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

Студентка гр. 3342 _____ Смирнова Е.С.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью работы является освоение работы с линейными списками, а также реализовать программу с их использованием.

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.

Основные теоретические положения

Двусвязный список – это список с возможностью идти в обе стороны, в отличие от односвязного. Каждый элемент двусвязного списка имеет указатель на предыдущий элемент и на следующий.

Выполнение работы

Структура MusicalComposition имеет поля: name, author, year, next, а также next и prev — указатели на предыдущий и следующий элементы списка композиций.

Функция MusicalComposition* createMusicalComposition() возвращает указатель на новую «композицию».

Функция MusicalComposition* createMusicalCompositionList() создает список из структур, каждая композиция добавляется с помощью функции push().

Функция void push() добавляет элемент в конец списка.

В функции void removeEl() используется функция strcmp() для сравнения названия текущей композиции и той, что удалится. Значения полей предыдущего и последующего элементов next и prev меняются, память удаляемого элемента очищается с помощью free().

Функция int count() возвращает количество элементов списка.

Функция void print_names() выводит названия всех композиций.

Разработанный программный код см. в приложении А.

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

Изучены и освоены работы с линейными списками, а также реализована программа с их использованием.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu.c

```
#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);

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

    MusicalComposition    *    creation    =    (MusicalComposition    *)
malloc(sizeof(MusicalComposition));
    if (creation == NULL){
        exit(1);
    }
    creation->name = name;
    creation->author = author;
    creation->year = year;
    creation->next = NULL;
    creation->prev = NULL;
    return creation;
}

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

void push(MusicalComposition* head, MusicalComposition* element) {
```

```
    while (head->next) head = head->next;
    head->next = element;
    element->prev = head;
}

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

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

int count(MusicalComposition* head) {
    int n = 0;
    while (head) {
        n++;
        head = head->next;
    }
    return n;
}

void print_names(MusicalComposition* head) {
    while (head) {
        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;
}
```