САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Программирование»

Тема: Лабораторная работа № 2: Линейные списки

Студент гр. 3341 _____ Игнатьев К.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Изучить и научиться применять двунаправленные линейные списки на языке Си для хранения данных полей структуры. Создать программный интерфейс для работы со списками, например, добавления и удаления элементов.

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

Описание функций:

- int main(): главная функция программы, возвращает 0. Принимает на вход и обрабатывает данные для создания и изменения списка.
- MusicalComposition* createMusicalComposition(char* name, char* author, int year): создает элемент списка из поданных в качестве аргументов значений.
- MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n): создает список музыкальных композиций.
- void push(MusicalComposition* head, MusicalComposition* element): добавляет элемент (element) в конец списка (head).
- 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.

Описание структур:

- struct MusicalComposition - структура элемента списка, имеет следующие поля:
  - char name[80] — строка, не превышающая длиной 80 символов, название композиции.
  - author — строка, не превышающая длиной 80 символов, автор композиции/ музыкальная группа.
  - int year - целое число, год создания.
  - struct MusicalComposition* next — указатель на следующий элемент списка.

Разработанный программный 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 Floods Pantera 1996 Flag of Hate Kreator 1985 In the Army Now Status Quo 1986 Idle Hands Kerry King 2024 One Metallica 1989 Midnight Sun Kreator 2022 Agent Orange Sodom 1989 Need Money for Beer Tankard 2002 Flag of Hate	Floods Pantera 1996 7 8 Floods In the Army Now Idle Hands One Midnight Sun Agent Orange Need Money for Beer 7	Выходные данные соответствуют ожиданиям.

2.	1 Flag of Hate Kreator 1985 One Metallica 1989 Flag of Hate	Flag of Hate Kreator 1985 1 2 One 1	Выходные данные соответствуют ожиданиям.
3.	2 Floods Pantera 1996 Flag of Hate Kreator 1985 One Metallica 1989 Midnight Sun Kreator 2022 Flag of Hate	Floods Pantera 1996 2 3 Floods Flag of Hate One 3	Выходные данные соответствуют ожиданиям.

Выводы

В ходе выполнения лабораторной работы были освоены необходимые навыки для создания двунаправленных списков на языке Си, а также программных интерфейсов для работы с ними.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

typedef struct MusicalComposition {
    char* name;
    char* author;
    int year;

    struct MusicalComposition *next;
    struct MusicalComposition *prev;
} MusicalComposition;

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

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 * composition=(MusicalComposition*)
malloc(sizeof (MusicalComposition));

    composition->name= malloc((strlen(name)+1)*sizeof(char));
    composition->author= malloc((strlen(author)+1)*sizeof(char));

    strcpy(composition->name, name);
    strcpy(composition->author, author);
    composition->year=year;

    composition->next=NULL;
    composition->prev=NULL;
```

```
    return composition;
}

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

    if (n == 0) {
        return head;
    }

    MusicalComposition* list = createMusicalComposition(*array_names,
*array_authors, *array_years);

    head = list;

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

    return head;
}

void push(MusicalComposition* head, MusicalComposition* element){
    MusicalComposition * list = head;

    if (head == NULL){
        head = element;
        return;
    }

    while (list->next != NULL){
        list = list->next;
    }
```

```
    list->next = element;
    list->next->prev=list;
}

void removeEl(MusicalComposition* head, char* name_for_remove) {
    MusicalComposition* list = head;

    while (list != NULL) {
        if (strcmp(name_for_remove, list->name) == 0) {

            if (list->next != NULL) {
                list->next->prev = list->prev;
            }

            if (list->prev != NULL) {
                list->prev->next = list->next;
            }

            free(list);
        }

        list=list->next;
    }
}

int count(MusicalComposition* head) {
    int quantity = 0;
    MusicalComposition* list = head;

    while (list != NULL) {
        quantity++;
        list = list->next;
    }

    return quantity;
}

void print_names(MusicalComposition* head) {
    MusicalComposition* list = head;
```

```
    while (list != NULL) {
        printf("%s\n", list->name);
        list = list->next;
    }
}
```