

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. 3344

Щербак.М.С.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Освоение работы с линейными списками на языке Си на примере использующей их программы.

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.

Выполнение работы

Функции, представленные в данном коде, предназначены для работы с двусвязным списком структур `MusicalComposition`, которые содержат информацию о музыкальных произведениях: название (`name`), автора (`author`) и год создания (`year`). Рассмотрим каждую функцию подробнее:

1. `createMusicalComposition` - функция создания новой структуры `MusicalComposition`. Принимает на вход название, автора и год создания музыкального произведения, выделяет память под новую структуру и инициализирует её поля переданными значениями.

2. `createMusicalCompositionList` - функция создания двусвязного списка структур `MusicalComposition`. Принимает на вход массивы названий, авторов и годов создания музыкальных произведений, а также их количество. Для каждого элемента массивов создается новая структура `MusicalComposition`, которая затем добавляется в конец списка.

3. `push` - функция добавления нового элемента в конец списка. Принимает указатель на голову списка и указатель на новый элемент, который нужно добавить. Перебирает список до конечного элемента и добавляет новый элемент в конец.

4. `removeEl` - функция удаления элемента из списка по названию музыкального произведения. Принимает указатель на голову списка и название произведения для удаления. Перебирает список до тех пор, пока не найдет элемент с указанным названием, затем удаляет этот элемент из списка и освобождает выделенную память.

5. `count` - функция подсчета количества элементов в списке. Принимает указатель на голову списка и возвращает количество элементов в списке, перебирая его до конца.

6. `print_names` - функция вывода названий музыкальных произведений из списка. Принимает указатель на голову списка и последовательно выводит названия всех произведений.

Эти функции позволяют создавать, модифицировать и управлять списком музыкальных произведений, обеспечивая различные операции с данными в удобной структуре двусвязного 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	-

Выводы

Была освоена работа с линейными списками на языке Си на примере использующей их программы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.c

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

// Описание структуры MusicalComposition

typedef struct MusicalComposition{
    char* name;
    char* author;
    int year;
    struct MusicalComposition* prev;
    struct MusicalComposition* next;
} MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* autor, int
year)
{
    MusicalComposition* MusComp =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    MusComp->name = name;
    MusComp->author = autor;
    MusComp->year = year;
    return MusComp;
}

// Функции для работы со списком MusicalComposition

MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n)
{
    MusicalComposition* head = NULL;
    MusicalComposition* tail = NULL;
    for(int i = 0; i < n; ++i){
        MusicalComposition* MusComp =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        if(NULL == head){
            head = MusComp;
            tail = MusComp;
        } else {
            tail->next=MusComp;
            MusComp->prev=tail;
            tail = MusComp;
        }
    }
    return head;
}
```



```

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

void removeEl(MusicalComposition* head, char* name_for_remove)
{
    MusicalComposition* pointer = head;
    while(strcmp(pointer->name, name_for_remove)){
        pointer = pointer->next;
    }
    pointer->prev->next = pointer->next;
    pointer->next->prev = pointer->prev;
    free(pointer);
}

int count(MusicalComposition* head)
{
    int counter = 0;
    MusicalComposition* pointer = head;
    do counter++; while((pointer=pointer->next) != NULL);
    return counter;
}

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

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;
}

```