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

Студент гр. 7381 _____ Смирнов М.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

Цель работы:

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, 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.

Основные теоретические положения.

1. Линейный однонаправленный список.

Список – некий упорядоченный набор элементов одной природы. Линейный однонаправленный (односвязный) список – список, каждый элемент которого хранит помимо значения указатель на следующий элемент. В последнем элементе указатель ан следующий элемент равен *NULL* (константа нулевого указателя). Чтобы использовать *NULL*, необходимо подключить заголовочный файл . Чтобы не писать каждый раз *struct MusicalComposition*, можно воспользоваться оператором *typedef*.

Стандартный синтаксис использования:
typedef ;

type – любой тип *name* – новое имя типа (при этом можно использовать и старое имя).

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*
addmusic=(MusicalComposition*)malloc(sizeof(MusicalComposition));
   addmusic->name = name;
   addmusic->author = author;
   addmusic->year=year;
   addmusic->next=NULL;
   addmusic->prev=NULL;
   return addmusic;
   }


// Функции для работы со списком MusicalComposition

void push(MusicalComposition* head, MusicalComposition* element)
   {
      MusicalComposition* tp = (MusicalComposition*)malloc(sizeof(MusicalComposition));
      if ( head->next == NULL )
      {
      element->next = NULL;
      element->prev = head;
      head->next = element;
      return;
      }
      tp = head->next;

      while (tp->next)
      {
      tp = tp->next;
      }
      element->next = NULL;
      element->prev = tp;
      tp->next = element;
      return;
   }

MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years, int n)
   {
      int g;
      if (n==0) return NULL;
         MusicalComposition
*head=createMusicalComposition(array_names[0],array_authors[0], array_years[0]);
      MusicalComposition *tp=(MusicalComposition*)malloc(sizeof(MusicalComposition));
                              for(g=1;g<n;g++)
               {

tp=createMusicalComposition(array_names[g],array_authors[g],array_years[g]);
```

```
                push(head,tp);
                tp=head;
            }

                return tp;
    }
void removeEl(MusicalComposition* head, char* name_for_remove)
                {
                if(head== NULL) return;
                MusicalComposition *tp=head;

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

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

int count(MusicalComposition* head)
    {
      int n=1;
      MusicalComposition* tp=(MusicalComposition*)malloc(sizeof(MusicalComposition));
      tp = head->next;
      while (tp)
      {
            tp = tp->next;
            n++;

      }
      return n;
   }



void print_names(MusicalComposition* head)
        {
       if(head== NULL) return;
       MusicalComposition *tp=head;
        do {
```

```
            printf("%s\n",tp->name);
            tp = tp->next;
        } while (tp !=NULL);
         }


int main(){
    int length; int i;
    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 ( 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 ( i=0;i<length;i++)
        {
        free(names[i]);
        free(authors[i]);
    }
    free(names);
    free(authors);
    free(years);

    return 0;
}
```

Вывод: В процессе выполнения лабораторной работы были получены знания о структуре список (односвязный и двусвязный). Кроме того был освоен оператор *typedef*. Полученные знания поспособствовали созданию программы, выполняющей добавление, удаление информации в(из) список(ка).