**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МОЭВМ**

# ОТЧЕТ

**по лабораторной работе №4**

**по дисциплине «Программирование»**

**Тема: Линейные списки**

Студентка гр. 7381 _____ Алясова А.Н.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

**Цель работы:**

Познакомиться со структурой данных: двусвязный список.
Изучить её реализацию, расположение в динамической памяти, применение.
Реализовывать основные функции для работы со списком (конструктор, добавление/удаление элемента и т.д.).

**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:

**Структура** – это одна или несколько переменных (возможно, различных типов), которые для удобства работы с ними сгруппированы под одним именем.

Объявление структуры:

```
struct {
    int x;
    int y;
};
```

**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). Для использования NULL необходимо подключить библиотеку **stddef.h.**

С помощью оператора **typedef** можно изменить имя типа.

Синтаксис использования оператора:

**typedef  struct** {*определения_элементов*}
*обозначение_структурного_типа;*

Для создания линейного списка необходимо создать структуру, содержащую хотя-бы один указатель (на следующий элемент списка) в качестве поля, а затем связать между собой различные экземпляры структуры, используя этот(и) указатель(и).

## Вывод:

В данной лабораторной работе была изучена структура список (односвязный и двусвязный). Закреплены знания по работе с указателями.  Были реализованы основные функции для работы со списком: выделение памяти для элемента списка, удаление элемента по ключу, вставка элемента в конец списка, подсчет количества элементов списка, перебор элементов списка. Были освоены константа нулевого указателя и оператор **typedef**.

**Исходный код программы:**

**Makifile:**

```
all: main.c
	gcc main.o
main.o: main.c
	gcc –c main.c
```

**main.c**

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
// Описание структуры MusicalComposition
// Создание структуры MusicalComposition
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* element =
(MusicalComposition*)malloc(sizeof(MusicalComposition));

    element->name=(char*)malloc(sizeof(char)*81);
    element->author=(char*)malloc(sizeof(char)*81);

    strcpy(element->name , name );
    strcpy(element->author, author);
    element->year = year;
    element->next = NULL;
    element->prev = NULL;

    return element;
}

void push(MusicalComposition* head, MusicalComposition* element)
{
    MusicalComposition* two_element =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    if ( head->next == NULL )
    {
```

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

    two_element = head->next;

    while (two_element->next)
    {
        two_element = two_element->next;
    }

    element->next = NULL;
    element->prev = two_element;
    two_element->next = element;

}


MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years, int n)
{
    if ( n == 0 )
       return NULL;

    MusicalComposition* head  =
(MusicalComposition*)malloc(sizeof(MusicalComposition));

    head->name = (char*)malloc(sizeof(char)*81);
    head->author = (char*)malloc(sizeof(char)*81);

    strcpy(head->name , array_names[0]);
    strcpy(head->author , array_authors[0]);
    head->year = array_years[0];

    MusicalComposition* two_element;//не выделяю память malloc

    for ( int i = 1; i < n ; i++ )
    {
        two_element  = createMusicalComposition( array_names[i] ,
array_authors[i] , array_years[i] );
         push(head , two_element);
    }
```

```
        return head;
}


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


int count(MusicalComposition* head){
   int i=1;
   MusicalComposition* two_element =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
        two_element = head->next;
        while (two_element != NULL)
        {
                two_element = two_element->next;
                i++;

        }
        return i;
   }

void print_names(MusicalComposition* head){
   do{
      puts(head->name);
      head = head->next;
}
   while(head != NULL);
}

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;   //Функция strstr() возвращает указатель на первое вхождение в строку, на которую указывает name, строки, указанной str2 (исключая завершающий нулевой символ). Если совпадений не обнаружено, возвращается NULL.
        (*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;

}
```