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: Структуры данных, линейные списки.

Студент гр. 0382 Осинкин Е.А.

Преподаватель Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Научиться работать со структурами и линейными списками на языке СИ.

Задание.

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface - в данном слу- чае 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.

Список - некоторый упорядоченный набор элементов любой природы. Линейный однонаправленный (односвязный) список - список, каждый элемент которого хранит помимо значения указатель на следующий элемент. В последнем элементе указатель на следующий элемент равен NULL (константа нулевого указателя).

Выполнение работы.

Struct MusicalComposition.

С помощью оператора *typedef* для удобства использования структуры *MusicalComposition* был определен одноименный тип данных. Структура имеет 5 полей:

- *char name[80]* – название композиции;
- *char author[80]* – автор композиции;
- *int year* – год создания;
- *struct MusicalComposition *next* – содержит указатель на следующий элемент списка;
- *struct MusicalComposition *prev* – содержит указатель на предыдущий элемент списка;

У первого элемента списка поле *prev* и у последнего элемента списка поле *next* содержат нулевой указатель NULL.

Функции:

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

Функция для создания элемента списка, принимает на вход название композиции, ее автора и год создания. С помощью функции *malloc*выделяется динамическая память для элемента типа *MusicalComposition,* указатель на который записывается в переменную *node*. Далее заполняются поля структуры соответствующими данными, и функция возвращает указатель на новую композицию.

- **MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)**

Функция создает список музыкальных композиций, принимает на вход массивы с названиями композиций, их авторов и годов создания, а также целое число – длину этих массивов. С помощью функции *createMusicalComposition* создается первый элемент списка *head* типа *MusicalComposition*. Указатель на него присваивается переменной *tmp*. Далее с помощью этой же функции и цикла *for* создаются новые элементы списка, который связаны между собой указателями на следующий и предыдущий элементы. Функция возвращает указатель на первый элемент списка.

- **void push(MusicalComposition* head, MusicalComposition* element)**

Функция добавляет элемент в конец списка, на вход принимает указатель на первый элемент списка и указатель на элемент, который нужно добавить. С помощью цикла *while* находится последний элемент списка и его полю *next* присваивается указатель на новый. Полю *prev* нового элемента присваивается указатель на последний элемент списка, а полю *next* – нулевой указатель. Функция ничего не возвращает.

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

Функция принимает на вход указатель на первый элемент списка и название композиции *name_for_remove* и удаляет из списка элемент, у которого значение *name* равно значению *name_for_remove***.** С помощью цикла *while* и функции *strcmp* поле *name* каждого элемента списка сравнивается с *name_for_remove*. Если их значения одинаковые, элемент удаляется из списка, с помощью изменений указателей предыдущего и следующего элементов. Функция ничего не возвращает.

- **int count(MusicalComposition* head)**

Функция принимает на вход указатель на первый элемент списка и возвращает количество элементов списка. С помощью цикла *while* происходит перебор всех элементов списка и находится их количество, записанное в переменную *count*.

- **void print_names(MusicalComposition* head)**

Функция принимает на вход указатель на первый элемент списка и печатает названия всех композиций.

Разработанный программный код см. в приложении А

Тестирование.

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 верно.

Выводы.

Были изучена структуры и линейные списки языка СИ.

Разработана программа, которая создает двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface - в данном случае набор функций*) для работы со списком.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

// Описание структуры MusicalComposition
typedef struct MusicalComposition {
   char name[80];
   char author[80];
   int year;
   struct MusicalComposition *next;
   struct MusicalComposition *prev;
}MusicalComposition;

// Создание структуры MusicalComposition
MusicalComposition* createMusicalComposition(char* name, char*
author, int year) {
   MusicalComposition*             node             =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
   node->next = NULL;
   node->prev = NULL;
   node->year = year;
   strcpy(node->name, name);
   strcpy(node->author, author);
   return node;
}

// Функции для работы со списком MusicalComposition
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* tmp = head;
   for (int i = 1; i < n; i++) {
         MusicalComposition* node =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
         tmp->next = node;
         node->prev = tmp;
         tmp = node;
   }
   return head;
}

void push(MusicalComposition* head, MusicalComposition* element) {
   while (head->next != NULL) {
         head = head->next;
```

```
   }
   head->next = element;
   element->prev = head;
   element->next = NULL;
}

void removeEl(MusicalComposition* head, char* name_for_remove) {
   while (head) {
         if (strcmp(head->name, name_for_remove) == 0) {
               head->next->prev = head->prev;
               head->prev->next = head->next;
         }
         head = head->next;
   }
}

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

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;

}
```