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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: «Линейные списки»

Студент гр. 7381 _____ Тарасенко Е.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

Цель работы

Создать двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface* - *в данном случае набор функций*) для работы со списком.

Дополнительная информация для выполнения лабораторной работы

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.

Вывод

В ходе данной работы был создан двунаправленный список музыкальных композиций. Были получены навыки, необходимые для работы со структурами данных и по созданию двунаправленных списков.

Приложение

Код программы:

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

```
#define N 81


typedef struct MusicalComposition

{

        char* name;

        char* author;

        int year;

        struct MusicalComposition* next;

        struct MusicalComposition* prev;

} MusicalComposition;


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

{

        MusicalComposition* element_for_push =
(MusicalComposition*)malloc(sizeof(MusicalComposition));


        element_for_push->name = (char*)malloc(sizeof(char)*N);

        element_for_push->author = (char*)malloc(sizeof(char)*N);


        strcpy(element_for_push->name , name );

        strcpy(element_for_push->author,author);

        element_for_push->year = year;

        element_for_push->next = NULL;

        element_for_push->prev = NULL;


        return element_for_push;

}


void push(MusicalComposition* head, MusicalComposition* element)

{

        MusicalComposition* tmp =
(MusicalComposition*)malloc(sizeof(MusicalComposition));

        if ( head->next == NULL )

        {

                element->next = NULL;
```

```
                element->prev = head;
                head->next = element;
                return;
        }
        tmp = head->next;

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

MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years,int length)
{
        if ( length == 0 )
            return NULL;

        MusicalComposition* head  =
(MusicalComposition*)malloc(sizeof(MusicalComposition));

        head->name = (char*)malloc(sizeof(char)*N);
        head->author = (char*)malloc(sizeof(char)*N);

        strcpy(head->name , array_names[0]);
        strcpy(head->author , array_authors[0]);
        head->year = array_years[0];

        MusicalComposition* tmp;

        for ( int i = 1; i < length ; i++ )
        {
```

```
                tmp = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
                push(head , tmp);
        }
        return head;
}



void removeEl(MusicalComposition* head, char* name_for_remove)
{
        MusicalComposition *tmp ;
        tmp = head;

        while ( tmp )
        {
                if ( strcmp( tmp->name , name_for_remove ) == 0 )
                {
                        tmp->next->prev = tmp->prev;
                        tmp->prev->next = tmp->next;
                        free(tmp);
                }
                tmp = tmp->next;
        }
}


int count(MusicalComposition* head)
{
        int i=1;
        MusicalComposition* tmp =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
        tmp = head->next;
        while (tmp)
        {
                tmp = tmp->next;
                i++;
        }
```

```
        return i;
}

void print_names(MusicalComposition* head)
{
      MusicalComposition* tmp;
    tmp = head->next;
            printf("%s\n", head->name);
            printf("%s\n", tmp->name);
        while (tmp->next)
        {
                if ( tmp->next->year != -1 ) printf("%s\n", tmp->next->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;
}
```