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. 1304 _____ Дешура Д.В.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Научиться создавать структуры данных и api к ним для решения конкретных задач.

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 программе используются функции стандартной библиотеки из заголовочных файлов <stdio.h>, <stdlib.h>, <string.h>. Была описана структура *MusicalComposition* и создана функция, создающая экземпляры этой структуры. Кроме того была реализована функция *createMusicalComposition()*, создающая двунаправленный связный список из элементов структуры, и написаны функции *count()* для вычисления длины списка, удаления элемента по его названию *removeEl()* и добавления элемента в конец списка *push()* и вывода названия всех песен, содержащихся в списке, *print_names()*. Список реализован через структуру содержащую ссылки на соседние элементы, а в качестве самого списка передаётся первый элемент структуры.

Разработанный программный код см. в приложении А.

Тестирование.

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	Пройден
2.	5 Fields of Verdun Sabaton 2019 War of Change / russian cover Radio Tapok 2021 The Attack of the Dead Men / russian cover Radio Tapok 2019 Feel Invincible Skillet 2016 Experience Ludovico Einaudi 2016 Dvadtsat Shortparis 2021 Feel Invincible	Fields of Verdun Sabaton 2019 5 6 Fields of Verdun War of Change / russian cover The Attack of the Dead Men / russian cover Experience Dvadtsat 5	Пройден

Выводы.

Выполнив лабораторную работу №2, мы создали программу, содержащую в себе двунаправленный линейный список структур, и написали функции для работы с этим списком. Освоили общие принципы работы с такими структурами.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: 1304_PR_Дешура_ДВ_ЛР2.c

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


// Описание структуры MusicalComposition

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

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char*
author,int year)
{
    char *n = malloc(80 * sizeof(char)), *a = malloc(80 *
sizeof(char));
    strcpy(n, name);
    strcpy(a, author);
    MusicalComposition *mc = (MusicalComposition
*)malloc(sizeof(MusicalComposition));
    mc -> name = n;
    mc -> author = a;
    mc -> year = year;
    return mc;
}

// Функции для работы со списком MusicalComposition

MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n){
    void *pointer, *head;
    MusicalComposition* list;
    list = createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
    list -> previuos = NULL;
    head = list;
    for(int i = 1; i < n; i ++){
        pointer = (void *)list;
        list = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        list -> previuos = (MusicalComposition *)pointer;
        list -> previuos -> next = (MusicalComposition *)list;
    }
    list -> next = NULL;
    return head;
}
```

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

void removeEl(MusicalComposition* head, char* name_for_remove){
    while(strcmp(head -> name, name_for_remove) != 0)
        head = head -> next;
    head -> previuos -> next = head -> next;
    head -> next -> previuos = head -> previuos;
    free(head -> name);
    free(head -> author);
    free(head);
}

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

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

    }
```