

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЕТ
по лабораторной работе № 10
по дисциплине «Программирование»
ТЕМА: Кольцевые списки

Студент гр. 3311

Баймухамедов Р.Р

Преподаватель

Хахаев И.А.

Санкт-Петербург

2024

Цель работы

Целью работы является изучение и освоение работы с кольцевыми списками.

Задание (вариант 15)

Для выбранной предметной области создать линейный список структур, содержащих характеристики объектов предметной области.

Обязательный набор полей:

- динамический массив символов, включая пробелы (name)
- кольцевой список одной характеристики предметной области
- числовые поля типов `int` и `float` (не менее двух полей каждого типа)
- поле с числовым массивом.

Написать программу, обеспечивающую начальное формирование линейного и кольцевого списков структур при чтении из файлов (текст с разделителями — CSV), (сначала формируется кольцевой список одной характеристики предметной области, потом в поле исходной структуры записывается адрес с нужным значением), а также разработать подалгоритм и написать функцию сортировки двусвязного кольцевого списка по значению информационного поля. Направление сортировки задается параметром функции.

Постановка задачи и описание решения

Для выполнения данной лабораторной работы необходимо разработать программу, которая будет читать данные из двух файлов и записывать их в линейные двусвязный и кольцевой список. Причём кольцевой список является характеристикой объекта предметной области. Предметная область будет фильмы: идентификационный номер фильма, название фильма, режиссёр (эта характеристика объекта будет являться списком), год выхода, длительность, рейтинг КиноПоиска, личный рейтинг и дата просмотра. В

качестве информационного поля для сортировки кольцевого двусвязного списка выбрано поле с именем режиссёра. Таким образом сортировка будет в алфавитном порядке и обратном алфавитном порядке.

Для достижения этой цели необходимо выполнить следующие шаги:

1. Запросить у пользователя символ-разделитель
2. Сформировать кольцевой двусвязный список режиссеров, прочтением данных строк файла, разделенных ранее введенным символом-разделителем, до тех пор, пока данные всех строк не будут занесены в список
3. Сформировать линейный двусвязный список, прочтением строк файла, разделенных ранее введенным символом-разделителем, до тех пор, пока данные всех строк не будут занесены в список, в поле режиссёра записывать адрес элемента с нужным значением из уже сформированного кольцевого двусвязного списка
4. Запросить у пользователя направление, в котором кольцевой двусвязный список будет отсортирован
5. Отсортировать кольцевой двусвязный список в направлении, выбранном пользователем

Описание функций

№	Функция	Тип	Назначение
1	main	int	Запускает программу, обрабатывает пользовательский ввод, вызывает другие функции в зависимости от выбора пользователя, и завершает программу при необходимости.
2	make_movie_head	MHD	Инициализация головы списка фильмов
3	make_director_head	DHD	Инициализация головы списка режиссёров
4	find_director	DIR	Поиск элемента в списке режиссёров по ID
5	create_movie	MOV	Создание элемента списка фильмов
6	create_director	DIR	Создание элемента списка режиссёров
7	add_movie	void	Привязка нового элемента к списку фильмов
8	add_director	void	Привязка нового элемента к списку режиссёров
9	menu	void	Выводит варианты выбора пользователя
10	print_head	void	Вывод оглавления списка
11	output_movie	void	Вывод списка фильма

12	output_director	void	Вывод списка режиссёров
13	split_string	void	Разделяет строку по символу-разделителю и записывает элементы строки в массив
14	add_movie_to_list	void	Чтение из файла и запись элементов строк в линейный двусвязный список фильмов
15	add_director_to_list	void	Чтение из файла и запись элементов строк в линейный двусвязный список режиссёров
16	swap_director	int	Меняет местами элементы кольцевого двусвязного списка режиссёров
17	sort_director_list	void	Сортировка кольцевого двусвязного списка режиссёров

Описание переменных

int main()

№	Переменная	Тип	Назначение
1	*mph	MHD (struct movie_head)	Голова списка фильмов, которая хранит в себе указатель на первый и последний элемент списка
2	*dph	DHD (struct director_head)	Голова списка режиссёров, которая хранит в себе указатель на первый и последний элемент списка
3	option	int	Опция, выбираемая пользователем
4	sep	char	Символ-разделитель

struct movie

№	Название	Тип	Назначение
1	id	int	Идентификационный номер фильма
2	*name	char	Название фильма
3	*director	DIR (struct director)	Имя и фамилия режиссёра фильма
4	year	int	Год выхода фильма
5	duration	int	Длительность фильма
6	kpr	float	Оценка фильма на КиноПоиске

7	plr	float	Личная оценка фильма
8	date[3]	int	Дата просмотра фильма
9	*prev	MOV (struct movie)	Указатель на предыдущий элемент списка
10	*next	MOV (struct movie)	Указатель на следующий элемент списка

struct movie_head

№	Название	Тип	Назначение
1	*first	MOV (struct movie)	Указатель на первый элемент списка фильмов
2	*last	MOV (struct movie)	Указатель на последний элемент списка фильмов

struct director

№	Название	Тип	Назначение
1	id	int	Идентификационный номер режиссёра
2	*name	char	Имя и фамилия режиссёра
3	*prev	DIR (struct director)	Указатель на предыдущий элемент списка
4	*next	DIR (struct director)	Указатель на следующий элемент списка

struct movie_head

№	Название	Тип	Назначение
1	*first	DIR (struct director)	Указатель на первый элемент списка режиссёров
2	*last	DIR (struct director)	Указатель на последний элемент списка режиссёров

MHD *make_movie_head()

№	Переменная	Тип	Назначение
1	*mph	MHD	Голова линейного двусвязного списка фильмов

DHD *make_director_head()

№	Переменная	Тип	Назначение
1	*dph	DHD	Голова линейного двусвязного списка режиссёров

DIR *find_director(DHD *dph, int id)

№	Переменная	Тип	Назначение
1	*current	DIR (struct director)	Указатель на текущий элемент списка режиссёров
2	flag	int	Результат на то, был ли найден элемент с заданным ID

MOV *create_movie(int id_mov, char *movie_name, int id, int movie_year, int movie_duration, float movie_kpr, float movie_plr, int watch_date[3], DHD *dph)

№	Переменная	Тип	Назначение
1	*new_movie	MOV	Новый элемент списка фильмов
2	*name	char	указатель на строку для хранения названия фильма
3	*movie_director	DIR	Элемент списка режиссёров

void output_movie (MHD *mph, int n)

№	Переменная	Тип	Назначение
1	*current	MOV	Указатель на текущий элемент

void output_director (DHD *dph, int n)

№	Переменная	Тип	Назначение
1	*current	DIR	Указатель на текущий элемент

void split_string(char *inputString, char **words, int *wordCount, char delimiter)

№	Переменная	Тип	Назначение
1	wordIndex	char	индекс текущего слова в массиве words
2	wordStart	char	индекс начала текущего слова в строке inputString
3	wordLength	char	длина текущего слова
4	inWord	int	флаг, указывающий на то, находится ли функция внутри слова или не внутри
5	i	int	переменная для итерации по символам в строке inputString.

void add_movie_to_list(char *filename, MHD *mph, char sep, DHD *dph)

№	Переменная	Тип	Назначение
1	*words[10]	char	Временный массив, куда записываются данные из строки файла
2	line[maxlen]	char	Хранение информации строки из файла
3	wordcount	int	Кол-во элементов, на которые строка была разделена
4	i	int	Переменная в цикле, отвечающая за номер элемента, на которые была разделена строка
5	date[3]	int	Массив для хранения даты просмотра фильма

6	*new_movie	MOV	Указатель на структуру MOV, представляющую новый фильм
---	------------	-----	--

void add_director_to_list(char *filename, DHD *dph, char sep)

№	Переменная	Тип	Назначение
1	*words[2]	char	Временный массив, куда записываются данные из строки файла
2	line[maxlen]	char	Хранение информации строки из файла
3	wordcount	int	Кол-во элементов, на которые строка была разделена
4	i	int	Переменная в цикле, отвечающая за номер элемента, на которые была разделена строка
5	date[3]	int	Массив для хранения даты просмотра фильма
6	*new_director	DIR	Указатель на структуру DIR, представляющую нового режиссёра

int swap_director (DHD *dph, DIR *current)

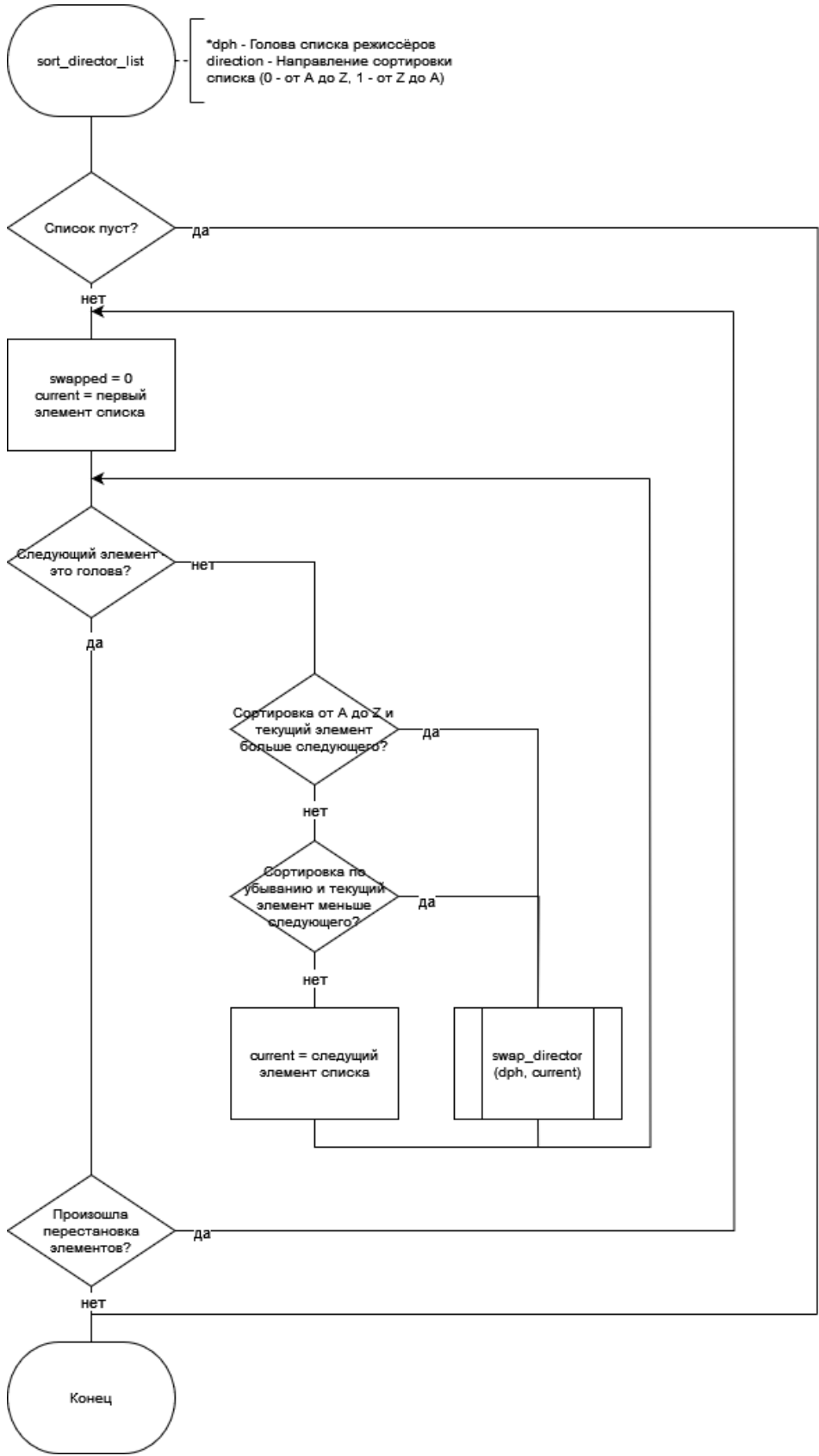
№	Переменная	Тип	Назначение
1	*temp	DIR	Временное хранение данных элемента кольцевого двусвязного списка режиссёров

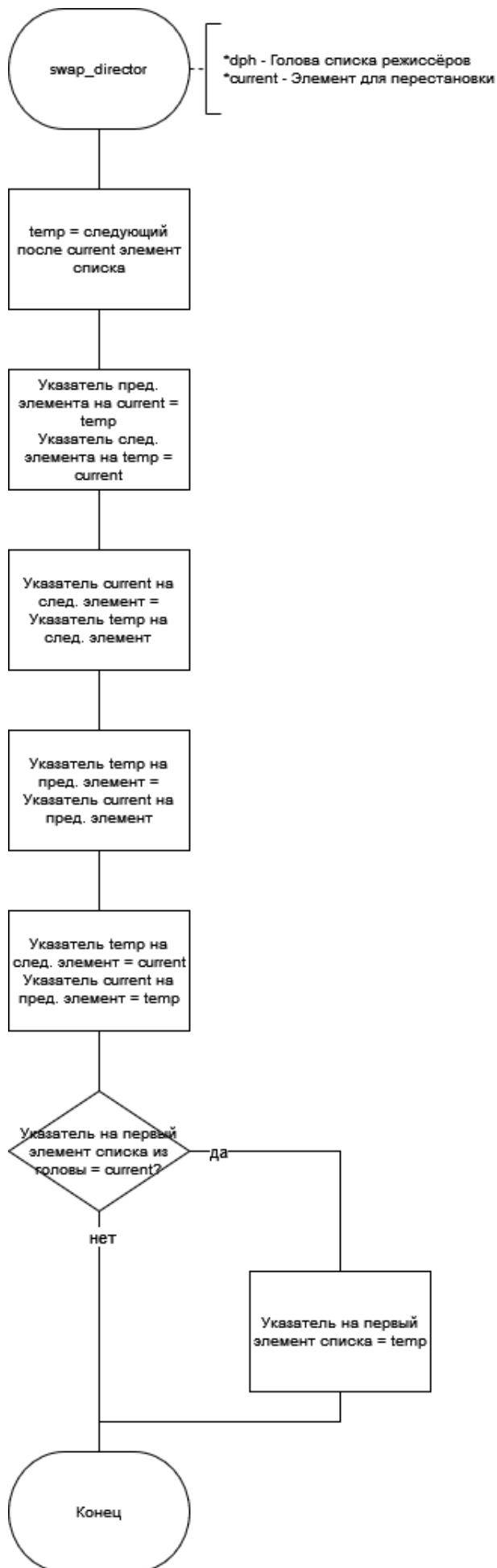
void sort_director_list (DHD *dph, int direction)

№	Переменная	Тип	Назначение
1	swapped	int	Факт, замены элементов списка

2	*current	DIR	Текущий элемент кольцевого двусвязного списка
---	----------	-----	---

Схемы алгоритмов





Пример 1:

Исходные данные:

```
Enter the separator of structure data: ;
Choose the option
0 - for EXIT program
1 - for SHOW THE MOVIE DATA
2 - for SHOW THE DIRECTOR DATA
3 - for SORT DIRECTOR LIST IN A-Z DIRECION
4 - for SORT DIRECTOR LIST IN Z-A DIRECTION
Enter the option: 2
```

Your selection is SHOW THE DIRECTOR DATA

ID	Director
0	Hayao Miyazaki
1	Francis Ford Coppola
2	Cristopher Nolan
3	Greta Gerwig
4	Denis Villeneuve
5	Boris Khlebnikov
6	Andrey Zvyagintsev
7	Zack Snyder
8	David Fincher

...

Enter the option: 3

Your selection is SORT DIRECTOR LIST IN A-Z DIRECION

The director list was sorted

Результат:

...

Enter the option: 2

Your selection is SHOW THE DIRECTOR DATA

ID	Director
6	Andrey Zvyagintsev
5	Boris Khlebnikov
2	Cristopher Nolan
8	David Fincher
4	Denis Villeneuve
1	Francis Ford Coppola
3	Greta Gerwig
0	Hayao Miyazaki
7	Zack Snyder

Пример 2:

Исходные данные:

Choose the option

```
0 - for EXIT program
1 - for SHOW THE MOVIE DATA
2 - for SHOW THE DIRECTOR DATA
3 - for SORT DIRECTOR LIST IN A-Z DIRECION
4 - for SORT DIRECTOR LIST IN Z-A DIRECTION
Enter the option: 2
```

Your selection is SHOW THE DIRECTOR DATA

ID	Director
0	Hayao Miyazaki
1	Francis Ford Coppola
2	Cristopher Nolan
3	Greta Gerwig
4	Denis Villeneuve
5	Boris Khlebnikov
6	Andrey Zvyagintsev
7	Zack Snyder
8	David Fincher

...

Enter the option: 4

Your selection is SORT DIRECTOR LIST IN Z-A DIRECION

The director list was sorted

Результат:

...

Enter the option: 2

Your selection is SHOW THE DIRECTOR DATA

ID	Director
7	Zack Snyder
0	Hayao Miyazaki
3	Greta Gerwig
1	Francis Ford Coppola
4	Denis Villeneuve
8	David Fincher
2	Cristopher Nolan
5	Boris Khlebnikov
6	Andrey Zvyagintsev

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define maxlen 128

/* Structure ----- */
```

```

struct movie; /* Define structure of Movie */

struct movie_head; /* Define structure of Movie List Head*/

struct director; /* Define structure of Director */

struct director_head; /* Define structure of Director List Head */

typedef struct movie MOV; /* Define Structured Type of Movie */

typedef struct movie_head MHD; /* Define Structured Type of Movie List Head*/

typedef struct director DIR; /* Define Structured Type of Director */

typedef struct director_head DHD; /* Define Structured Type of Director List Head*/

/* Node and List ----- */

MHD *make_movie_head(); /* Movie Head Initialization */

DHD *make_director_head(); /* Director Head Initialization */

DIR *find_director(DHD *dph, int id); /* Get pointer to Node in Director List by ID */

MOV *create_movie(int id_mov, char *movie_name, int id, int movie_year, int movie_duration, float movie_kpr, float
    movie_plr, int watch_date[3], DHD *dph); /* Movie Node Initialization */

DIR *create_director(int id, char *movie_director); /* Director Node Initialization */

void add_movie(MHD *ph, MOV *new_node, MOV *current_node); /* Add new Movie Node to List */

void add_director(DHD *dph, DIR *new_node); /* Add new Director Node to List */

/* Interface -----*/

void menu(); /* Output main menu */

void print_head(int n); /* Output title of sheet (0 - for MOVIE LIST, 1 - for DIRECTOR LIST) */

void output_movie(MHD *mph, int n); /* Output list with MOV structure (n for PRINT HEAD) */

void output_director(DHD *dph, int n); /* Output list with DIR structure (n for PRINT HEAD) */

void clear_screen(); /* Clear the console */

/* Sort, Form and other Stuff */

void split_string(char *inputString, char **words, int *wordCount, char delimiter); /* Split string by separator */

void add_movie_to_list(char *filename, MHD *mph, char sep, DHD *dph); /* Adding the movie data of file to list */

void add_director_to_list(char *filename, DHD *dph, char sep); /* Adding the director data of file to list */

```

```

int swap_director(DHD *dph, DIR *current); /* Swap elements of director list */

void sort_director_list(DHD* dph, int option); /* Sort director list */

/* Main Program */

int main() {
    char sep;
    int option;
    DHD *dph;
    MHD *mph;

    printf("Enter the separator of structure data: ");
    scanf("%c", &sep);
    dph=make_director_head();
    mph=make_movie_head();
    add_director_to_list("struct-data2-win.txt",dph,sep);
    add_movie_to_list("struct-data-win.txt",mph,sep,dph);

    do{
        menu();
        scanf("%i", &option);

        switch(option){
            case 0:{
                puts("\nYour selection is EXIT");
                getchar();
                break;
            }

            case 1:{
                puts("\nYour selection is SHOW THE MOVIE DATA\n");
                getchar();
                output_movie(mph,0);
                break;
            }

            case 2:{
                puts("\nYour selection is SHOW THE DIRECTOR DATA\n");
                getchar();
                output_director(dph,1);
                break;
            }

            case 3:{
                puts("\nYour selection is SORT DIRECTOR LIST IN A-Z DIRECTION\n");
                getchar();
                sort_director_list(dph, 0);
                break;
            }

            case 4:{
                puts("\nYour selection is SORT DIRECTOR LIST IN Z-A DIRECTION\n");

```

```

        getchar();
        sort_director_list(dph,1);
        break;
    }

    default:{
        puts("\nIncorrect key");
        getchar();
    }
}

puts("\nPress ENTER to continue");
getchar();
clear_screen();
} while (option!=0);

return 0;
}

/* Functions and their description ----- */

struct movie{
    int id; /* ID of movie */
    char *name; /* Name of movie */
    DIR *director; /* Director of movie */
    int year; /* Year of movie release */
    int duration; /* duration of movie in minutes*/
    float kpr; /* Movie rating on KinoPoisk */
    float plr; /* Movie rating on my opinion */
    int date[3]; /* Day/Month/Year of watch the movie */
    struct movie *prev; /* Link to previous node */
    struct movie *next; /* Link to next node */
};

struct movie_head{
    struct movie *first;
    struct movie *last;
};

struct director{
    int id; /* ID of the director */
    char *name; /* Director name */
    struct director *next; /* Link to previous node */
    struct director *prev; /* Link to next node */
};

struct director_head{
    struct director *first;
    struct director *last;
};

MHD *make_movie_head(){
    MHD *ph=NULL; /* Define and init Head */
    ph=(MHD*)malloc(sizeof(MHD));

```

```

    ph->first=NULL;
    ph->last=NULL;
    return ph;
}

DHD *make_director_head(){
    DHD *ph=NULL; /* Define and init Head */
    ph=(DHD*)malloc(sizeof(DHD));
    ph->first=NULL;
    ph->last=NULL;
    return ph;
}

DIR *find_director(DHD *dph, int id){
    DIR *current = NULL;
    int flag=0;
    current = dph->first;

    while(current!=NULL && flag==0){
        if(current->id==id){
            flag=1;
        } else current=current->next;
    }
    return current;
}

MOV *create_movie(int id_mov, char *movie_name, int id, int movie_year, int movie_duration, float movie_kpr, float
    movie_plr, int watch_date[3], DHD *dph){
    MOV *new_movie = NULL; /* Pointer to new node */
    DIR *movie_director = NULL; /* Pointer to director of movie */
    char *name = NULL;
    new_movie = (MOV*)malloc(sizeof(MOV));
    name = (char*)malloc((strlen(movie_name) + 1) * sizeof(char));
    movie_director = (DIR *)malloc(sizeof(DIR));

    movie_director=find_director(dph, id);

    if (new_movie && name) { /* Data is not empty */
        new_movie->id = id_mov;
        new_movie->name = name;
        new_movie->director = movie_director;
        new_movie->year = movie_year;
        new_movie->duration = movie_duration;
        new_movie->kpr = movie_kpr;
        new_movie->plr = movie_plr;
        memcpy(name, movie_name, strlen(movie_name) + 1);
        memcpy(new_movie->date, watch_date, sizeof(new_movie->date));
        new_movie->next = NULL;
        new_movie->prev = NULL;
    }

    return new_movie; /* Return adress of node */
}

```



```

DIR *create_director(int id, char *movie_director){
    DIR *new_director = NULL; /* Pointer to new node */
    char *director = NULL;

    new_director = (DIR*)malloc(sizeof(DIR));
    director = (char*)malloc((strlen(movie_director) + 1) * sizeof(char));

    if (new_director && director){ /* Data is not empty */
        new_director->id = id;
        new_director->name = director;
        memcpy(director, movie_director, strlen(movie_director) + 1);
        new_director->prev = NULL;
        new_director->next = NULL;
        director[strlen(movie_director)] = '\0';
    }

    return new_director; /* return adress of node */
}

```

```

void add_movie(MHD *ph, MOV *new_node, MOV *current_node){
    if (ph && new_node) {
        if (current_node == NULL) { /* Add first node of list */
            ph->first = new_node;
            ph->last = new_node;
            new_node->prev = NULL;
            new_node->next = NULL;
        } else {
            current_node->next = new_node;
            new_node->prev = current_node;
            new_node->next = NULL;
            ph->last = new_node;
        }
    }
}

```

```

void add_director(DHD *dph, DIR *new_node){
    if (dph && new_node) {
        if (dph->first == NULL) { /* Add first node of list */
            dph->first = new_node;
            new_node->prev = new_node;
            new_node->next = new_node;
        } else {
            new_node->prev = dph->first->prev;
            new_node->next = dph->first;
            dph->first->prev->next = new_node;
            dph->first->prev = new_node;
        }
    }
}

```

```

void menu(){
    puts("Choose the option");
    puts("0 - for EXIT program");
}

```

```

puts("1 - for SHOW THE MOVIE DATA");
puts("2 - for SHOW THE DIRECTOR DATA");
puts("3 - for SORT DIRECTOR LIST IN A-Z DIRECION");
puts("4 - for SORT DIRECTOR LIST IN Z-A DIRECTION");
printf("Enter the option: ");

}

void print_head(int n){
    if(n==0){
        printf("|      %2s      |      %25s      |      %25s      |      %4s      |      %3s      |      %5s      |      %5s      |      %10s\n", "ID", "Name", "Director", "Year", "Dur", "KPR", "PLR", "Watchdate");
        printf("+-----+-----+-----+-----+-----+-----+-----+-----+\n");
    } else {
        printf("| %2s | %25s |\n", "ID", "Director");
        printf("+-----+-----+\n");
    }
}

void output_movie(MHD *mph, int n) {
    MOV *current;
    current = mph->first;
    print_head(n);
    while (current!=NULL) {
        if(current->director!=NULL){
            printf("| %2d | %25s | %25s | %d | %3d | %5.1f | %5.1f | %.2d%.2d.%d |\n", current->id, current->name,
                current->director->name, current->year, current->duration, current->kpr, current->plr, current->date[0],
                current->date[1], current->date[2]);
            current = current->next;
        }
    }
}

void output_director(DHD *dph, int n) {
    DIR *current=dph->first;
    print_head(n);

    do {
        printf("| %2d | %25s |\n", current->id, current->name);
        current = current->next;
    } while (current != dph->first);
}

void clear_screen(){
#ifdef _WIN32 || defined(_WIN64)
    system("cls");
#else
    system("clear");
#endif
}

void split_string(char *inputString, char **words, int *wordCount, char delimiter) {
```

```

int wordIndex = 0, wordStart=0, wordLength=0, inWord=0, i;

for (i = 0; i <= strlen(inputString); i++) {
    if ((inputString[i] == delimiter || inputString[i] == '\0') && (inWord==1)) {
        words[wordIndex] = (char *)malloc(wordLength + 1);
        strncpy(words[wordIndex], inputString + wordStart, wordLength);
        words[wordIndex][wordLength] = '\0';
        wordIndex++;
        inWord = 0;
    } else {
        if (inWord==0) {
            wordStart = i;
            wordLength = 1;
            inWord = 1;
        } else {
            wordLength++;
        }
    }
}

*wordCount = wordIndex;
}

void add_movie_to_list(char *filename, MHD *mph, char sep, DHD *dph){
    char line[maxlen], *words[10];
    int wordCount, date[3], i;
    MOV *new_movie;
    FILE *file = fopen(filename, "r");
    if (file == NULL) printf("Error opening file.\n");

    while (fgets(line, sizeof(line), file)){
        line[strcspn(line, "\n")] = 0;
        split_string(line, words, &wordCount, sep);

        if (wordCount == 10){
            int id_mov = atoi(words[0]);
            char *name = words[1];
            int id = atoi(words[2]);
            int year = atoi(words[3]);
            int duration = atoi(words[4]);
            float kpr = atof(words[5]);
            float plr = atof(words[6]);
            date[0] = atoi(words[7]);
            date[1] = atoi(words[8]);
            date[2] = atoi(words[9]);

            new_movie = create_movie(id_mov, name, id, year, duration, kpr, plr, date, dph);
            add_movie(mph, new_movie, mph->last);
        }
        else printf("Invalid number of attributes in line: %s\n", line);

        for (i = 0; i < wordCount; i++){
            free(words[i]);
        }
    }
}

```

```

    }
    fclose(file);
}

void add_director_to_list(char *filename, DHD *dph, char sep){
    char line[maxlen], *words[2];
    int wordCount, i;
    DIR *new_director;
    FILE *file = fopen(filename, "r");
    if (file == NULL) printf("Error opening file.\n");

    while (fgets(line, sizeof(line), file)){
        line[strcspn(line, "\n")] = 0;
        split_string(line, words, &wordCount, sep);

        if (wordCount == 2){
            int id = atoi(words[0]);
            char *name = words[1];

            new_director = create_director(id, name);
            add_director(dph, new_director);
        }
        else printf("Invalid number of attributes in line: %s\n", line);

        for (i = 0; i < wordCount; i++){
            free(words[i]);
        }
    }
    fclose(file);
}

int swap_director(DHD *dph, DIR *current){
    DIR *temp=current->next;

    current->prev->next = temp;
    temp->next->prev = current;
    current->next = temp->next;
    temp->prev = current->prev;
    temp->next = current;
    current->prev = temp;

    if (dph->first == current) { /* If current was first element of head -> change head pointer to changed element */
        dph->first = temp;
    }
    return 1;
}

void sort_director_list(DHD *dph, int direction) {
    int swapped;
    DIR *current = NULL;

    if (dph == NULL){
        printf("The list is empty");
    }
}

```

```
} else {
    do {
        swapped = 0;
        current = dph->first;

        while (current->next != dph->first) {
            if (direction==0 && strcmp(current->name, current->next->name) > 0){
                swapped = swap_director(dph,current);
            } else if (direction==1 && strcmp(current->name, current->next->name) < 0) {
                swapped = swap_director(dph,current);
            } else {
                current = current->next;
            }
        }
    } while (swapped == 1);
}

printf("The director list was sorted\n");
}
```

Примеры выполнения программы

Исходный текстовые файлы и данные:

```
0;The Boy and Heron;0;2023;124;7.8;8.0;17;12;2024
1;Princess Mononoke;0;1997;134;8.3;8.5;11;2;2024
2;Oppenheimer;2;2023;181;8.2;8.0;18;9;2023
3;The Wind Rises;0;2013;126;7.8;7.0;22;06;2022
4;Barbie;3;2023;114;6.6;7.2; 4;11;2023
5;Loveless;6;2017;127;7.4;7.0;23;03;2020
6;Arrhythmia;5;2017;112;7.2;8.2;1;12;2023
7;Watchmen;7;2009;162;7.7;9.5;25;12;2019
8;The Godfather;1;1972;175;9.0;8.6;10;01;2021
9;Dune: Part 2;4;2024;166;8.8;8.5;10;03;2024
10;Fight Club;8;1999;139;8.7;10.0;12;11;2019
```

ID	Name	Director	Year	Dur	KPR	PLR	Watchdate
0	The Boy and Heron	Hayao Miyazaki	2023	124	7.8	8.0	17.12.2024
1	Princess Mononoke	Hayao Miyazaki	1997	134	8.3	8.5	11.02.2024
2	Oppenheimer	Cristopher Nolan	2023	181	8.2	8.0	18.09.2023
3	The Wind Rises	Hayao Miyazaki	2013	126	7.8	7.0	22.06.2022
4	Barbie	Greta Gerwig	2023	114	6.6	7.2	04.11.2023
5	Loveless	Andrey Zvyagintsev	2017	127	7.4	7.0	23.03.2020
6	Arrhythmia	Boris Khlebnikov	2017	112	7.2	8.2	01.12.2023
7	Watchmen	Zack Snyder	2009	162	7.7	9.5	25.12.2019
8	The Godfather	Francis Ford Coppola	1972	175	9.0	8.6	10.01.2021
9	Dune: Part 2	Denis Villeneuve	2024	166	8.8	8.5	10.03.2024
10	Fight Club	David Fincher	1999	139	8.7	10.0	12.11.2019

0;Hayao Miyazaki	ID	Director
1;Francis Ford Coppola	0	Hayao Miyazaki
2;Cristopher Nolan	1	Francis Ford Coppola
3;Greta Gerwig	2	Cristopher Nolan
4;Denis Villeneuve	3	Greta Gerwig
5;Boris Khlebnikov	4	Denis Villeneuve
6;Andrey Zvyagintsev	5	Boris Khlebnikov
7;Zack Snyder	6	Andrey Zvyagintsev
8;David Fincher	7	Zack Snyder
	8	David Fincher

Пример 1:

```
Choose the option
0 - for EXIT program
1 - for SHOW THE MOVIE DATA
2 - for SHOW THE DIRECTOR DATA
3 - for SORT DIRECTOR LIST IN A-Z DIRECION
4 - for SORT DIRECTOR LIST IN Z-A DIRECTION
Enter the option: 2

Your selection is SHOW THE DIRECTOR DATA

| ID | Director |
+---+-----+
| 0 | Hayao Miyazaki |
| 1 | Francis Ford Coppola |
| 2 | Cristopher Nolan |
| 3 | Greta Gerwig |
| 4 | Denis Villeneuve |
| 5 | Boris Khlebnikov |
| 6 | Andrey Zvyagintsev |
| 7 | Zack Snyder |
| 8 | David Fincher |

Press ENTER to continue
```

```
Choose the option
0 - for EXIT program
1 - for SHOW THE MOVIE DATA
2 - for SHOW THE DIRECTOR DATA
3 - for SORT DIRECTOR LIST IN A-Z DIRECION
4 - for SORT DIRECTOR LIST IN Z-A DIRECTION
Enter the option: 3

Your selection is SORT DIRECTOR LIST IN A-Z DIRECION

The director list was sorted

Press ENTER to continue
```

```
Choose the option
0 - for EXIT program
1 - for SHOW THE MOVIE DATA
2 - for SHOW THE DIRECTOR DATA
3 - for SORT DIRECTOR LIST IN A-Z DIRECION
4 - for SORT DIRECTOR LIST IN Z-A DIRECTION
Enter the option: 2

Your selection is SHOW THE DIRECTOR DATA

| ID | Director |
+---+-----+
| 6 | Andrey Zvyagintsev |
| 5 | Boris Khlebnikov |
| 2 | Cristopher Nolan |
| 8 | David Fincher |
| 4 | Denis Villeneuve |
| 1 | Francis Ford Coppola |
| 3 | Greta Gerwig |
| 0 | Hayao Miyazaki |
| 7 | Zack Snyder |

Press ENTER to continue
```

Пример 2:

```
Choose the option
0 - for EXIT program
1 - for SHOW THE MOVIE DATA
2 - for SHOW THE DIRECTOR DATA
3 - for SORT DIRECTOR LIST IN A-Z DIRECION
4 - for SORT DIRECTOR LIST IN Z-A DIRECTION
Enter the option: 2
```

Your selection is SHOW THE DIRECTOR DATA

ID	Director
0	Hayao Miyazaki
1	Francis Ford Coppola
2	Cristopher Nolan
3	Greta Gerwig
4	Denis Villeneuve
5	Boris Khlebnikov
6	Andrey Zvyagintsev
7	Zack Snyder
8	David Fincher

Press ENTER to continue

```
Choose the option
0 - for EXIT program
1 - for SHOW THE MOVIE DATA
2 - for SHOW THE DIRECTOR DATA
3 - for SORT DIRECTOR LIST IN A-Z DIRECION
4 - for SORT DIRECTOR LIST IN Z-A DIRECTION
Enter the option: 4
```

Your selection is SORT DIRECTOR LIST IN Z-A DIRECTION

The director list was sorted

Press ENTER to continue

```
Choose the option
0 - for EXIT program
1 - for SHOW THE MOVIE DATA
2 - for SHOW THE DIRECTOR DATA
3 - for SORT DIRECTOR LIST IN A-Z DIRECION
4 - for SORT DIRECTOR LIST IN Z-A DIRECTION
Enter the option: 2
```

Your selection is SHOW THE DIRECTOR DATA

ID	Director
7	Zack Snyder
0	Hayao Miyazaki
3	Greta Gerwig
1	Francis Ford Coppola
4	Denis Villeneuve
8	David Fincher
2	Cristopher Nolan
5	Boris Khlebnikov
6	Andrey Zvyagintsev

Press ENTER to continue

Выводы.

В результате выполнения работы изучена и освоена работа с кольцевыми двусвязными списками.