

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: «Линейные списки»

Студент гр. 7381

Лауцюс М.

Преподаватель

Берленко Т.А.

Санкт-Петербург

2017

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Лауцюс М.

Группа 7381

Тема работы: Линейные списки

Исходные данные:

Создать двунаправленный список музыкальных композиций и `api` (application programming interface - в данном случае набор функций) для работы со списком. Написать функцию, меняющую местами 1й и `n`й, 2й и `n-1` и т.д. элементы списка.

Содержание пояснительной записки:

«Содержание», «Введение», «Заключение», «Описание структуры `struct MusicalComposition`», «Функции программы», «Приложение».

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 23.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студент

Лауцюс М.

Преподаватель

Берленко Т.А.

АННОТАЦИЯ

В результате выполнения курсовой работы была создана программа на языке СИ, которая позволяет работать со списком и с набором функций для действий с ним. Для функционирования списка были созданы и описаны необходимые функции, позволяющие добавлять, удалять, и выводить поле элементов списка и их количество на консоль, а также описана структура элемента списка. Также была создана функция, меняющая местами 1й и пй, 2й и п-1 и т.д. элементы списка. Был создан интерфейс для пользования программой.

СОДЕРЖАНИЕ

Оглавление

Введение	5
1. Описание структуры struct MusicalComposition	6
2. Функции программы	7
2.1. Создание элемента списка	7
2.2. Создание двунаправленного списка связанных элементов	7
2.3. Добавление элемента в список	7
2.4. Удаление элемента из списка	8
2.5. Подсчет количества элементов в списке	8
2.6. Вывод названий композиций в консоль	8
2.7. Смена местами 1-ого и n-ого, 2-ого и n-1 и т.д. элементов списка	8
2.8. Интерфейс (функция main)	9
3. Примеры работы программы	10
Заключение	13
Список использованных источников	14
Приложение	15
1 Makefile	15
2 main.c	15
3. header.h	19
4. reverse.h	24

Введение

Целью данной работы является изучение понятия структуры в языке Си, изучение структуры данных, именуемой списком, создание программы с функциями для работы с двунаправленным списком и интерфейса для вызова этих функций.

Задание - создать двунаправленный список музыкальных композиций MusicalComposition и api (application programming interface - в данном случае набор функций) для работы со списком.

Структура элемента списка (тип - MusicalComposition)

- name - строка неизвестной длины (гарантируется, что длина не может быть больше 80 символов), название композиции.
- author - строка неизвестной длины (гарантируется, что длина не может быть больше 80 символов), автор композиции/музыкальная группа.
- year - целое число, год создания.

Функции, необходимые для работы программы

- Создание элемента типа MusicalComposition
- Создание двунаправленного списка связанных элементов типа MusicalComposition
- Добавление элемента в конец списка
- Удаление определенных элементов по имени из списка
- Подсчет количества элементов в списке
- Вывод поля элементов списка в консоль
- Функция, меняющая местами 1й и nй, 2й и n-1 и т.д. элементы списка.

1. Описание структуры struct MusicalComposition

struct MusicalComposition – структура, экземпляры которой являются элементами двусвязного списка. Содержит в себе поля музыкальной композиции и указатели на соседние элементы списка.

Поля struct MusicalComposition:

char* name - название музыкальной композиции

char* author - имя автора музыкальной композиции

int year - год публикации музыкальной композиции

struct MusicalComposition *next - указатель на следующую музыкальную композицию

struct MusicalComposition *previous - указатель на предыдущую музыкальную композицию

2. Функции программы

2.1. Создание элемента списка

```
MusicalComposition* createMusicalComposition(char* name, char* author,  
int year);
```

Функция принимает в качестве аргументов указатели на название композиции (**char*** name) и ее автора (**char*** author), а также год написания (**int** year). Происходит выделение памяти для структуры типа MusicalComposition и заполняются ее переменные.

Возвращаемое значение – указатель на созданный элемент.

2.2. Создание двунаправленного списка связанных элементов

```
MusicalComposition* createMusicalCompositionList(MusicalComposition*  
head, char** array_names, char** array_authors, int* array_years, int n);
```

Функция принимает в качестве аргументов указатели на указатель на массив названий композиций (char** array_names) и их авторов (char** array_authors), указатель на массив лет написания (int* array_years), размер массива (int n). Сначала создается голова списка и происходит заполнение ее переменных данными, затем аналогичные операции происходят для всех последующих n-1 элементов списка.

Возвращаемое значение - указатель на начало списка.

2.3. Добавление элемента в список

```
void push(MusicalComposition** head, MusicalComposition* element);
```

Функция принимает в качестве аргументов указатель на указатель на голову списка (MusicalComposition** head) и элемент (MusicalComposition* element), который нужно добавить в конец списка. В первую очередь

происходит поиск последнего элемента списка, далее новый элемент вставляется после текущего.

2.4. Удаление элемента из списка

```
void removeEl(MusicalComposition** head, char* name_for_remove);
```

Функция принимает в качестве аргументов указатель на указатель голову списка (`MusicalComposition** head`) и название композиций (`char* name_for_remove`), которые нужно удалить из списка. Для каждого его элемента происходит сравнение названия композиции с именем произведений, которые нужно удалить. Если проверка даёт положительный результат, элемент удаляется.

2.5. Подсчет количества элементов в списке

```
int count(MusicalComposition* head);
```

Функция принимает в качестве аргументов указатель на голову списка (`MusicalComposition* head`). Далее происходит подсчет элементов в списке, пока следующий элемент существует.

Возвращаемое значение - количество элементов списка.

2.6. Вывод названий композиций в консоль

```
void print_names(MusicalComposition* head);
```

Данная функция принимает указатель на первый элемент (указатель на голову) списка. Перебирая все элементы, функция печатает названия каждой композиции.

2.7. Смена местами 1-ого и n-ого, 2-ого и n-1 и т.д. элементов списка

```
void reverse(MusicalComposition** head);
```


Функция принимает в качестве аргументов указатель на указатель голову списка (MusicalComposition** head). Перебирая все элементы, функция меняет значение указателя элемента на предыдущий элемент на значение указателя элемента на следующий элемент и значение указателя элемента на следующий элемент на значение указателя элемента на предыдущий элемент. Головой списка становится последний элемент.

2.8. Интерфейс (функция main)

Запрашивается количество элементов списка.

Динамически выделяется память под массивы для хранения названий композиций, авторов и годов.

Запрашиваются данные массивов.

Создается двунаправленный список по данным из массивов.

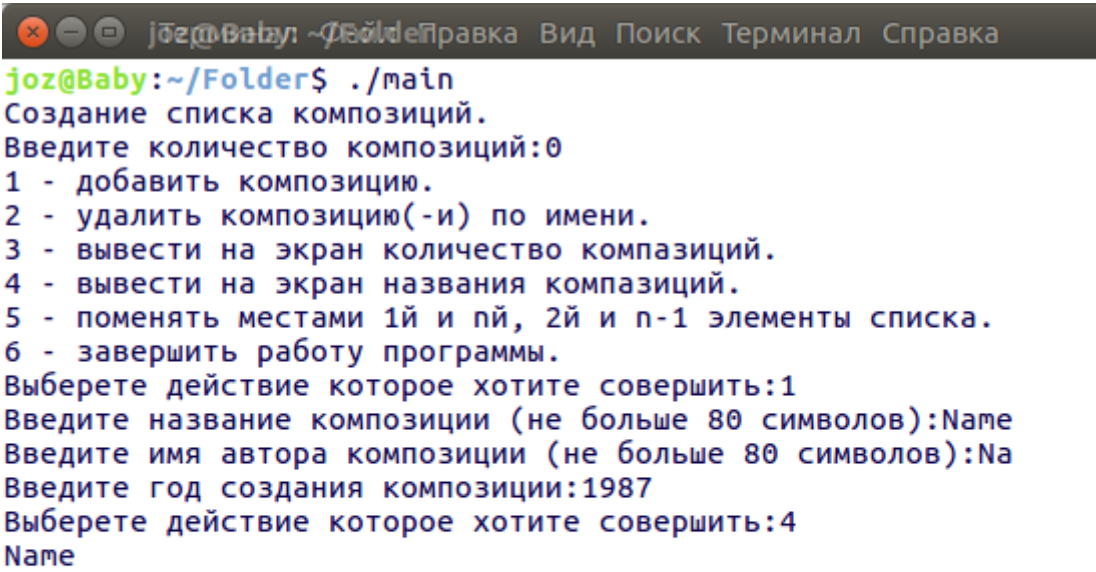
Пользователю предлагают многократный выбор ниже перечисленных действий

- 1 - Добавить элемент в конец списка.
- 2 - Удалить элемент с данным названием.
- 3 - Вывести количество элементов списка.
- 4 - Вывести названия композиций.
- 5 - Сменить местами 1-ого и n-ого, 2-ого и n-1 и т.д. элементов списка.
- 6 – Прекратить процесс.

Освобождается выделенная динамически память.

3. Примеры работы программы

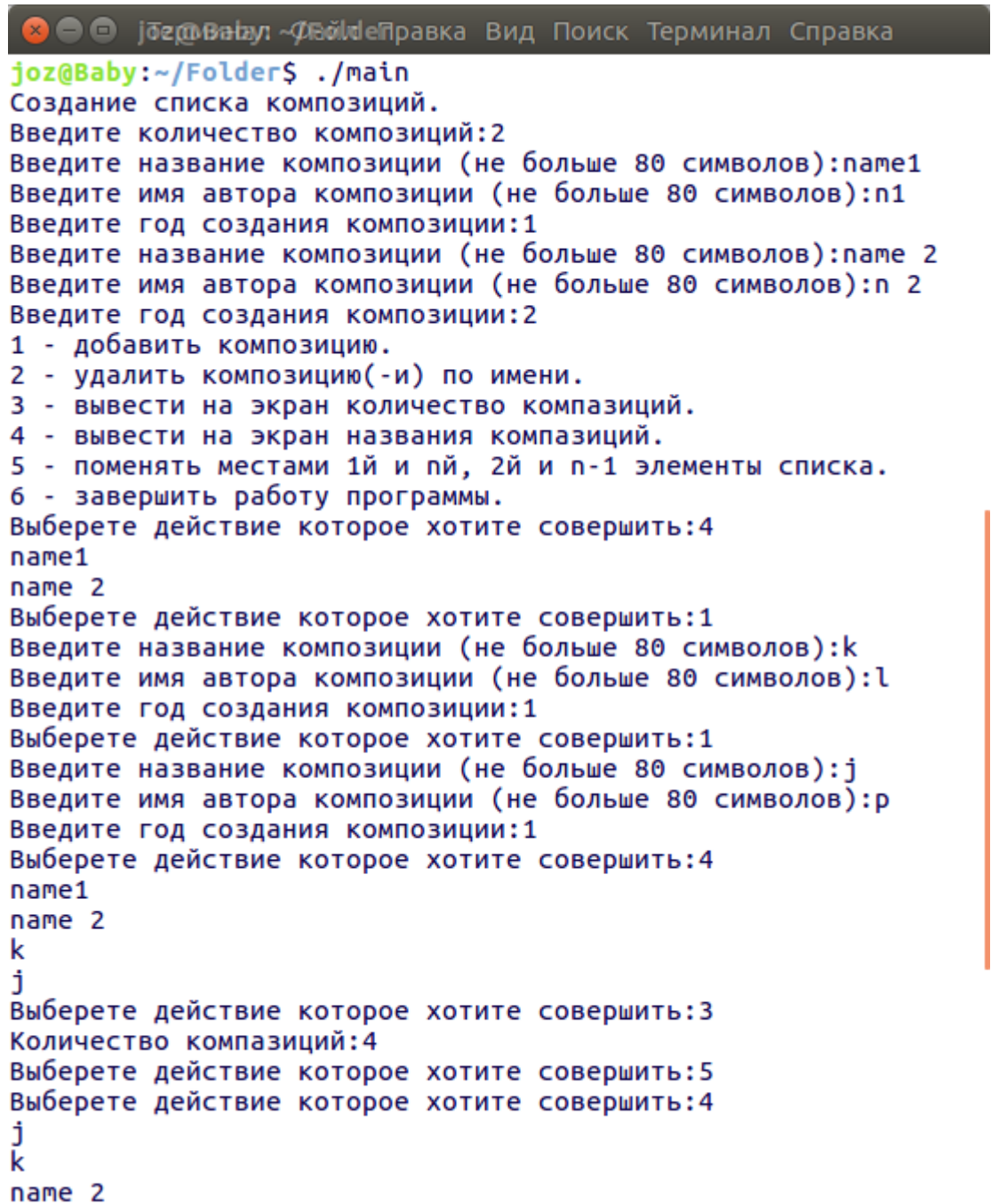
Тест 1



```
joz@Baby:~/Folder$ ./main
Создание списка композиций.
Введите количество композиций:0
1 - добавить композицию.
2 - удалить композицию(-и) по имени.
3 - вывести на экран количество композиций.
4 - вывести на экран названия композиций.
5 - поменять местами 1й и пй, 2й и п-1 элементы списка.
6 - завершить работу программы.
Выберете действие которое хотите совершить:1
Введите название композиции (не больше 80 символов):Name
Введите имя автора композиции (не больше 80 символов):Na
Введите год создания композиции:1987
Выберете действие которое хотите совершить:4
Name
Выберете действие которое хотите совершить:3
Количество композиций:1
Выберете действие которое хотите совершить:2
Введите название композиции:Name
Выберете действие которое хотите совершить:4
Выберете действие которое хотите совершить:3
Количество композиций:0
Выберете действие которое хотите совершить:6
joz@Baby:~/Folder$ clear
```

Рисунок 1

Тест 2



```
joz@Baby: ~/Folder
joz@Baby:~/Folder$ ./main
Создание списка композиций.
Введите количество композиций:2
Введите название композиции (не больше 80 символов):name1
Введите имя автора композиции (не больше 80 символов):n1
Введите год создания композиции:1
Введите название композиции (не больше 80 символов):name 2
Введите имя автора композиции (не больше 80 символов):n 2
Введите год создания композиции:2
1 - добавить композицию.
2 - удалить композицию(-и) по имени.
3 - вывести на экран количество композиций.
4 - вывести на экран названия композиций.
5 - поменять местами 1й и пй, 2й и n-1 элементы списка.
6 - завершить работу программы.
Выберете действие которое хотите совершить:4
name1
name 2
Выберете действие которое хотите совершить:1
Введите название композиции (не больше 80 символов):k
Введите имя автора композиции (не больше 80 символов):l
Введите год создания композиции:1
Выберете действие которое хотите совершить:1
Введите название композиции (не больше 80 символов):j
Введите имя автора композиции (не больше 80 символов):p
Введите год создания композиции:1
Выберете действие которое хотите совершить:4
name1
name 2
k
j
Выберете действие которое хотите совершить:3
Количество композиций:4
Выберете действие которое хотите совершить:5
Выберете действие которое хотите совершить:4
j
k
name 2
```

Рисунок 2

```
name1
Выберете действие которое хотите совершить:2
Введите название композиции:j
Выберете действие которое хотите совершить:4
k
name 2
name1
Выберете действие которое хотите совершить:2
Введите название композиции:name 2
Выберете действие которое хотите совершить:4
k
name1
Выберете действие которое хотите совершить:2
Введите название композиции:name1
Выберете действие которое хотите совершить:4
k
Выберете действие которое хотите совершить:3
Количество композиций:1
Выберете действие которое хотите совершить:6
joz@Baby:~/Folder$ |
```

Рисунок 3

Заключение

При выполнении курсовой работы, были освоены и закреплены на практике навыки написания программы с применением структур в качестве элементов двунаправленного линейного списка, а также API для работы с ними, в частности, функции вставки, удаления, подсчета, вывода названий элементов списка и смены местами 1-ого и n-ого, 2-ого и n-1 и т.д. элементов списка. Также были закреплены знания по выделению и очищению динамической памяти.

Список использованных источников

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство "Невский Диалект", 2001. 352 с.
2. Stepik Программирование (ЛЭТИ) 1 семестр/ URL: <http://east-front.narod.ru/memo/latchford.html>

Приложение

1 Makefile:

```
all: main.c header.h reverse.h
    gcc -o main main.c
```

2 main.c:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include "header.h"
#include "reverse.h"

int main(){
    int length;
    printf("Создание списка композиций.\nВведите количество
композиций:");
    scanf("%d", &length);
    char** names = (char**)malloc(sizeof(char*)*length);
    char** authors = (char**)malloc(sizeof(char*)*length);
    int* years = (int*)malloc(sizeof(int)*length);
    //char** years = (char**)malloc(sizeof(char*)*length);
    int i;
    for (i=0;i<length;i++)
    {
        char name[80];
        char author[80];
        //char year[10]
        printf("Введите название композиции (не больше 80
символов):");
        getchar();
```

```

        fgets(name, 80, stdin);
        printf("Введите имя автора композиции (не больше 80
СИМВОЛОВ):");
        getchar();
        fgets(author, 80, stdin);
        printf("Введите год создания композиции:");
        fscanf(stdin, "%d", &years[i]);
        //fgets(year, 10, stdin);

        (*strstr(name, "\n"))=0;
        (*strstr(author, "\n"))=0;
        //( *strstr(year, "\n"))=0;

        names[i] = (char*)malloc(sizeof(char*) * (strlen(name)+1));
        authors[i] = (char*)malloc(sizeof(char*) * (strlen(author)+1));
        //years[i] = (char*)malloc(sizeof(char*) * (strlen(year)+1));
        strncpy(names[i], name, 80);
        strncpy(authors[i], author, 80);
        //strcpy(years[i], year);

    }
    MusicalComposition* head = createMusicalCompositionList(names,
authors, years, length);

    for (int i=0;i<length;i++){
        free(names[i]);
        free(authors[i]);
    }

    if(names!=NULL)
        free(names);
    if(authors!=NULL)
        free(authors);
    if(years!=NULL)

```



```
free(years);
```

```
char name_for_remove[80];
int what_you_want=0;
printf("1 - добавить композицию.\n");
printf("2 - удалить композицию(-и) по имени.\n");
printf("3 - вывести на экран количество композиций.\n");
printf("4 - вывести на экран названия композиций.\n");
printf("5 - поменять местами 1й и nй, 2й и n-1 элементы списка.\n");
printf("6 - завершить работу программы.\n");
int k=0;
getchar();
char name_for_push[80];
char author_for_push[80];
int year_for_push;
while(what_you_want!=6)
{
    printf("Выберете действие которое хотите совершить:");
    scanf("%d",&what_you_want);
    if(!((int)what_you_want >= 1 && (int)what_you_want <= 6)){
        printf("wrong\n");
        getc(stdin);
        continue;
    }
    switch(what_you_want)
    {
    case 1:
        //author_for_push = (char*)malloc(sizeof(char*) * 80);
        //name_for_push = (char*)malloc(sizeof(char*) * 80);
        printf("Введите название композиции (не больше 80
СИМВОЛОВ):");
```

```

        getchar();
fgets(name_for_push, 80, stdin);
printf("Введите имя автора композиции (не больше 80
символов):");

        getchar();
fgets(author_for_push, 80, stdin);
printf("Введите год создания композиции:");
        fscanf(stdin, "%d", &year_for_push);
        (*strstr(name_for_push, "\n"))=0;
        (*strstr(author_for_push, "\n"))=0;
        MusicalComposition* element_for_push =
createMusicalComposition(name_for_push, author_for_push, year_for_push);
        push(&head, element_for_push);
break;
case 2:
        printf("Введите название композиции:");
        getchar();
fgets(name_for_remove, 80, stdin);
        (*strstr(name_for_remove, "\n"))='\0';
        removeEl(&head, name_for_remove);
break;
case 3:
k = count(head);
        printf("Количество композиций:%d\n", k);
        getchar();
break;
case 4:
        print_names(head);
        getchar();
break;
case 5:
        reverse(&head);
        getchar();
break;

```

```

        }
    }
    /*if(head!=NULL){
        while(head->next)
        {
            head=head->next;
            free(head->previous);
            head->previous=NULL;
        }
        if(head!=NULL)
            free(head);
        head=NULL;
    }*/
    delete_list(&head);
return 0;
}

```

3. header.h:

```

#pragma once
typedef struct MusicalComposition
{
    char* name;
    char* author;
    int year;
    struct MusicalComposition *next;
    struct MusicalComposition *previous;
}MusicalComposition;

```

```

MusicalComposition* createMusicalComposition(char* name, char*
author, int year)

```

```

{

    MusicalComposition* music =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    music->name=(char*)malloc(sizeof(char)*strlen(name));
    strncpy(music->name, name, 80);

    music->author=(char*)malloc(sizeof(char)*strlen(author));
    strncpy(music->author, author, 80);
    music->year = year;
    music->next = NULL;
    music->previous = NULL;
    return music;
}

```

```

MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n)
{
    MusicalComposition* head = NULL;
    if(n>0){
        head = createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
        MusicalComposition* current;
        MusicalComposition* previous_el=head;
        int i;
        for(i = 1; i < n; i++)
        {
            current = createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);

```

```

        current->previous = previous_el;
        previous_el->next = current;
        previous_el = current;
    }
}
return head;
}

```

```

void push(MusicalComposition** head, MusicalComposition* element)
{
    MusicalComposition* current = *head;
    if(*head==NULL)
        *head=element;
    else{
        while(current->next)
        {
            current=current->next;
        }
        current->next=element;
        element->previous=current;
    }
}

```

```

void removeEl(MusicalComposition** head, char* name_for_remove)
{
    MusicalComposition* current = *head;
    MusicalComposition* pointer=NULL;
    while(current)
    {
        if(strcmp(current->name, name_for_remove) == 0)

```

```

{ //όääëåíèå ýëåìåíòå ååç neõt è previous
if(current->next == NULL && current->previous == NULL){
    free(*head);
    *head=NULL;
    head=NULL;
    return;
}

//όääëåíèå ýëåìåíòå ñ neõt è previous
if(current->next != NULL && current->previous != NULL)
{
    pointer=current;
    current->previous->next = current->next;
    current->next->previous = current->previous;
    current=current->next;
    free(pointer);
    pointer=NULL;
    continue;
}

//όääëåíèå ýëåìåíòå ååç neõt
if(current->next == NULL)
{
    current->previous->next = NULL;
    free(current);
    current=NULL;
    return;
}

//όääëåíèå ýëåìåíòå ååç previous
else if(current->previous == NULL)
{
    pointer=current;

```

```

        current->next->previous = NULL;
        *head = current->next;
        current=*head;
        free(pointer);
        pointer=NULL;
        continue;
    }
}
current = current->next;

}

}

```

```

int count(MusicalComposition* head)
{
    MusicalComposition* current = head;
    int n = 0;
    while (current)
    {
        n++;
        current = current->next;
    }
    return n;
}

```

```

void print_names(MusicalComposition* head)
{
    MusicalComposition* current = head;
    while (current)

```

```

    {
        printf("%s\n", current->name);
        current = current->next;
    }
}

void delete_list(MusicalComposition** head)
{
    if(*head!=NULL){
        while((*head)->next)
        {
            (*head)=(*head)->next;
            free((*head)->previous->name);
            free((*head)->previous->author);
            free((*head)->previous);
            (*head)->previous=NULL;
        }
        if(*head!=NULL)
            free(*head);
        *head=NULL;
    }
}

```

4. reverse.h

```

#pragma once

void reverse(MusicalComposition** head)
{
    MusicalComposition* current = *head;
    MusicalComposition* pointer=NULL;
    while(current!=NULL)

```



```
{  
    *head=current;  
    pointer=current->next;  
    current->next=current->previous;  
    current->previous=pointer;  
    current=current->previous;  
}  
}
```