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. 8303 _____ Гришин К. И.

Преподаватель _____ Чайка К. В.

Санкт-Петербург
2019

Цель работы

Изучить принципы работы с адресами структур. Смоделировать такую структуру данных, как линейный двунаправленный список.

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

Ход выполнения работы

Помимо описанной в задании функции *main()*, программа использует:

- структуру *MusicalComposition*, она содержит три поля для данных пользователя и два поля для хранения ссылок на следующий и предыдущий элементы списка.
- Функцию *createMusicalComposition*, она принимает на вход значения полей некоторой композиции и возвращает адрес сформированной структуры. В поля *name* и *author* с помощью функции *strcpy* копируются значения соответствующих полей в вызове функции, в поле *year* просто присваивается значение соответствующего поля. Поля для ссылок получают значение NULL.
- Функцию *createMusicalCompositionList*, она принимает на вход массив значений из которых должен быть составлен список и возвращает адрес первого элемента. Создается первый элемент который именуется *head*, а все дальнейшие как *tmp*, для перехода к следующему элементу без потери предыдущего инициализурется переменная *prevTmp*, которая хранит адрес предыдущей структуры, затем иттеративно создается *«следующая»*

структура по адресу полю next текущей tmp, затем в цикле значение поля *tmp → next → prev* принимает ранее записанное значение *prevTmp,* в конце цикла в *tmp* записывается новый адрес, а именно *tmp→ next.*

- Функцию *Push*, которая получает на вход голову списка и новый элемент и добавляет этот элемент в конец списка. Цикл используя голову списка двигается по элементам пока не будет найден такой, что значение следующего равно NULL, и в этот следующий добавляется элемент который был подан на вход.
- Функцию *removeEl*, которая получает на вход голову списка и некоторую строку, которая определяет какой элемент надо удалить, а именно по полю *name*. Функция циклично шагает по элементам, пока не найдет требуемое совпадение и не отвяжет соответствующие ссылки от требуемого элемента, соединив при этом соседние, либо пока не дойдет до конца списка.
- Функцию *Count,* которая получает на вход голову списка и просто идет до конца считая количество иттераций и возвращая это количество.
- Функцию *print_names*, которая получает на вход голову списка, она циклично идет по элементам списка, выводя при этом значение в поле *name* посредством функции *printf* стандартной библиотеки.

Код программы находится в приложении А к лабораторной работе.

Выводы

В ходе лабораторной работы были изучены способы управления структурой, используя ее адрес, а также была смоделирована структура данных «линейный двунаправленный список».

ПРИЛОЖЕНИЕ А

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


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


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

// Функции для работы со списком MusicalComposition


MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n){
    MusicalComposition* head = createMusicalComposition(*array_names, *array_authors, *array_years);
    MusicalComposition* tmp;
    MusicalComposition* prevTmp = head;

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


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


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

int count(MusicalComposition* head){
    MusicalComposition* tmp = head;
    int k;
    for (k = 1; tmp->next != NULL; k++) tmp = tmp->next;
    return k;
}

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

}
```