

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

Валиев Р.А.

Преподаватель

Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Научиться работать с динамическими структурами данных, в частности, с двунаправленными списками, а также освоить создание и использование API для управления такими структурами.

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

В ходе выполнения лабораторной работы был разработан тип `MusicalComposition`, представляющий собой элемент двунаправленного списка. Каждый элемент списка содержит информацию о музыкальной композиции: название (`name`), автора или музыкальную группу (`author`) и год создания (`year`). Для связи элементов списка между собой используются указатели на предыдущий (`prev`) и следующий (`next`) элементы. композиций:

- `createMusicalComposition` — создает новый элемент списка;
- `createMusicalCompositionList` — создает список из массивов названий, авторов и годов выпуска;
- `push` — добавляет элемент в конец списка;
- `count` — выводит количество элементов списка;
- `print_names` — выводит названия всех структур в списке;
- `removeEl` — удаляет элемент из списка по названию композиции;

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	
2.	3 Floods Pantera 1996 Points of Authority Linkin Park 2000 Seek and Destroy Metallica 1982 Angel of Death Slayer 1986	Floods Pantera 1996 3 4 Floods Seek and Destroy Angel of Death 3	

	Points of Authority		
--	---------------------	--	--

Выводы

Лабораторная работа успешно выполнена. Был реализован двунаправленный список музыкальных композиций с базовым набором функций для его управления. Эта работа позволила закрепить навыки работы с динамическими структурами данных в С, а также разработки и использования API для управления такими структурами. .

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

typedef struct MusicalComposition {
    char name[81];
    char author[81];
    int year;
    struct MusicalComposition* next;
    struct MusicalComposition* prev;
} MusicalComposition;

MusicalComposition* createMusicalComposition(char* name, char*
author, int year) {
    MusicalComposition* newComposition =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    strcpy(newComposition->name, name);
    strcpy(newComposition->author, author);
    newComposition->year = year;
    newComposition->next = NULL;
    newComposition->prev = NULL;
    return newComposition;
}

MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n) {
    if (n == 0) return NULL;
    MusicalComposition* head =
createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
    MusicalComposition* current = head;
    for (int i = 1; i < n; ++i) {
        MusicalComposition* newElement =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        current->next = newElement;
        newElement->prev = current;
        current = newElement;
    }
    return head;
}

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

```

void removeEl(MusicalComposition* head, char* name_for_remove) {
    MusicalComposition* current = head;
    while (current != NULL) {
        if (strcmp(current->name, name_for_remove) == 0) {
            if (current->prev != NULL) {
                current->prev->next = current->next;
            }
            if (current->next != NULL) {
                current->next->prev = current->prev;
            }
            free(current);
            break;
        }
        current = current->next;
    }
}

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

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

```