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		**Сьомак Д.А.**
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 C.

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

Были описана структура MusicalComposition. Далее был создан набор функций для работы со списком:

CreateMusicalComposition - функция, внутри которой выделяется память для структуры Composition, после чего в неё заносятся считанные данные. Функция возвращает указатель на структуру, для которой была выделена память.

Push - функции для добавления элемента в конец списка, путём перемещения в конец списка и изменения prev и next.

CreateMusicalCompositionList - функция создания двунаправленного списка композиций с помощью функции push. Также внутри этой функции создаётся структура head, которая служит началом списка.

RemoveEl - функция, удаляющая элемент, имеющий имя, которое совпадает с поданным на вход функции. Это происходит путём перемещения по списку и сравнения имён элементов, при совпадении элемент удаляется и переназначаются prev и next.

Count - функция подсчёта количества элементов посредством прохода по списку циклом while, который останавливается при нулевом указателе на следующий элемент.

Print_names - функция вывода имён элементов списка с помощью того же цикла while, который останавливается при получении нулевого элемента.

После функций api следует функция main, которая была заранее прописана с целью проверки корректности api.

Исходный код см. в приложении А.

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	-

Выводы

Были освоены правила работы с линейными списками, получены навыки их составления. Был получен практический опыт использования двунаправленных линейных списков и api, который требуется для работы с ними, посредством написания программы на языке программирования C.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Somak_Demid_lb2.c

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


// Описание структуры MusicalComposition
typedef struct MusicalComposition {

    struct MusicalComposition *prev;
    struct MusicalComposition *next;
    char *name;
    char *author;
    int year;

} MusicalComposition;


// Создание структуры MusicalComposition
MusicalComposition*    createMusicalComposition(char*    name,    char*
author,int year){
    MusicalComposition*                  Composition                  =
(MusicalComposition*)malloc(sizeof(MusicalComposition));

    Composition->name = name;
    Composition->author = author;
    Composition->year = year;
    Composition->next = NULL;
    Composition->prev = NULL;

    return Composition;
}

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

    for(int i = 1; i < n; i++){
        element       =       createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        push(head,element);
    }

    return head;
}

// Функции для добавления элемента в конец списка
```

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

    element -> prev = head;
    head-> next = element;

}

int count(MusicalComposition* head);

// Функции для удаления элемента списка,
void removeEl(MusicalComposition* head, char* name_for_remove){

    while(head != NULL){
        if (strcmp(head->name,name_for_remove)==0){
            head->next->prev = head->prev;
            head->prev->next = head->next;
            free(head);
            head = NULL;
        }else{
            head = head->next;
        }
    }
}

// Функции для подсчёта количества элементов списка
int count(MusicalComposition* head){
    int count = 1;

    while(head->next != NULL){
        head = head->next;
        count+= 1;
    }

    return count;
}

// Функции для вывода названия композиций
void print_names(MusicalComposition* head){
    while(head != NULL) {
        printf("%s\n", head->name);
        head = head->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;

}
```