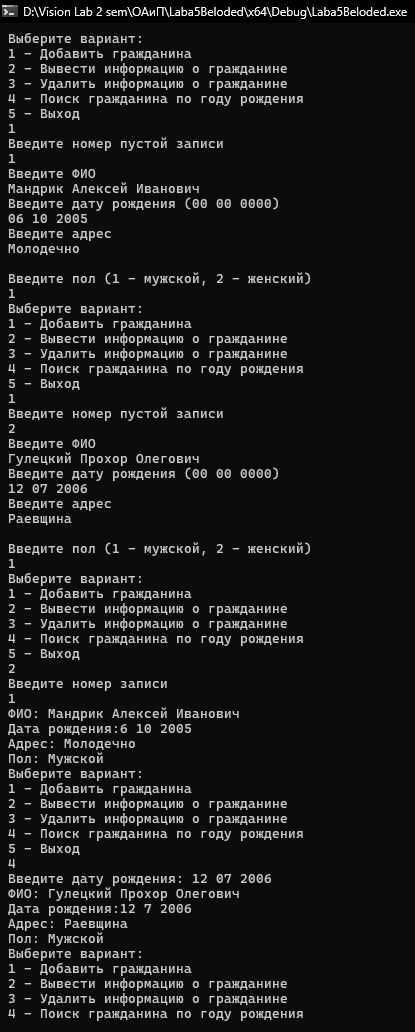
}

}

} while (choice != 5);

}

**Результат выполнения:**



**Задание 2**

****

**Код программы:**

#include <iostream>

#include <string>

#include <Windows.h>

#define SIZE 5

using namespace std;

union TIME // Объединение для времени

{

unsigned int hours; // Часы

unsigned int minutes; // Минуты

};

struct TRAINS // Структура для хранения данных о поездах

{

string number; // Номер поезда

string station; // Пункт назначения

string travel\_days; // Дни следования

struct DEPARTURE\_TIME // Структура для хранения времени отправления

{

unsigned int hours : 5; // Часы (5 бит)

unsigned int minutes : 6; // Минуты (6 бит)

} departure\_time = { 0,0 }; // Инициализация времени отправления нулями

struct ARRIVAL\_TIME // Структура для хранения времени прибытия

{

unsigned int hours : 5; // Часы (5 бит)

unsigned int minutes : 6; // Минуты (6 бит)

} arrival\_time = { 0,0 }; // Инициализация времени прибытия нулями

void input() // функция для ввода данных о поезде

{

cin.ignore(); // Очистка буфера

cout << "Введите номер поезда" << endl;

getline(cin, number);

cout << "Введите пункт назначения" << endl;

getline(cin, station);

cout << "Введите дни следования" << endl;

getline(cin, travel\_days);

TIME time{}; // Создание объекта времени

cout << "Введите время отправления" << endl;

cout << "Введите час" << endl;

cin >> time.hours;

if (time.hours < 25)

{

departure\_time.hours = time.hours;

}

cout << "Введите минуты" << endl;

cin >> time.minutes;

if (time.minutes < 61)

{

departure\_time.minutes = time.minutes;

}

cout << "Введите время прибытия" << endl;

cout << "Введите час" << endl;

cin >> time.hours;

if (time.hours < 25)

{

arrival\_time.hours = time.hours;

}

cout << "Введите минуты" << endl;

cin >> time.minutes;

if (time.minutes < 61)

{

arrival\_time.minutes = time.minutes;

}

}

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

{

cout << "Номер поезда: " << number << endl;

cout << "Пункт назначения: " << station << endl;

cout << "Дни следования: " << travel\_days << endl;

cout << "Время отправления: " << departure\_time.hours << ":" << departure\_time.minutes << endl;

cout << "Время прибытия: " << arrival\_time.hours << ":" << arrival\_time.minutes << endl;

}

bool search(string station) // функция для поиска поезда по пункту назначения

{

bool flag = false;

if (this->station == station) // Если пункт назначения совпадает с искомым

{

flag = true;

}

return flag; // Возврат результата поиска

}

};

int main()

{

SetConsoleCP(1251); // Для русского алфавита

SetConsoleOutputCP(1251);

TRAINS train[SIZE]; // Создание массива поездов

int choice;

// Меню

do

{

cout << "Выберите вариант: " << endl;

cout << "1 - Добавить информацию о поезде" << endl;

cout << "2 - Вывести информацию" << endl;

cout << "3 - Поиск поезда по пункту назначения" << endl;

cout << "4 - Выход" << endl;

cin >> choice; // Ввод выбранной опции

switch (choice)

{

case 1: // Добавление информации о поезде

{

cout << "Введите номер пустой записи" << endl;

int number;

cin >> number;

if (number <= SIZE)

{

train[number - 1].input(); // Вызов функции для ввода данных

}

else

{

cout << "Ошибка!" << endl; // Вывод сообщения об ошибке

}

break;

}

case 2: // Вывод информации о поезде

{

cout << "Введите номер записи" << endl;

int number;

cin >> number;

if (number <= SIZE)

{

train[number - 1].output(); // Вызов функции для вывода данных

}

else

{

cout << "Ошибка!" << endl; // Вывод сообщения об ошибке

}

break;

}

case 3: // Поиск поезда по пункту назначения

{

cin.ignore(); // Очистка буфера

cout << "Введите пункт назначения: ";

string station;

getline(cin, station); // Ввод пункта назначения

bool flag = false; //

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

{

if (train[i].search(station)) // Если поезд найден

{

train[i].output(); // Вывод информации о поезде

flag = true;

}

}

if (!flag)

{

cout << "Поезд не найден!" << endl; // Вывод сообщения об отсутствии результатов

}

break;

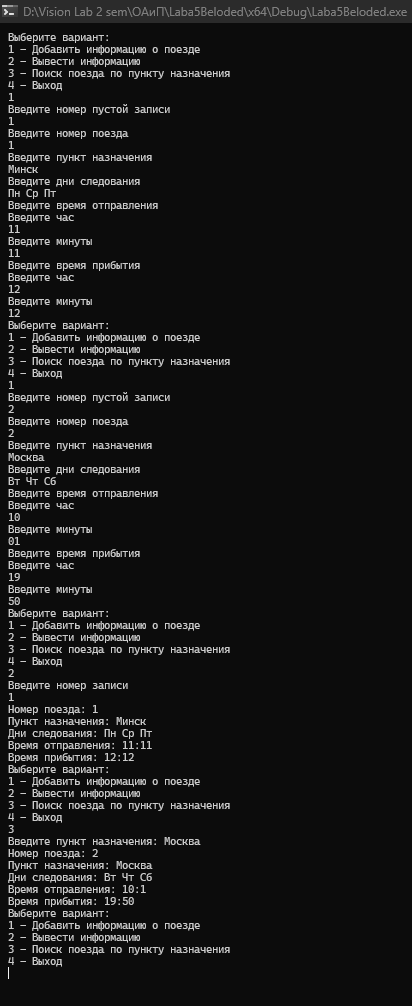
}

}

} while (choice != 4);

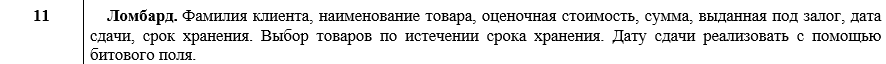
}

**Результат выполнения:**



**Вариант 11**

**Задание 1**

****

**Код программы:**

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;

enum SHELF\_LIFE { // Перечисление для срока хранения

month = 1, halfYear, year, twoYear, fiveYear

};

struct Pawnshop { // Структура для данных о залогах

string surname; // Фамилия клиента

string nameProduct; // Название товара

int gradePrice; // Оценочная сумма

int price; // Сумма

string shelfLife; // Срок хранения

struct DATE { // Структура для хранения даты

unsigned int day : 5; // 5 бит для дня

unsigned int month : 4; // 4 бита для месяца

unsigned int year : 11; // 11 бит для года

} date;

void input() { // Функция для ввода данных

cin.ignore(); // Игнорирование символа новой строки

cout << "Фамилия клиента: ";

getline(cin, surname);

cout << "Выберите срок хранения\n(1 - Месяц, 2 - Полгода, 3 - Год, 4 - Два года, 5 - Пять лет)\n";

unsigned int variable;

cin >> variable;

switch (variable)

{

case SHELF\_LIFE::month:

shelfLife = "Месяц";

break;

case SHELF\_LIFE::halfYear:

shelfLife = "Полгода";

break;

case SHELF\_LIFE::year:

shelfLife = "Год";

break;

case SHELF\_LIFE::twoYear:

shelfLife = "Два года";

break;

case SHELF\_LIFE::fiveYear:

shelfLife = "Пять лет";

break;

default:

cout << "Вы ввели неверное значение";

break;

}

cin.ignore(); // Игнорирование символа новой строки

cout << "Введите название товара: ";

getline(cin, nameProduct);

cout << "Введите оценочная сумма: ";

cin >> gradePrice;

cout << "Введите сумму: ";

cin >> price;

unsigned int day, month, year;

repeatDate: // Метка для повторного ввода даты в случае некорректного ввода

cout << "Введите дату дд.мм.гггг: "; // Ввод даты

cin >> day >> month >> year;

if (day < 25 && month < 13 && year < 2049) { // Проверка корректности даты

date.day = day;

date.month = month;

date.year = year;

}

else {

cout << "Вы неверно ввели дату";

goto repeatDate;

};

}

void output() { // Функция для вывода данных

if (surname != "") { // Проверка наличия данных

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

cout << "Фамилия: " << surname << endl;

cout << "Название продукта: " << nameProduct << endl;

cout << "Оценочная цена: " << gradePrice << "$" << endl;

cout << "Цена:" << price << "$" << endl;

cout << "Дата: " << date.day << "." << date.month << "." << date.year << endl;

cout << "Срок хранения: " << shelfLife << endl;

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

}

}

void del() { // Функция для удаления данных

surname = "";

nameProduct = "";

gradePrice = 0;

price = 0;

date.day = 0;

date.month = 0;

date.year = 0;

}

// Функция для поиска товаров по сроку хранения

bool searh(string shelfLife) {

if (this->shelfLife == shelfLife) {

return true;

}

else {

return false;

}

}

};

int main()

{

SetConsoleCP(1251); // Для русского алфавита

SetConsoleOutputCP(1251);

int variable, num;

Pawnshop pawnshop[10]; // Создаем массив объектов

do

{

// Меню

cout << "1 - Ввод новых залогов\n"

<< "2 - Вывод всех залогов\n"

<< "3 - Удалить залог\n"

<< "4 - Поиск залога\n"

<< "0 - Выход\n";

cout << "Выберите опцию: ";

cin >> variable;

switch (variable)

{

case 1:

{

cout << "Введите количество залогов, которое хотите ввести: ";

cin >> num;

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

{

pawnshop[i].input(); // Вызываем функция ввода для каждого залога

}

break;

}

case 2:

{

for (int i = 0; i < 10; i++)

{

pawnshop[i].output(); // Вызываем функция вывода для каждого залога

}

break;

}

case 3:

{

cout << "Введите порядковый номер, который хотите удалить: ";

cin >> num;

for (int i = 0; i < 10; i++)

{

if (num == i)

pawnshop[i].del(); // Вызываем функция удаления для выбранного залога

}

break;

}

case 4: // Поиск залога по сроку хранения

{

string shelfLife;

cin.ignore();

cout << "Введите срок хранения: ";

getline(cin, shelfLife);

for (int i = 0; i < 10; i++)

{

if (pawnshop[i].searh(shelfLife)) { // Вызываем функция поиска для каждого залога

pawnshop[i].output(); // Если срок хранения совпадает, выводим информацию о залоге

}

}

break;

}

default:

cout << "Вы ввели неверное значение\n";

break;

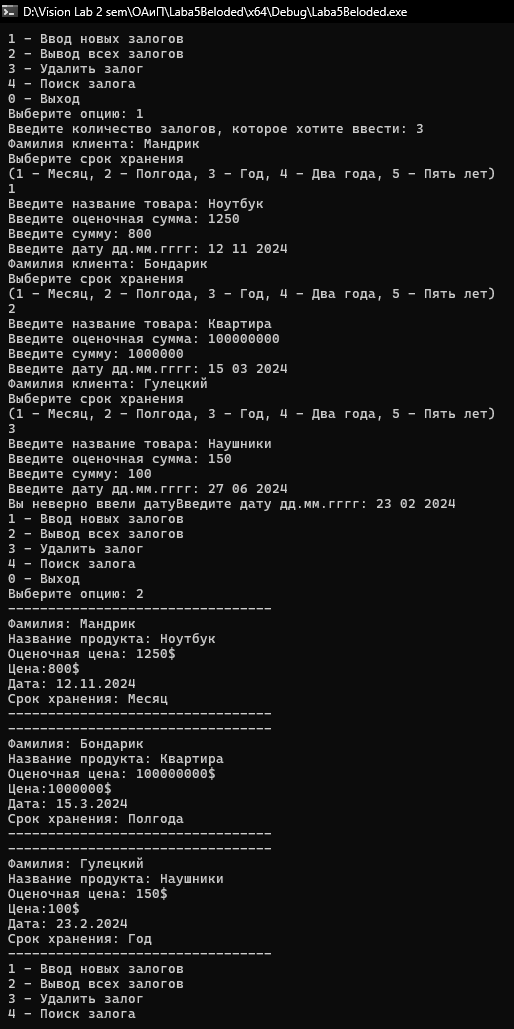
}

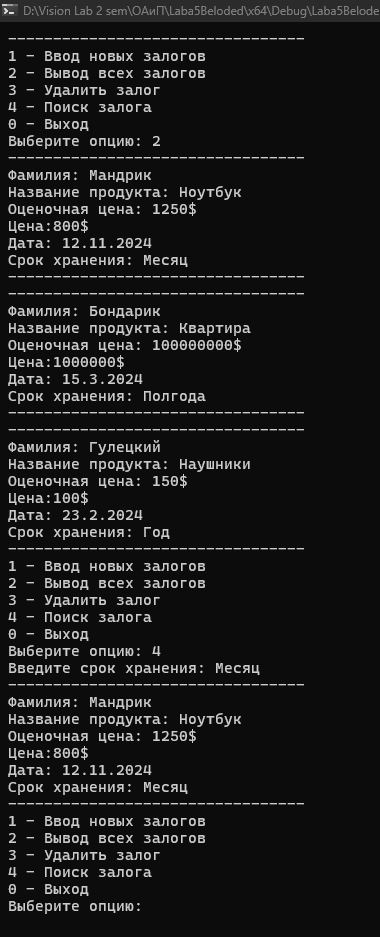
} while (variable != 0); // Повторяем цикл, пока пользователь не выберет выход

return 0;

}

**Результат выполнения:**





**Задание 2**

****

**Код программы:**

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;

// Определение объединения для даты

union DATE\_ISSUE {

unsigned int day; // День

unsigned int month; // Месяц

unsigned int year; // Год

};

// Определение структуры для данных о машине

struct CAR\_DATA

{

string modelCar; // Модель машины

string color; // Цвет

int numberFactory; // Заводской номер

string typeBodyCar; // Тип кузова

string owner; // Владелец

// Вложенная структура для даты выпуска

struct DATE\_RELEASE {

unsigned int day : 5; // День

unsigned int month : 4; // Месяц

unsigned int year : 11; // Год

} date\_data = { 0, 0, 0 }; // Инициализация даты выпуска нулями

// Вложенная структура для даты технического осмотра

struct DATE\_TECH\_INSPECTION {

unsigned int day : 5; // День

unsigned int month : 4; // Месяц

unsigned int year : 11; // Год

} date\_tech = { 0, 0, 0 }; // Инициализация даты тех. осмотра нулями

// Функция для ввода данных о машине

void input() {

cout << "Напишите марку машины: ";

cin.ignore();

getline(cin, modelCar);

cout << "Введите цвет: ";

getline(cin, color);

cout << "Введите тип кузова: ";

getline(cin, typeBodyCar);

cout << "Введите дату выпуска (дд.мм.гггг): ";

DATE\_ISSUE date{};

repeatDate:

cin >> date.day;

if (date.day < 33) {

date\_data.day = date.day;

}

else {

cout << "Вы неверно ввели дату\n";

goto repeatDate;

};

cin >> date.month;

if (date.month < 13)

{

date\_data.month = date.month;

}

else {

cout << "Вы неверно ввели дату\n";

goto repeatDate;

};

cin >> date.year;

if (date.year < 2049)

{

date\_data.year = date.year;

}

else {

cout << "Вы неверно ввели дату\n";

goto repeatDate;

};

cout << "Введите дату тех. осмотра (дд.мм.гггг): ";

cin >> date.day;

if (date.day < 33) {

date\_tech.day = date.day;

}

else {

cout << "Вы неверно ввели дату\n";

};

cin >> date.month;

if (date.month < 13)

{

date\_tech.month = date.month;

}

else {

cout << "Вы неверно ввели дату\n";

};

cin >> date.year;

if (date.year < 2049)

{

date\_tech.year = date.year;

}

else {

cout << "Вы неверно ввели дату\n";

};

cout << "Введите владельца: ";

cin.ignore();

getline(cin, owner);

cout << "Введите заводской номер: ";

cin.ignore();

cin >> numberFactory;

}

// Функция для вывода данных о машине

void output() {

if (modelCar != "") {

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

cout << "Марка машины: " << modelCar << endl;

cout << "Заводской номер: " << numberFactory << endl;

cout << "Тип кузова: " << typeBodyCar << endl;

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

cout << "Дата выпуска: " << date\_data.day << " " << date\_data.month << " " << date\_data.year << endl;

cout << "Дата тех. осмотра: " << date\_tech.day << " " << date\_tech.month << " " << date\_tech.year << endl;

cout << "Владелец: " << owner << endl;

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

}

}

// Функция для поиска машины по хозяину

bool search(string destination) {

if (destination == this->owner) {

return true;

}

else {

return false;

}

}

};

int main() {

SetConsoleCP(1251); // Для русского алфавита

SetConsoleOutputCP(1251);

int variable, num;

string destination;

CAR\_DATA cars[10]; // Создание массива объектов "Машина"

// Меню

do

{

cout << "1 - Ввод новых машин\n"

<< "2 - Вывод имеющихся машин\n"

<< "3 - Найти машину\n"

<< "0 - Выход\n";

cout << "Выберите опцию: ";

cin >> variable;

switch (variable)

{

case 1: // Ввод новых машин

{

cout << "Введите количество машин, которое хотите ввести: ";

cin >> num;

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

{

cars[i].input(); // Вызов функции ввода для каждой машины

}

break;

}

case 2: // Вывод имеющихся машин

{

for (int i = 0; i < 10; i++)

{

cars[i].output(); // Вызов функция вывода для каждой машины

}

break;

}

case 3: // Поиск машины

cin.ignore();

getline(cin, destination);

for (int i = 0; i < 10; i++)

{

if (cars[i].search(destination)) {

cars[i].output(); // Если машина найдена, вызываем функцию вывода

}

}

break;

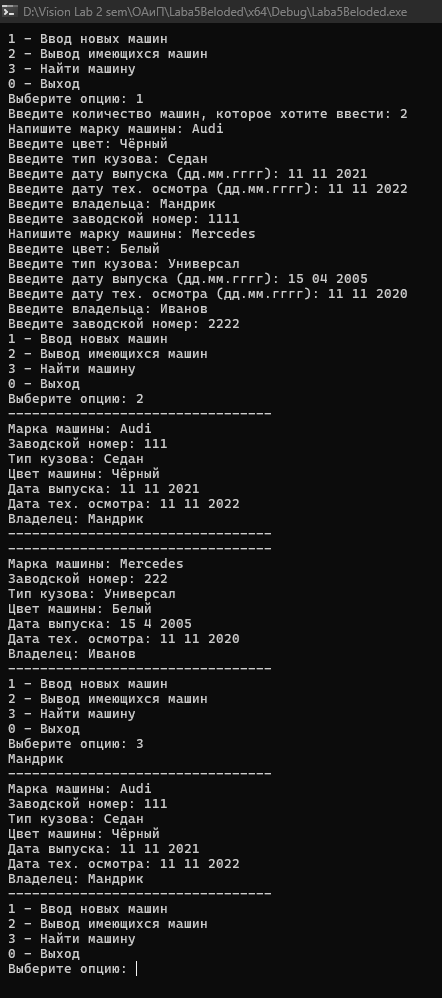
}

} while (variable != 0);

return 0;

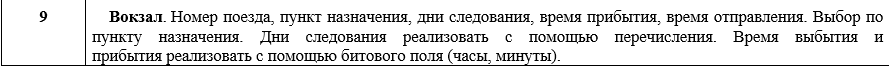
}

**Результат выполнения:**

****

**Вариант 9**

**Задание 1**

****

**Код программы:**

#include <iostream>

#include <cstring>

#include <Windows.h>

using namespace std;

// Перечисление для дней следования поезда

enum Days {

MONDAY, // Дни

TUESDAY,

WEDNESDAY,

THURSDAY,

FRIDAY,

SATURDAY,

SUNDAY

};

// Структура для данных о поезде

struct Train {

int number; // Номер

char destination[50]; // Пункт назначения

bool days[7]; // Дни следования

int departureTime : 11; // Битовое поле для времени отправления (11 битов: 5 битов для часов, 6 битов для минут)

int arrivalTime : 11; // Битовое поле для времени прибытия (11 битов: 5 битов для часов, 6 битов для минут)

};

// Функция для ввода данных о поезде с клавиатуры

void inputTrain(Train& train) {

cout << "Введите номер поезда: ";

cin >> train.number;

cin.ignore();

cout << "Введите пункт назначения: ";

cin.getline(train.destination, 50);

cout << "Введите дни следования (0 - нет, 1 - да) в формате Пн Вт Ср Чт Пт Сб Вс:\n";

for (int i = 0; i < 7; ++i) {

cout << "Следует ли поезд в " << (i == 0 ? "Пн" : i == 1 ? "Вт" : i == 2 ? "Ср" : i == 3 ? "Чт" : i == 4 ? "Пт" : i == 5 ? "Сб" : "Вс") << ": ";

cin >> train.days[i];

}

cout << "Введите время отправления (часы и минуты через пробел): ";

int hours, minutes;

cin >> hours >> minutes;

train.departureTime = (hours << 6) | minutes;

cout << "Введите время прибытия (часы и минуты через пробел): ";

cin >> hours >> minutes;

train.arrivalTime = (hours << 6) | minutes;

}

// Функция для вывода данных о поезде на экран

void outputTrain(const Train& train) {

cout << "Номер поезда: " << train.number << endl;

cout << "Пункт назначения: " << train.destination << endl;

cout << "Дни следования:\n";

const char\* dayNames[] = { "Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Вс" };

for (int i = 0; i < 7; ++i) {

if (train.days[i]) {

cout << dayNames[i] << ": Да\n";

}

else {

cout << dayNames[i] << ": Нет\n";

}

}

cout << "Время отправления: " << (train.departureTime >> 6) << ":" << (train.departureTime & 0b111111) << endl;

cout << "Время прибытия: " << (train.arrivalTime >> 6) << ":" << (train.arrivalTime & 0b111111) << endl;

}

// Функция для выбора поезда по пункту назначения

void selectTrainByDestination(const Train trains[], int numTrains, const char\* destination) {

cout << "Поезда с пунктом назначения \"" << destination << "\":\n";

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

if (strcmp(trains[i].destination, destination) == 0) {

outputTrain(trains[i]);

}

}

}

int main() {

SetConsoleCP(1251); // Для русского алфавита

SetConsoleOutputCP(1251);

const int MAX\_TRAINS = 100;

Train trains[MAX\_TRAINS];

int numTrains = 0;

int choice;

char destination[50];

// Меню

do {

cout << "1. Добавить поезд\n";

cout << "2. Вывести информацию о поезде\n";

cout << "3. Выбрать поезда по пункту назначения\n";

cout << "0. Выйти из программы\n";

cout << "Выберите действие: ";

cin >> choice;

cin.ignore();

switch (choice) {

case 1: // Ввод данных

if (numTrains < MAX\_TRAINS) {

inputTrain(trains[numTrains]);

numTrains++;

}

else {

cout << "Достигнуто максимальное количество поездов.\n";

}

break;

case 2: // Вывод данных

cout << "Введите номер поезда: ";

int trainNumber;

cin >> trainNumber;

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

if (trains[i].number == trainNumber) {

outputTrain(trains[i]);

break;

}

}

break;

case 3: // Поиск

cout << "Введите пункт назначения: ";

cin.getline(destination, 50);

selectTrainByDestination(trains, numTrains, destination);

break;

case 0:

cout << "Программа завершена.\n";

break;

default:

cout << "Некорректный ввод. Попробуйте еще раз.\n";

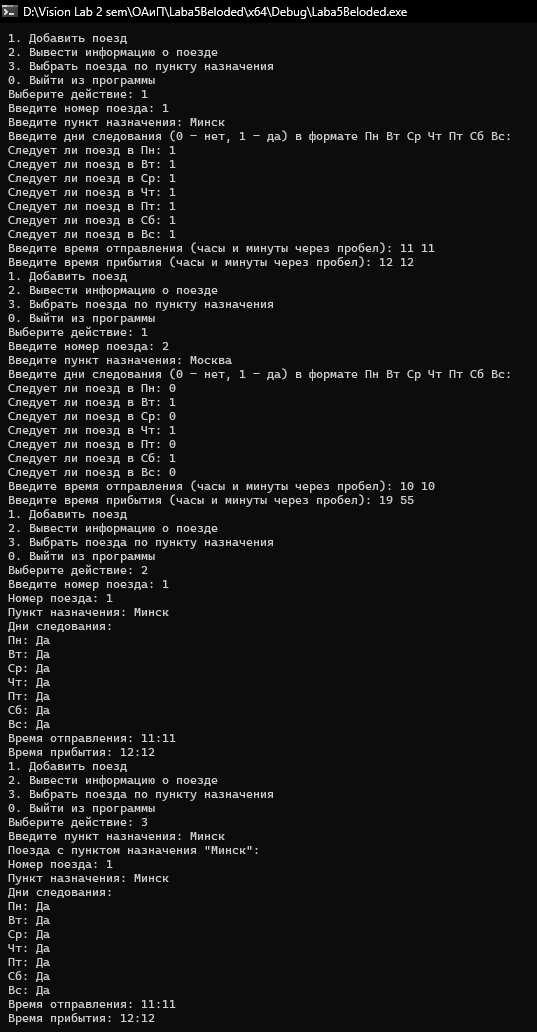
}

} while (choice != 0);

return 0;

}

**Результат выполнения:**

****

**Задание 2**

****

**Код программы:**

#include <iostream>

#include <cstring>

#include <Windows.h>

using namespace std;

const int MAX\_SUBJECTS = 5;

const int MAX\_NAME\_LENGTH = 50;

const int MAX\_STUDENTS = 10;

// Перечисление для предметов

enum Subject {

MATH,

PHYSICS,

CHEMISTRY,

BIOLOGY,

ENGLISH

};

// Для данных об ученике

union StudentData {

struct {

char name[MAX\_NAME\_LENGTH]; // ФИО

int grade;

bool subjects[MAX\_SUBJECTS]; // Выбор нужных предметов

float grades[MAX\_SUBJECTS]; // Отметки

float average; // Средний балл

};

};

// Структура для ученика

struct Student {

StudentData data;

};

// Функция для ввода данных об ученике с клавиатуры

void inputStudent(Student& student) {

cout << "Введите Ф.И.О. ученика: ";

cin.getline(student.data.name, MAX\_NAME\_LENGTH);

cout << "Введите класс: ";

cin >> student.data.grade;

cin.ignore();

cout << "Введите оценки по предметам:\n";

for (int i = 0; i < MAX\_SUBJECTS; ++i) {

cout << "Оценка по предмету " << i + 1 << ": ";

cin >> student.data.grades[i];

}

// Расчет среднего балла

float sum = 0;

for (int i = 0; i < MAX\_SUBJECTS; ++i) {

sum += student.data.grades[i];

}

student.data.average = sum / MAX\_SUBJECTS;

}

// Функция для вывода данных об ученике на экран

void outputStudent(const Student& student) {

cout << "Ф.И.О.: " << student.data.name << endl;

cout << "Класс: " << student.data.grade << endl;

cout << "Предметы:\n";

const char\* subjects[] = { "Математика", "Физика", "Химия", "Биология", "Английский" };

for (int i = 0; i < MAX\_SUBJECTS; ++i) {

cout << subjects[i] << ": " << student.data.grades[i] << endl;

}

cout << "Средний балл: " << student.data.average << endl;

}

// Функция для поиска ученика по фамилии

void findStudent(const Student students[], int numStudents, const char\* lastName) {

bool found = false;

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

if (strstr(students[i].data.name, lastName) != nullptr) {

outputStudent(students[i]);

found = true;

}

}

if (!found) {

cout << "Ученик с фамилией '" << lastName << "' не найден.\n";

}

}

int main() {

// Для русского алфавита

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Student students[MAX\_STUDENTS];

int numStudents = 0;

// Меню

int choice;

do {

cout << "\nМеню:\n";

cout << "1. Добавить ученика\n";

cout << "2. Вывести информацию об учениках\n";

cout << "3. Поиск ученика по фамилии\n";

cout << "0. Выйти из программы\n";

cout << "Выберите действие: ";

cin >> choice;

cin.ignore();

switch (choice) {

case 1: // Ввод данных

if (numStudents < MAX\_STUDENTS) {

cout << "Ученик " << numStudents + 1 << ":\n";

inputStudent(students[numStudents]);

++numStudents;

}

else {

cout << "Достигнуто максимальное количество учеников.\n";

}

break;

case 2: // Вывод данных

cout << "\nИнформация о введенных учениках:\n";

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

cout << "Ученик " << i + 1 << ":\n";

outputStudent(students[i]);

cout << endl;

}

break;

case 3: // Поиск

char lastName[MAX\_NAME\_LENGTH];

cout << "Введите фамилию для поиска: ";

cin.getline(lastName, MAX\_NAME\_LENGTH);

cout << "\nРезультаты поиска:\n";

findStudent(students, numStudents, lastName);

break;

case 0:

cout << "Программа завершена.\n";

break;

default:

cout << "Некорректный ввод. Попробуйте еще раз.\n";

}

} while (choice != 0);

return 0;

}

**Результат выполнения:**

