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 _____ Сербиновский Ю.М.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Освоение работы с линейными списками, получение навыков их составления. Получение практического опыта создания и обработки линейных списков на языке программирования С.

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 двунаправленного списка.

Функцию main менять не нужно.

Выполнение работы

Были описана структура MusicalComposition. Были созданы функции для работы со списком:

CreateMusicalComposition - функция, внутри которой выделяется память для структуры MusicalComposition, элементам структуры присваиваются поданные данные. Функция возвращает указатель на структуру.

Push - функции для добавления экземпляра MusicalComposition в конец списка, путём монипуляций с указателями на структуру.

CreateMusicalCompositionList - функция создания двунаправленного списка, элементы которого связаны между собой линейно при помощи указателей на структуру MusicalComposition. Также внутри функции создаётся структура head (первый элемент списка).

RemoveEl - функция, удаляющая элемент, имеющий имя, которое совпадает с поданным на вход функции. Удаление происходит посредством освобождения памяти при помощи free(). Чтобы не нарушить структуру списка, указатель на удаленный элемент перенаправляется на следующий после него.

Count - функция подсчёта количества элементов с помощью do-while, который останавливается при нулевом указателе на следующий элемент.

Print_names - функция вывода имён элементов списка с помощью цикла while.

Функция main была дана изначально с целью проверки функциональности api.

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 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

Выводы

Был получен опыт работы с линейными спиками и были усовершенствованы навыки работы с указателями. Также был реализован api для работы с линейными списками, который позволяет с ними эфективно работать.

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;
}MusicalComposition;



// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* autor,int
year);

// Функции для работы со списком MusicalComposition

MusicalComposition*   createMusicalCompositionList(char**   array_names,
char** array_authors, int* array_years, int n);

void push(MusicalComposition* head, MusicalComposition* element);

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

int count(MusicalComposition* head);

void print_names(MusicalComposition* head);


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;

}
```

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

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 = createMusicalComposition(array_names[1],
array_authors[1], array_years[1]);
    head->next = tmp;
    for (int i = 2; i < n; i++){
        tmp->next       =       createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        tmp->next->next = NULL;
        tmp = tmp->next;
    }
    return head;
}


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

void removeEl(MusicalComposition* head, char* name_for_remove) {
    if (strcmp(head->name, name_for_remove)==0) {
        MusicalComposition* a = head;
        head = head->next;
        free(a);
    }
    else {
        MusicalComposition* tmp = head;
        while(1) {
            if (strcmp(tmp->next->name, name_for_remove) == 0) {
                MusicalComposition* a = tmp->next;
                tmp->next = tmp->next->next;
                free(a);
                break;
            }
            tmp = tmp->next;
        }
    }
}

int count(MusicalComposition* head) {
    int c = 0;
```

```
    if (head != NULL) {
        MusicalComposition* tmp = head;
        do
        {
            c++;
            tmp = tmp->next;
        } while (tmp != NULL);
    }
    return c;
}

void print_names(MusicalComposition* head){
    if (head != NULL) {
        puts(head->name);
        MusicalComposition* tmp = head->next;
        while(tmp != 0) {
            puts(tmp->name);
            tmp = tmp->next;
        }
    }
}
```