

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: структуры данных, линейные списки

Студент гр. 7381

Кортев Ю.В.

Преподаватель

Берленко Т.А.

Санкт-Петербург

2017

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Кортев Ю.В.

Группа 7381

Тема работы: структуры данных, линейные списки

Исходные данные: двунаправленный линейный список на основе списка из 4й лабораторной работы, набор функция для работы с этим списком.

Содержание пояснительной записки:

«Содержание», «Введение», «Заключение», «Примеры работы».

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент

Кортев Ю.В.

Преподаватель

Берленко Т.А.

АННОТАЦИЯ

В результате данной курсовой работы написана программа на языке СИ, составляющая из вводимых с клавиатуры данных список и выполняющая на выбор предложенные функции, такие как: добавление нового элемента в список, удаление элемента из списка, выведение на экран кол-во элементов списка и сортировка элементов, 1й половины списка по убыванию, а второй по возрастанию.

СОДЕРЖАНИЕ

Введение	5
1.Создание списка	6
1.1.Инициализация двумерного динамического массива	6
1.2.Описание функций для работы со структурами	6
2.Сортировка	8
2.1.Разделение списка на два	8
2.2.Сортировка списка по возрастанию	9
3.Вывод и освобождение памяти	10
3.1.Вывод на экран	10
3.2.Освобождение	10
Заключение	11

ВВЕДЕНИЕ

Цель работы.

На практике закрепить навыки работы с двунаправленными списками.

Задача.

Написать программу выполняющую следующие функции:

- 1) создание двунаправленного списка из полученных данных;
- 2) сортировка первой половины списка по убыванию года, второй по возрастанию;
- 3) освобождение всей выделенной за работу программы памяти.

Методы решения поставленных задач.

- 1) Для начала полученные данные сохраняются в двумерный динамический массив, кол-во строк такого массива равняется кол-ву элементов списка. Затем эти данные попадают в функцию, инициализирующую элементы списка и связывающую их.
- 2) Для сортировки списка написана функция `sort`, она принимает на вход элемент списка и кол-во элементов в списке, затем она находит первый элемент второй половины списка и разделяет список на 2 равных половины. Таким образом задача сводится к сортировке 2х списков меньших первого списка в 2 раза. После сортировки половины первоначального списка соединяются.
- 3) Каждый элемент списка содержит по 2 указателя на массивы, поэтому для полного освобождения памяти необходимо пройти каждый элемент списка. Затем освободить список и динамические массивы.

1. СОЗДАНИЕ СПИСКА

1.1. Инициализация двумерного динамического массива.

Написанная программа должна принять на вход названия фильмов, имена режиссеров и даты их выходов и составить из них двунаправленный список. Для названий фильмов и имен режиссеров выделяются динамические массивы с указателями на тип `char`, длина такого массива равна кол-ву фильмов. Для каждого указателя такого массива выделяется массив типа `char` (его длина не может превышать 80) и туда сохраняется полученные от пользователя названия фильмов и имена режиссеров. Для годов выхода достаточно выделить массив типа `int`, размер которого равен кол-ву фильмов.

1.2. Описание функций для работы со структурами.

Для начала необходимо описать шаблон структуры `Film`.

```
typedef struct Film
{
    int year;
    char* name;
    char* director;
    struct Film* next;
    struct Film* back;
}Film;
```

Он содержит 2 указателя, переменную типа `int` и 2 указателя необходимых для перемещения по списку.

Функция makeFilm из полученных данных инициализирует структуру.

```
Film* makeFilm(char* name, char* director, int year)
{
    Film* make=(Film*)malloc(sizeof(Film));
    make->name = name;
    make->director = director;
    make->year = year;
    make->next = NULL;
    make->back = NULL;
    return make;
}
```

Функция makeFilmList, используя функцию makeFilm, создает элементы списка и связывает их.

```
Film* makeFilmList(char** array_name, char** array_director, int* array_year, int n)
{
    Film* head = makeFilm(array_name[0], array_director[0], array_year[0]);
    Film* tmp;
    Film* prev=head;
    int i;
    for(i = 1; i < n; i++)
    {
        tmp = makeFilm(array_name[i], array_director[i], array_year[i]);
        tmp->back = prev;
        prev->next = tmp;
        prev = tmp;
    }
    return head;
}
```

2. СОРТИРОВКА

2.1. Разделение списка на два.

Сортировкой списка занимается функция **sort**, она принимает на вход указатель на первый элемент списка и кол-во элементов. Для начала инициализируется два указателя на структуру **half1** и **half2**, **half2** указывает на начало 2й половины списка, а **half1** на конец первой половины. Указатели между элементами **half1** и **half2** приравниваются

NULL, таким образом список делится на два, и задача сводится к сортировке двух списков по возрастанию и убыванию.

```
void sort(Film* head,int n)
{
    Film* half1;
    Film* half2;
    Film* tmp=head;
    int i=0;
    int j,c;
    for(i=0;i<n/2;i++)
    {
        tmp=tmp->next;
    }

    half1=tmp->back;//убывание
    half2=tmp;//возрастание
    half2->back->next=NULL;
    half2->back=NULL;
```


2.2. Сортировка по возрастанию

Сортировка списка аналогична сортировке массива, в данной программе сортировка производится методом “пузырька”.

Инициализируется 2 указателя на структуру **a** и **b**. **b** указывает на первый элемент списка, а **a** на второй. Функция сравнивает значения годов структур **a** и **b** и меняет элементы структур местами до тех пор, пока года не будут находиться в порядке возрастания.

```
Film* a;
Film* b;
int flag = 1;

if(!half2)
return;

while(flag)
{
    b=half2;
    a=half2->next;
    flag=0;

    while(a)
    {
        if((b->year)>(a->year))
        {
            if(b--head)
            {
                *head=a;
                b->next=a->next;
                a->next=a->back;
                a->back=NULL;
                a->back=a;
                b->next->back=b;
            }
            b->back->next=b->next;
            a->back=b->back;
            a->back=a;
            b->next=a->next;
            a->next=b;
            b->next->back=b;

            flag=1;
        }
        b=b->next;
        a=a->next;
    }
}
```

Сортировка 1й половины по убыванию происходит аналогично, за исключением того, что указатели **a** и **b** двигаются с конца к началу.

2.3. Соединение

После сортировки элементы **half1** и **half2** соединяются.

```
//соединение
half1->next=half2;
half2->back=half1;
```

3.Вывод и освобождение памяти.

3.1. Вывод

В теле цикла **for** указатель **tmp** проходит каждый элемент списка, и **printf** печатает его поля.

```
for(h=0;h<n;h++)
{
    printf("%s %s %d\n", tmp->name, tmp->director, tmp->year);
    tmp=tmp->next;
}
```

3.2. Освобождение

Указатель **tmp** проходит каждый элемент списка и освобождает предыдущий, после цикла освобождается последний элемент на, который указывает **tmp** и первый на который указывает **head**.

Затем, в цикле **for**, освобождается каждая строка массивов **name_array** и **director_array**, следом и сами массивы.

```

Film* tmp=head;
while(tmp->next!=NULL)
{
    tmp=tmp->next;
    tmp->back->next=NULL;
    tmp->back=NULL;
}
free(tmp);
free(head);
for(i=0;i<n;i++)
{
    free(name_array[i]);
    free(director_array[i]);
}
free(name_array);
free(director_array);
free(year_array);

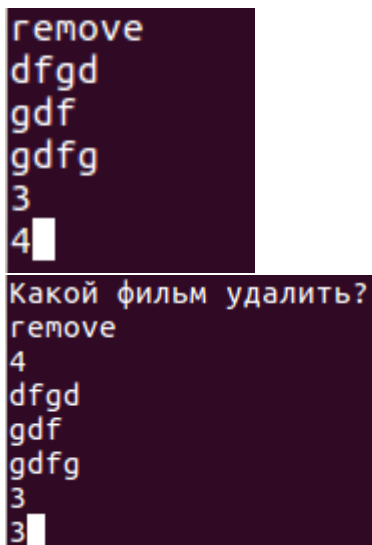
```

ЗАКЛЮЧЕНИЕ

В итоге курсовой работы написана функция для списка из 4й лабораторной работы, сортирующая первую половину элементов по убыванию года, а вторую по возрастанию.

ПРИМЕРЫ РАБОТЫ

Проверка удаления 1го элемента списка



```

remove
dfgd
gdf
gdfg
3
4

Какой фильм удалить?
remove
4
dfgd
gdf
gdfg
3
3

```

Сортировка списка

список до сортировки:

```
film2  
film1  
film4  
film3
```

список после сортировки:

```
film1  
film2  
film3  
film4
```