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. 3343 _____ Иванов П.Д.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург
2024

Цель работы
Написать реализацию двунаправленного списка на языке C.

Задание

Создайте двунаправленный список музыкальных композиций MusicalComposition и api(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.

  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.

Выполнение работы

Была написана структура, которая помимо полей, описанных в задании, имеет также указатели на предыдущий и следующий элементы структуры *prev* и *next* соответственно.

Также были реализованы необходимые функции:

- *createMusicalComposition():*

Создает экземпляр структуры MusicalComposition.

- *createMusicalCompositionList():*

Создает двунаправленный список экземпляров MusicalComposition из элементов входящих массивов.

- *push():*

Добавляет новый элемент в конец списка.

- *removeEl():*

Удаляет элемент из списка по имени.

- *count():*

Возвращает количество элементов в списке.

- *print_names():*

Печатает имена всех элементов в списке.

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	Тестирование работы программы

Выводы

В ходе выполнения поставленной задачи была написана структура для реализации двунаправленного списка *MusicalCompositionList* и API для него.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


typedef struct MusicalComposition {
    char name[80];
    char author[80];
    int year;
    struct MusicalComposition* next;
    struct MusicalComposition* prev;
}MusicalComposition;

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

// Функции для работы со списком MusicalComposition

MusicalComposition*              createMusicalCompositionList(char**
array_names, char** array_authors, int* array_years, int n){
    MusicalComposition*          MusicalCompositionList          =
malloc(sizeof(MusicalComposition) * n);
    for (int i = 0; i < n; ++i) {
        strcpy(MusicalCompositionList[i].name, array_names[i]);
        strcpy(MusicalCompositionList[i].author, array_authors[i]);
        MusicalCompositionList[i].year = array_years[i];
    }
    for (int i = 0; i < n; ++i) {
        if (i == 0){
            MusicalCompositionList[i].prev = NULL;
            MusicalCompositionList[i].next                          =
&MusicalCompositionList[i+1];
            continue;
        }
        if (i == n-1){
            MusicalCompositionList[i].prev                          =
&MusicalCompositionList[i - 1];
            MusicalCompositionList[i].next = NULL;
            continue;
        }
        MusicalCompositionList[i].prev  =  &MusicalCompositionList[i
- 1];
        MusicalCompositionList[i].next                              =
&MusicalCompositionList[i+1];
    }
```

```
    return MusicalCompositionList;
}

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

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

int count(MusicalComposition* head){
    MusicalComposition* temp = head;
    int ct = 1;
    while (temp->next != NULL){
        ct++;
        temp = temp->next;
    }
    return ct;
}

void print_names(MusicalComposition* head){
    MusicalComposition* temp = head;
    while (temp->next != NULL){
        puts(temp->name);
        temp = temp->next;
    }
    puts(temp->name);
}


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;
}
```