

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

Басыров В.А.

Преподаватель

Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Изучение структуры списка, а также углубленное изучение структур, и их взаимодействия друг с другом. Практика работы с односвязными, двусвязными списками.

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_years` (**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.

Основные теоретические положения.

Структура двунаправленный список чаще всего состоит из значений, который принимает каждый узел списка, а также из указателя на структуры следующего и предыдущего. Чаще всего базовые операции для них это добавление в конец, удаление элемента по значению, инициализация и освобождение памяти для этой структуры.

Выполнение работы.

Все функции выполнены в соответствии с условием задания.

При создании нового узла заполняются все поля, кроме полей указателей.

При инициализации списка в переменных указателях на структуры а и b хранятся соответственно текущее и предыдущее значение узла, таким образом, чтобы список был полностью проинициализирован.

При добавлении элемента ищется самый последний элемент, а дальше в поле указателя на следующий элемент этому элементу.

При удалении элемента, как и в инициализации, хранится предыдущий и текущий узел списка, обеспечивающий таким образом целостность списка.

Подсчет и распечатывание списка выполняется аналогичным образом.

Тестирование.

Здесь результаты тестирования, которые помещаются на одну страницу.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.		Fields of Gold	Пример из условия.

	<p>7</p> <p>Fields of Gold</p> <p>Sting</p> <p>1993</p> <p>In the Army Now</p> <p>Status Quo</p> <p>1986</p> <p>Mixed Emotions</p> <p>The Rolling Stones</p> <p>1989</p> <p>Billie Jean</p> <p>Michael Jackson</p> <p>1983</p> <p>Seek and Destroy</p> <p>Metallica</p> <p>1982</p> <p>Wicked Game</p> <p>Chris Isaak</p> <p>1989</p>	<p>Sting 1993</p> <p>7</p> <p>8</p> <p>Fields of Gold</p> <p>In the Army Now</p> <p>Mixed Emotions</p> <p>Billie Jean</p> <p>Seek and Destroy</p> <p>Wicked Game</p> <p>Sonne</p> <p>7</p>	
--	---	--	--

	Points Authority Linkin Park 2000 Sonne Rammstein 2001 Points Authority	of of		
2.	2 Help Beatles 1976 Yesterday Beatles 1975 Ulet PH 2000		Help Beatles 1976 2 3 Help Yesterday Ulet 3	-
3.	2 Atl Dance 2015 Slava Vladimir 2017 XACKu Panelka 2019		Atl Dance 2015 2 3 Atl Slava XACKu 3	-

Выводы.

Были изучены основы работы со структурами, а также изучена работа со списком. Ознакомлен с двусвязными и односвязными списками.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Сначала указываем имя файла, в котором код лежит в репозитории:

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 *pred;
    struct MusicalComposition *next;
}MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char*
author,int year)
{
    MusicalComposition *my=malloc(sizeof(MusicalComposition));
    my->name=name;
    my->author=author;
    my->year=year;
    my->pred=NULL;
    my->next=NULL;
    return my;
}
// 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_y
ears[0]);
    MusicalComposition *a;
    MusicalComposition *b;
    a=head;
    for (int i=1;i<n;i++)
    {
        b=createMusicalComposition(array_names[i],array_authors[i],array_years
[i]);
        a->next=b;
        b->pred=a;
        a=a->next;
    }
    return head;
}
```



```

void push(MusicalComposition* head, MusicalComposition* element)
{
    if (!head)
        head=element;
    else{
        while (head->next)
            head=head->next;
        head->next=element;
        element->pred=head;
    }
}

void removeEl(MusicalComposition* head, char* name_for_remove)
{
    MusicalComposition* a;
    if (head)
        while (head->next)
        {
            if (!strcmp(head->name, name_for_remove))
            {
                a=head;
                head=head->next;
                head->pred=a->pred;
                free(a);
                a=head->pred;
                a->next=head;
            }
            else
                head=head->next;
        }
}

int count(MusicalComposition* head)
{
    int kol=0;
    if (!head)
        return 0;
    while (head->next)
    {
        head=head->next;
        kol++;
    }
    return kol+1;
}

void print_names(MusicalComposition* head)
{
    while (head)
    {
        printf("%s\n", 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;
```