

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. 0382

Куликов М.Д.

Преподаватель

Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Создание двунаправленного списка и функций для работы с ним.

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.

Основные теоретические положения.

Двунаправленный список — структура данных, которая хранит в себе некоторые поля с данными , а также 2 поля типа **struct struct_name*** ,хранящие в себе ссылку на прошлый элемент списка и на следующий элемент списка. Эти два поля позволяют последовательно обрабатывать элементы списка.

Выполнение работы.

В начале была определена структура **MusicalComposition** для хранения элементов списка. В ней были определены 3 поля, предназначенных для

хранящихся в элементах данных и 2 поля, предназначенных для хранения ссылок на предыдущий и следующий элемент в списке.

Была создана функция

`MusicalComposition *createMusicalComposition(char *name, char *author, int year)` , которая создавала элемент списка из переданных аргументов (не считая ссылок на соседние элементы)

Была создана функция

`MusicalComposition *createMusicalCompositionList(char **array_names, char **array_authors, int *array_years, int n)` , которая с помощью переданных в массивах данных и количества элементов списка генерировала список и возвращала ссылку на первый элемент списка, используя цикл `for`.

Была создана функция `void push(MusicalComposition *head, MusicalComposition *element)`, которая переходила в конец списка и вставляла в конце новый элемент списка.

Была создана функция `void removeEl(MusicalComposition *head, char *name_for_remove)`, которая удаляла элементы, у которых поле `name` совпадало с переданным аргументом. В этой функции были учтены исключительные ситуации (удаление первого и удаление последнего элемента)

Были созданы функции `int count(MusicalComposition *head)`

и `void print_names(MusicalComposition *head)`. Первая функция возвращает количество элементов в списке, вторая выводит на экран поля `name` каждого элемента.

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	Программа работает корректно.

2.	2	ttt kkk 12	Программа выполняет
	ttt	2	корректно
	kkk	3	
	12	ttt	
	lll	fff	
	ppp	2	
	30		
	fff		
	sss		
	777		
	lll		

Выводы.

В результате выполнения задания была написана программа с использованием двунаправленного списка и функции для работы с ним.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

```
struct MusicalComposition {
    char *name;
    char *author;
    int year;
    struct MusicalComposition *next;
    struct MusicalComposition *before;
};
```

```
typedef struct MusicalComposition MusicalComposition;
```

```
MusicalComposition *createMusicalComposition(char *name, char *author,
int year) {
    MusicalComposition *elem = malloc(sizeof(struct MusicalComposition));
    elem->name = name;
    elem->author = author;
    elem->year = year;
    elem->before = NULL;
    elem->next = NULL;

    return elem;
}
```

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

```

        cur->before = before;
        before = cur;
    }
    cur->next = NULL;

    return head;
}

void push(MusicalComposition *head, MusicalComposition *element) {
    MusicalComposition *cur = head;
    while (cur->next != NULL) {
        cur = cur->next;
    }
    cur->next = element;
    element->before = cur;
    element->next = NULL;
}

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

```



```

        }
    }
}
else{
    cur = cur -> next;
}
}

}

int count(MusicalComposition *head) {
    int n = 1;
    MusicalComposition *cur = head;
    while(cur->next != NULL){
        cur = cur-> next;
        n+=1;
    }

    return n;
}

void print_names(MusicalComposition *head) {
    MusicalComposition *cur = head;
    while (cur != NULL) {
        printf("%s\n", cur->name);
        cur = cur->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;

}
```