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. 1304 _____ Климов Г.А.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы.

Начиться работать с линейными списками

Задание.

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (**a***pplication* **p***rogramming* **i***nterface - 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

Создаём структуру MusicalComposition, у неё объявлены нужные поля из условия.

CreateMusicalComposition создаёт экземпляр структуры.

CreateMusicalCompositionList инициализирует список и возвращает головной элемент.

Push добавляет новый элемент списка.

RemoveEl удаляет нужный элемент.

Count возвращает число элементов в списке.

Print_names выводит names для каждого элемента 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	Пример из условия, выполнился верно

Выводы.

Была создана программа создающая двунаправленный список.

Также написаны функции, позволяющие работать с ним.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb1.c

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


typedef struct MusicalComposition

{
   char name[100];
   char author[100];
   int year;

   struct MusicalComposition *next;


} MusicalComposition;

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

   MusicalComposition *Composition = (MusicalComposition *) malloc (sizeof (MusicalComposition));
   strcpy (Composition->name, name);
```

```
    strcpy (Composition->author, author);
    Composition->year = year;
    Composition->next = NULL;

    return Composition;
}
```

```
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 = createMusicalComposition(array_names[1], array_authors[1],array_years[1]);
head->next = tmp;
for (int i = 2; i<n;i++){
        tmp->next = (MusicalComposition*)malloc(sizeof(MusicalComposition));
        strcpy(tmp->next->name, array_names[i]);
        strcpy(tmp->next->author, array_authors[i]);
        tmp->next->year = array_years[i];
        tmp->next->next = NULL;
```

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



    }


    void push(MusicalComposition*  head, MusicalComposition* element){



    MusicalComposition *tmp = (MusicalComposition*)malloc(sizeof(MusicalComposition));
    tmp = head->next;
    while (tmp->next != NULL){
        tmp = tmp->next;
    }
    tmp->next = element;



    }
```

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



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

   return count;
}
```

```
void print_names(MusicalComposition*  head){
    MusicalComposition *tmp =
(MusicalComposition*)malloc(sizeof(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;

}
```