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. 3342 _____ Роднов И.С.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью работы является ознакомление с двунаправленным списком и его создание, а так же реализация API 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]).
- 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.

Выполнение работы

В программе реализована структура MusicalComposition, в которой находятся поля: char* name, char* author, int year, struct MusicalComposition* next, struct MusicalComposition* prev.

Функции createMusicalComposition передается на вход название муз. композиции, имя автора и год выхода. Выделяется память под элемент списка и инициализируется значение соответсвующей структуры переданными в функцию аргументами. Устанавливаются в NULL указатели на следующий и предыдущий элементы. В конце функция возвращает указатель на структуру.

Функции createMusicalCompositionList передается на вход массивы названий, авторов и годов для музыкальных композиций, а также количество элементов в этих массивах. Затем при помощи цикла инициализируются элементы списка. Функция возвращает указатель на первый его элемент.

Функции push передается на вход указатель на первый элемент списка и элемент типа MusicalComposition*, который необходимо добавить в конец двунаправленного списка. При помощи цикла функция начинает перебирать элементы списка, пока не встретит последний элемент и добавляет указатели next и prev соответстующих элементов, так, чтобы список продолжился передаваемым элементом.

Функции removeEl передается на вход указатель на голову списка и name композиции. При совпадении поля name с названием композиции, элемент необходимо удалить. При помощи цикла элементы перебираются, и при совпадении указатели next и prev меняются соотвествующим образом.

Функции count передается на вход указатель на первый элемент списка. Перебираются все элементы списка до тех пор, пока указатель на текущий элемент не станет равным NULL. Вместе с этим идет подсчет элементов списка.

Функции print_names передается на вход голова списка. Функция выводит все его элементы, пока не встретит указатель на текущий элемент, равный NULL.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ 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	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

	2001 Points of Authority	

Выводы

Ознакомились с двунаправленным списком и создали его, реализовали API для работы с ним.

ПРИЛОЖЕНИЕ А
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* element =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    if(element == NULL){
        exit(1);
    }
    element->name = name;
    element->author = autor;
    element->year = year;
    element->next = NULL;
    element->prev = NULL;
    return element;
}

// 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* tmp = head;

    for(int i = 1; i < n; i++){
        MusicalComposition* element =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        element->prev = tmp;
        tmp->next = element;
        tmp = element;
    }
    if (tmp != NULL) {
    tmp->next = NULL;
    }
```

```
    return head;
}

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

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

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

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

}
```