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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Программирование»
Тема: Линейные списки

Студентка гр. 2384 _____ Валеева А. А.

Преподаватель _____ Гаврилов А.В.

Санкт-Петербург
2023

Цель работы.

Изучить принцип работы с линейными списками. Получить практические навыки работы с двунаправленными 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.

Объявим заголовки требуемых библиотек.

Для начала рассмотрим структуру *MusicalComposition*, в которой содержится название композиции (*name*), ее автор (*author*), год выпуска, а так же указатель на структуру предыдущего и следующего элементов, так как мы создаем двунаправленный список.

Рассмотрим функцию *creatMusicalComposition,* на вход передаем название, автора и год композиции, для заполнения структуры выделяем динамически память с помощью функции *malloc()* и заполняем поля структуры. Указатели на следующий и предыдущий элемент делаем *NULL.*

Следующая функция для работы со списком — *push(),* добавляет в конец списка новый элемент. На вход получает самый первый элемент и просто элемент списка. Принцип работы: мы создаем элемент *music*, который изначально приравниваем самому первому (*head*), далее, «проходя» по списку мы будем менять его на текущие элементы списка. Прохождение по списку осуществим с помощью цикла *while(),* ориентируясь на значение указателя на следующий элемент, ведь у самого последнего элемента он равен *NULL.* В цикле не забываем менять значение элемента *music.* Когда был найден конец списка, то добавляем переданный *element,* заполняя у него поле *back,* а у предыдущего поле *forward.*

Далее используем для создания списка функцию *creatMusicalCompositionList,* создаем начальный элемент *head, вызывая функцию creatMusicalComposition,* передавая на вход нулевой элемент массива с именами, а также автора композиции и года. Далее, начиная со следующего элемента, запускаем цикл *for,* в котором «идем» до конца количества элементов (переменная *n*). Создаем *elem*, вызывая функцию *creatMusicalComposition.* После вызываем функцию push, ккоторой передаем наш первый элемент *head,* и текущий *elem,* создавая таким способом двунаправленный список, который в конце возвращаем.

После реализуем функцию удаления элемента из списка при совпадении с искомым — *removeEl().* На вход получает самый первый элемент и искомый. Принцип работы: мы создаем элемент *music*, который изначально приравниваем самому первому (*head*), далее, «проходя» по списку мы будем менять его на текущие элементы списка. Прохождение по списку осуществим с помощью цикла *while(),* ориентируясь на значение указателя на следующий элемент. Внутри цикла проверяем на совпадение элемент с искомым с помощью функции *strcmp().* Если совпадение было, то удаляем элемент следующим образом: если он не первый, то в таком случае у

предыдущего элемента указатель на следующий элемент приравниваем к указателю на следующий элемент у удаляемого элемента. Тот же принцип с не последним элементом. В конце проверки не забываем менять значение перебираемого элемента *music.*

Функция подсчета количества элементов — *count()*. Тот же принцип с созданием элемента *music*, пока он не *NULL*, увеличиваем переменную количества переменных *count_real,* в конце возвращаем ее.

Последняя функция — функция печати *print_name(),* на вход которой передается указатель на нулевой элемент *head.* Тот же принцип с созданием элемента *music*, *пока он не NULL,* печатаем название композиции.

После идет прописанная основная функция *main.*

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	Fields of Gold Sting 1993 7 8 Fields of Gold In the Army Now Mixed Emotions Billie Jean Seek and Destroy Wicked Game Sonne	Верный ответ

	Wicked Game Chris Isaak 1989 Points of Authority Linkin Park 2000 Sonne Rammstein 2001 Points of Authority	7	

Выводы.

Были изучены принципы работы с линейными списками.

Разработан **api**, создающий двунаправленный список, удаляющий элементы из него и выполняющий подсчет количества элементов в списке.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Valeyeva_Alina_lb2.c

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

//структура элемента
typedef struct Musical{
    char *name;
    char *author;
    int year;
    struct Musical *forward;
    struct Musical *back;
}MusicalComposition;

// заполнение полей структуры
MusicalComposition*  createMusicalComposition(char*  name,  char*
autor,int year){
    MusicalComposition *trak = malloc(sizeof(MusicalComposition)*
1);
    trak->name = name;
    trak->author = autor;
    trak->year = year;
    trak->back = NULL;
    trak->forward = NULL;
    return trak;
}

// добавление элемент в конец списка
void push(MusicalComposition* head, MusicalComposition* element){
    MusicalComposition *music = head;
    while (music->forward != NULL){
        music = music->forward;
    }
    music->forward = element;
    element->back = music;
}

// создание списка элементов
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]);
    for (int n2 = 1; n2 < n; n2++){
                                    MusicalComposition    *elem    =
createMusicalComposition(array_names[n2],           array_authors[n2],
array_years[n2]);
```

```
        push(head, elem);
    }
    return head;
}

// удаление элемента
void removeEl(MusicalComposition* head, char* name_for_remove){
    MusicalComposition *music = head;
    while (music->forward != NULL){
        if (strcmp(music->name, name_for_remove) == 0){
            if (music->back != NULL){
                music->back->forward = music->forward;
            }
            if (music->forward != NULL){
                music->forward->back = music->back;
            }
        }
        music = music->forward;
    }
}

// количество элементов
int count(MusicalComposition* head){
    MusicalComposition *music = head;
    int count_real = 0;
    while (music != NULL){
        count_real ++;
        music = music->forward;
    }
    return count_real;
}

// печать элементов
void print_names(MusicalComposition* head){
    MusicalComposition *music = head;
    for (int i = 0; music!=NULL; i++){
        printf("%s\n", music->name);
        music = music->forward;
    }
}

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;

}
```