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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

ТЕМА: «Структуры данных, линейные списки»

Студент гр. 7381 _____ Павлов А.П.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

Цель работы

Изучить структуры данных, линейные списки и работу с ними; закрепить, полученные данные на практике.

Создать 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. 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

Основные теоретические положения

Структура- совокупность переменных, объединенных под одним именем.
Синтаксис объявления структуры

```
 struct <имя> {<тип1> <поле1>;
                <тип2> <поле2>;
                    ...
               <типN> <полеN>;
};
```

Линейный список-список, каждый элемент которого хранит помимо значения указатель на следующий элемент. В последнем элементе указатель на следующий элемент равен NULL (константа нулевого указателя). Чтобы использовать NULL, необходимо подключить #include <stddef.h>

Чтобы не писать каждый раз "struct Node", воспользуемся оператором typedef. Стандартный синтаксис использования: typedef <type> <name>; где type - любой тип name - новое имя типа (при этом можно использовать и старое имя) Вывод В ходе лабораторной работы были освоены структуры данных, линейные списки и работа с ними.

Вывод

В ходе выполнения лабораторной были освоены структуры данных, линейные списки и работа с ними.

Исходный код

Файл main.c

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


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


// Создание структуры MusicalComposition

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

```
    {
     MusicalComposition*
my_music=(MusicalComposition*)malloc(sizeof(MusicalComposition));
     my_music->name = name;
     my_music->author = author;
     my_music->year=year;
     my_music->next=NULL;
     my_music->prev=NULL;
     return my_music;
     }
// Функции для работы со списком MusicalComposition

MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years, int n)
{
    int i;
     MusicalComposition* current, *prev = NULL;
     MusicalComposition* head = NULL;

   for(i=0; i < n; i++){
     MusicalComposition* tmp = (MusicalComposition*)
malloc(sizeof(MusicalComposition));
     if (head == NULL)
        head = tmp;
     else
        prev->next = tmp;

     tmp->next = NULL;
     tmp->name = array_names[i];
     tmp->author = array_authors[i];
     tmp->year = array_years[i];
     tmp->prev = prev;
     prev = tmp;
   }
   return head;
}
void push(MusicalComposition* head, MusicalComposition* element){
    if (head == NULL)
     head = element;
   else {
     for(;head->next != NULL; head = head->next);
     head->next = element;
     }
}
void removeEl(MusicalComposition* head, char* name_for_remove){
```

```
        MusicalComposition *tmp ;
        tmp = head;

        while (tmp)
        {
              if(strcmp(tmp->name, name_for_remove) == 0)
              {
                    tmp->next->prev = tmp->prev;
                    tmp->prev->next = tmp->next;
                    free(tmp);
              }
              tmp = tmp->next;
        }
 }

int count(MusicalComposition* head){

         int i = 0;
         while(head){
           i++;
           head = head->next;
         }
         return i;
}


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;

}
```

Файл Makefile

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