

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

**ОТЧЕТ**  
**по лабораторной работе №2**  
**по дисциплине «Программирование»**  
**Тема: Линейные списки**

Студентка гр. 3341

Яковлева А.А.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

## **Цель работы**

Целью работы является освоение работы с линейными списками.

Для достижения поставленной цели требуется решить следующие задачи:

- ознакомиться со структурой «список»;
- ознакомиться со списком операций используемых для списков;
- изучить способы реализации этих операций на языке С;
- написать программу, реализующую двусвязный линейный список и решающую задачу в соответствии с индивидуальным заданием.

## Задание

Создайте 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.

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]*).

Аналогично для второго, третьего, ... *n*-1-го элемента массива.

!длина массивов *array\_names*, *array\_authors*, *array\_years* одинаковая и равна *n*, это проверять не требуется.

Функция возвращает указатель на первый элемент списка.

- *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.

В функции *main* написана некоторая последовательность вызова команд для проверки работы вашего списка.

Функцию *main* менять не нужно.

## Выполнение работы

Элемент списка типа *MusicalComposition* состоит из:

- указателя на *char name* - названия композиции
- указателя на *char author* - автора композиции
- целого числа *year* - года создания композиции
- указателя на тип *MusicalComposition prev* - указателя на предыдущую композицию
- указателя на тип *MusicalComposition next* - указателя на следующую композицию

Функции:

- *MusicalComposition\* createMusicalComposition(char\* name, char\* author, int year)* принимает указатель на название композиции, автора композиции и год создания, с помощью функции *malloc* выделяет память для одного элемента типа *MusicalComposition*, полям *name*, *author*, *year* данного элемента присваивает соответствующие значения, полям *prev*, *next* присваивает *NULL*, возвращает указатель на созданный элемент.
- *MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)* принимает указатель на массив названий композиций, авторов композиций, лет создания и количество элементов в массивах; *head* присваивает нулевой элемент списка, созданный с помощью функции *createMusicalComposition*, предыдущей музыкальной композиции *prev\_musical\_composition* присваивает *head*, с помощью цикла *for* текущей музыкальной композиции *curr\_musical\_composition* присваивает *i*-ый элемент списка, созданный с помощью функции *createMusicalComposition*, полю *next* предыдущей музыкальной композиции присваивает указатель на текущую музыкальную композицию, полю *prev* текущей музыкальной композиции присваивает указатель на предыдущую музыкальную композицию, *prev\_musical\_composition* присваивает *curr\_musical\_composition*; возвращает *head*.

- *void push(MusicalComposition\* head, MusicalComposition\* element)* принимает указатель на нулевой элемент списка и на композицию, которую нужно добавить в список; с помощью цикла *while* проходит по списку до последнего элемента, т.е. пока поле *next* текущего элемента не равно *NULL*; после цикла полю *next* текущей музыкальной композиции (последней) присваивает *element*, полю *prev* добавленной музыкальной композиции присваивает *curr\_musical\_composition*.
- *void removeEl (MusicalComposition\* head, char\* name\_for\_remove)* принимает указатель на нулевой элемент списка и на название композиции, которую нужно удалить из списка; текущей музыкальной композиции *curr\_musical\_composition* присваивает *head*, с помощью цикла *do-while* проходит по списку пока указатель на текущий элемент не равен *NULL* и на каждом шаге цикла функцией *strcmp* проверяет совпадение поля *name* текущей композиции с *name\_for\_remove*, при совпадении удаляет из списка текущую композицию: полю *next* предыдущей музыкальной композиции (*curr\_musical\_composition->prev*) присваивает указатель на следующую музыкальную композицию *next\_musical\_composition*, полю *prev* следующей музыкальной композиции, если указатель на неё не *NULL*, т.е. текущая композиция не последняя, присваивает указатель на предыдущую музыкальную композицию, освобождает память, выделенную для текущей композиции.
- *int count(MusicalComposition\* head)* принимает указатель на нулевой элемент списка; количеству композиций *counting* присваивает 0, с помощью цикла *while* проходит по списку пока указатель на текущий элемент не равен *NULL* и добавляет 1 к *counting* на каждом шаге цикла; возвращает *counting*.
- *void print\_names(MusicalComposition\* head)* принимает указатель на нулевой элемент списка; с помощью цикла *do-while* проходит по списку пока указатель на текущий элемент не равен *NULL* и на каждом шаге цикла выводит поле *name* текущей композиции.
- *main* содержит некоторую последовательность вызова команд для проверки работы списка.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	3 Fields of Gold Sting 1993 In the Army Now Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Sonne Rammstein 2001 Mixed Emotions	Fields of Gold Sting 1993 3 4 Fields of Gold In the Army Now Sonne 3	<i>main</i> считывает в массив 3 названия композиций, авторов и лет; название, автора и год композиции, которую нужно будет добавить в список, затем название композиции, которую нужно удалить. Выводит название, автора и год первой композиции, количество элементов в списке (3), затем добавляет элемент и снова выводит количество элементов (4), удаляет элементы с заданным названием, после выводит названия композиций и количество элементов в списке.
2.	3 In the Army Now Sting 1993 Mixed Emotions Status Quo	In the Army Now Sting 1993 3 4 In the Army Now	В данном примере удаляется не одна, а 3 композиции с заданным именем.



	1986 Mixed Emotions The Rolling Stones 1989 Mixed Emotions Rammstein 2001 Mixed Emotions	1	
--	---	---	--

## **Выводы**

В ходе выполнения работы были изучены:

- основные принципы работы с линейными списками;
- структура списков и операции, применяемые к ним;
- способы реализации этих операций на языке С;
- написана программа, реализующая двусвязный линейный список и решающая задачу в соответствии с индивидуальным заданием.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

struct MusicalComposition
{
    char* name;
    char* author;
    int year;
    struct MusicalComposition* prev;
    struct MusicalComposition* next;
};

struct MusicalComposition* createMusicalComposition(char* name, char*
author, int year)
{
    struct MusicalComposition* musical_composition = (struct
MusicalComposition*)malloc(sizeof(struct MusicalComposition));
    musical_composition->name = name;
    musical_composition->author = author;
    musical_composition->year = year;
    musical_composition->prev = NULL;
    musical_composition->next = NULL;
    return musical_composition;
}

struct MusicalComposition* createMusicalCompositionList(char**
array_names, char** array_authors, int* array_years, int n)
{
    struct MusicalComposition* head =
createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
    struct MusicalComposition* prev_musical_composition = head;
    struct MusicalComposition* curr_musical_composition;
    for(int i = 1; i < n; i++)
    {
        curr_musical_composition =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        prev_musical_composition->next = curr_musical_composition;
        curr_musical_composition->prev = prev_musical_composition;
        prev_musical_composition = curr_musical_composition;
    }
    return head;
}

void push(struct MusicalComposition* head, struct MusicalComposition*
element)
{
    struct MusicalComposition* curr_musical_composition = head;
    while (curr_musical_composition->next != NULL)
        curr_musical_composition = curr_musical_composition->next;
```

```

    curr_musical_composition->next = element;
    element->prev = curr_musical_composition;
}

void removeEl(struct MusicalComposition* head, char* name_for_remove)
{
    struct MusicalComposition* curr_musical_composition = head;
    struct MusicalComposition* next_musical_composition =
curr_musical_composition->next;
    do
    {
        if (strcmp(curr_musical_composition->name, name_for_remove) == 0)
        {
            curr_musical_composition->prev->next =
next_musical_composition;
            next_musical_composition->prev =
curr_musical_composition->prev;
            free(curr_musical_composition);
        }
        curr_musical_composition = next_musical_composition;
        if (next_musical_composition != NULL) next_musical_composition =
next_musical_composition->next;
    }while (curr_musical_composition != NULL);
}

int count(struct MusicalComposition* head)
{
    int counting = 0;
    struct MusicalComposition* curr_musical_composition = head;
    while (curr_musical_composition != NULL)
    {
        curr_musical_composition = curr_musical_composition->next;
        counting++;
    }
    return counting;
}

void print_names(struct MusicalComposition* head)
{
    struct MusicalComposition* curr_musical_composition = head;
    do
    {
        puts(curr_musical_composition->name);
        curr_musical_composition = curr_musical_composition->next;
    }while (curr_musical_composition != NULL);
}

int main()
{
    int length;
    scanf("%d\n", &length);

    char** names = (char**)malloc(sizeof(char*)*length);
    char** authors = (char**)malloc(sizeof(char*)*length);
    int* years = (int*)malloc(sizeof(int)*length);

```

```

for (int i=0;i<length;i++)
{
    char name[80];
    char author[80];

    fgets(name, 80, stdin);
    fgets(author, 80, stdin);
    fscanf(stdin, "%d\n", &years[i]);

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

    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);

}
struct MusicalComposition* head = createMusicalCompositionList(names,
authors, years, length);
char name_for_push[80];
char author_for_push[80];
int year_for_push;

char name_for_remove[80];

fgets(name_for_push, 80, stdin);
fgets(author_for_push, 80, stdin);
fscanf(stdin, "%d\n", &year_for_push);
(*strstr(name_for_push, "\n"))=0;
(*strstr(author_for_push, "\n"))=0;

struct MusicalComposition* element_for_push =
createMusicalComposition(name_for_push, author_for_push, year_for_push);

fgets(name_for_remove, 80, stdin);
(*strstr(name_for_remove, "\n"))=0;

printf("%s %s %d\n", head->name, head->author, head->year);
int k = count(head);

printf("%d\n", k);
push(head, element_for_push);

k = count(head);
printf("%d\n", k);

removeEl(head, name_for_remove);
print_names(head);

k = count(head);
printf("%d\n", k);

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

```

```
        free(authors[i]);  
    }  
    free(names);  
    free(authors);  
    free(years);  
  
    return 0;  
}
```