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. 9383 _____ Ноздрин В.В.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2020

1 Цель работы.

Реализовать двусвязный список на языке программирования Си.

2 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.

3 Выполнение работы.

Функции:

createMusicalComposition	***removeEl***
createMusicalCompositionList	***count***
push	***print_names***

Во всех функциях реализован следующий алгоритм:

выполняется проход в цикле по всем элементам списка и делается соответствующее действие с полями структуры, отвечающими за ссылки на следующий или предыдущий элемент двусвязного списка.

4 Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица – Результаты тестирования

№ п/п	Входные данные	Выходные данные

1. 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

5 Вывод.

Была изучена структура двусвязного списка и возможность ее реализации на языке программирования Си.

Разработана программа в которой реализована структура двусвязного списка, а так же реазилованы некоторые метода для работы с ним.

1 ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: **main.c**

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

// Описание структуры MusicalComposition
typedef struct track {
   /* 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.
    */
   char name[80];
   char author[80];
   int year;

   struct track *next;
   struct track *prev;

} MusicalComposition;

// Создание структуры MusicalComposition

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

// Функции для работы со списком MusicalComposition

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 %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;
```

```
}


MusicalComposition* createMusicalComposition(char* name, char* author,int year)
{
   MusicalComposition *track = malloc(sizeof(MusicalComposition));
   strcpy(track->name, name);
   strcpy(track->author, author);
   track->year = year;

   track->next = NULL;
   track->prev = NULL;

   return track;
}


MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)
{
        /* 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.
        */
        MusicalComposition *head = createMusicalComposition(array_names[0], array_authors[0], array_years[0]);
```

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

void push(MusicalComposition* head, MusicalComposition* element)
{
    /*
     * void push(MusicalComposition*  head, MusicalComposition* element);
     * добавляет element  в конец списка musical_composition_list
     */
    MusicalComposition *tmp = head;
    while (tmp->next)
        tmp = tmp->next;
    tmp->next = element;
    element->prev = tmp;
    element->next = NULL; // ?? без него тоже работает, но пусть будет так
}

void removeEl(MusicalComposition* head, char* name_for_remove)
{
    /*
     * void removeEl (MusicalComposition*  head, char* name_for_remove);
        * удаляет элемент element списка, у которого значение name равно значению name_for_remove
     */
    MusicalComposition *tmp = head;
    while (tmp)
    {
        if (!strcmp (tmp->name, name_for_remove))
```

```
        {
            if (tmp->prev)
            {
                if (tmp->next)
                {
                    tmp->prev->next = tmp->next;
                    tmp->next->prev = tmp->prev;
                    free(tmp);
                }
                else
                {
                    tmp->prev->next = NULL;
                    free(tmp);
                }
            } else {
                if (tmp->next)
                {
                    tmp->next->prev = NULL;
                    // MusicalComposition *new_head = tmp->next;
                    free(tmp);
                    // return new_head;
                }
                /*
                else
                {
                    free(tmp);
                    return NULL;
                }
                */
            }
            break;
        }
        tmp = tmp->next;
    }
}

int count(MusicalComposition* head)
{
```

```
    /*
     * int count(MusicalComposition*  head);
     * возвращает количество элементов списка
     */
    int counter = 0;
    MusicalComposition *tmp = head;
    while (tmp)
    {
        counter++;
        tmp = tmp->next;
    }
    return counter;
}

void print_names(MusicalComposition* head)
{
    MusicalComposition *tmp = head;
    while (tmp)
    {
        printf("%s\n", tmp->name);
        tmp = tmp->next;
    }
    /*
     * void print_names(MusicalComposition*  head);
     * Выводит названия композиций
     */
}
```