

**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**

Студент гр. 3342

Иванов С. С.

Преподаватель

Глазунов С. А.

Санкт-Петербург

2024

## **Цель работы**

Изучить двунаправленные списки и их реализацию на языке программирования С. Реализовать двунаправленный список с несколькими полями и написать для работы с ним `api`.

## 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:

1) 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]).

2)void push(MusicalComposition\* head, MusicalComposition\* element); // добавляет element в конец списка musical\_composition\_list

3)void removeEl (MusicalComposition\* head, char\* name\_for\_remove); // удаляет элемент element списка, у которого значение name равно значению name\_for\_remove

4)int count(MusicalComposition\* head); //возвращает количество элементов списка

5)void print\_names(MusicalComposition\* head); //Выводит названия композиций.

## Выполнение работы

Необходимо было создать структуру двунаправленного списка, содержащего следующие поля: `char* name`(название композиции), `char* author`(автор композиции), `int year`(год создания), `struct MusicalComposition* next`(указатель на следующую композицию), `struct MusicalComposition* prev`(указатель на прошлую).

Дальше идет написание функций для этого списка:

1) `MusicalComposition* createMusicalComposition(char* name, char* author, int year)` – создание элемента списка. Выделяется память, заполняются поля через функцию `->` и данный элемент возвращается функцией.

2) `MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)` - создает список музыкальных композиций. Для начала создается “голова” списка из первого элемента. Потом циклом происходит обход поданного списка, создается новая структура и связывается с остальными элементами через поля `next` и `prev`.

3) `void push(MusicalComposition* head, MusicalComposition* element)` – добавления элемента в конец. Идет цикл до того момента, когда элемент будет ссылаться на `NULL`. Затем эта структура через `next` добавляется в список.

4) `void removeEl(MusicalComposition* head, char* name_for_remove)` - удаление элемента по названию композиции. Циклом находим нужный элемент. Освобождаем память и меняем связь в списке через поля `next` и `prev`.

5) `int count(MusicalComposition* head)` – подсчет количества элементов в списке. Циклом происходит обход элементов до ссылки на `NULL` и увеличение счетчика.

6) `void print_names(MusicalComposition* head)` – вывод всех композиций из списка.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	7 Fields of Gold Sting 1993 In the Army Now Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Billie Jean Michael Jackson 1983 Seek and Destroy Metallica 1982 Wicked Game Chris Isaak 1989 Points of Authority Linkin Park 2000 Sonne Rammstein 2001 Points of Authority	Fields of Gold Sting 1993 7 8 Fields of Gold In the Army Now Mixed Emotions Billie Jean Seek and Destroy Wicked Game Sonne 7	Верный вывод

## **Выводы**

Была разработана программа, создающая двунаправленный список из музыкальных композиций и выполняющая с ним определенные функции. Изучена работа с линейными списками, со структурами и реализация их на языке Си.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define BAD_ALLOC "Error: bad alloc\n"

typedef struct MusicalComposition
{
    struct MusicalComposition *next;
    struct MusicalComposition *prev;
    char name[80];
    char author[80];
    int year;
} MusicalComposition;

MusicalComposition *
createMusicalComposition(char *name, char *author, int year)
{
    MusicalComposition *comp
    (MusicalComposition*)malloc(sizeof(MusicalComposition));
    if (comp == NULL)
    {
        fprintf(stderr, BAD_ALLOC);
        exit(EXIT_FAILURE);
    }

    strncpy(comp->name, name, sizeof(comp->name)-1);
    strncpy(comp->author, author, sizeof(comp->author)-1);
    comp->year = year;
    comp->next = NULL;
    comp->prev = NULL;

    return comp;
}

MusicalComposition *
createMusicalCompositionList(
    char **array_names,
    char **array_authors,
    int *array_years,
    int n
)
{
    if (n < 1)
        return NULL;
```



```

    MusicalComposition *head = createMusicalComposition(array_names[0],
array_authors[0], array_years[0]);
    MusicalComposition *prev = head;
    MusicalComposition *curr = head;

    for(size_t i = 1; i < n; i++)
    {
        curr = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        prev->next = curr;
        curr->prev = prev;
        prev = curr;
    }

    return head;
}

void
push(MusicalComposition *head, MusicalComposition *element)
{
    if (head == NULL)
    {
        element->next = NULL;
        element->prev = NULL;
        return;
    }

    while(head->next != NULL)
        head = head->next;

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

void
removeEl(MusicalComposition *head, char *name_for_remove)
{
    if (head == NULL)
        return;

    while(strcmp(head->name, name_for_remove) != 0 && head->next != NULL)
        head = head->next;

    if (head->next == NULL && strcmp(head->name, name_for_remove) != 0)
        return;

    if(head->next != NULL && head->prev != NULL)
    {
        head->prev->next = head->next;
        head->next->prev = head->prev;
    } else if(head->next != NULL && head->prev == NULL)
        head->next->prev = NULL;
}

```

```

        else if(head->next == NULL && head->prev != NULL)
            head->prev->next = NULL;

        free(head);
    }

    int
    count(MusicalComposition *head)
    {
        if(head == NULL)
            return 0;

        int cnt = 1;
        while(head->next != NULL)
        {
            cnt++;
            head = head->next;
        }

        return cnt;
    }

    void
    print_names(MusicalComposition *head)
    {
        while (head != NULL)
        {
            fprintf(stdout, "%s\n", head->name);
            head = head->next;
        }
    }

    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);

    }
    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;

    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;
}

```