

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

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

Студент гр. 3343

Наумкин А.Д.

Преподаватель

Государкин Я. С.

Санкт-Петербург

2024

## **Цель работы**

Реализовать двунаправленный список музыкальных композиций MusicalComposition и API для работы с ним.

## 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**

### **main():**

- Главная функция программы.
- Считывает количество композиций (length).
- Считывает данные (название, автор, год) для каждой композиции.
- Создает список композиций.
- Добавляет новую композицию.
- Удаляет композицию по имени.
- Печатает информацию о композициях.

### **getSentence():**

- Считывает текст с консоли посимвольно.

### **createMusicalComposition():**

- Создает экземпляр MusicalComposition.

### **createMusicalCompositionList():**

- Создает связный список экземпляров MusicalComposition из предоставленных массивов.

### **push():**

- Добавляет новый элемент в конец списка.

### **removeEl():**

- Удаляет элемент из списка по имени.

### **count():**

- Возвращает количество элементов в списке.

### **print\_names():**

- Печатает имена всех элементов в списке.

Разработанный программный код см. в приложении А.

## **Выводы**

В ходе выполнения лабораторной работы была поставлена задача реализовать двунаправленный список музыкальных композиций MusicalComposition и API для работы с ним.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

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

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

MusicalComposition* createMusicalComposition(char* name, char*
autor, 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);

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 %zu\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;
}

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

MusicalComposition* createMusicalCompositionList(char**
array_names, char** array_authors, int* array_years, int n) {
    MusicalComposition* composition_list =
(MusicalComposition*)malloc(sizeof(MusicalComposition) * n);
    for (int i = 0; i < n; i++) {
        composition_list[i].name = array_names[i];
        composition_list[i].author = array_authors[i];
    }
}

```

```

        composition_list[i].year = array_years[i];
        if (i != 0) composition_list[i].prev = &composition_list[i
- 1];
        if (i != n - 1) composition_list[i].next =
&composition_list[i + 1];
    }
    composition_list[0].prev = NULL;
    composition_list[n - 1].next = NULL;
    return composition_list;
}

void push(MusicalComposition* head, MusicalComposition* element) {
    MusicalComposition* composition = head;
    while (composition->next != NULL) composition =
composition->next;
    composition->next = element;
    element->prev = head;
}

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

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

```

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