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 _____ Хангулян С. К.

Преподаватель _____ Глазунов С. А.

Санкт-Петербург

2024

Цель работы

Изучение основ работы односвязных и двусвязных линейных списков, написание программы, создающей двунаправленный линейный список и работающей с ним.

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:

1. n - длина массивов array_names, array_authors, array_years.
2. поле name первого элемента списка соответствует первому элементу списка array_names (array_names[0]).
3. поле author первого элемента списка соответствует первому элементу списка array_authors (array_authors[0]).
4. 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

Вначале была создана структура MusicalComposition с полями name, author, year, а также prev и next, необходимыми для работы с двусвязным списком. Функция createMusicalComposition непосредственно создает структуру и инициализирует ее полями переданными в функцию значениями, возвращая указатель на структуру.

Функция push добавляет в конец списка элемент. Создается копия «головы» - temp, после чего ищется последний элемент списка. Полем next объявляется добавляемый элемент, у самого элемента поле next – NULL, а поле prev – последний элемент старого списка.

Функция createMusicalCompositionList создает список. Нулевой элемент списка – «голова». Далее с помощью цикла и функции push создаются и добавляются последующие элементы списка. Функция возвращает указатель на нулевой элемент.

Функция removeEl удаляет элемент из списка. Пробегаясь по списку, функция ищет совпадение данного имени и имени текущей песни. В случае совпадения, полем next предыдущего элемента становится следующий, полем prev следующего – предыдущий, текущий элемент удаляется, итерация цикла прекращается.

Функция count подсчитывает и выводит количество элементов списка. Пока текущий элемент temp не равен NULL, счетчик увеличивается.

Функция print_names выводит названия всех песен из списка. Пока текущий элемент temp не равен NULL, имена выводятся на экран.

Тестирование

Результаты тестирования представлены в таблице 1.

Таблица 1 – Результаты тестирования

№	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	Корректно

Выводы

Были изучены основы работы односвязных и двусвязных линейных списков, была написана программа, создающая двунаправленный линейный список и работающая с ним. Все поставленные задачи выполнены.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Khangulyan_Sargis_lb2

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

typedef struct MusicalComposition{
    char* name;
    char* author;
    int year;
    struct MusicalComposition* prev;
    struct MusicalComposition* next;
} MusCom;



MusCom* createMusicalComposition(char* name, char* author, int year){
    MusCom* node = (MusCom*)malloc(sizeof(MusCom));
    node->name = name;
    node->author = author;
    node->year = year;

    return node;
}



void push(MusCom* head, MusCom* node){
    MusCom* temp = head;
    while(temp->next != NULL)
        temp = temp->next;
    temp->next = node;
    node->prev = temp;
    node->next = NULL;
}



MusCom*    createMusicalCompositionList(char**    array_names,    char**
array_authors, int* array_years, int n){
    MusCom*     head     =     createMusicalComposition(array_names[0],
array_authors[0], array_years[0]);
    head->prev = NULL;

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

    return head;
}
```

```
void removeEl(MusCom* head, char* name_for_remove){
    MusCom* temp = head;
    while(temp){
        if (!strcmp(temp->name, name_for_remove)){
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
            free(temp);
            break;
        }
        temp = temp->next;
    }
}



int count(MusCom* head){
    MusCom* temp = head;
    int count = 0;

    while(temp){
        count += 1;
        temp = temp->next;
    }

    return count;
}



void print_names(MusCom* head){
    MusCom* temp = head;
    while(temp){
        printf("%s\n", temp->name);
        temp = temp->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);

    }
    MusCom* 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;

    MusCom* 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;
}
```