МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра информационно-аналитических систем безопасности

имени Л.С. Берштейна

**Отчет по лабораторной работе №5**

по дисциплине  
Основы алгоритмизации и программирования  
(наименование дисциплины)  
На тему  
Работа со структурами  
(наименование темы)

Вариант 8

Выполнил:  
студент гр. КТбо1-8

Кравцов В.Д.

«\_\_\_» 2017 г.

Проверил:

Профессор кафедры ИАСБ   
Беляков С.Л.

«\_\_\_» 2017 г.

Таганрог – 2017

*Оглавление*

[Вариант задания 2](#_Toc496296733)

[Алгоритм работы 3](#_Toc496296734)

[Примеры входных и выходных данных 4](#_Toc496296735)

[Код программы 5](#_Toc496296736)

[Заключение 5](#_Toc496296737)

# Вариант задания

Вариант №8

Задание. Написать программу, которая записывает с клавиатуры в файл структуру согласно выданному варианту задания. В качестве разделителя полей структуры использовать символ табуляции. В программе реализовать:

а) дополнение существующего массива структур новыми структурами;

б) поиск структуры с заданным значением выбранного элемента;

в) вывод на экран содержимого массива структур;

г) упорядочение массива структур по заданному полю (элементу), например

государство по численности.

Вариант 8. «Фильм»: название, режиссер (фамилия, имя, отчество), страна, год выпуска, стоимость (расходы на выпуск), кассовые сборы (доход).

# Описание работы алгоритма:

1. Программа получает на вход число (количество структур);
2. В цикле от 0 до полученного числа программа считывает все элементы каждой структуры массива структур.
3. После получения всех структур, программа выводит все элементы массива в виде списка (для того, чтобы можно было проверить правильность введенных данных)
4. Программа выводит список возможных действий (Поиск фильма, Вывод всех фильмов на экран, Сортировка фильмов, Выход из программы), каждому из которых присвоена команда (номер, число).
5. Программа получает на вход команду.
6. Программа проверяет введенное пользователем значение:

* Если пользователь выбрал функцию поиска фильмов, то программа запрашивает очередную команду, которая определяет то, по какому элементу должен выполниться поиск (По названию, По Имени\Фамилии\Отчеству продюсера, По дате выхода на экраны, По себестоимости, По кассовым сборам, По стране [где проводились съёмки]). Получив команду, программа запрашивает значение соответствующего элемента структуры. (Если пользователь ввёл неверную команду, то программа выводит соответствующее предупреждение). Получив команду, она выводит все найденные структуры с совпадающим элементом. Если по заданному значению не было найдено структур, то программа выводит сообщение “Нет результатов”
* Если пользователь выбрал функцию сортировки, то программа запрашивает очередную команду, которая определяет по какому элементу нужно отсортировать структуры (По себестоимости, По кассовым сборам, По дате выхода на экраны). (Если пользователь ввёл неверную команду, то программа выводит соответствующее предупреждение). Получив команду, программа сортирует структуры по заданному пользователем элементу.
* Если пользователь выбрал функцию вывода данных, то программа выводит данные всех структур в виде списка.
* Если пользователь выбрал функцию выхода, программа завершает работу.
* Если пользователь ввёл неверную команду, то программа выводит соответствующее предупреждение.

# Примеры входных и выходных данных

Тестирование программы с разными значениями на входе

|  |  |
| --- | --- |
| **Входные данные** | **Вывод программы** |
|  |  |
|  |  |
|  |  |

**Код программы**

#include "stdafx.h"

#include<string>

struct Date

{

int day;

int month;

int year;

};

struct Dir

{

char name[100];

char surname[100];

char dadname[100];

};

struct Mov

{

char name[100];

char country[100];

int price;

int proffit;

Date dt;

Dir director;

};

int find\_film\_name(Mov\*, int);

int find\_director\_surname(Mov\*, int);

int find\_director\_name(Mov\*, int);

int find\_director\_dadname(Mov\*, int);

int find\_film\_date(Mov\*, int);

int find\_film\_price(Mov\*, int);

int find\_film\_proffit(Mov\*, int);

int find\_film\_country(Mov\*, int);

int find\_film\_name(Mov\*, int);

void sort\_price(Mov\*, int);

void sort\_proffit(Mov\*, int);

void sort\_date(Mov\*, int);

void get\_films(Mov\*, int);

void print\_films(Mov\*, int);

int main(){

struct Mov Movie[100];

int n, command = 0, command\_to\_find = 0, command\_to\_find2 = 0, len = 0, k = 0, day = 0, v = 0;

int month = 0, year = 0, price = 0, proffit = 0, command\_to\_sort = 0;

char s[100] = "", object\_of\_finding[101], vs[100];

bool find = false;

printf\_s("Enter count of films: ");

scanf\_s("%d", &n);

get\_films(Movie, n);

print\_films(Movie, n);

while (command != 4) {

printf\_s("\nChoose command:\n1. To find a film type '1'\n2. To sort films type '2'\n3. To print all films type '3'\n4. To exit type '4'\nCommand: ");

scanf\_s("%d", &command);

switch (command) {

case 1: {

printf("\nCommand accepted.\nNow choose aim of searching:\nTo find in film names type '1'\nTo find in director names/surnames/dadnames type '2'\nTo find in date type'3'\nTo find in price type '4'\nTo find in proffit type '5'\nTo find in country type '6'\nCommand: ");

scanf\_s("%d", &command\_to\_find);

switch (command\_to\_find) {

case 1: {

if (find\_film\_name(Movie, n)) {

find = true;

}

break;

}

case 2: {

printf\_s("\nCommand accepted.\nTo find in directors surname type '1'\nTo find in directors name type '2'\nTo find in directors dadsname type '3'\nCommand: ");

scanf\_s("%d", &command\_to\_find2);

switch (command\_to\_find2) {

case 1: {

if (find\_director\_surname(Movie, n)) {

find = true;

}

break;

}

case 2: {

if (find\_director\_name(Movie, n)) {

find = true;

}

break;

}

case 3: {

if (find\_director\_dadname(Movie, n)) {

find = true;

}

break;

}

default: {printf\_s("\nWrong command!\n"); }

}

break;

}

case 3: {

if (find\_film\_date(Movie, n)) {

find = true;

}

break;

}

case 4: {

if (find\_film\_price(Movie, n)) {

find = true;

}

break;

}

case 5: {

if (find\_film\_proffit(Movie, n)) {

find = true;

}

break;

}

case 6: {

if (find\_film\_country(Movie, n)) {

find = true;

}

break;

}

default: {printf("Wrong command!"); }

}

if (find == false)

{

printf\_s("\n====================================\n");

printf("No results!");

printf\_s("\n====================================\n");

}

find = false;

break;

}

case 2: {

printf\_s("\nCommand accepted.\nTo sort in price type '1'\nTo sort in proffit type '2'\nTo sort in date type '3'\nCommand: ");

scanf\_s("%d", &command\_to\_sort);

switch (command\_to\_sort)

{

case 1: {sort\_price(Movie, n); break; }

case 2: {sort\_proffit(Movie, n); break; }

case 3: {sort\_date(Movie, n); break; }

}

break;

}

case 3: {

print\_films(Movie, n);

break;

}

case 4: {

printf\_s("\nCommand accepted. Goodbye!\n\n");

return 0;

break;

}

default: printf("\nWrong command!\n");

}

}

return 0;

}

int find\_film\_name(Mov \*Movie, int n)

{

char object\_of\_finding[100], s[100];

int len, k = 0;

bool find = false;

printf\_s("\nType a film name: ");

fgets(s, 100, stdin);

fgets(object\_of\_finding, 100, stdin);

len = strlen(object\_of\_finding);

object\_of\_finding[len - 1] = '\0';

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

{

for (int j = 0; j < strlen(object\_of\_finding); j++)

{

if (Movie[i].name[j] == object\_of\_finding[j])

{

k++;

}

}

if (k == strlen(object\_of\_finding))

{

printf\_s("\n====================================");

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf\_s("\n====================================\n");

find = true;

}

}

return find;

}

int find\_director\_surname(Mov \*Movie, int n)

{

char object\_of\_finding[100], s[100];

int len, k = 0;

bool find = false;

printf\_s("\nType a directors surname: ");

fgets(s, 100, stdin);

fgets(object\_of\_finding, 100, stdin);

len = strlen(object\_of\_finding);

object\_of\_finding[len - 1] = '\0';

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

{

for (int j = 0; j < strlen(object\_of\_finding); j++)

{

if (Movie[i].director.surname[j] == object\_of\_finding[j])

{

k++;

}

}

if (k == strlen(object\_of\_finding))

{

printf\_s("====================================");

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf\_s("\n====================================\n");

find = true;

}

}

return find;

}

int find\_director\_name(Mov \*Movie, int n)

{

char object\_of\_finding[100], s[100];

int len, k = 0;

bool find = false;

printf\_s("\nType a directors name: ");

fgets(s, 100, stdin);

fgets(object\_of\_finding, 100, stdin);

len = strlen(object\_of\_finding);

object\_of\_finding[len - 1] = '\0';

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

{

for (int j = 0; j < strlen(object\_of\_finding); j++)

{

if (Movie[i].director.name[j] == object\_of\_finding[j])

{

k++;

}

}

if (k == strlen(object\_of\_finding))

{

printf\_s("\n====================================");

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf\_s("\n====================================\n");

find = true;

}

}

return find;

}

int find\_director\_dadname(Mov \*Movie, int n)

{

char object\_of\_finding[100], s[100];

int len, k = 0;

bool find = false;

printf\_s("\nType a directors dadname: ");

fgets(s, 100, stdin);

fgets(object\_of\_finding, 100, stdin);

len = strlen(object\_of\_finding);

object\_of\_finding[len - 1] = '\0';

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

{

for (int j = 0; j < strlen(object\_of\_finding); j++)

{

if (Movie[i].director.dadname[j] == object\_of\_finding[j])

{

k++;

}

}

if (k == strlen(object\_of\_finding))

{

printf\_s("\n====================================");

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf\_s("\n====================================\n");

find = true;

}

}

return find;

}

int find\_film\_date(Mov \*Movie, int n)

{

int day, month, year;

bool find = false;

printf\_s("\nType date: ");

scanf\_s("%d %d %d", &day, &month, &year);

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

{

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

{

printf\_s("\n====================================");

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf\_s("\n====================================\n");

find = true;

}

}

return find;

}

int find\_film\_price(Mov \*Movie, int n)

{

int price;

bool find = false;

printf\_s("\nType price: ");

scanf\_s("%d", &price);

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

{

if (Movie[i].price == price)

{

printf\_s("\n====================================");

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf\_s("\n====================================\n");

find = true;

}

}

return find;

}

int find\_film\_proffit(Mov \* Movie, int n)

{

int proffit;

bool find = false;

printf\_s("\nType proffit: ");

scanf\_s("%d", &proffit);

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

{

if (Movie[i].proffit == proffit)

{

printf\_s("\n====================================");

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf\_s("\n====================================\n");

find = true;

}

}

return find;

}

int find\_film\_country(Mov \*Movie, int n)

{

char object\_of\_finding[100], s[100];

int len, k = 0;

bool find = false;

printf\_s("\nType a country: ");

fgets(s, 100, stdin);

fgets(object\_of\_finding, 100, stdin);

len = strlen(object\_of\_finding);

object\_of\_finding[len - 1] = '\0';

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

{

for (int j = 0; j < strlen(object\_of\_finding); j++)

{

if (Movie[i].country[j] == object\_of\_finding[j])

{

k++;

}

}

if (k == strlen(object\_of\_finding))

{

printf\_s("\n====================================");

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf\_s("\n====================================\n");

find = true;

}

}

return find;

}

void sort\_price(Mov \*Movie, int n)

{

char vs[100];

int v;

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

for (int j = 0; j < n - 1; j++)

{

if (Movie[j].price > Movie[j + 1].price)

{

v = Movie[j].price;

Movie[j].price = Movie[j + 1].price;

Movie[j + 1].price = v;

v = Movie[j].proffit;

Movie[j].proffit = Movie[j + 1].proffit;

Movie[j + 1].proffit = v;

v = Movie[j].dt.day;

Movie[j].dt.day = Movie[j + 1].dt.day;

Movie[j + 1].dt.day = v;

v = Movie[j].dt.month;

Movie[j].dt.month = Movie[j + 1].dt.month;

Movie[j + 1].dt.month = v;

v = Movie[j].dt.year;

Movie[j].dt.year = Movie[j + 1].dt.year;

Movie[j + 1].dt.year = v;

strcpy\_s(vs, Movie[j].name);

strcpy\_s(Movie[j].name, Movie[j + 1].name);

strcpy\_s(Movie[j + 1].name, vs);

strcpy\_s(vs, Movie[j].country);

strcpy\_s(Movie[j].country, Movie[j + 1].country);

strcpy\_s(Movie[j + 1].country, vs);

strcpy\_s(vs, Movie[j].director.name);

strcpy\_s(Movie[j].director.name, Movie[j + 1].director.name);

strcpy\_s(Movie[j + 1].director.name, vs);

strcpy\_s(vs, Movie[j].director.surname);

strcpy\_s(Movie[j].director.surname, Movie[j + 1].director.surname);

strcpy\_s(Movie[j + 1].director.surname, vs);

strcpy\_s(vs, Movie[j].director.dadname);

strcpy\_s(Movie[j].director.dadname, Movie[j + 1].director.dadname);

strcpy\_s(Movie[j + 1].director.dadname, vs);

}

}

}

void sort\_proffit(Mov \*Movie, int n)

{

int v;

char vs[100];

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

for (int j = 0; j < n - 1; j++)

{

if (Movie[j].proffit > Movie[j + 1].proffit)

{

v = Movie[j].price;

Movie[j].price = Movie[j + 1].price;

Movie[j + 1].price = v;

v = Movie[j].proffit;

Movie[j].proffit = Movie[j + 1].proffit;

Movie[j + 1].proffit = v;

v = Movie[j].dt.day;

Movie[j].dt.day = Movie[j + 1].dt.day;

Movie[j + 1].dt.day = v;

v = Movie[j].dt.month;

Movie[j].dt.month = Movie[j + 1].dt.month;

Movie[j + 1].dt.month = v;

v = Movie[j].dt.year;

Movie[j].dt.year = Movie[j + 1].dt.year;

Movie[j + 1].dt.year = v;

strcpy\_s(vs, Movie[j].name);

strcpy\_s(Movie[j].name, Movie[j + 1].name);

strcpy\_s(Movie[j + 1].name, vs);

strcpy\_s(vs, Movie[j].country);

strcpy\_s(Movie[j].country, Movie[j + 1].country);

strcpy\_s(Movie[j + 1].country, vs);

strcpy\_s(vs, Movie[j].director.name);

strcpy\_s(Movie[j].director.name, Movie[j + 1].director.name);

strcpy\_s(Movie[j + 1].director.name, vs);

strcpy\_s(vs, Movie[j].director.surname);

strcpy\_s(Movie[j].director.surname, Movie[j + 1].director.surname);

strcpy\_s(Movie[j + 1].director.surname, vs);

strcpy\_s(vs, Movie[j].director.dadname);

strcpy\_s(Movie[j].director.dadname, Movie[j + 1].director.dadname);

strcpy\_s(Movie[j + 1].director.dadname, vs);

}

}

}

void sort\_date(Mov \*Movie, int n)

{

int v;

char vs[100];

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

for (int j = 0; j < n - 1; j++)

{

if (Movie[j].dt.year > Movie[j + 1].dt.year)

{

v = Movie[j].price;

Movie[j].price = Movie[j + 1].price;

Movie[j + 1].price = v;

v = Movie[j].proffit;

Movie[j].proffit = Movie[j + 1].proffit;

Movie[j + 1].proffit = v;

v = Movie[j].dt.day;

Movie[j].dt.day = Movie[j + 1].dt.day;

Movie[j + 1].dt.day = v;

v = Movie[j].dt.month;

Movie[j].dt.month = Movie[j + 1].dt.month;

Movie[j + 1].dt.month = v;

v = Movie[j].dt.year;

Movie[j].dt.year = Movie[j + 1].dt.year;

Movie[j + 1].dt.year = v;

strcpy\_s(vs, Movie[j].name);

strcpy\_s(Movie[j].name, Movie[j + 1].name);

strcpy\_s(Movie[j + 1].name, vs);

strcpy\_s(vs, Movie[j].country);

strcpy\_s(Movie[j].country, Movie[j + 1].country);

strcpy\_s(Movie[j + 1].country, vs);

strcpy\_s(vs, Movie[j].director.name);

strcpy\_s(Movie[j].director.name, Movie[j + 1].director.name);

strcpy\_s(Movie[j + 1].director.name, vs);

strcpy\_s(vs, Movie[j].director.surname);

strcpy\_s(Movie[j].director.surname, Movie[j + 1].director.surname);

strcpy\_s(Movie[j + 1].director.surname, vs);

strcpy\_s(vs, Movie[j].director.dadname);

strcpy\_s(Movie[j].director.dadname, Movie[j + 1].director.dadname);

strcpy\_s(Movie[j + 1].director.dadname, vs);

}

}

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

for (int j = 0; j < n - 1; j++)

{

if (Movie[j].dt.month > Movie[j + 1].dt.month)

{

v = Movie[j].price;

Movie[j].price = Movie[j + 1].price;

Movie[j + 1].price = v;

v = Movie[j].proffit;

Movie[j].proffit = Movie[j + 1].proffit;

Movie[j + 1].proffit = v;

v = Movie[j].dt.day;

Movie[j].dt.day = Movie[j + 1].dt.day;

Movie[j + 1].dt.day = v;

v = Movie[j].dt.month;

Movie[j].dt.month = Movie[j + 1].dt.month;

Movie[j + 1].dt.month = v;

v = Movie[j].dt.year;

Movie[j].dt.year = Movie[j + 1].dt.year;

Movie[j + 1].dt.year = v;

strcpy\_s(vs, Movie[j].country);

strcpy\_s(Movie[j].country, Movie[j + 1].country);

strcpy\_s(Movie[j + 1].country, vs);

}

}

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

for (int j = 0; j < n - 1; j++)

{

if (Movie[j].dt.day > Movie[j + 1].dt.day)

{

v = Movie[j].price;

Movie[j].price = Movie[j + 1].price;

Movie[j + 1].price = v;

v = Movie[j].proffit;

Movie[j].proffit = Movie[j + 1].proffit;

Movie[j + 1].proffit = v;

v = Movie[j].dt.day;

Movie[j].dt.day = Movie[j + 1].dt.day;

Movie[j + 1].dt.day = v;

v = Movie[j].dt.month;

Movie[j].dt.month = Movie[j + 1].dt.month;

Movie[j + 1].dt.month = v;

v = Movie[j].dt.year;

Movie[j].dt.year = Movie[j + 1].dt.year;

Movie[j + 1].dt.year = v;

strcpy\_s(vs, Movie[j].country);

strcpy\_s(Movie[j].country, Movie[j + 1].country);

strcpy\_s(Movie[j + 1].country, vs);

}

}

}

void get\_films(Mov \*Movie, int n)

{

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

{

char s[100];

int len;

printf\_s("\n[ %d film ]\n", i + 1);

printf\_s("\nEnter film name: ");

fgets(s, 100, stdin);

fgets(Movie[i].name, 100, stdin);

len = strlen(Movie[i].name);

Movie[i].name[len - 1] = '\0';

printf("\nEnter country: ");

fgets(Movie[i].country, 100, stdin);

printf("\nEnter surname of director: ");

fgets(Movie[i].director.surname, 100, stdin);

printf("\nEnter name of director: ");

fgets(Movie[i].director.name, 100, stdin);

printf("\nEnter dadsname of director: ");

fgets(Movie[i].director.dadname, 100, stdin);

printf("\nEnter date: ");

scanf\_s("%d %d %d", &Movie[i].dt.day, &Movie[i].dt.month, &Movie[i].dt.year);

printf("\nEnter film price (in mln $): ");

scanf\_s("%d", &Movie[i].price);

printf("\nEnter film proffit (in mln $): ");

scanf\_s("%d", &Movie[i].proffit);

}

}

void print\_films(Mov\* Movie, int n)

{

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

{

printf("\n====================================");

printf\_s("\n[ %d film ]\n", i + 1);

printf\_s("\nFilm: %s\nDate: %d.%d.%d \nCountry: %sSurname: %sName: %sDadsname: %sFilm price (in mln $): %d \nFilm proffit (in mln $): %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].country, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].price, Movie[i].proffit);

printf("\n====================================\n");

}

}

**Заключение**

В результате выполнения лабораторной №5 были освоены методы работы со структурами, такие как создание элементов, запись конкретных из них.