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. 6304 Курков Д. В.

Преподаватель Берленко Т. А.

Санкт-Петербург

2016

Цель работы:

Освоить приемы работы с линейными списками.

Задание:

Создайте двунаправленный список музыкальных композиций MusicalComposition и api (*application programming interface - 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

Ход работы.

1. Создание структуры элементов списка

```
typedef struct MusicalComposition
{
    char* name;
    char* author;
    int year;
    struct MusicalComposition *next;
}MusicalComposition;
```

2. Функции для работы со списком

1. Создание нового элемента

```
MusicalComposition* createMusicalComposition (char* name, char* author, int year)
{
    MusicalComposition* composition =
    (MusicalComposition*)malloc(sizeof(MusicalComposition));
    composition->name = name;
    composition->author = author;
    composition->year = year;
    composition->next = NULL;
    return composition;
}
```

2. Создание нового списка

```
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++)
    {
        tmp->next = createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        tmp = tmp->next;
    }
    return head;
}
```

3. Добавление нового элемента в список

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

4. Удаление элемента из списка

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

		}
			bef = bef->next;
			tmp = tmp->next;
	}
}
```

5. Подсчет элементов в списке

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

6. Вывод имен элементов списка.

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

3. Функция main

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

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);

        return 0;
}
```

4. Работы программы проверена на корректность с помощью предложенных тестов.

5. Исходный код программы размещен в репозитории группы.

Вывод: *В ходе лабораторной работы была создана программа, отвечающая требованию задания, получены навыки работы с линейными списками.*