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 _____ Афанасьев Н. С.

Преподаватели _____ Берленко Т. А.

Санкт-Петербург

2021

Цель работы.

Изучение работы со структурами и линейными списками в языке С.

Задание.

Создайте двунаправленный список музыкальных композиций MusicalComposition и api (application programming interface - в данном случае набор функций) для работы со списком.

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

Функцию main менять не нужно.

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 *prev* 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 *prev* (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. Сначала происходит поиск этого элемента, через цикл *while*. Далее, если это первый элемент списка, то происходит копирование данных из второго элемента в первый, затем второй элемент удаляется путём изменения полей *next* и *after* предыдущего и следующего элемента соответственно (мы не можем удалить первый элемент напрямую, так как он является началом списка); если это последний элемент, то в предпоследнем элементе в качестве следующего элемента указывается *NULL*; в остальных случаях элемент удаляется так, как было описано ранее. В конце освобождается память под удалённым элементом.

Функция *int* **count***(MusicalComposition* head)* и *void* **print_names***(MusicalComposition* head)* выполняют подсчёт элементов в списке и их вывод соответственно через цикл *while*.

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 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	Верно

Выводы.

Был изучена работа со структурами и линейными списками в языке С.

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

```
#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* prev;
} MusicalComposition;

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

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]);
    MusicalComposition* prev = head;
    for(int i = 1; i < n; i++){
        MusicalComposition*                  current                  =
createMusicalComposition(array_names[i],             array_authors[i],
array_years[i]);
        prev->next = current;
        current->prev = prev;
        prev = current;
    }
    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)) head = head->next;
    if(head->prev == NULL){
        head = head->next;
        head->prev->name = head->name;
```

```
        head->prev->author = head->author;
        head->prev->year = head->year;
        head->prev->next = head->next;
    }
    else if(head->next == NULL) head->prev->next = NULL;
    else{
        head->prev->next = head->next;
        head->next->prev = head->prev;
    }
    free(head);
};

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               =
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;

}
```