Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Лабораторная работа 4

По дисциплине «Основы алгоритмизации и программирования»

На тему «Представление информации в виде структуры»

 Выполнила:

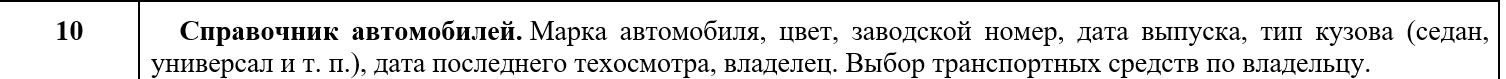
Студентка1 курса 6 группы

Литвинчук Дарья Валерьевна

Преподаватель: асс. Андронова М.В.

2024, Минск

**Вариант 10 (основной)**



#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <Windows.h>

using namespace std;

// Структура автомобиля

struct Car {

string brand;

string color;

string serialNumber;

string manufactureDate;

string bodyType;

string lastInspectionDate;

string owner;

};

// Вектор для хранения автомобилей

vector<Car> cars;

// Функция для ввода данных об автомобиле

void addCar() {

Car car;

cout << "Марка автомобиля: ";

cin.ignore();

getline(cin, car.brand);

cout << "Цвет: ";

getline(cin, car.color);

cout << "Заводской номер: ";

getline(cin, car.serialNumber);

cout << "Дата выпуска: ";

getline(cin, car.manufactureDate);

cout << "Тип кузова: ";

getline(cin, car.bodyType);

cout << "Дата последнего техосмотра: ";

getline(cin, car.lastInspectionDate);

cout << "Владелец: ";

getline(cin, car.owner);

cars.push\_back(car);

cout << "Автомобиль успешно добавлен!" << endl;

}

// Функция для вывода данных об автомобилях

void displayCars() {

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

cout << "Автомобиль " << i + 1 << ":" << endl;

cout << "Марка автомобиля: " << cars[i].brand << endl;

cout << "Цвет: " << cars[i].color << endl;

cout << "Заводской номер: " << cars[i].serialNumber << endl;

cout << "Дата выпуска: " << cars[i].manufactureDate << endl;

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

cout << "Дата последнего техосмотра: " << cars[i].lastInspectionDate << endl;

cout << "Владелец: " << cars[i].owner << endl;

cout << endl;

}

}

// Функция для удаления автомобиля по заданному владельцу

void deleteCarByOwner() {

string owner;

cout << "Введите владельца автомобиля: ";

cin.ignore();

getline(cin, owner);

bool found = false;

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

if (cars[i].owner == owner) {

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

found = true;

}

}

if (found) {

cout << "Автомобили владельца " << owner << " успешно удалены!" << endl;

}

else {

cout << "Автомобили владельца " << owner << " не найдены!" << endl;

}

}

// Функция для поиска автомобиля

void searchCar() {

string owner;

cout << "Введите Владельца автомобиля: ";

cin.ignore();

getline(cin, owner);

bool found = false;

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

if (cars[i].owner == owner) {

cout << "Автомобиль " << i + 1 << ":" << endl;

cout << "Марка автомобиля: " << cars[i].brand << endl;

cout << "Цвет: " << cars[i].color << endl;

cout << "Заводской номер: " << cars[i].serialNumber << endl;

cout << "Дата выпуска: " << cars[i].manufactureDate << endl;

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

cout << "Дата последнего техосмотра: " << cars[i].lastInspectionDate << endl;

cout << "Владелец: " << cars[i].owner << endl;

cout << endl;

found = true;

}

}

if (!found) {

cout << "Автомобили владельца " << owner<< " не найдены!" << endl;

}

}

// Функция для записи данных об автомобилях в файл

void writeToFile() {

ofstream file("cars.txt");

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

file << "Автомобиль " << i + 1 << ":" << endl;

file << "Марка автомобиля: " << cars[i].brand << endl;

file << "Цвет: " << cars[i].color << endl;

file << "Заводской номер: " << cars[i].serialNumber << endl;

file << "Дата выпуска: " << cars[i].manufactureDate << endl;

file << "Тип кузова: " << cars[i].bodyType << endl;

file << "Дата последнего техосмотра: " << cars[i].lastInspectionDate << endl;

file << "Владелец: " << cars[i].owner << endl;

file << endl;

}

file.close();

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

}

// Функция для чтения данных об автомобилях из файла

void readFromFile() {

ifstream file("cars.txt");

if (!file) {

cout << "Файл не найден!" << endl;

return;

}

string line;

while (getline(file, line)) {

cout << line << endl;

}

file.close();

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "rus");

int choice;

do {

cout << "Меню:" << endl;

cout << "1. Ввод информации об автомобиле" << endl;

cout << "2. Вывод информации об автомобилях" << endl;

cout << "3. Удаление автомобиля по владельцу" << endl;

cout << "4. Поиск автомобиля по владельцу" << endl;

cout << "5. Запись информации в файл" << endl;

cout << "6. Чтение информации из файла" << endl;

cout << "0. Выход" << endl;

cout << "Введите номер пункта меню: ";

cin >> choice;

cout << endl;

switch (choice) {

case 1:

addCar();

break;

case 2:

displayCars();

break;

case 3:

deleteCarByOwner();

break;

case 4:

searchCar();

break;

case 5:

writeToFile();

break;

case 6:

readFromFile();

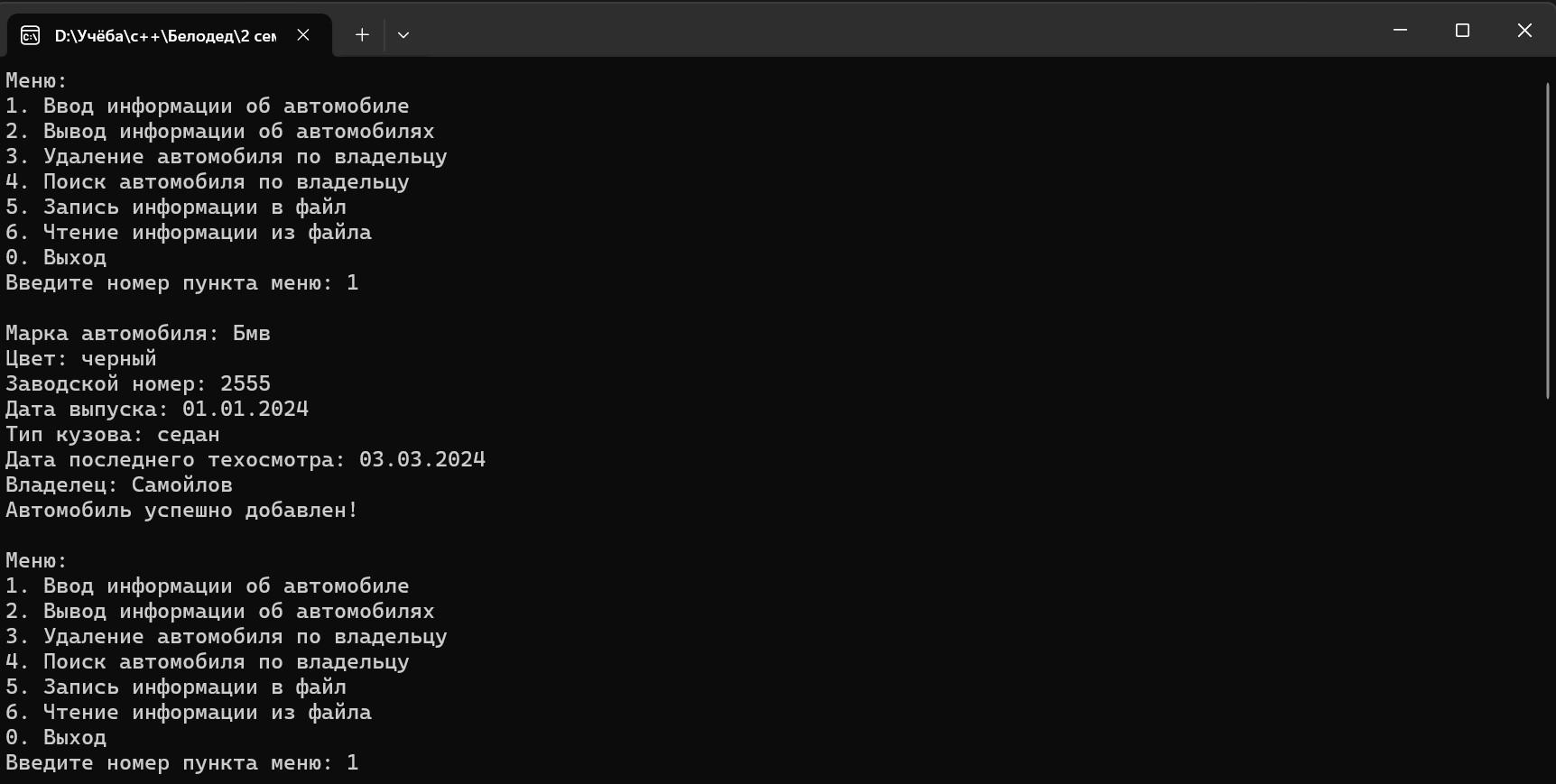
break;

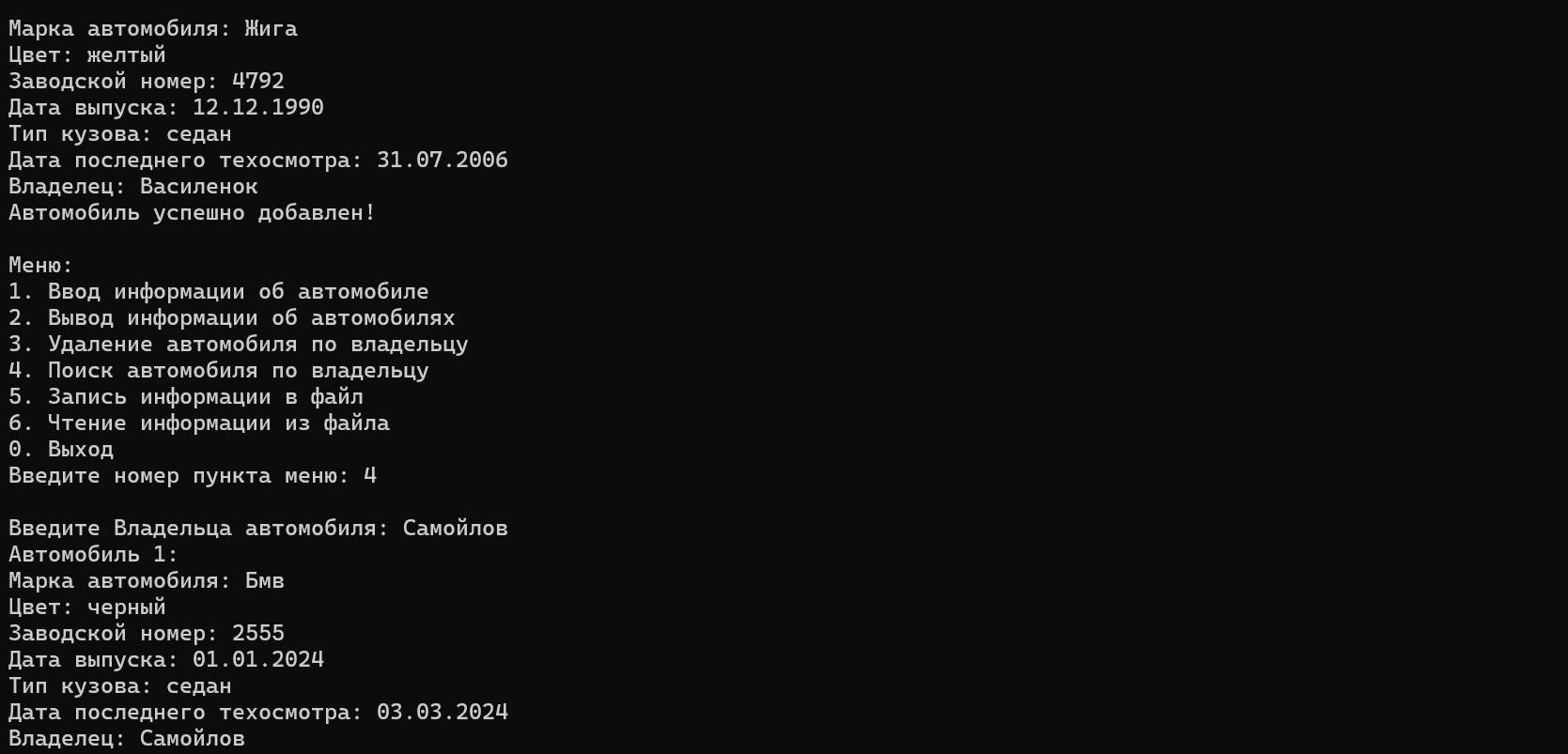
}

cout << endl;

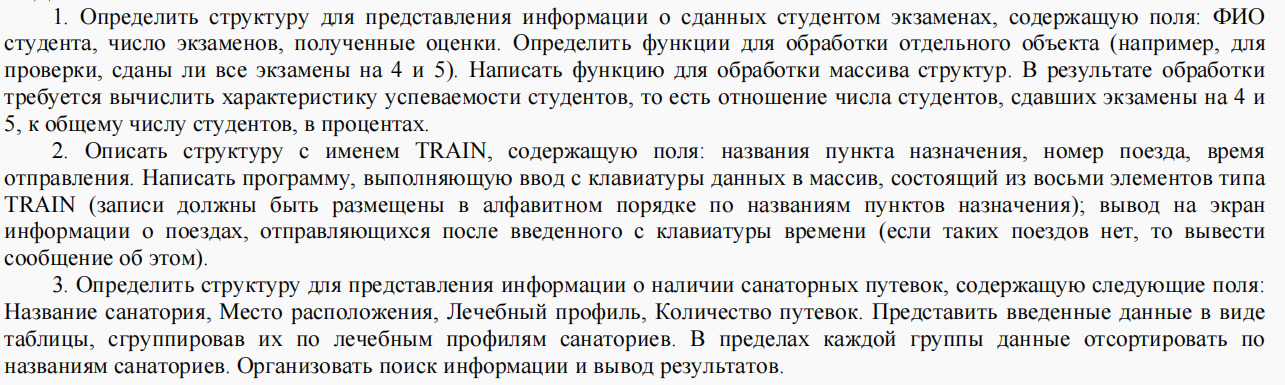
} while (choice != 0);

return 0;

}



**Доп. задания:**

1.  


#include <iostream>

#include <Windows.h>

//прототипы функций

using namespace std;

bool isExamPassed(int);

void getStatistics();

// структура

struct Marks

{

char FIO[80]; //фио ученика

const int amount = 3; //количество эзкаменов

int marks[3]; //массив с оценкам

};

//определяем структуру для студентов и их оценками по экзамену

const int students = 3;

Marks exam\_list[students] = {

{"Student1", 3, {4, 5, 4}},

{"Student2", 3, {5, 5, 3}},

{"Student3", 3, {4, 4, 4}}

};

// главная функция

int main()

{

SetConsoleOutputCP(1251); // для распознавания кириллицы при записи в массив

SetConsoleCP(1251);

//определяем, сдали ли студенты все экзамены

for (int i = 0; i < exam\_list[i].amount; i++) {

if (isExamPassed(i)) {

cout << "Студент №" << i + 1 << " сдал все экзамены" << endl;

}

else {

cout << "Студент №" << i + 1 << " сдал НЕ все экзамены" << endl;

}

}

// получаем статистику по предмету

getStatistics();

}

// фукнция для проверки сдачи всех экзаменов студентом

bool isExamPassed(int num){

for (int j = 0; j < exam\_list[num].amount; j++) { //цикл для обработки массива в массиве структур

if (exam\_list[num].marks[j] < 4) { //если найдена оценка меньше 4

return false;

}

}

return true;

}

// функция для сбора статистику по предмету

void getStatistics() {

for (int i = 0; i < exam\_list[i].amount; i++) {

double result\_4 = 0, result\_5 = 0; // создаем счетчики для подсчета отношений

for (int j = 0; j < exam\_list[i].amount; j++) {

if (exam\_list[j].marks[i] == 4) {

result\_4++;

}

if (exam\_list[j].marks[i] == 5) {

result\_5++;

}

}

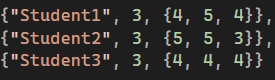
//выводим соотв. сообщения

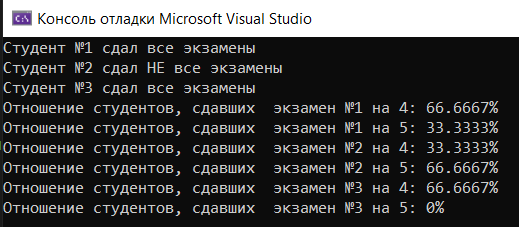
cout << "Отношение студентов, сдавших экзамен №" << i + 1 << " на 4: " << (result\_4 / students) \* 100 << "%" << endl;

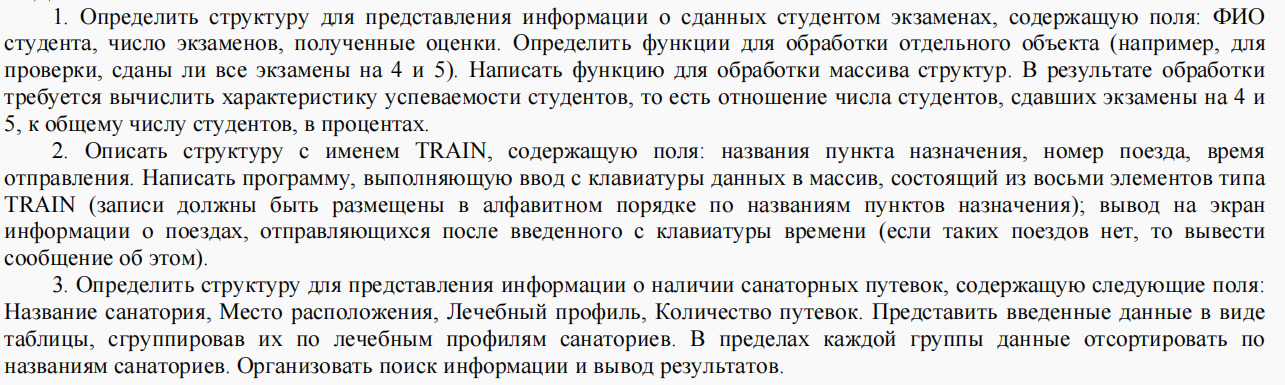
cout << "Отношение студентов, сдавших экзамен №" << i + 1 << " на 5: " << (result\_5 / students) \* 100 << "%" << endl;

}

}





2. 

#include <iostream>

#include <Windows.h>

using namespace std;

// структура

struct TRAIN

{

char destination[80]; // пункт назначения

unsigned int train\_number; //номер поезда

char arrival\_time[80]; //время прибытия

};

// прототипы функций

void sortStructures(TRAIN train[], int num);

void compareTime(TRAIN train[], char ask\_time[]);

//определяем структуру для поездов и информации и о них

TRAIN trains[8];

// главная функция

int main()

{

SetConsoleOutputCP(1251); // для распознавания кириллицы при записи в массив

SetConsoleCP(1251);

char ask\_time[8]; //время, которое запрашивает пользователь

//ввод данных в массив структуры

for (int i = 0; i < 8; i++) {

cout << "Поезд №" << i + 1 << ": " << endl;

cout << "Введите пункт назначения: ";

cin >> trains[i].destination;

cout << "Введите номер поезда: ";

cin >> trains[i].train\_number;

cout << "Введите время прибытия в формате ЧЧ:ММ: ";

cin >> trains[i].arrival\_time;

}

sortStructures(trains, 8); //сортируем элементы

cout << "Введите интересующее вас время в формате ЧЧ:ММ " << endl;

cin >> ask\_time;

compareTime(trains, ask\_time); //сравниваем время прибытия и введенное значение

return 0;

}

// функция для сортировку поездов по алфавиту

void sortStructures(TRAIN train[], int num) {

for (int i = 0; i < num - 1; i++) {

for (int j = 0; j < num - i - 1; j++) {

if (strcmp(train[j].destination, train[j + 1].destination) > 0) { // сравниваем строки пунктов назначения

swap(train[j], train[j + 1]);

}

}

}

}

// фукнция для сравнения времен

void compareTime(TRAIN train[], char ask\_time[]) {

bool flag = false; // по умолчанию считается, что поездов с подходящим временем не нашлось

for (int i = 0; i < 8; i++) { //сравнвиваем посимвольно время

if (strcmp(train[i].arrival\_time, ask\_time) > 0) { //если время поездов больше введенного времени

cout << "Вам подходит: " << endl; // выводим сообщение

cout << "Поезд №" << trains[i].train\_number << ". Пункт назначения: " << trains[i].destination << " . Время прибытия: " << trains[i].arrival\_time << endl;

flag = true; //устанавливаем флажок, что нашли

}

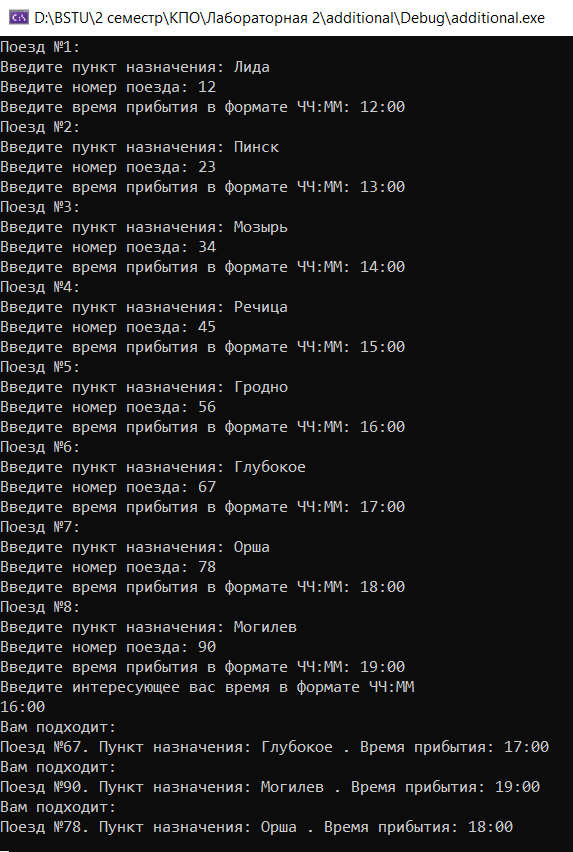
}

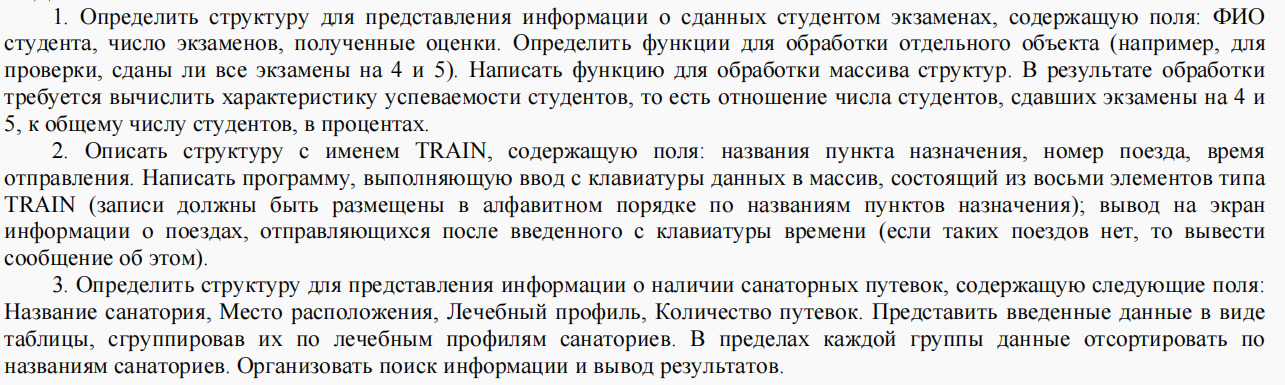
if (!flag) { //если время поездов меньше введенного времени

cout << "Поездов с подходящим временем не найдено" << endl;

}

}



3. 

#include <iostream>

#include <Windows.h>

using namespace std;

// структура

struct SANATORIUM

{

char name[40]; // название санатория

char location[40]; // местоположение

char profile[40]; //профиль

unsigned int amount\_package; //количество путевок

};

// прототипы функций

void sortStructures(SANATORIUM sanat[], int num);

void findSanat(SANATORIUM sanat[], char ask\_sanat[]);

void outputSanatoriums(SANATORIUM sanat[], int num);

//определяем структуру для санаториев и информации и о них

SANATORIUM sanat[5] = {

{"Пралеска", "Беларусь", "Профиль A", 13},

{"Зеленый бор", "Беларусь", "Профиль Б", 7},

{"Нарочь", "Беларусь", "Профиль A", 2},

{"Криница", "Беларусь", "Профиль A", 9},

{"Буг", "Беларусь", "Профиль Б", 6}

};

// главная функция

int main()

{

SetConsoleOutputCP(1251); // для распознавания кириллицы при записи в массив

SetConsoleCP(1251);

char ask\_sanat[40]; // переменная для запроса профиля

//ввод данных в массив структуры

sortStructures(sanat, 5); //сортируем элементы

outputSanatoriums(sanat, 5);

cout << "Какой санаторий вас интересует? " << endl;

cin >> ask\_sanat;

findSanat(sanat, ask\_sanat); //ищем нужный санаторий

return 0;

}

// функция для сортировки санаториев по алфавиту

void sortStructures(SANATORIUM sanat[], int num) {

for (int i = 0; i < num - 1; i++) {

for (int j = 0; j < num - i - 1; j++) {

if (strcmp(sanat[j].name, sanat[j + 1].name) > 0) { // сравниваем строки названий санаториев

swap(sanat[j].name, sanat[j + 1].name);

}

}

}

}

//вывод санаториев в виде таблицы

void outputSanatoriums(SANATORIUM sanat[], int num) {

// первая строка таблицы

cout << "Лечебный профиль" << "\t \t";

cout << "Название" << "\t \t";

cout << "Местоположение" << endl;

// вторая строка таблицы

cout << "Санатории по профилю А" << "\t \t";

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

if (strcmp(sanat[i].profile, "Профиль A") == 0) {

cout << sanat[i].name << "\t \t \t";

cout << sanat[i].location << endl << "\t \t \t \t";

}

}

// третья строка таблицы

cout << "\rСанатории по профилю Б" << "\t \t";

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

if (strcmp(sanat[i].profile, "Профиль Б") == 0) {

cout << sanat[i].name << "\t \t";

cout << sanat[i].location << endl << "\t \t \t \t";

}

}

cout << "\r"; //возвращение каретки в начало строки

}

// поиск путевок по названию санатория

void findSanat(SANATORIUM sanat[], char ask\_sanat[]) {

bool found = false; // флаг для отслеживания, был ли найден санаторий

for (int i = 0; i < 5; i++) {

if (strcmp(sanat[i].name, ask\_sanat) == 0) {

cout << "Количество доступных путевок: " << sanat[i].amount\_package << endl;

found = true;

break; // если нашли санаторий, то выходим из цикла

}

}

if (!found) { // если санаторий не был найден

cout << "Санаторий не был найден" << endl;

}

}

