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 _____ Фоминых Е.Г.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Написать программу на языке Си, в которой реализуется функционал работы с линейным списком.

Задание

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface - 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 начале описывается структура MusicalComposition, которая имеет поля name, author, year и указатели на предыдущий и следующий элемент, pred и next соответственно. Функции для работы со списком:

- createMusicalComposition – принимает на вход название песни, автора и год, внутри создается указатель на структуру t, возвращает функция этот указатель.
- createMusicalCompositionList – принимает на вход массив названий, авторов, годов, и n – количество элементов. Внутри функции при помощи цикла for происходит создание двунаправленного списка, возвращает функция указатель на первый элемент списка.
- push – принимает на вход указатель на первый элемент списка и указатель на элемент, который нужно добавить в конец списка, внутри функция добавляет элемент в конец списка при помощи цикла while, функция ничего не возвращает.
- removeEl – принимает на вход указатель на первый элемент списка и имя, по которому нужно удалить элемент списка с соответствующим именем. С помощью цикла while происходит удаление элемента, если его имя совпадает с name_for_remove. Функция ничего не возвращает.
- count – принимает на вход указатель на начало списка, внутри происходит подсчет длины списка при помощи цикла while, возвращает функция длину списка.
- print_names – принимает на вход указатель на начало списка, внутри происходит вывод названий композиций при помощи цикла while, функция ничего не возвращает.

В конце идет функция main, написанная изначально в задании.

Тестирование

Результаты тестирования представлены в Таблице 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	Верно

Выводы

В ходе лабораторной работы была написана программа на языке Си, которая позволяет создавать линейный список и взаимодействовать с ним.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: src.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 *pred;

} MusicalComposition;


// Создание структуры MusicalComposition

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

// 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*                    first                    =
createMusicalComposition(array_names[i],              array_authors[i],
array_years[i]);
        if(NULL == head){
            head = first;
            tail = first;
        } else {
            tail->next=first;
            first->pred=tail;
            tail = first;
        }
```

```
    }
    return head;
}

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

void removeEl(MusicalComposition* head, char* name_for_remove){

    MusicalComposition *first = head;
    while(strcmp(first->name, name_for_remove)){
        first = first->next;
    }
    first->pred->next = first->next;
    first->next->pred = first->pred;
    free(first);
}

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

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

}
```