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**

**Лабораторная работа №5**

**Основы алгоритмизации и программирования**

**«Работа со структурами»**

**Вариант №8**

Выполнил:

студент гр. КТбо1-8

Лаврухин К. М.

«\_\_\_» 2017 г.

Проверил:

Профессор кафедры ИАСБ

Беляков Станислав Леонидович

«\_\_\_» 2017 г.

Таганрог – 2017

Оглавление

[Цели работы 3](#_Toc499755574)

[Алгоритм выполнения работы 4](#_Toc499755575)

[Создание алгоритма 4](#_Toc499755576)

[Программный код 5](#_Toc499755577)

[Тестирование программы 7](#_Toc499755578)

[Вывод 8](#_Toc499755579)

# Цели работы

Цель данной лабораторной работы ознакомить студента с понятием структуры данных, научить его правильно использовать ее и работать с ней.

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

# Алгоритм выполнения работы

Когда программа запускается, сразу выводит на экран меню выбора: 1) Внести новый фильм в файл, 2) Поиск нового фильма в файле 3) Вывод всех фильмов на экран 4) Сортировка фильмов в файле 5) Завершение работы программы.

Рассмотрим работу каждого пункта программы на основе двух введенных фильмов.

1) Выбор цифры 1 перебрасывает нас в функцию enterFilm, где пользователь циклично вводит структуру, которая затем записывается в файл.

2) Выбор цифры 2 перебрасывает нас в функцию searchFilm. В этой самой функции выводится меню на экран с выбором, по какому элементу искать фильм. После выбора, функция searchFilmByElement берет значение элемента с экрана и ищет в файле в каждой структуре фильм, в котором выбранный элемент совпадает с поисковым элементом. Если совпадение найдено – выводит данную структуру на экран.

3) Выбор цифры 3 перебрасывает нас в функцию outputFilms. Функция циклично выводит по 1 структуре из файла.

4) Выбор цифры 4 перебрасывает нас в функцию sortingFilms. Сначала функция записывает все структуры из файла в один массив структур. Затем, применяя пузырьковую сортировку, выводи отсортированный массив структур на экран(сортирует по возрастнию).

5) Выбор цифры 5 перебрасывает нас обратно в функцию main, где программа просто завершается.

# Программный код

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS |
|  | #define MAX\_LENGTH\_NAME\_FILM 256 |
|  | #define MAX\_LENGTH\_NAME\_PRODUCER 50 |
|  | #define MAX\_LENGTH\_NAME\_COUNTRY 58 |
|  |  |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | #include <conio.h> |
|  | #include <stdlib.h> |
|  |  |
|  | void Choice(); |
|  | int Checkup(int max); |
|  | void EnterFilm(); |
|  | void SearchFilm(); |
|  | void SortingFilms(); |
|  | void OutputFilms(); |
|  | void ChoiceFunction(int inquiry); |
|  | void FillFilm(struct Film \*Movie, struct Producer \*FIO, struct Value \*MONEY); |
|  | void FillFile(FILE \*filmsFile, struct Film Movie, struct Producer FIO, struct Value MONEY); |
|  | void SearchFilmElement(); |
|  | void SearchFilmByElement(int choose); |
|  | void FillStructForSearch(struct Film \*Movie, struct Producer \*FIO, struct Value \*MONEY, FILE \*filmsFile); |
|  | int CheckForCoincidence(struct Film Movie, struct Producer FIO, struct Value MONEY, int inquiry, char \*search); |
|  | void checkFile(FILE \*filmsFile); |
|  | void SortingFunction(struct Film Movie[100], struct Producer FIO[100], struct Value MONEY[100], int j); |
|  | void OutputOnDisplay(struct Film Movie[100], struct Producer FIO[100], struct Value MONEY[100], int i); |
|  |  |
|  | struct Film |
|  | { |
|  | char name[MAX\_LENGTH\_NAME\_FILM]; |
|  | char country[MAX\_LENGTH\_NAME\_COUNTRY]; |
|  | int dateOfRelease = 0; |
|  | }; |
|  |  |
|  | struct Producer |
|  | { |
|  | char producerName[MAX\_LENGTH\_NAME\_PRODUCER]; |
|  | char producerSurname[MAX\_LENGTH\_NAME\_PRODUCER]; |
|  | char producerLastName[MAX\_LENGTH\_NAME\_PRODUCER] = ""; |
|  | }; |
|  |  |
|  | struct Value |
|  | { |
|  | int cost = 0; |
|  | int income = 0; |
|  | }; |
|  |  |
|  | int main() |
|  | { |
|  | system("chcp 1251"); |
|  | system("cls"); |
|  | int inquiry = 0; |
|  | do |
|  | { |
|  | printf("\nЧто вы хотите сделать?\n"); |
|  | Choice(); |
|  | inquiry = Checkup(5); |
|  | ChoiceFunction(inquiry); |
|  | } while (inquiry != 5); |
|  | return 0; |
|  | } |
|  |  |
|  | void Choice() |
|  | { |
|  | printf("1)Внести новый фильм в файл\n"); |
|  | printf("2)Поиск фильма в файле\n"); |
|  | printf("3)Вывод всех фильмов на экран\n"); |
|  | printf("4)Сортировка фильмов в файле\n"); |
|  | printf("5)Завершение работы программы\n"); |
|  | printf("Ответ: "); |
|  | } |
|  |  |
|  | int Checkup(int max) |
|  | { |
|  | int number = 0; |
|  | while (!scanf("%d", &number) || number < 1 || number > max) |
|  | { |
|  | while (getchar() != '\n'); |
|  | printf("Ошибка ввода. Пожалуйста, повторите снова: "); |
|  | } |
|  | if (getchar() != '\n') |
|  | { |
|  | while (getchar() != '\n'); |
|  | printf("Ошибка ввода. Пожалуйста, повторите снова: "); |
|  | number = Checkup(max); |
|  | } |
|  | return number; |
|  | } |
|  |  |
|  | void ChoiceFunction(int inquiry) |
|  | { |
|  | switch (inquiry) |
|  | { |
|  | case 1: |
|  | EnterFilm(); |
|  | break; |
|  | case 2: |
|  | SearchFilm(); |
|  | break; |
|  | case 3: |
|  | OutputFilms(); |
|  | break; |
|  | case 4: |
|  | SortingFilms(); |
|  | break; |
|  | case 5: |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | void EnterFilm() |
|  | { |
|  | Film Movie; |
|  | Producer FIO; |
|  | Value MONEY; |
|  |  |
|  | FILE\* filmsFile = fopen("Films.txt", "a"); |
|  | if (filmsFile == NULL) |
|  | { |
|  | filmsFile = fopen("Films.txt", "w"); |
|  | } |
|  | bool check = true; |
|  | int choice; |
|  | do |
|  | { |
|  | FillFilm(&Movie, &FIO, &MONEY); |
|  | FillFile(filmsFile, Movie, FIO, MONEY); |
|  | printf("\nЗаписать еще один фильм? (1 - да, 2 - нет): "); |
|  | choice = Checkup(2); |
|  | printf("\n"); |
|  | if (choice == 1) |
|  | { |
|  | check = true; |
|  | } |
|  | else |
|  | { |
|  | check = false; |
|  | } |
|  | } while (check == true); |
|  | fclose(filmsFile); |
|  | } |
|  |  |
|  | void FillFilm(struct Film \*Movie, struct Producer \*FIO, struct Value \*MONEY) |
|  | { |
|  | printf("\nПожалуйста, введите название фильма: "); |
|  | fgets(Movie->name, MAX\_LENGTH\_NAME\_FILM, stdin); |
|  |  |
|  | printf("\nПожалуйста, введите имя продюссера: "); |
|  | fgets(FIO->producerName, MAX\_LENGTH\_NAME\_PRODUCER, stdin); |
|  |  |
|  | printf("\nПожалуйста, введите фамилию продюссера: "); |
|  | fgets(FIO->producerSurname, MAX\_LENGTH\_NAME\_PRODUCER, stdin); |
|  |  |
|  | printf("\nПожалуйста, введите отчество продюссера: "); |
|  | fgets(FIO->producerLastName, MAX\_LENGTH\_NAME\_PRODUCER, stdin); |
|  |  |
|  | printf("\nПожалуйста, введите страну, где был сделан фильм: "); |
|  | fgets(Movie->country, MAX\_LENGTH\_NAME\_COUNTRY, stdin); |
|  |  |
|  | printf("\nПожалуйста, введите год выпуска фильма: "); |
|  | Movie->dateOfRelease = Checkup(2020); |
|  |  |
|  | printf("\nПожалуйста, введите затраты на фильм: "); |
|  | MONEY->cost = Checkup(100000000); |
|  |  |
|  | printf("\nПожалуйста, введите доходы фильма: "); |
|  | MONEY->income = Checkup(100000000); |
|  | } |
|  |  |
|  | void FillFile(FILE \*filmsFile, struct Film Movie, struct Producer FIO, struct Value MONEY) |
|  | { |
|  | fprintf(filmsFile, " %s", Movie.name); |
|  | fprintf(filmsFile, "%s", FIO.producerName); |
|  | fprintf(filmsFile, "%s", FIO.producerSurname); |
|  | fprintf(filmsFile, "%s", FIO.producerLastName); |
|  | fprintf(filmsFile, "%s", Movie.country); |
|  | fprintf(filmsFile, "%d\n", Movie.dateOfRelease); |
|  | fprintf(filmsFile, "%d\n", MONEY.cost); |
|  | fprintf(filmsFile, "%d\0\n", MONEY.income); |
|  | } |
|  |  |
|  | void OutputFilms() |
|  | { |
|  | Film Movie; |
|  | Producer FIO; |
|  | Value MONEY; |
|  |  |
|  | FILE\* filmsFile = fopen("Films.txt", "r"); |
|  | checkFile(filmsFile); |
|  | int n = 1; |
|  | do |
|  | { |
|  | printf("\nФильм №%d", n); |
|  |  |
|  | fgets(Movie.name, MAX\_LENGTH\_NAME\_FILM, filmsFile); |
|  | printf("\nНазвание фильма: %s", Movie.name); |
|  |  |
|  | fgets(FIO.producerName, MAX\_LENGTH\_NAME\_PRODUCER, filmsFile); |
|  | printf("Имя продюссера: %s", FIO.producerName); |
|  |  |
|  | fgets(FIO.producerSurname, MAX\_LENGTH\_NAME\_PRODUCER, filmsFile); |
|  | printf("Фамилия продюссера: %s", FIO.producerSurname); |
|  |  |
|  | fgets(FIO.producerLastName, MAX\_LENGTH\_NAME\_PRODUCER, filmsFile); |
|  | printf("Отчество продюссера: %s", FIO.producerLastName); |
|  |  |
|  | fgets(Movie.country, MAX\_LENGTH\_NAME\_COUNTRY, filmsFile); |
|  | printf("Страна, где был сделан фильм: %s", Movie.country); |
|  |  |
|  | fscanf(filmsFile, "%d", &Movie.dateOfRelease); |
|  | printf("Год выпуска фильма: %d", Movie.dateOfRelease); |
|  |  |
|  | fscanf(filmsFile, "%d", &MONEY.cost); |
|  | printf("\nЗатраты на фильм: %d", MONEY.cost); |
|  |  |
|  | fscanf(filmsFile, "%d", &MONEY.income); |
|  | printf("\nДоходы фильма: %d\n\n", MONEY.income); |
|  |  |
|  | n = n + 1; |
|  | } while (fgetc(filmsFile) != EOF); |
|  | } |
|  |  |
|  | void checkFile(FILE \*filmsFile) |
|  | { |
|  | if (filmsFile == NULL) |
|  | { |
|  | printf("Файл пустой, запустите программу снова, если хотите его заполнить."); |
|  | \_getch(); |
|  | exit(0); |
|  | } |
|  | } |
|  |  |
|  | void SearchFilm() |
|  | { |
|  | SearchFilmElement(); |
|  | int chooseElement = Checkup(6); |
|  | SearchFilmByElement(chooseElement); |
|  | } |
|  |  |
|  | void SearchFilmElement() |
|  | { |
|  | printf("\nПо какому элементу вы хотите найти фильм? "); |
|  | printf("\n1) Название фильма "); |
|  | printf("\n2) Фамилия продюссера"); |
|  | printf("\n3) Страна, где был сделан фильм "); |
|  | printf("\n4) Год выпуска фильма"); |
|  | printf("\n5) Затраты фильма"); |
|  | printf("\n6) Доходы фильма"); |
|  | printf("\nОтвет: "); |
|  | } |
|  |  |
|  | void SearchFilmByElement(int choose) |
|  | { |
|  | Film Movie; |
|  | Producer FIO; |
|  | Value MONEY; |
|  | FILE\* filmsFile = fopen("Films.txt", "r"); |
|  | int n = 1; |
|  | bool check; |
|  | printf("Введите значение: "); |
|  | char search[MAX\_LENGTH\_NAME\_FILM]; |
|  | fgets(search, MAX\_LENGTH\_NAME\_FILM, stdin); |
|  | do |
|  | { |
|  | FillStructForSearch(&Movie, &FIO, &MONEY, filmsFile); |
|  | check = CheckForCoincidence(Movie, FIO, MONEY, choose, search); |
|  | if (check == true) |
|  | { |
|  | printf("\nФильм №%d", n); |
|  | printf("\nНазвание фильма: %s", Movie.name); |
|  | printf("Имя продюссера: %s", FIO.producerName); |
|  | printf("Фамилия продюссера: %s", FIO.producerSurname); |
|  | printf("Отчество продюссера: %s", FIO.producerLastName); |
|  | printf("Страна, где был сделан фильм: %s", Movie.country); |
|  | printf("Год выпуска фильма: %d", Movie.dateOfRelease); |
|  | printf("\nЗатраты на фильм: %d", MONEY.cost); |
|  | printf("\nДоходы фильма: %d\n\n", MONEY.income); |
|  | n = n + 1; |
|  | } |
|  | } while (fgetc(filmsFile) != EOF); |
|  | } |
|  |  |
|  | void FillStructForSearch(struct Film \*Movie, struct Producer \*FIO, struct Value \*MONEY, FILE \*filmsFile) |
|  | { |
|  | fgets(Movie->name, MAX\_LENGTH\_NAME\_FILM, filmsFile); |
|  | fgets(FIO->producerName, MAX\_LENGTH\_NAME\_PRODUCER, filmsFile); |
|  | fgets(FIO->producerSurname, MAX\_LENGTH\_NAME\_PRODUCER, filmsFile); |
|  | fgets(FIO->producerLastName, MAX\_LENGTH\_NAME\_PRODUCER, filmsFile); |
|  | fgets(Movie->country, MAX\_LENGTH\_NAME\_COUNTRY, filmsFile); |
|  | fscanf(filmsFile, "%d", &Movie->dateOfRelease); |
|  | fscanf(filmsFile, "%d", &MONEY->cost); |
|  | fscanf(filmsFile, "%d", &MONEY->income); |
|  | } |
|  |  |
|  | int CheckForCoincidence(struct Film Movie, struct Producer FIO, struct Value MONEY, int inquiry, char \*search) |
|  | { |
|  | bool check = false; |
|  | int checkint; |
|  | switch (inquiry) |
|  | { |
|  | case 1: |
|  | if (strcmp(Movie.name, search) == 0) |
|  | { |
|  | check = true; |
|  | } |
|  | case 2: |
|  | if (strcmp(FIO.producerSurname, search) == 0) |
|  | { |
|  | check = true; |
|  | } |
|  | case 3: |
|  | if (strcmp(Movie.country, search) == 0) |
|  | { |
|  | check = true; |
|  | } |
|  | case 4: |
|  | checkint = atoi(search); |
|  | if (Movie.dateOfRelease == checkint) |
|  | { |
|  | check = true; |
|  | } |
|  | case 5: |
|  | checkint = atoi(search); |
|  | if (MONEY.cost == checkint) |
|  | { |
|  | check = true; |
|  | } |
|  | case 6: |
|  | checkint = atoi(search); |
|  | if (MONEY.income == checkint) |
|  | { |
|  | check = true; |
|  | } |
|  | } |
|  | return check; |
|  | } |
|  |  |
|  | void SortingFilms() |
|  | { |
|  | Film Movie[100]; |
|  | Producer FIO[100]; |
|  | Value MONEY[100]; |
|  | FILE\* filmsFile = fopen("Films.txt", "r"); |
|  | int amountStructs = 0, n = 1; |
|  | do |
|  | { |
|  | FillStructForSearch(&Movie[amountStructs], &FIO[amountStructs], &MONEY[amountStructs], filmsFile); |
|  | amountStructs++; |
|  | } while (fgetc(filmsFile) != EOF); |
|  |  |
|  | for (int i = 0; i < amountStructs; i++) |
|  | for (int j = 0; j < amountStructs - 1; j++) |
|  | { |
|  | if (Movie[j].dateOfRelease > Movie[j + 1].dateOfRelease) |
|  | { |
|  | SortingFunction(Movie, FIO, MONEY, j); |
|  | } |
|  | } |
|  | for (int i = 0; i < amountStructs; i++) |
|  | { |
|  | printf("\nФильм №%d", n); |
|  | OutputOnDisplay(Movie, FIO, MONEY, i); |
|  | n++; |
|  | } |
|  | } |
|  |  |
|  | void SortingFunction(struct Film Movie[100], struct Producer FIO[100], struct Value MONEY[100], int j) |
|  | { |
|  | int reserv; |
|  | char reservChar[100]; |
|  |  |
|  | reserv = MONEY[j].cost; |
|  | MONEY[j].cost = MONEY[j + 1].cost; |
|  | MONEY[j + 1].cost = reserv; |
|  |  |
|  | reserv = MONEY[j].income; |
|  | MONEY[j].income = MONEY[j + 1].income; |
|  | MONEY[j + 1].income = reserv; |
|  |  |
|  | reserv = Movie[j].dateOfRelease; |
|  | Movie[j].dateOfRelease = Movie[j + 1].dateOfRelease; |
|  | Movie[j + 1].dateOfRelease = reserv; |
|  |  |
|  | strcpy(reservChar, Movie[j].name); |
|  | strcpy(Movie[j].name, Movie[j + 1].name); |
|  | strcpy(Movie[j + 1].name, reservChar); |
|  |  |
|  | strcpy(reservChar, Movie[j].country); |
|  | strcpy(Movie[j].country, Movie[j + 1].country); |
|  | strcpy(Movie[j + 1].country, reservChar); |
|  |  |
|  | strcpy(reservChar, FIO[j].producerName); |
|  | strcpy(FIO[j].producerName, FIO[j + 1].producerName); |
|  | strcpy(FIO[j + 1].producerName, reservChar); |
|  |  |
|  | strcpy(reservChar, FIO[j].producerSurname); |
|  | strcpy(FIO[j].producerSurname, FIO[j + 1].producerSurname); |
|  | strcpy(FIO[j + 1].producerSurname, reservChar); |
|  |  |
|  | strcpy(reservChar, FIO[j].producerLastName); |
|  | strcpy(FIO[j].producerLastName, FIO[j + 1].producerLastName); |
|  | strcpy(FIO[j + 1].producerLastName, reservChar); |
|  | } |
|  |  |
|  | void OutputOnDisplay(struct Film Movie[100], struct Producer FIO[100], struct Value MONEY[100], int i) |
|  | { |
|  | printf("\nНазвание фильма: %s", Movie[i].name); |
|  | printf("Имя продюссера: %s", FIO[i].producerName); |
|  | printf("Фамилия продюссера: %s", FIO[i].producerSurname); |
|  | printf("Отчество продюссера: %s", FIO[i].producerLastName); |
|  | printf("Страна, где был сделан фильм: %s", Movie[i].country); |
|  | printf("Год выпуска фильма: %d", Movie[i].dateOfRelease); |
|  | printf("\nЗатраты на фильм: %d", MONEY[i].cost); |
|  | printf("\nДоходы фильма: %d\n\n", MONEY[i].income); |
|  | } |

# Тестирование программы

Пример входных и выходных данных, представлены на рисунках 1 и 2

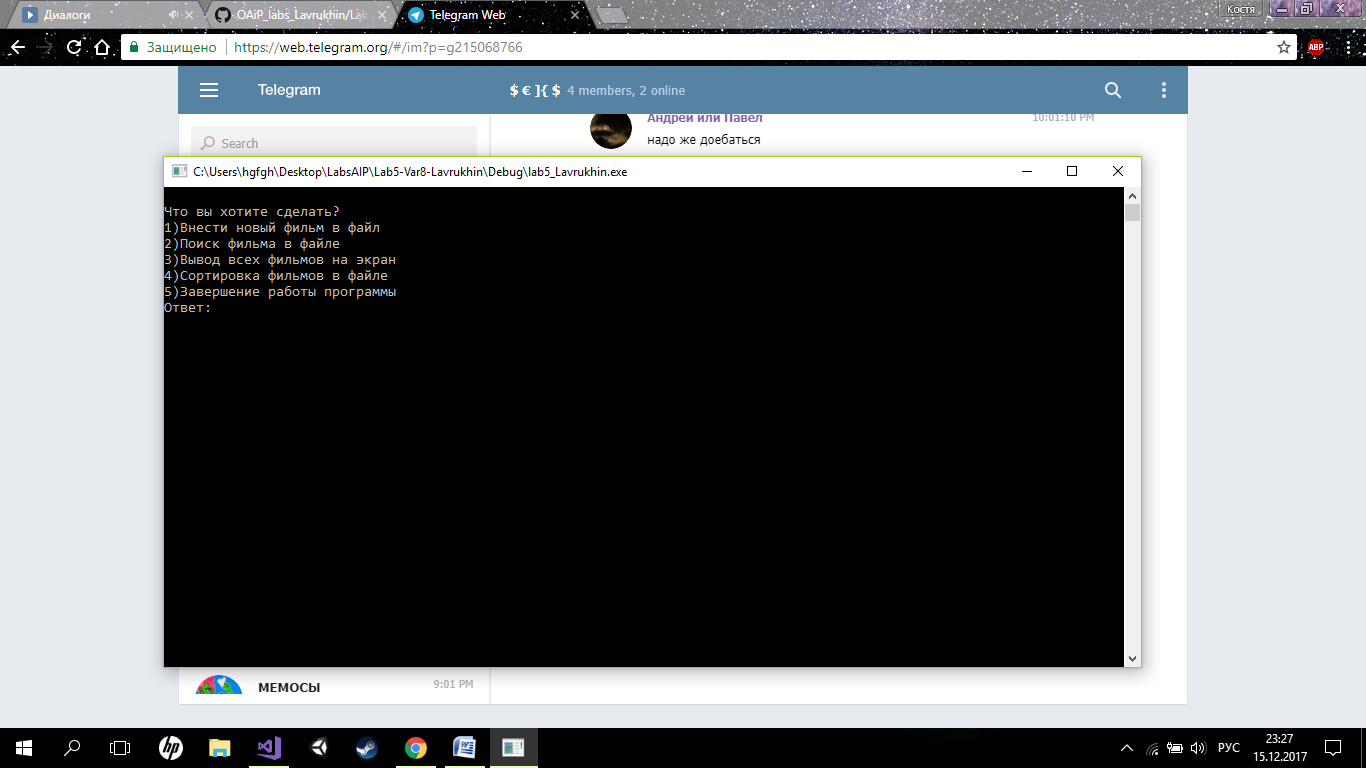


рис.1 Главное меню

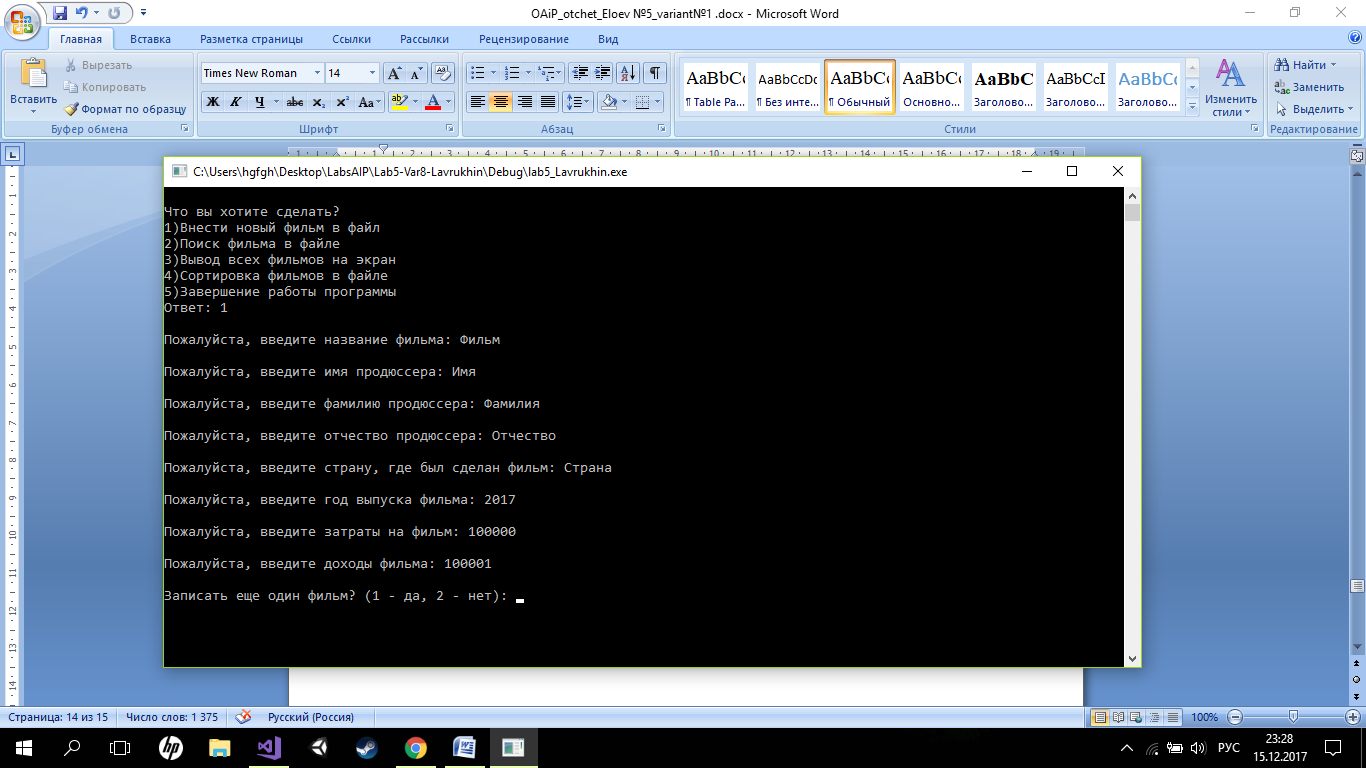


рис.2 Ввод нового фильма в файл

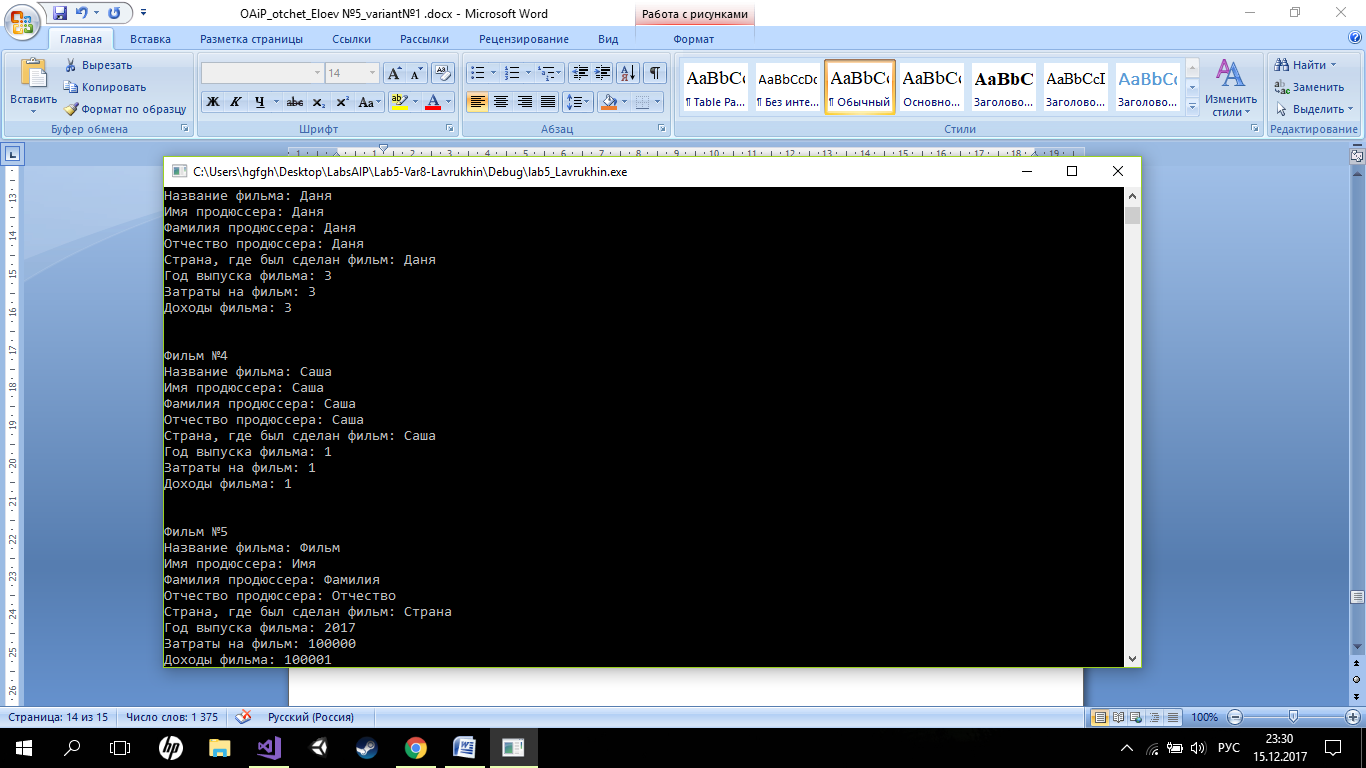
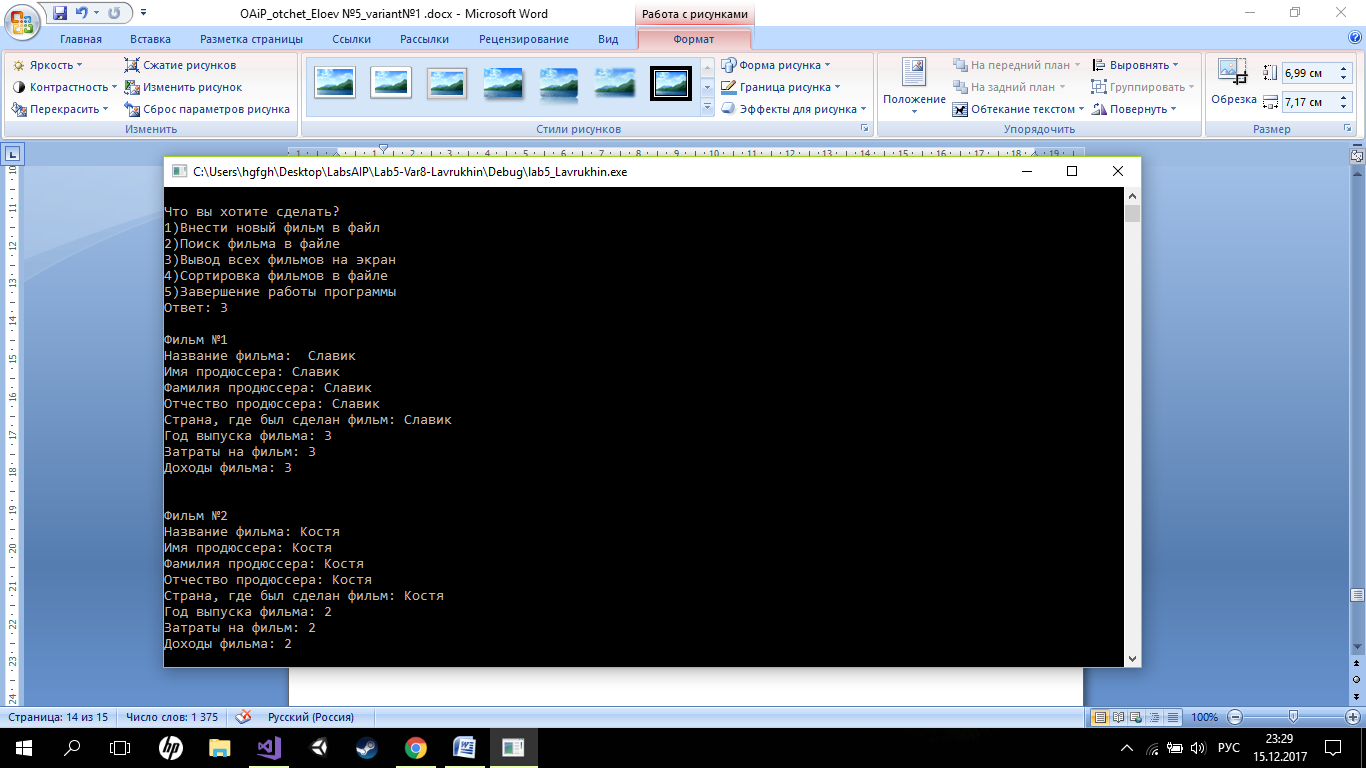


рис.3 Вывод всех фильмов на экран

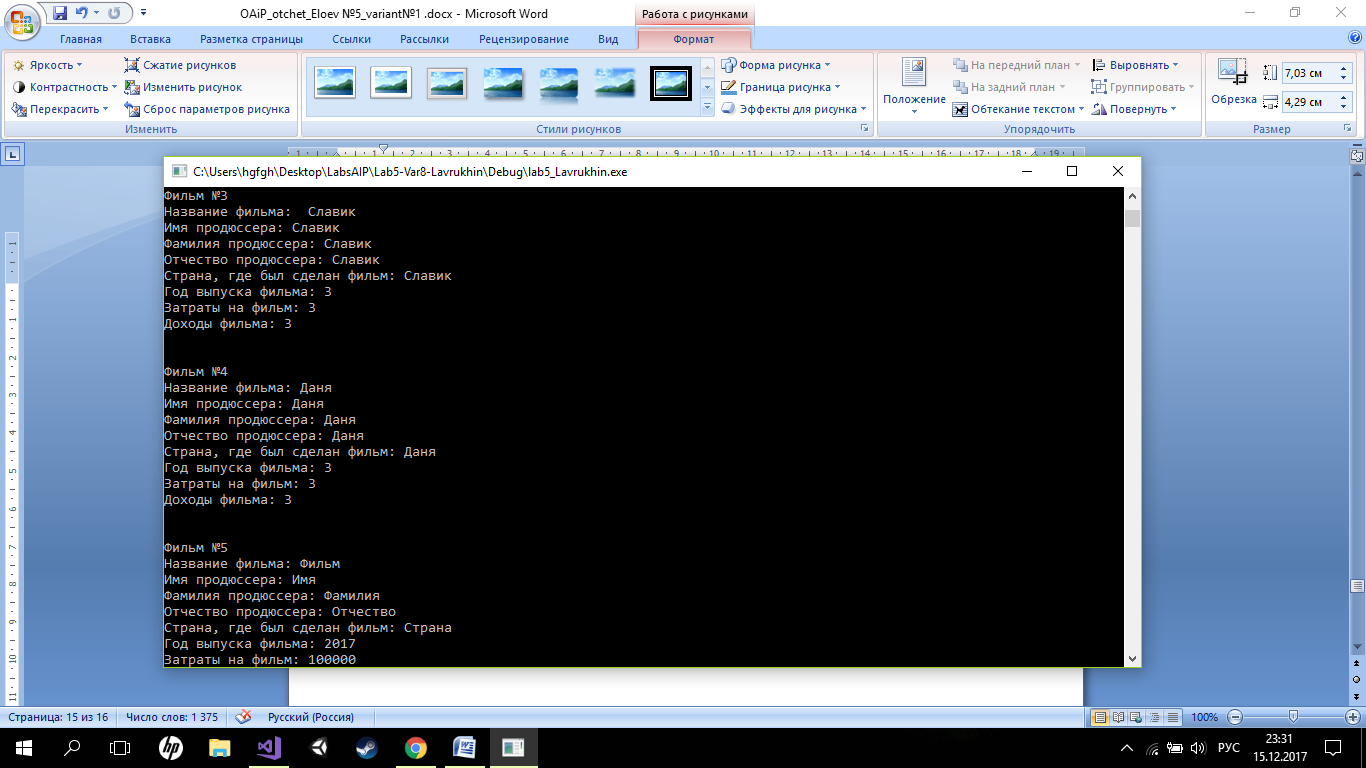
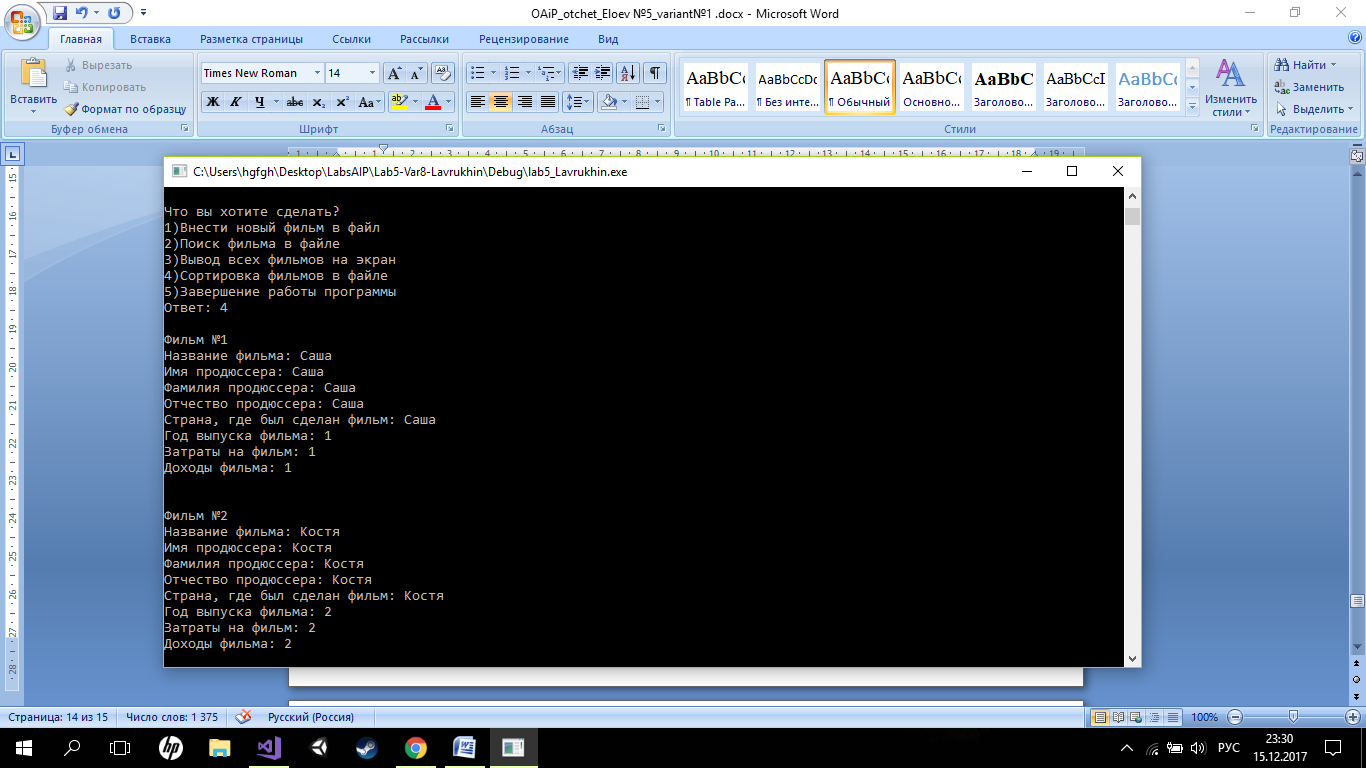


рис.4 Сортировка фильмов по дате по возрастанию

# Вывод

Выполняя данную лабораторной работы, мы ознакомились с понятием структуры, научились правильно использовать и работать с ней.