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. 1384 _____ Усачева Д.В.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург 2022

Цель работы.

Целью работы является создание линейного списка и оттачивание навыков работы со структурами и указателями.

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.

Изначально была создана структура элемента списка MusicalComposition, которая уже описана в задании.

Для реализации двунаправленного списка нужно добавить дополнительно указатель на предыдущий элемент этого списка и на следующий элемент этого списка.

В функции для создания элемента этого списка выделяется память под

структуру этого элемента, после чего в эту структуру присваивается указатель на переданную строку названия композиции, ее автора и года создания. Указатель на предыдущий и следующий элементы списка устанавливаются в NULL.

В функции для создания самого списка отдельно создается первый элемент этого списка, который позже будет возвращен как результат работы функции. Далее указатель на этот первый элемент копируется в head, который будет обновляться по мере заполнения этого списка и обозначает последний добавленный в список элемент. В цикле в этот самый head->next будут записываться указатель на каждый новый элемент списка. Взамен, в поле prev нового элемента списка будет записываться указатель на сам head, который после этого станет уже предпоследний, и, в качестве последнего шага в цикле, head перейдет в только что добавленный head->next.

В функцию добавления элемента в конец списка передается указатель на первый элемент этого списка, поэтому необходимо с помощью цикла while пройти до конца списка, и к последнему элементу присоединить новый, записав в поле next последнего элемента указатель на новый элемент, а в поле prev нового элемента – указатель на последний

В функции для удаления элемента из списка по названию перебираются элементы этого списка, в самом цикле с помощью функции strcmp сравниваются строки названия текущего элемента с той строкой, которую передали в функцию. В случае совпадения строк нужно удалить элемент из списка.

В функции для подсчета количества элементов списка count перебираются элементы этого списка, и с каждой новой итерацией значение счетчика, который будет возвращен как ответ, увеличивается на 1.

В функции для вывода названий композиций в списке перебираются элементы этого списка, и в цикле выводится значение поля name очередного элемента списка.

Разработанный программный код см. в приложении 2.

Результаты тестирования см. в приложении 1.

Выводы.

Был создан линейный список и проделана работа со структурами и указателями.

ПРИЛОЖЕНИЕ 1

ТЕСТИРОВАНИЕ

Таблица 1 - Пример тестовых случаев.

№ п/п	Входные данные	Выходные данные
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

ПРИЛОЖЕНИЕ 2
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

// Описание структуры MusicalComposition

typedef struct MusicalComposition {

    char *name;
    char *author;
    int year;

    struct MusicalComposition *next;
    struct MusicalComposition *prev;

} MusicalComposition;

// Создание структуры MusicalComposition

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

    MusicalComposition*        mc        =        (MusicalComposition*)
malloc(sizeof(MusicalComposition));
    mc->name = name;
    mc->author = autor;
    mc->year = year;
    mc->next = NULL;
    mc->prev = NULL;
    return mc;
}

// Функции для работы со списком MusicalComposition

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* prev = head;
    MusicalComposition* tmp;

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

```
}

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

    while (head->next)
            head = head->next;
        head->next = element;
        element->prev = head;
}

void removeEl(MusicalComposition* head, char* name_for_remove){

    while (head) {
        if (!strcmp(head->name, name_for_remove)) {
            MusicalComposition *prev = head->prev, *next = head->next;
            if (prev) prev->next = next;
            if (next) next->prev = prev;
            free(head);
            return;
        }
        head = head->next;
    }

}

int count(MusicalComposition* head){

    int count = 0;
        while (head) {
            count++;
            head = head->next;
        }
        return count;
}

void print_names(MusicalComposition* head){

    while (head) {
        puts(head->name);
        head = head->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;
}}
```