

**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. 3341

Преподаватель

\_\_\_\_\_  
\_\_\_\_\_

Шаповаленко

Е.В

Глазунов С.А.

Санкт-Петербург

2024

### **Цель работы**

Целью работы является изучение создания линейных списков и их применение на практике.

## Задание

### Вариант 1

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

Структура данных *MusicalComposition* содержит информацию о музыкальной композиции (название, автор, год выпуска) и указатели на следующий и предыдущий элементы в двусвязном списке.

Функция *createMusicalComposition* которая выделяет память под новую музыкальную композицию, заполняет поля структуры данными и возвращает указатель на созданную композицию.

Функция *createMusicalCompositionList* создает двусвязный список из массивов названий, авторов и годов выпуска композиций, которые подаются на вход. Функция создает элементы списка функцией *createMusicalComposition*, а затем связывает их указателями *prev* и *next*.

Функция *count* подсчитывает количество элементов в списке, начиная с головного узла.

Функция *push* добавляет новый элемент в конец списка и связывает его с предыдущим.

Функция *removeEl* удаляет элемент из списка по заданному названию.

Функция *printNames* выводит на экран названия всех композиций из списка, начиная с головного элемента.

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.	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	Тест с e.moevm

## **Выводы**

Изучение создания линейных списков и их применение на практике является важным этапом для понимания принципов работы с данными структурами. Разработка линейных списков позволяет эффективно организовывать данные и обращаться к ним в процессе выполнения алгоритмов. Практическое применение линейных списков широко распространено в различных областях информатики и программирования, так как они обеспечивают удобный доступ к элементам, упрощают процесс обработки данных и повышают эффективность работы с информацией. Изучение данной темы позволяет углубить знания о структурах данных и оптимизировать процессы обработки информации в различных программах и приложениях.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

// Описание структуры MusicalComposition
typedef struct MusicalComposition{
    char *name;
    char *author;
    int year;
    struct MusicalComposition *next;
    struct MusicalComposition *prev;
}MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char*
author,int year){
    MusicalComposition *new_musical_composition =
(MusicalComposition*)calloc(1, sizeof(MusicalComposition));
    new_musical_composition->name = name;
    new_musical_composition->author = author;
    new_musical_composition->year = year;
    new_musical_composition->next = NULL;
    new_musical_composition->prev = NULL;
    return new_musical_composition;
}

// Функции для работы со списком MusicalComposition

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* tmp = head;
    for (int i=1; i<n; i++) {
        tmp->next = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        tmp->next->prev = tmp;
        tmp = tmp->next;
    }
    return head;
}

void push(MusicalComposition* head, MusicalComposition* element){
    MusicalComposition* tmp = head;
```



```

        while(tmp->next != NULL){
            tmp = tmp->next;
        }
        tmp->next = element;
        tmp->next->prev = tmp;
    }

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

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

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

```