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. 1304 _____ Шаврин А.П

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Изучить, научиться создавать и применять на практике линейные списки.

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.

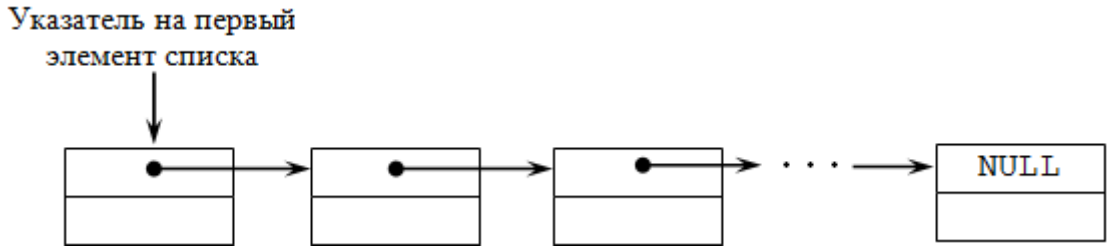
- void push(MusicalComposition* head, MusicalComposition* element); // добавляет **element** в конец списка **musical_composition_list**
- void removeEl (MusicalComposition* head, char* name_for_remove); // удаляет элемент **element** списка, у которого значение **name** равно значению **name_for_remove**
- 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.

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.

1. Создана 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.

1. Создана функция для создания элемента списка (тип элемента *MusicalComposition*):

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

1. Создана функция создания списка музыкальных композиций *MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n);*

2. Создана функция добавления элемента списка в конец

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

3. Создана функция удаления элемента из списка

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

4. Создана функция подсчета кол-ва элементов

   int count(MusicalComposition head)*

5. Создана функция вывода названий композиций

   void print_names(MusicalComposition head);*

   Разработанный программный код см. в приложении А.

   Результаты тестирования см. в приложении Б.

   Выводы.

В ходе работы были изучены основные методы работы со списками, реализован двунаправленный список, созданы необходимые структуры и функции (для создания и работы со списком).

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

// Описание структуры MusicalComposition

typedef struct MusicalComposition{
   char name[80];
   char author[80];
   int year;

   struct MusicalComposition *next;
   struct MusicalComposition *previous;
} MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* autor,int year);

// Функции для работы со списком MusicalComposition

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);

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;

}

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

    return element;
}

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

```
    MusicalComposition* current;
    MusicalComposition* previous;
    MusicalComposition* head;

        head = createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
    previous = head;

    for (int i = 1; i < n; i++){
        current = createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        previous->next = current;
        current->previous = previous;
        previous = current;
    }

    return head;
}

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

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

```
            tmp = head -> next -> next;
            free(head -> next);
            head -> next = tmp;
            head -> next -> previous = head;
        }else
            head = head->next;
    }
}

int count(MusicalComposition* head){
    int count = 1;
    while (head -> next){
        count++;
        head = head->next;
    }
    return count;
}

void print_names(MusicalComposition* head){
    do{
        printf("%s\n", head -> name);
        head = head->next;
    }while (head);
}
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Таблица Б.2 - Примеры тестовых случаев

№ 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 Author- ity 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.
2.	5 In the Army Now Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Seek and Destroy Metallica 1982 Wicked Game Chris Isaak 1989 Points of Author- ity Linkin Park 2000 Sonne Rammstein 2001 Points of	In the Army Now Status Quo 1986 5 6 In the Army Now Mixed Emotions Seek and Destroy Wicked Game Sonne 5	Программа работает корректно.

	Authority		