Банкузов Михаил 7 группа Лабораторная работа №4 Вариант 1



#include <iostream>

#include <fstream>

#include <string>

using namespace std;

const int MAX\_CITIZEN = 100;

//создаем структуру

struct Citizen {

string name;

string gender;

int year;

string adress;

};

void addCitizen(Citizen Citizens[], int& CitizenCount) { //добавляем элемент структуры в массив Она принимает два параметра: массив структур и количество элементов в массиве

// функция добавления новой структуры в массив Citizens

if (CitizenCount < MAX\_CITIZEN) {

cout << "Citizen name: ";

cin >> Citizens[CitizenCount].name;

cout << "Enter gender: ";

cin >> Citizens[CitizenCount].gender;

cout << "Enter year: ";

cin >> Citizens[CitizenCount].year;

cout << "Enter adress: ";

cin >> Citizens[CitizenCount].adress;

//после того, как данные введены, элемент массива Citizens с индексом CitizenCount заполняется полученными значениями

//в массиве стало на один элемент больше

CitizenCount++;

}

else {

cout << "The array is full" << endl;

}

}

void printCitizens(Citizen Citizens[], int CitizenCount) {

// функция вывода всех структур из массива в консольное окно

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

cout << "Citizen name: " << Citizens[i].name << endl;

cout << "Gender: " << Citizens[i].gender << endl;

cout << "Serial number: " << Citizens[i].year << endl;

cout << "Adress: " << Citizens[i].adress << endl;

cout << endl;

}

}

void deleteCitizen(Citizen Citizens[], int& CitizenCount) {

// функция удаления заданной структуры из массива

int year;

cout << "Enter year of Citizen to delete: ";

cin >> year;

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

if (Citizens[i].year == year) {

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

Citizens[j] = Citizens[j + 1];

}

CitizenCount--;

}

}

}

void searchCitizen(Citizen Citizens[], int CitizenCount) {

// функция поиска информации по заданным параметрам

int year;

cout << "Enter year of Citizen to search: ";

cin >> year;

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

if (Citizens[i].year == year) {

cout << "Citizen name: " << Citizens[i].name << endl;

cout << "Gender: " << Citizens[i].gender << endl;

cout << "Year: " << Citizens[i].year << endl;

cout << "Adress: " << Citizens[i].adress << endl;

}

}

}

void writeToFile(Citizen Citizens[], int CitizenCount) {

// функции записи всех структур из массива в файл

ofstream fout("Citizens.txt");

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

fout << "Citizen name: " << Citizens[i].name << endl;

fout << "Gender: " << Citizens[i].gender << endl;

fout << "Year: " << Citizens[i].year << endl;

fout << "Adress: " << Citizens[i].adress << endl;

fout << endl;

}

fout.close();

}

void readFromFile(Citizen Citizens[], int& CitizenCount) {

// функция чтения всех структур из файла и записи их в массив

ifstream fin("Citizens.txt"); //создаем файл

// чтение из файла

string line;

while (getline(fin, line)) { //функция читает файл построчно

cout << line << endl;

}

fin.close();

}

int main() {

Citizen Citizens[MAX\_CITIZEN];

int CitizenCount = 0;

int choice;

do {

cout << "MENU" << endl;

cout << "1. Add Citizen" << endl;

cout << "2. Print all Citizens" << endl;

cout << "3. Delete Citizen" << endl;

cout << "4. Search by year" << endl;

cout << "5. Write to file" << endl;

cout << "6. Read from file" << endl;

cout << "0. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

addCitizen(Citizens, CitizenCount);

break;

case 2:

printCitizens(Citizens, CitizenCount);

break;

case 3:

deleteCitizen(Citizens, CitizenCount);

break;

case 4:

searchCitizen(Citizens, CitizenCount);

break;

case 5:

writeToFile(Citizens, CitizenCount);

break;

case 6:

readFromFile(Citizens, CitizenCount);

break;

case 0:

break;

default:

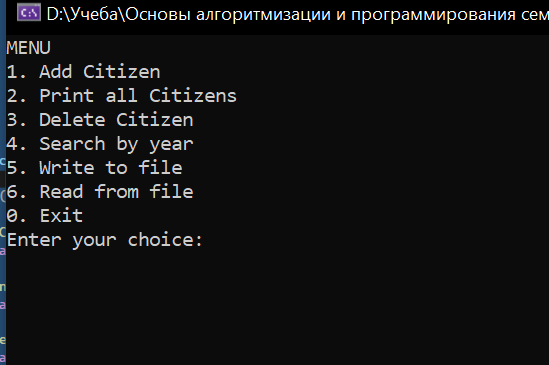
cout << "Invalid choice" << endl;

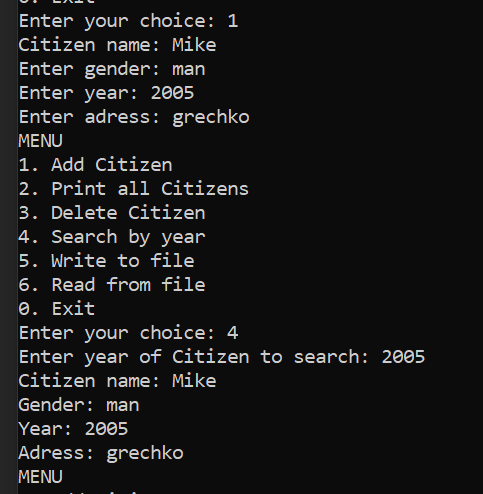
}

} while (choice != 0);

return 0;

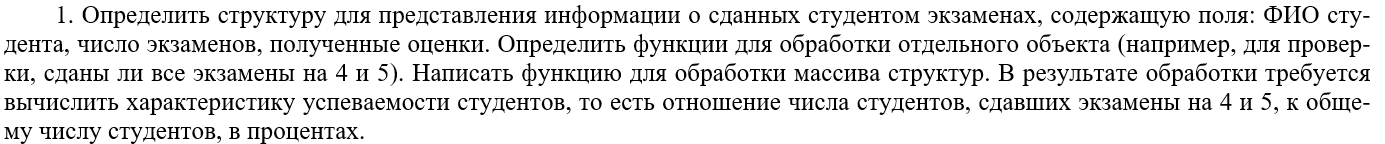
}





Дополнительные задания

Задание 1



#include <iostream>

#include <string>

using namespace std;

// Определение структуры для представления информации о студенте

struct Student {

string name; // ФИО студента

int examsCount; // Число экзаменов

int\* grades; // Массив оценок

};

// Функция для проверки, сдали ли все экзамены на 4 и 5

bool passedWithHighMarks(Student student) {

for (int i = 0; i < student.examsCount; i++) {

if (student.grades[i] < 4) {

return false;

}

}

return true;

}

// Функция для обработки массива структур

float calculatePerformance(Student\* students, int count) {

int passedCount = 0;

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

if (passedWithHighMarks(students[i])) {

passedCount++;

}

}

return (float)passedCount / (float)count \* 100.0;

}

int main() {

// Пример использования функций

Student students[4] = {

{"Ivanov Ivan", 3, new int[3] {3, 4, 5}},

{"Petrov Petr", 4, new int[4] {4, 4, 4, 4}},

{"Sidorov Sidr", 2, new int[2] {5, 5}},

{"Olegovich Oleg", 3, new int[3] {5, 5, 2}}

};

float performance = calculatePerformance(students, 3);

cout << "Performance: " << performance << "%" << endl;

// Освобождение памяти, занятой массивом оценок

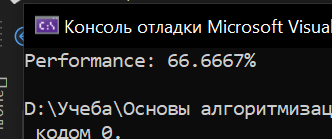
for (int i = 0; i < 3; i++) {

delete[] students[i].grades;

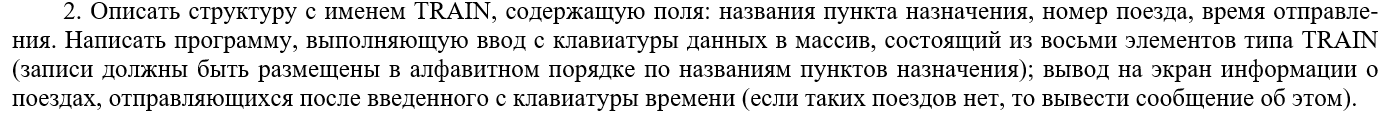
}

return 0;

}



Задание 2



#include <iostream>

#include <string>

using namespace std;

// Определение структуры TRAIN

struct TRAIN {

string destination;

int trainNumber;

string departureTime;

};

int main() {

setlocale(LC\_ALL, "Russian");

const int numTrains = 8; // количество поездов

TRAIN trains[numTrains]; // массив структур TRAIN

string searchTime; // время, которое мы будем искать

// Ввод данных о поездах

cout << "Введите данные о " << numTrains << " поездах:" << endl;

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

cout << "Поезд #" << i + 1 << endl;

cout << "Название пункта назначения: ";

cin >> trains[i].destination;

cout << "Номер поезда: ";

cin >> trains[i].trainNumber;

cout << "Время отправления: ";

cin >> trains[i].departureTime;

}

// Сортировка массива по названию пунктов назначения

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

for (int j = i + 1; j < numTrains; j++) {

if (trains[i].destination > trains[j].destination) {

swap(trains[i], trains[j]);

}

}

}

// Вывод информации о поездах, отправляющихся после введенного времени

cout << "Введите время, после которого будут отправляться поезда: ";

cin >> searchTime;

bool found = false;

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

if (trains[i].departureTime > searchTime) {

cout << "Поезд #" << i + 1 << endl;

cout << "Название пункта назначения: " << trains[i].destination << endl;

cout << "Номер поезда: " << trains[i].trainNumber << endl;

cout << "Время отправления: " << trains[i].departureTime << endl;

found = true;

}

}

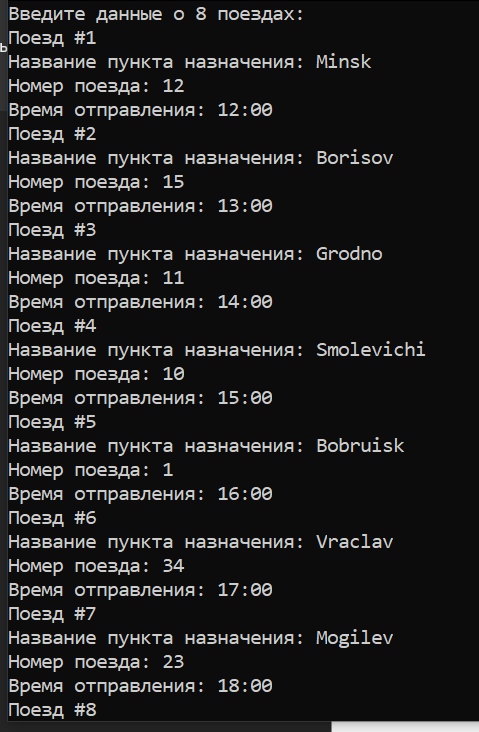
if (!found) {

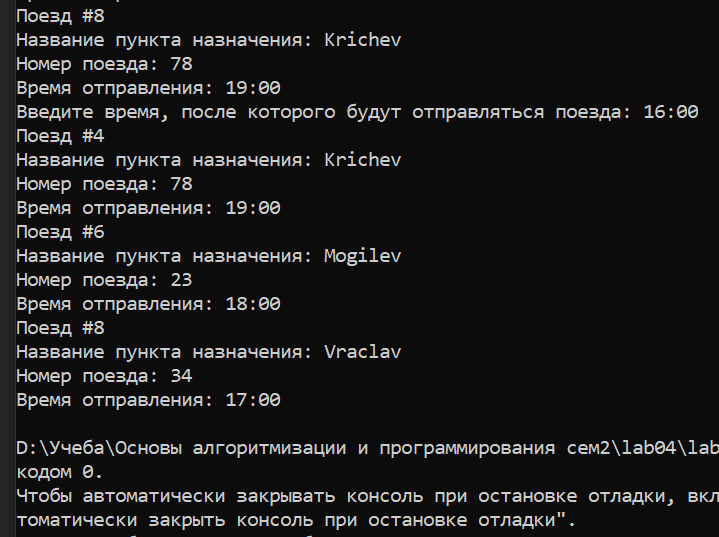
cout << "Поездов, отправляющихся после введенного времени, не найдено." << endl;

}

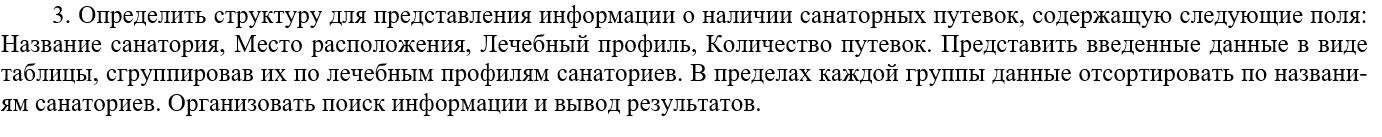
return 0;

}





Задание 3



#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

// определите структуру для представления санатория с названием, местоположением, медицинским профилем и количеством ваучеров

struct Sanatorium {

string name;

string location;

string medical\_profile;

int num\_vouchers;

};

// определите функцию для сравнения двух санаториев по названию

bool compareSanatoriumByName(const Sanatorium& s1, const Sanatorium& s2) {

return s1.name < s2.name;

}

int main() {

setlocale(LC\_ALL, "Russian");

// определите вектор санаториев

vector<Sanatorium> sanatoriums = {

{"Ynost", "Minsk", "Profile 1", 10},

{"Andorium", "Borisov", "Profile 3", 5},

{"Greck", "Vraclav", "Profile 2", 15},

{"Earth", "Mogilev", "Profile 1", 20}

};

// // определите вектор для хранения уникальных медицинских профилей санатория

vector<string> medicalProfiles;

for (const auto& sanatorium : sanatoriums) {

if (find(medicalProfiles.begin(), medicalProfiles.end(), sanatorium.medical\_profile) == medicalProfiles.end()) {

medicalProfiles.push\_back(sanatorium.medical\_profile);

}

}

// отсортировать вектор медицинских профилей в алфавитном порядке

sort(medicalProfiles.begin(), medicalProfiles.end());

// выполнить итерацию по вектору медицинских профилей

for (const auto& medicalProfile : medicalProfiles) {

// вывести название медицинского профиля

cout << "Medical Profile: " << medicalProfile << endl;

// распечатать строку заголовка для таблицы санаториев для текущего медицинского профиля

cout << "------------------------------------------------------------------" << endl;

cout << "Название | Место | Количество " << endl;

cout << "------------------------------------------------------------------" << endl;

// определите вектор для сохранения санатория для текущего медицинского профиля

vector<Sanatorium> sanatoriumsForProfile;

for (const auto& sanatorium : sanatoriums) {

// если текущий санаторий имеет текущий медицинский профиль, добавьте его в вектор санаториев для текущего профиля

if (sanatorium.medical\_profile == medicalProfile) {

sanatoriumsForProfile.push\_back(sanatorium);

}

}

// отсортировать вектор санаториев для текущего профиля по названию

sort(sanatoriumsForProfile.begin(), sanatoriumsForProfile.end(), compareSanatoriumByName);

for (const auto& sanatorium : sanatoriumsForProfile) {

printf("| %-20s | %-20s | %-18d |\n", sanatorium.name.c\_str(), sanatorium.location.c\_str(), sanatorium.num\_vouchers);

}

cout << "------------------------------------------------------------------" << endl;

cout << endl;

}

return 0;

}

