**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. 6304 |  | Зубов К.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Зубов Константин | | |
| Группа 6304 | | |
| Тема работы: Линейные списки | | |
| Содержание пояснительной записки:   * Содержание * Введение * Описание функций * Работа с репозиторием * Примеры работы программы * Заключение * Список использованных источников * Приложение: Исходный код программы | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 02.12.2016 | | |
| Дата сдачи реферата: 29.12.2016 | | |
| Дата защиты реферата: 29.12.2016 | | |
| Студент |  | Зубов К.А. |
| Преподаватель |  | Берленко Т.А. |

# **Аннотация**

В данной работе был создан проект на языке программирования C, который позволяет работать с набором функций, отвечающих за список музыкальных композиций. Для функционирования списка были созданы и описаны необходимые функции, позволяющие добавлять, удалять, сортировать и выводить элементы списка и их количество на консоль, а также описана структура элемента списка. Помимо этого, была проведена работа над оптимизацией исходного кода программы для ускорения ее быстродействия и оптимального использования памяти и ресурсов. Приведено полное описание исходного кода.

**Содержание**

[Аннотация 3](#_Toc470137644)

[Введение 5](#_Toc470137645)

[Цель работы 5](#_Toc470137646)

[Формулировка задачи 5](#_Toc470137647)

[Индивидуальное задание 6](#_Toc470137648)

[Решение задачи 6](#_Toc470137649)

[1. Создание структуры данных 6](#_Toc470137650)

[2. Функция создания нового экземпляра данного типа 6](#_Toc470137651)

[3. Функция создания списка 7](#_Toc470137652)

[4. Функция считывания значений списка 7](#_Toc470137653)

[5. Функция добавления элемента в конец списка 8](#_Toc470137654)

[6. Функция удаления элемента](#_Toc470137656) 8

[7. Функция подсчета элементов списка](#_Toc470137657) 9

[8. Функция вывода элементов списка](#_Toc470137658) 9

9. Функция перестановки частей списка местами………………………………………………………...…….....10

10.Функция разделения относительно n, разделение по m, перестановка………………………………….………11

[11. Makefile .13](#_Toc470137664)

12.Заключение…………………………………………………………………………………………………………...15

13.Источники……………………………………………………………………………………………………..……..15

14.Приложение…………………………………………………………………………………………………………..16

# **Введение**

## Цель работы

Практика применения сложных типов (struct) в языке С. Использование их для реализации сложных структур данных. В частности, одно- и дву-направленных линейных списков. Создание API для работы с ними. Закрепление знаний об указателях (в том числе на сложные типы), динамической памяти, массивах, стандартном вводе-выводе и основных функций библиотек “stdio.h”, “stdlib.h”, “string.h”.

**Формулировка задачи**

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (application***p****rogramming* ***i****nterface - 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.

Функция для создания элемента списка (тип элемента MusicalComposition)

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

Функции для работы со списком:

* MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n); // создает список музыкальных композиций MusicalCompositionList, в котором:
* ***n****- длина массивов* ***array\_names****,* ***array\_authors****,* ***array\_years****.*
* поле **name** первого элемента списка соответствует первому элементу списка array\_names (**array\_names[0]**).
* поле **author** первого элемента списка соответствует первому элементу списка array\_authors (**array\_authors[0]**).
* поле **year** первого элемента списка соответствует первому элементу списка array\_authors (**array\_years [0]**).
* void push (MusicalComposition\* head, MusicalComposition\* element); // добавляет **element** в конец списка **musical\_composition\_list**
* void removeEl (MusicalComposition\*  head, char\* name\_for\_remove); // удаляет элемент **element** списка, у которого значение **name** равно значению name**\_for\_remove**
* int count(MusicalComposition\*  head); //возвращает количество элементов списка
* void print\_names (MusicalComposition\* head); //Выводит названия композиций

## Индивидуальное задание

1.Разделить список на две части (индекс центрального элемента округлить в большую сторону), сделать копию первой части и вставить между разделенными частями, соединить все части.   
2.Разделить список относительно индекса m, вторую часть разделить по n элементов, поменять первый и последний элемент группы местами.

**Решение задачи**

## Создание структуры данных

С помощью типа *struct* объявим новый сложный тип *struct MusicalComposition*, экземплярами которого и будем заполнять список.

struct MusicalComposition{

char name[80];

char author[80];

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

};

typedef struct MusicalComposition MusicalComposition;

## Функция создания нового экземпляра данного типа

Выделим память для нового элемента и проинициализируем поля переданными в функцию параметрами с помощью указателя на элемент.

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

MusicalComposition\* s = (MusicalComposition\*)malloc(sizeof(MusicalComposition));

strcpy(s->name, name);

strcpy(s->author, author);

s->year = year;

s->next = NULL;

s->prev = NULL;

return s;

}

## Функция создания списка

Создадим сначала один элемент и проинициализируем его поля, а затем с помощью еще двух указателей в цикле создадим остальные элементы и свяжем их и вернем указатель на первый элемент в списке.

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n){

MusicalComposition\* head = createMusicalComposition( array\_names[0], array\_authors[0], array\_years[0]);

MusicalComposition\* ptr = head;

MusicalComposition\* point = NULL;

int i;

for(i = 1; i<n; i++){

point = createMusicalComposition( array\_names[i], array\_authors[i], array\_years[i]);

point->prev = ptr;

ptr->next = point;

ptr = point;

}

return head;

}

## Функция считывания значений списка

Первоначально выполняем цикл пока пользователь не введет подходящее значение количества элементов - целое неотрицательное число (length). Получив значение переменной length, начинаем последовательно считывать названия композиций (name), авторов композиций(author) и года (&years[i]) и записываем их в массивы (names,authors,years) соответственно. Передаем функции createMusicalCompositionList() данные массивы для создания линейного списка.

int main(){

int sw=1;

int key = 0;

int length;

int year;

char name[80];

char author[80];

printf("Let's start creating list.\n Enter the length of the list:\n");

scanf("%d", &length);

char\*\* names = (char\*\*)malloc(sizeof(char\*)\*length);

char\*\* authors = (char\*\*)malloc(sizeof(char\*)\*length);

int\* years = (int\*)malloc(sizeof(int)\*length);

int i;

for (i=0;i<length;i++)

{ printf("Enter the name of song, the author and the year of creation\n");

scanf("%s%s%d", name, author, &year);

names[i] = (char\*)malloc(sizeof(char\*) \* (strlen(name)+1));

authors[i] = (char\*)malloc(sizeof(char\*) \* (strlen(author)+1));

strcpy(names[i], name);

strcpy(authors[i], author);

}

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

for(i=0; i<length; i++){

free(names[i]);

free(authors[i]);

}

free(years);

## Функция добавления элемента в конец списка

Функция принимает в качестве аргумента указатель на новый элемент (head).

void push(MusicalComposition\* head, MusicalComposition\* element){

while(head->next)

head = head->next;

head->next = element;

element->prev = head;

element->next = NULL;

}

## 

## Функция удаления элемента

void removeEl(MusicalComposition\* head, char\* name\_for\_remove){

MusicalComposition \*ptr = head;

while(head){

if(strcmp(head->name, name\_for\_remove) == 0){

if((head->next == NULL) && (head->prev == NULL)){

free(head);

head = NULL;

return;

}

if(head->prev == 0){

head->next->prev = NULL;

return;

}

## 

## 7.Функция подсчета элементов списка

Чтобы подсчитать количество элементов, необходимо использовать счетчик и смещать с помощью цикла указатель до последнего элемента.

int count(MusicalComposition\* head){

int c = 0;

while(head){

c++;

head = head->next;

}

return c;

}

## 8.Функция вывода элементов списка

Функция вывода элементов списка аналогична предыдущей функции, только вместо счетчика выполняется функция вывода названия композиции, автора и года выпуска.

void print\_names(MusicalComposition\* head){

int i = 1;

while(head){

printf(" %d. %s %s %d\n", i++, head->name, head->author, head->year);

head = head->next;

}

}

**9.Функция разделения списка на 2 части.**

*void pushmid(MusicalComposition \* head, MusicalComposition\* element) {*

*MusicalComposition \* ptr = head;*

*int schet = 0;*

*while ( ptr->next != NULL) {*

*++schet;*

*ptr = ptr->next;*

*}*

*int j;*

*for ( j=0; j<((schet/2)+1); ++j) {*

*ptr = ptr->prev;*

*}*

*MusicalComposition\* second = ptr->next ;*

*MusicalComposition\* pp= createMusicalComposition(element->name, element->author, element->year);*

*ptr->next = pp;*

pp->next=second;

}

void function(MusicalComposition\* head){

int len = count(head);

MusicalComposition \*current = head;

int i;

if(len%2==0){

i = len/2;

}

else{

i = len/2 +1;

}

int j;

for(j = 0; j<i; j++){

MusicalComposition\* part = (MusicalComposition\*)malloc(sizeof(MusicalComposition));

strcpy(part->name, current->name);

strcpy(part->author, current->author);

part->year = current->year;

pushmid(head, part);

current = current->next;

}

}

## 10.Функция перестановки элементов частей списка местами

## void exchange(MusicalComposition\* a, MusicalComposition\* b){

## char t[80];

## int r;

## strcpy(t, a->name);

## strcpy(a->name, b->name);

## strcpy (b->name, t);

## strcpy(t, a->author);

## strcpy(a->author, b->author);

## strcpy (b->author, t);

## r=a->year;

## a->year=b->year;

## b->year=r;}

## //разбить список на группы и поменять в нем начальный и конечный элементы местами

## MusicalComposition\* ChangeList1(MusicalComposition\* head, int m){

## int i=0;

## int j=0;

## int n=count(head);

## MusicalComposition\*\* arr = (MusicalComposition\*\*)malloc((n+1)\*sizeof(MusicalComposition\*));

## for (i = 0; i<n; i++) arr[i] = (MusicalComposition\*)malloc(sizeof(MusicalComposition\*));

## for (i = 0; i<n; i++){

## arr[i] = head;

## head = head->next;

## }

## if (n%m==0){

## for(i = 0; i<n-1; i = i + m){

## arr[i+m-1]->next = NULL;

## arr[i]->prev = NULL;

## exchange(arr[i], arr[i+m-1]);

## }

## for (i = 1; i<n-1; i = i + m){

## arr[i+m-2]->next = arr[i+m-1];

## arr[i+1]->prev = arr[i];

## }

## }

## else {

## for(i = 0; i<n-2; i = i + m){

## arr[i+m-1]->next = NULL;

## arr[i]->prev = NULL;

## exchange(arr[i], arr[i+m-1]);

## }

## for (i = 1; i<n-2; i = i + m){

## arr[i+m-2]->next = arr[i+m-1];

## arr[i+1]->prev = arr[i];

## }

## i=n-2;

## arr[i+1]->next=arr[i+2];

## arr[i+1]->prev=arr[i];

## }

## head = arr[0];

## return head;

## }

## //разбить список на две части относительно индекса n, вторую часть разбить группы по m элементов, поменять в каждой начальный и конечный элемент местами, соединить обратно

## MusicalComposition\* ChangeList2(MusicalComposition\* head, int n, int m){

## MusicalComposition \*tmp = head;

## int j;

## for(j=0;j<(n-1);j++) tmp=tmp->next;

## MusicalComposition \*curr=tmp->next;

## tmp->next=NULL;

## ChangeList1(curr, m);

## tmp->next=curr;

## return head

## }

## 11.Makefile

Создадим поочерёдно объектные файлы функций и главной функции, после скомпилируем их вместе, также предусмотрена функция очистки после сборки проекта.

**curs: curs.o func.o**

**gcc curs.c -o curs.out func.c**

**rm -rf \*.o**

**curs.o :curs.c**

**gcc -c curs.c**

**func.o: func.c func.h**

**gcc -c func.c**

# **Работа с репозиторием**

1. Скопируем репозиторий группы 6304

*Git clone https://github.com/moevm/pr1-2016-6304.git*

1. От ветки master создаем локальную ветку с названием работы и именем:

*Git checkout –b “zubov\_coursework”*

1. В папку с клонированным репозиторием скопируем файлы с работой с помощью команды *cp –r*
2. Добавляем папку с файлом на ветку командой

*git add zubov\_coursework*

1. Коммитим изменения:

*Git commit –m “zubov\_coursework”*

1. Отправляем изменения в репозиторий командой

*Git push origin*

1. На сайте github создаем *pull request* в ветку *master.*

**примеры работы программы:**

Для демонстрации всех возможностей программы используется следующий формат ввода:

* 1. Выбор пункта меню
  2. Количество элементов списке
  3. Ввод названия, имени автора и год создания элемента списка
  4. Выбор в меню следующего действия
  5. Новый элемент списка
  6. Выбор в меню следующего действия
  7. Название композиции, которую нужно удалить
  8. Индекс, относительно которого будет просходить перестановка элементов

Для вывода:

1. Название, автор и год написания элементов списка
2. Количество элементов в результате каждого действия
3. Список названий композиций после добавления
4. Количество элементов после удаления
5. Список элементов после удаления

|  |  |  |
| --- | --- | --- |
| № | ВВОд | вЫВОД |
| 1 | 3 // количество элементов списка  // элементы для формирования списка  Rae Sremmurd  No Type  2015  ASAP Rocky  Canal St.  2015  50 Cent  Get Up  2010  Tory Lanez Line Up The Flex 1//выбор в меню добавления нового элемента  Line Up The Flex // элемент для добавления в список  2015  2//выбор в меню удаления элемента  Line Up The Flex //имя элемента для удаления из списка  4// запрос на вывод списка на экран  3// количество элементов для формирования списка | No Type //список названий композиций  Canal St.  //после добавления и удаления элемента  Get up  3 //количество элементов после добавления и удаления |
| 2 | Name1 // элементы для формирования списка  author1  1  Name2  author2  2  Name3  author3  3  5// разделение частей и копирование 2 части | Name1 // имена элементов списка создания  Name2 //  Name3  // список элемента после разделения и копирования частей  Name1  Name1  Name2  Name3 |
| 3 | Name1  Author1  1  …  Name4  Author4  4 | //список после группировки и перестановки  Name2  Name1  Name4  Name3  4// количество элементов списка |

# **Заключение**

Выполнив данную курсовую работу, мы освоили и закрепили на практике написание программы с применением структур в качестве элементов сложных типов представления данных таких, как одно- и дву-направленные линейные списки, а также API для работы с ними, в частности, функции вставки, удаления, подсчета и вывода элементов списка. Повторили работу с проектом в целом: разбиение его на заголовки (header files) и файлы кода (source files), а также написание makefile для сборки программы, так же темы: указатели, динамическая память, массивы и функции стандартных библиотек.

***Список использованных источников***

Язык программирования СИ / Керниган Б., Ритчи Д. СПб: Издательство «Невский Диалект», 2001. 352 с.

UNIX. Программное окружение / Керниган Б., Пайк Р. СПб: Символ Плюс, 2003. 416с.

# **Приложение**

# **Исходный код программы**

main.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "func.h"

int main(){

int sw=1;

int key = 0;

int length;

int year;

char name[80];

char author[80];

printf("Start creating list.\n Enter the length of the list:\n");

scanf("%d", &length);

char\*\* names = (char\*\*)malloc(sizeof(char\*)\*length);

char\*\* authors = (char\*\*)malloc(sizeof(char\*)\*length);

int\* years = (int\*)malloc(sizeof(int)\*length);

int i;

for (i=0;i<length;i++)

{ printf("Enter the name of song, the author and the year of creation\n");

scanf("%s%s%d", name, author, &year);

names[i] = (char\*)malloc(sizeof(char\*) \* (strlen(name)+1));

authors[i] = (char\*)malloc(sizeof(char\*) \* (strlen(author)+1));

strcpy(names[i], name);

strcpy(authors[i], author);

}

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

for(i=0; i<length; i++){

free(names[i]);

free(authors[i]);

}

free(years);

while(sw){

printf("1.Add element to list\n2.Remove element\n3.Count elements\n4.Print names of the list\n5.Make a copy of the 1st part\n6.Group elements and swap\n7.Quit\n");

scanf("%d", &key);

if(head == 0){

while((key != 1) || (key != 0)){

}

if(key == 0){

break;

sw = 0;

}

else{

printf("Enter the length of the list:\n");

scanf("%d", &length);

printf("Enter the name of song, the author and the year of creation\n");

scanf("%s%s%d", name, author, &year);

head = createMusicalComposition(name, author, year);

int i;

for (i=0;i<length-1;i++) {

printf("Enter the name of song, the author and the year of creation\n");

scanf("%s%s%d", name, author, &year);

push(head, createMusicalComposition(name, author, year));

}

}

}

else

switch(key){

case 1:

{

printf("Enter the song, the author and the year of creation\n");

scanf("%s%s%d", name, author, &year);

push(head, createMusicalComposition(name, author, year));

break;

}

case 2:

{

printf("Enter the name of the song to remove\n");

scanf("%s", name);

removeEl(head, name);

break;

}

case 3:

{

int c = count(head);

if(c>1)

printf("There are %d elements in the list\n", c);

else

printf("There is %d element in the list\n", c);

break;

}

case 4:

{

print\_names(head);

break;

}

case 5:

{

function(head);

print\_names(head);

break;

}

case 6:

{int m;

int n;

printf("Введите m и n");

scanf("%d %d",&n, &m);

ChangeList2(head, n, m);

print\_names(head);

break;

}

case 7:

{

sw = 0;

break;

}

default: ;

}

printf("If you want to create a list PRESS 1, exit PRESS 0\n");

scanf("%d", &key);

if(key == 0) break;

}

return 0;

}

func.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct MusicalComposition{

char name[80];

char author[80];

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

};

typedef struct MusicalComposition MusicalComposition;

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

MusicalComposition\* s = (MusicalComposition\*)malloc(sizeof(MusicalComposition));

strcpy(s->name, name);

strcpy(s->author, author);

years[i]=year;

s->year = year;

s->next = NULL;

s->prev = NULL;

return s;

}

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n){

MusicalComposition\* head = createMusicalComposition( array\_names[0], array\_authors[0], array\_years[0]);

MusicalComposition\* ptr = head;

MusicalComposition\* point = NULL;

int i;

for(i = 1; i<n; i++){

point = createMusicalComposition( array\_names[i], array\_authors[i], array\_years[i]);

point->prev = ptr;

ptr->next = point;

ptr = point;

}

return head;

}

void push(MusicalComposition\* head, MusicalComposition\* element){

while(head->next)

head = head->next;

head->next = element;

element->prev = head;

element->next = NULL;

}

void removeEl(MusicalComposition\* head, char\* name\_for\_remove){

MusicalComposition \*ptr = head;

while(head){

if(strcmp(head->name, name\_for\_remove) == 0){

if((head->next == NULL) && (head->prev == NULL)){

free(head);

head = NULL;

return;

}

if(head->prev == 0){

head->next->prev = NULL;

return;

}

else if (head->next == NULL){

head->prev->next = NULL;

return;

}

head->prev->next = head->next;

head->next->prev = head->prev;

break;

}

head = head->next;

}

head = ptr;

}

int count(MusicalComposition\* head){

int c = 0;

while(head){

c++;

head = head->next;

}

return c;

}

void print\_names(MusicalComposition\* head){

int i = 1;

while(head){

printf(" %d. %s %s %d\n", i++, head->name, head->author, head->year);

head = head->next;

}

}

void pushmid(MusicalComposition \* head, MusicalComposition\* element) {

MusicalComposition \* ptr = head;

int schet = 0;

while ( ptr->next != NULL) {

++schet;

ptr = ptr->next;

}

int j;

for ( j=0; j<((schet/2)+1); ++j) {

ptr = ptr->prev;

}

MusicalComposition\* second = ptr->next ;

MusicalComposition\* pp= createMusicalComposition(element->name, element->author, element->year);

ptr->next = pp;

pp->next=second;

}

void function(MusicalComposition\* head){

int len = count(head);

MusicalComposition \*current = head;

int i;

if(len%2==0){

i = len/2;

}

else{

i = len/2 +1;

}

int j;

for(j = 0; j<i; j++){

MusicalComposition\* part = (MusicalComposition\*)malloc(sizeof(MusicalComposition));

strcpy(part->name, current->name);

strcpy(part->author, current->author);

part->year = current->year;

pushmid(head, part);

current = current->next;

}

}

void exchange(MusicalComposition\* a, MusicalComposition\* b){

char t[80];

int r;

strcpy(t, a->name);

strcpy(a->name, b->name);

strcpy (b->name, t);

strcpy(t, a->author);

strcpy(a->author, b->author);

strcpy (b->author, t);

r=a->year;

a->year=b->year;

b->year=r;

}

//разбить список на группы и поменять в нем начальный и конечный элементы местами

MusicalComposition\* ChangeList1(MusicalComposition\* head, int m){

int i=0;

int j=0;

int n=count(head);

MusicalComposition\*\* arr = (MusicalComposition\*\*)malloc((n+1)\*sizeof(MusicalComposition\*));

for (i = 0; i<n; i++) arr[i] = (MusicalComposition\*)malloc(sizeof(MusicalComposition\*));

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

arr[i] = head;

head = head->next;

}

if (n%m==0){

for(i = 0; i<n-1; i = i + m){

arr[i+m-1]->next = NULL;

arr[i]->prev = NULL;

exchange(arr[i], arr[i+m-1]);

}

for (i = 1; i<n-1; i = i + m){

arr[i+m-2]->next = arr[i+m-1];

arr[i+1]->prev = arr[i];

}

}

else {

for(i = 0; i<n-2; i = i + m){

arr[i+m-1]->next = NULL;

arr[i]->prev = NULL;

exchange(arr[i], arr[i+m-1]);

}

for (i = 1; i<n-2; i = i + m){

arr[i+m-2]->next = arr[i+m-1];

arr[i+1]->prev = arr[i];

}

i=n-2;

arr[i+1]->next=arr[i+2];

arr[i+1]->prev=arr[i];

}

head = arr[0];

return head;

}

//разбить список на две части относительно индекса n, вторую часть разбить группы по m элементов, поменять в каждой начальный и конечный элемент местами, соединить обратно

MusicalComposition\* ChangeList2(MusicalComposition\* head, int n, int m){

MusicalComposition \*tmp = head;

int j;

for(j=0;j<(n-1);j++) tmp=tmp->next;

MusicalComposition \*curr=tmp->next;

tmp->next=NULL;

ChangeList1(curr, m);

tmp->next=curr;

return head;

}

func.h:

typedef struct MusicalComposition MusicalComposition;

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

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n);

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

void removeEl(MusicalComposition\* head, char\* name\_for\_remove);

int count(MusicalComposition\* head);

void print\_names(MusicalComposition\* head);

void function(MusicalComposition\* head);

MusicalComposition\* ChangeList2(MusicalComposition\* head, int n, int m);

void exchange(MusicalComposition\* a, MusicalComposition\* b);

MusicalComposition\* ChangeList1(MusicalComposition\* head, int m);