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»

Выполнил:

Студент 1 курса 8 группы

Лужецкий Владислав Константинович

Преподаватель: асс. Андронова М.В.

2024, Минск

**Основа**:



#include <iostream>

#include <cstdlib>

#include <cstdio>

#include <string>

#include <fstream>

using namespace std;

const int MAX\_NAME\_SIZE = 100;

struct product

{

char name[MAX\_NAME\_SIZE];

float price;

int percent;

int amount;

};

void input(int size);

void output();

void find(float price);

void deleteProduct(const string& productName);

void variant6()

{

int choice, number;

float price;

string prodName;

do

{

cout << "\n1.Ввод данных с клавиатуры и запись в файл\n";

cout << "2.Вывод данных из файла\n";

cout << "3.Поиск по цене\n";

cout << "4.Удаление товара\n";

cout << "0.Выход из программы\n\n";

cout << "Введите номер операции: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Введите количество товаров: ";

cin >> number;

input(number);

break;

case 2:

output();

break;

case 3:

cout << "Введите цену: ";

cin >> price;

find(price);

break;

case 4:

cout << "Введите название: ";

cin >> prodName;

deleteProduct(prodName);

break;

case 0:

exit(0);

break;

default:

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

break;

}

} while (choice != 0);

}

void input(int size)

{

FILE\* f;

product buf;

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

{

for (int p = 0; p < size; p++)

{

cout << "Название: ";

cin >> buf.name;

cout << "Количество: ";

cin >> buf.amount;

cout << "Цена: ";

cin >> buf.price;

if (buf.amount >= 0 && buf.amount < 5)

buf.percent = 30;

if (buf.amount >= 5 && buf.amount < 10)

buf.percent = 25;

if (buf.amount >= 10 && buf.amount < 15)

buf.percent = 20;

if (buf.amount >= 15 && buf.amount < 20)

buf.percent = 15;

if (buf.amount >= 20 && buf.amount < 25)

buf.percent = 10;

if (buf.amount >= 25)

buf.percent = 5;

cout << "Процент торговой надбавки: " << buf.percent << "\n\n";

fwrite(&buf, sizeof(buf), 1, f); // Записываем структуру в файл

}

fclose(f);

}

else

{

cout << "Ошибка открытия файла";

return;

}

}

void output()

{

FILE\* f;

product buf;

if (fopen\_s(&f, "base.txt", "rb") == 0) // Открываем файл в режиме чтения в двоичном режиме

{

cout << "\nНазвание количество цена процент\n";

fread(&buf, sizeof(buf), 1, f); // Читаем структуру из файла

while (!feof(f))

{

cout << buf.name << " " << buf.amount << " " << buf.price << " " << buf.percent << endl;

fread(&buf, sizeof(buf), 1, f); // Читаем следующую структуру из файла

}

cout << endl;

fclose(f);

}

else

{

cout << "Ошибка открытия файла";

return;

}

}

void find(float price)

{

FILE\* f;

product buf;

bool found = false;

if (fopen\_s(&f, "base.txt", "rb") == 0) // Открываем файл в режиме чтения в двоичном режиме

{

while (fread(&buf, sizeof(buf), 1, f) == 1)

{

if (buf.price == price)

{

cout << "\nНазвание Количество\n";

cout << buf.name << "\t " << buf.amount << endl;

found = true;

}

}

fclose(f);

if (!found)

cout << "Товар не найден\n";

}

else

{

cout << "Ошибка открытия файла";

return;

}

}

void deleteProduct(const string& productName)

{

ifstream inputFile("base.txt", ios::binary); // Открываем входной файл в двоичном режиме

ofstream tempFile("temp.bin", ios::binary); // Открываем временный файл для записи в двоичном режиме

if (!inputFile || !tempFile)

{

cout << "Ошибка открытия файлов" << endl;

return;

}

product buf;

bool deleted = false;

while (inputFile.read(reinterpret\_cast<char\*>(&buf), sizeof(buf)))

{

if (buf.name == productName)

{

deleted = true;

continue; // Пропускаем запись товара, который нужно удалить

}

tempFile.write(reinterpret\_cast<const char\*>(&buf), sizeof(buf)); // Записываем структуру во временный файл

}

inputFile.close();

tempFile.close();

if (deleted)

{

if (remove("base.txt") != 0) // Удаляем исходный файл

{

cout << "Ошибка удаления файла" << endl;

return;

}

if (rename("temp.bin", "base.txt") != 0) // Переименовываем временный файл в исходное имя

{

cout << "Ошибка переименования файла" << endl;

return;

}

cout << "Товар успешно удален" << endl;

}

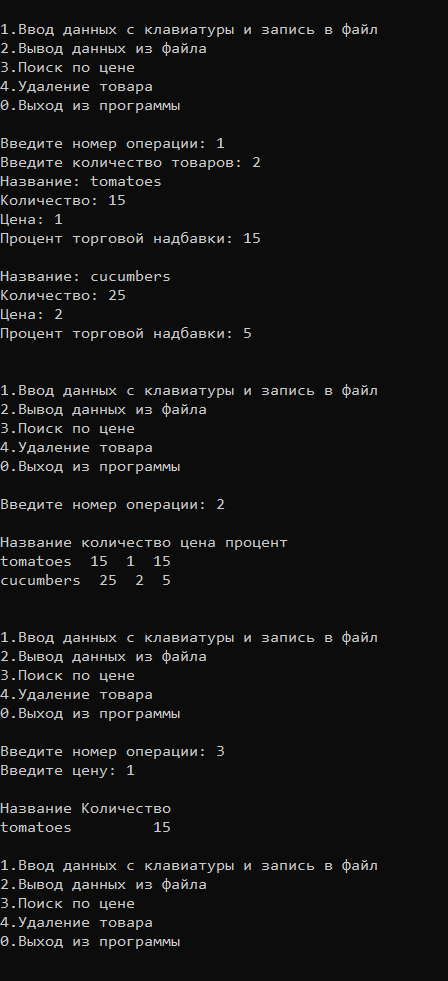
else

{

cout << "Товар не найден" << endl;

}

}



**Допы:**

**1**)

#include <iostream>

#include <string>

#include <Windows.h>

#define MAX\_STUDENTS 30

#define MAX\_EXAMS 6

using namespace std;

void inputStudents();

void outputStudents();

void checkSuccess();

void calculateSuccessRate();

int choice = 0;

int currentStudent = 0;

struct Student

{

string name; // Фамилия студента

int numExams; // Количество экзаменов

int marks[MAX\_EXAMS]; // Оценки за экзамены

};

Student listOfStudents[MAX\_STUDENTS];

int dop1()

{

do

{

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

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

cout << "3 - Проверка на успешность" << endl;

cout << "4 - Характеристика успеваемости" << endl;

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

cin >> choice; // Считываем выбор пользователя

switch (choice)

{

case 1:

inputStudents(); // Вызываем функцию для ввода данных о студентах

break;

case 2:

outputStudents(); // Вызываем функцию для вывода данных о студентах

break;

case 3:

checkSuccess(); // Вызываем функцию для проверки успешности студента

break;

case 4:

calculateSuccessRate(); // Вызываем функцию для расчета процента успешных студентов

break;

}

} while (choice != 5); // Повторяем цикл, пока пользователь не выберет выход

return 0;

}

void inputStudents()

{

int numStudents;

cout << "Введите количество студентов: ";

cin >> numStudents; // Считываем количество студентов

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

{

if (currentStudent < MAX\_STUDENTS)

{

cin.ignore();

cout << endl << "Фамилия студента: ";

getline(cin, listOfStudents[currentStudent].name); // Считываем фамилию студента

cout << "Количество экзаменов: ";

cin >> listOfStudents[currentStudent].numExams; // Считываем количество экзаменов

cout << "Оценки: " << endl;

for (int j = 0; j < listOfStudents[currentStudent].numExams; j++)

{

cout << "Введите оценку за " << j + 1 << " экзамен: ";

cin >> listOfStudents[currentStudent].marks[j]; // Считываем оценку за каждый экзамен

cout << endl;

}

currentStudent++;

}

}

}

void outputStudents()

{

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

{

cout << endl << "Фамилия студента: " << listOfStudents[i].name << endl; // Выводим фамилию студента

cout << "Количество экзаменов: " << listOfStudents[i].numExams << endl; // Выводим количество экзаменов

cout << "Оценки: " << endl;

for (int j = 0; j < listOfStudents[i].numExams; j++)

{

cout << "Оценка за " << j + 1 << " экзамен: " << listOfStudents[i].marks[j] << endl; // Выводим оценку за каждый экзамен

}

}

}

void checkSuccess()

{

int count = 0;

bool found = false;

string surname;

cout << "Введите фамилию студента: ";

cin.ignore();

getline(cin, surname);

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

{

if (listOfStudents[i].name == surname)

{

for (int j = 0; j < listOfStudents[i].numExams; j++)

{

if (listOfStudents[i].marks[j] >= 4)

{

count++;

}

}

if (count == listOfStudents[i].numExams)

{

cout << endl << "Студент сдал все экзамены на 4 или выше." << endl;

}

else

{

cout << endl << "Студент не сдал все экзамены на 4 или выше." << endl;

}

found = true;

break;

}

}

if (!found)

{

cout << "Студента с такой фамилией не найдено!" << endl;

}

}

void calculateSuccessRate()

{

int successfulStudents = 0;

string surname;

cout << "Введите фамилию студента: ";

cin.ignore();

getline(cin, surname); // Считываем фамилию студента

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

{

if (listOfStudents[i].name == surname) // Проверяем, есть ли студент с введенной фамилией

{

int count = 0;

for (int j = 0; j < listOfStudents[i].numExams; j++)

{

if (listOfStudents[i].marks[j] >= 4) // Проверяем оценку за каждый экзамен

{

count++;

}

}

if (count == listOfStudents[i].numExams) // Проверяем, все ли оценки выше 4

{

successfulStudents++;

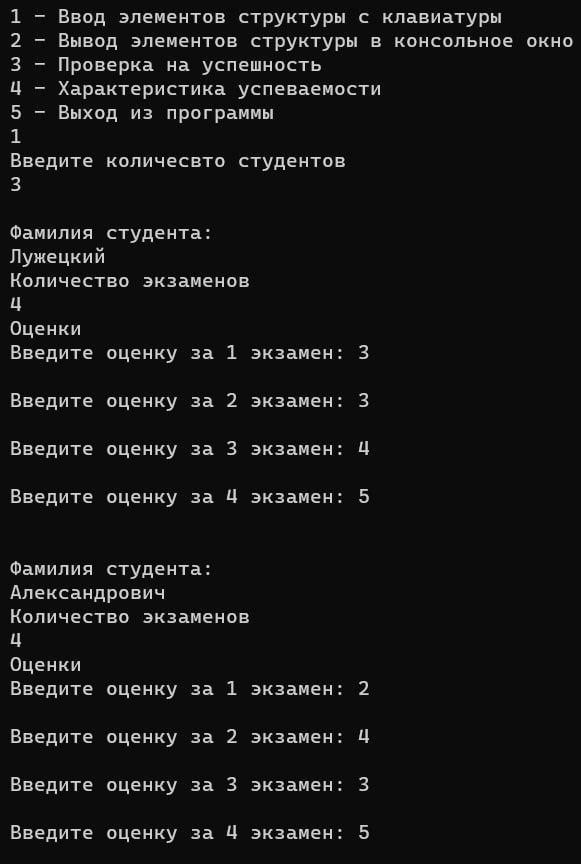
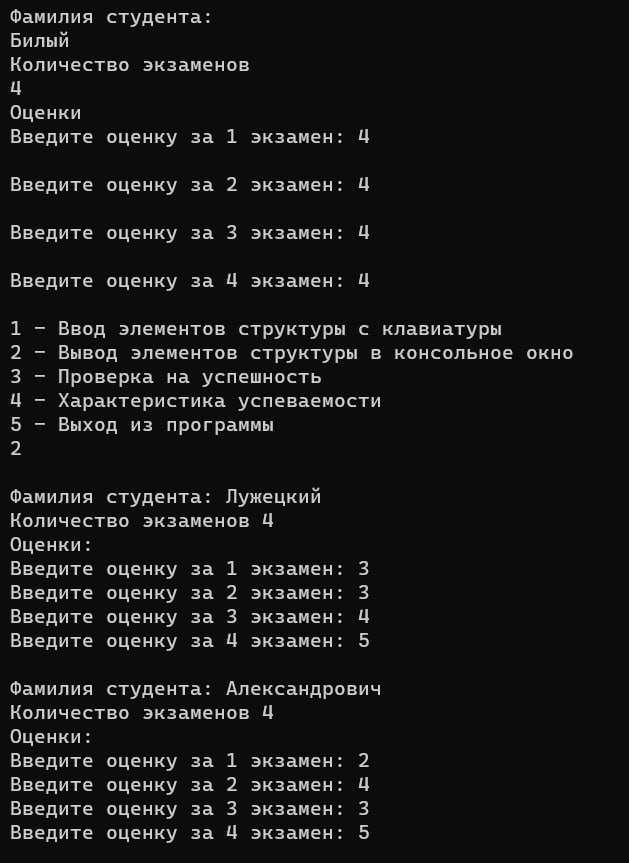
}

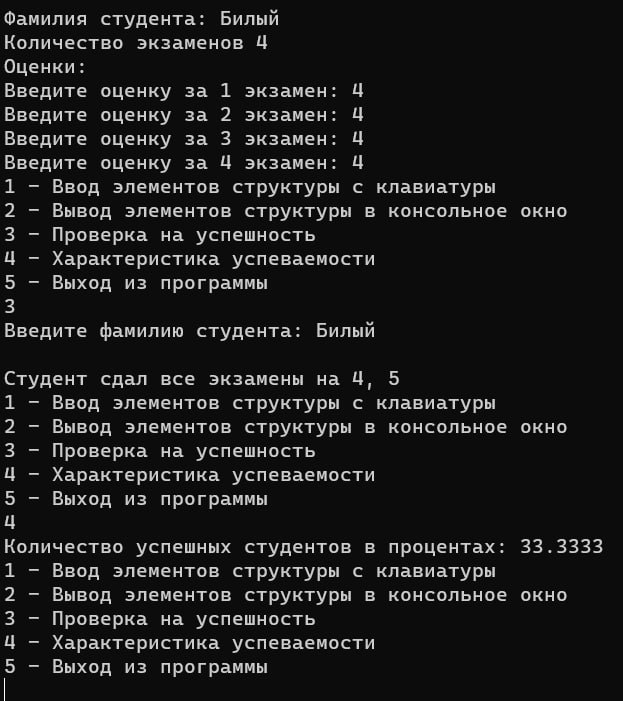
}

}

cout << "Количество успешных студентов: " << successfulStudents << endl;

}



**2)**

#define MAX\_TRAINS 10 **//на самом деле комментарии тут особо не нужны, также как и к 3 допу. Так как они все похожи и выполняют +- одно и тоже.**

struct Train

{

string destination; // пункт назначения

int number; // номер поезда

int departureTime; // время отправления

};

Train listOfTrains[MAX\_TRAINS];

void inputTrain();

void outputTrain();

void checkTrain();

int currentTrains = 0;

int dop2()

{

do

{

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

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

cout << "3 - Проверка на время" << endl;

cout << "0 - Выход из программы" << endl;

cin >> choice; // Считываем выбор пользователя

switch (choice)

{

case 1:

inputTrain();

break;

case 2:

outputStudents();

break;

case 3:

checkTrain();

break;

}

} while (choice != 0);

return 0;

}

void inputTrain()

{

int numTrains;

cout << "Введите количество поездов: ";

cin >> numTrains; // Считываем количество поездов

cin.ignore(); // Игнорируем символ новой строки после ввода числа

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

{

if (currentTrains < MAX\_TRAINS)

{

cout << endl << "Пункт назначения: ";

getline(cin, listOfTrains[currentTrains].destination);

cout << "Номер поезда: ";

cin >> listOfTrains[currentTrains].number;

cout << "Время отправления: ";

cin >> listOfTrains[currentTrains].departureTime;

cout << endl;

cin.ignore(); // Игнорируем символ новой строки после ввода времени

currentTrains++;

}

}

}

void outputTrain()

{

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

{

cout << endl << "Пункт назначения: " << listOfTrains[i].destination << endl;

cout << "Номер поезда: " << listOfTrains[i].number << endl;

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

}

}

void checkTrain()

{

int departureTime;

cout << "\nВведите текущее время: ";

cin >> departureTime;

bool foundTrain = false;

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

{

if (departureTime <= listOfTrains[i].departureTime)

{

foundTrain = true;

cout << endl << "Пункт назначения: " << listOfTrains[i].destination << endl;

cout << "Номер поезда: " << listOfTrains[i].number << endl;

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

}

}

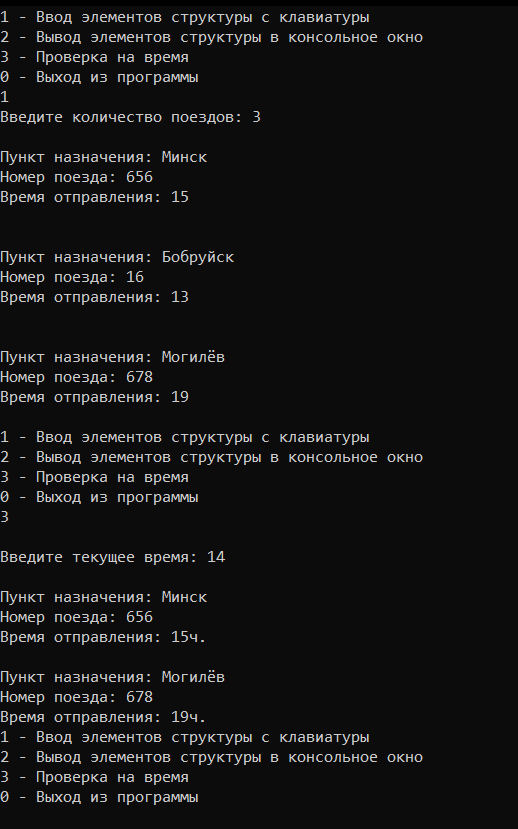
if (!foundTrain)

{

cout << "\nНет подходящих поездов.";

}

}

****

**3)**

#define MAX\_SANATORIUMS 10

struct Sanatorium

{

string name;

string location;

string direction; //направление

string amountPermit;

};

Sanatorium listOfSanatorium[MAX\_SANATORIUMS];

void inputSanatorium();

void outputSanatorium();

void foundSanatorium();

int currentSanatorium = 0;

int dop3()

{

do

{

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

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

cout << "3 - Поиск по названию" << endl;

cout << "0 - Выход из программы" << endl;

cin >> choice; // Считываем выбор пользователя

switch (choice)

{

case 1:

inputSanatorium();

break;

case 2:

outputSanatorium();

break;

case 3:

foundSanatorium();

break;

}

} while (choice != 0);

return 0;

}

void inputSanatorium()

{

int numSanatorium;

cout << "Введите количество санаториев: ";

cin >> numSanatorium; // Считываем количество санаториев

cin.ignore(); // Игнорируем символ новой строки после ввода числа

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

{

if (currentSanatorium < MAX\_SANATORIUMS)

{

cout << "\nНазвание санатория: ";

getline(cin, listOfSanatorium[currentSanatorium].name);

cout << "\nМесто расположения: ";

getline(cin, listOfSanatorium[currentSanatorium].location);

cout << "\nЛечебный профиль: ";

getline(cin, listOfSanatorium[currentSanatorium].direction);

cout << "\nКол-во путёвок: ";

cin >> listOfSanatorium[currentSanatorium].amountPermit;

cout << endl;

cin.ignore(); // Игнорируем символ новой строки после ввода времени

currentSanatorium++;

}

}

}

void outputSanatorium()

{

int choice;

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

{

cout << "\nНазвание санатория: " << listOfSanatorium[i].name;

cout << "\nМесто расположения: " << listOfSanatorium[i].location;

cout << "\nЛечебный профиль: " << listOfSanatorium[i].direction;

cout << "\nКол-во путёвок: " << listOfSanatorium[i].amountPermit;

cout << endl;

}

cin.ignore();

}

void foundSanatorium()

{

string name;

cout << "\nВведите название санатория: ";

cin >> name;

bool foundSanatorium = false;

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

{

if (listOfSanatorium[i].name == name)

{

foundSanatorium = true;

cout << "\nНазвание санатория: " << listOfSanatorium[i].name;

cout << "\nМесто расположения: " << listOfSanatorium[i].location;

cout << "\nЛечебный профиль: " << listOfSanatorium[i].direction;

cout << "\nКол-во путёвок: " << listOfSanatorium[i].amountPermit;

cout << endl;

}

}

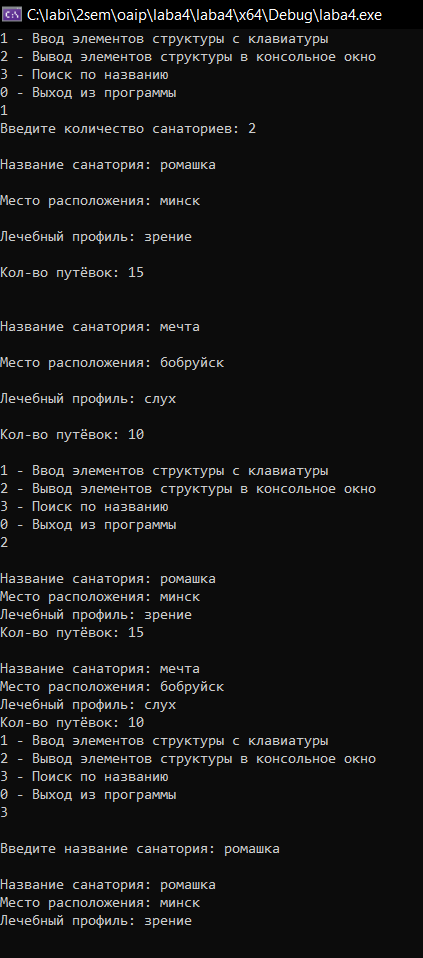
if (!foundSanatorium)

{

cout << "\nНет санаториев с указанным названием." << endl;

}

}

****