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. 0382 Мукатанов А.В.

Преподаватель Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Создание двунаправленного списка и функций для работы с ним.

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

В функции main написана некоторая последовательность вызова команд для проверки работы вашего списка.

Функцию main менять не нужно.

Основные теоретические положения.

Двунаправленный список — структура данных, которая хранит в себе некоторые поля с данными , а также 2 поля типа struct struct_name* ,хранящие в себе ссылку на прошлый элемент списка и на следующий элемент списка. Эти два поля позволяют последовательно обрабатывать элементы списка.

Выполнение работы.

Создана структура элемента списка *struct MusicalComposition* с именем типа *MusicalComposition* (через оператор *typedef*). Структура состоит из полей *char* name* (название песни), *char* author* (автор), *int year* (год создания). Также присутствую поля *previous* и *next* – указатели на соответственно предыдущий и следующий элемент списка.

Функция *MusicalComposition* createMusicalComposition(char* name, char* author, int year)* является конструктором экземпляра *MusicalComposition*, принимающим данные о композиции, и возвращающим указатель на готовый экземпляр.

Функция *MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)* создаёт двунаправленный список из элементов *MusicalComposition*. Через поля *previous* (у первого элемента - *NULL*) и *next* (у последнего - *NULL*) создаётся направленная связь между элементами списка. Функция принимает массивы с именами, авторами и годами и возвращает указатель на первый элемент списка.

Функция *void push(MusicalComposition* head, MusicalComposition* element)* добавляет элемент *element* в конец списка, добавляя в поле *next* последнего элемента списка указатель на *element*.

Функция *void removeEl(MusicalComposition* head, char* name_for_remove)* удаляет элемент из списка по его названию, которая удаляла элементы, у которых поле *name* совпадало с переданным аргументом. В этой функции были учтены исключительные ситуации (удаление первого и удаление последнего элемента).

Функция *int count(MusicalComposition* head)* и *void print_names(MusicalComposition *head).*Первая функция возвращает количество элементов в списке, вторая выводит на экран поля name каждого элемента.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

Тестирование.

Здесь результаты тестирования, которые помещаются на одну страницу.

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	Программа работает корректно

Выводы.

Был изучена работа со структурами и линейными списками в языке C. Разработана программа с API для работы с двунаправленным списком музыкальных композиций: создание одного элемента и списка, добавление в список элемента, удаление из списка элемента, подсчёт элементов в списке, вывод всех элементов списка.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


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

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

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

MusicalComposition *createMusicalCompositionList(char **array_names, char **ar-
ray_authors, int *array_years, int n) {
    MusicalComposition *head = createMusicalComposition(array_names[0], ar-
ray_authors[0], array_years[0]);
    MusicalComposition *current = NULL;
    for (int i = 1; i < n; i++) {
        current = createMusicalComposition(array_names[i], array_authors[i], ar-
ray_years[i]);
        push(head, current);
    }
    return head;
}

void removeEl(MusicalComposition *head, char *name_for_remove) {
    MusicalComposition *current = head;
    while (current != NULL) {
        if (!strcmp(current->name, name_for_remove)) {
            if (current->previous == NULL) {
                //current->name = current->next->name;
                //current->author = current->next->author;
                //current->year = current->next->year;
                //current = current->next;
                //current->next->previous = current->previous;
                //current->previous->next = current->next;
```

```
                //current = current->next;
                current->previous = current->next->previous;
                current = current->next;
                free(head);
            } else {
                if (current->next == NULL) {
                    current->previous->next = NULL;
                    current = current->previous;
                } else {
                    current->next->previous = current->previous;
                    current->previous->next = current->next;
                    current = current->next;
                }
            }
        } else {
            current = current->next;
        }
    }
}

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

void print_names(MusicalComposition *head) {
    while (head) {
        puts(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 = createMusicalComposi-
tion(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;

}
```