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 3

Выполнил:  
студент гр. КТбо1-8

Князев Д.С.

«\_\_\_» 2017 г.

Проверил:

Профессор кафедры ИАСБ   
Беляков С.Л.

«\_\_\_» 2017 г.

Таганрог – 2017

*Оглавление*

[Вариант задания 2](#_Toc501650490)

[Описание работы алгоритма: 3](#_Toc501650491)

[Примеры входных и выходных данных 4](#_Toc501650492)

[Код программы 5](#_Toc501650493)

[Заключение 18](#_Toc501650494)

# Вариант задания

Вариант №3

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.

3. «Пациент»: фамилия, имя, отчество, год рождения, телефон, адрес, номер медицинской карты, группа крови, резус-фактор, диагноз.

# 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 струтур, 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**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define MaximumArray 100

#include <string>

#include <windows.h>

struct Date

{

int day;

int month;

int year;

};

struct Dir

{

char name[MaximumArray];

char surname[MaximumArray];

char dadname[MaximumArray];

};

struct Mov

{

char name[MaximumArray];

char address[MaximumArray];

int medСard;

int phone;

Date dt;

Dir director;

};

int find\_diagnosis(Mov\*, int);

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

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

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

int find\_year\_of\_birth(Mov\*, int);

int find\_med\_card(Mov\*, int);

int find\_phone(Mov\*, int);

int find\_address(Mov\*, int);

void sort\_med\_card(Mov\*, int);

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

void sort\_year\_of\_birth(Mov\*, int);

void new\_patient(Mov\*, int);

void conclusion\_patients(Mov\*, int);

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

struct Mov Movie[MaximumArray];

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

bool find = false;

printf\_s("Введите количество пациентов: ");

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

new\_patient(Movie, n);

conclusion\_patients(Movie, n);

while (command != 4) {

printf\_s("\nВыберите команду:\n1.Чтобы найти пациента '1'\n2.Для сортировки пациентов введите '2'\n3Чтобы напечатать всех пациентов '3'\n4.Для выхода нажмите '4'\nКоманда: ");

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

switch (command) {

case 1: {

printf("\nКоманда принята.\nТеперь выберите цель поиска:\nПоиск по диагнозу '1'\nПоиск по ФИО '2'\nПоиск по дате рождения '3'\nПоиск по номеру мед. карты '4'\nПоиск по номеру телефона '5'\nПоиск по адресу '6'\nКоманда: ");

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

switch (command\_to\_find) {

case 1: {

if (find\_diagnosis(Movie, n)) {

find = true;

}

break;

}

case 2: {

printf\_s("\nКоманда принята.\nНайти фамилию пациента '1'\nНайти имя пациента '2'\nНайти отчество пациента '3'\n Команда: ");

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

switch (command\_to\_find2) {

case 1: {

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

find = true;

}

break;

}

case 2: {

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

find = true;

}

break;

}

case 3: {

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

find = true;

}

break;

}

default: {printf\_s("\nНеправильная команда!\n"); }

}

break;

}

case 3: {

if (find\_year\_of\_birth(Movie, n)) {

find = true;

}

break;

}

case 4: {

if (find\_med\_card(Movie, n)) {

find = true;

}

break;

}

case 5: {

if (find\_phone(Movie, n)) {

find = true;

}

break;

}

case 6: {

if (find\_address(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("\nКоманда принята.\nСортировать по номеру мед. карты '1'\nСортировать по номеру телефона '2'\nСортировать по дате рождения '3'\n Команда: ");

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

switch (command\_to\_sort)

{

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

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

case 3: {sort\_year\_of\_birth(Movie, n); break; }

}

break;

}

case 3: {

conclusion\_patients(Movie, n);

break;

}

case 4: {

printf\_s("\nКоманда принята. Досвидание!\n\n");

return 0;

break;

}

default: printf("\nНеправильная команда!\n");

}

}

return 0;

}

int find\_diagnosis(Mov \*Movie, int n)

{

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

int len, k = 0;

bool find = false;

printf\_s("\nВведите диагноз пациента: ");

fgets(s, MaximumArray, stdin);

fgets(object\_of\_finding, MaximumArray, 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("\nДиагноз: %s\nДата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед. карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

find = true;

}

}

return find;

}

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

{

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

int len, k = 0;

bool find = false;

printf\_s("\nВведите фамилию пациента: ");

fgets(s, MaximumArray, stdin);

fgets(object\_of\_finding, MaximumArray, 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("\nДиагноз: %s\nДата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

find = true;

}

}

return find;

}

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

{

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

int len, k = 0;

bool find = false;

printf\_s("\nВведите имя пациента: ");

fgets(s, MaximumArray, stdin);

fgets(object\_of\_finding, MaximumArray, 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("\nДиагноз: %s\n Дата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед. карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

find = true;

}

}

return find;

}

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

{

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

int len, k = 0;

bool find = false;

printf\_s("\nВедиет отчество пациента: ");

fgets(s, MaximumArray, stdin);

fgets(object\_of\_finding, MaximumArray, 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("\nДиагноз: %s\nДата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед. карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

find = true;

}

}

return find;

}

int find\_year\_of\_birth(Mov \*Movie, int n)

{

int day, month, year;

bool find = false;

printf\_s("\nДата рождения: ");

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("\nДиагноз: %s\nДата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед. карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

find = true;

}

}

return find;

}

int find\_med\_card(Mov \*Movie, int n)

{

int med\_card;

bool find = false;

printf\_s("\nНомер мед. карты: ");

scanf\_s("%d", &med\_card);

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

{

if (Movie[i].medСard == med\_card)

{

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

printf\_s("\nДиагноз: %s\nДата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед. карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

find = true;

}

}

return find;

}

int find\_phone(Mov \* Movie, int n)

{

int phone;

bool find = false;

printf\_s("\nНомер телефона пациента: ");

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

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

{

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

{

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

printf\_s("\nДиагноз: %s\nДата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед. карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

find = true;

}

}

return find;

}

int find\_address(Mov \*Movie, int n)

{

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

int len, k = 0;

bool find = false;

printf\_s("\nВведите адрес пациента: ");

fgets(s, MaximumArray, stdin);

fgets(object\_of\_finding, MaximumArray, 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].address[j] == object\_of\_finding[j])

{

k++;

}

}

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

{

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

printf\_s("\nДиагноз: %s\nДата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед. карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

find = true;

}

}

return find;

}

void sort\_med\_card(Mov \*Movie, int n)

{

char vs[MaximumArray];

int v;

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

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

{

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

{

v = Movie[j].medСard;

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

Movie[j + 1].medСard = v;

v = Movie[j].phone;

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

Movie[j + 1].phone = 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].address);

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

strcpy\_s(Movie[j + 1].address, 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\_phone(Mov \*Movie, int n)

{

int v;

char vs[MaximumArray];

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

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

{

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

{

v = Movie[j].medСard;

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

Movie[j + 1].medСard = v;

v = Movie[j].phone;

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

Movie[j + 1].phone = 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].address);

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

strcpy\_s(Movie[j + 1].address, 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\_year\_of\_birth(Mov \*Movie, int n)

{

int v;

char vs[MaximumArray];

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].medСard;

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

Movie[j + 1].medСard = v;

v = Movie[j].phone;

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

Movie[j + 1].phone = 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].address);

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

strcpy\_s(Movie[j + 1].address, 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].medСard;

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

Movie[j + 1].medСard = v;

v = Movie[j].phone;

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

Movie[j + 1].phone = 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].address);

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

strcpy\_s(Movie[j + 1].address, 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].medСard;

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

Movie[j + 1].medСard = v;

v = Movie[j].phone;

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

Movie[j + 1].phone = 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].address);

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

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

}

}

}

void new\_patient(Mov \*Movie, int n)

{

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

{

char s[MaximumArray];

int len;

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

printf\_s("\nВведите диагноз: ");

fgets(s, MaximumArray, stdin);

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

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

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

printf("\nВведите адрес пациента: ");

fgets(Movie[i].address, MaximumArray, stdin);

printf("\nВведиет фамилию пациента: ");

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

printf("\nВведимет имя пациента: ");

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

printf("\nВведите отчество пациента: ");

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

printf("\nВведиет дату рождения пациента: ");

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

printf("\nВведите номер мед. карты пациента: ");

scanf\_s("%d", &Movie[i].medСard);

printf("\nВведиет номер телефона пациента: ");

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

}

}

void conclusion\_patients(Mov\* Movie, int n)

{

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

{

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

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

printf\_s("\nДиагноз: %s\nДата рождения: %d.%d.%d \nАдрес пациента: %sФамилия: %sИмя: %sОтчество: %sНомер мед. карты: %d \nНомер телефона: %d", Movie[i].name, Movie[i].dt.day, Movie[i].dt.month, Movie[i].dt.year, Movie[i].address, Movie[i].director.surname, Movie[i].director.name, Movie[i].director.dadname, Movie[i].medСard, Movie[i].phone);

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

}

}

**Заключение**

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