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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «программирование»

Тема: «Структруы данных, линейные списки»

Студент гр. 7381 _____ Ильясов А. В.
Преподаватель _____ Берленко Т. А.

Санкт-Петербург
2017

Цель работы:

Научиться работать с двунаправленными списками. Научиться добавлять, удалять, создавать, выводить элементы списка и выводить их количество.

Задача:

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:

Создана структура MusicalComposition с членами типов char*, char*, int и 2 члена с типом указателей на структуру, хранящих адреса следующих и предыдущих элементов.

Функции, которые необходимо написать:

- **createMusicalComposition**

Функция создает элемент списка и возвращает указатель на первый элемент списка. На вход подается 3 параметра: имя, автор и год создания.

- **Push**

Функция добавляет элемент списка в его конец. На вход подается указатель на первый элемент и элемент, который нужно добавить. Функция ничего не возвращает.

- **CreateMusicalCompositionList**

Функция создает целый список. На вход подается массив имен, названий и годов создания, а также количество элементов в списке. Если количество равно 0, то возвращается NULL. Иначе функция возвращает указатель на первый элемент списка.

- **RemoveEl**

Функция удаляет элемент списка, если значение элемента name совпало с полученной функцией строкой. На вход подается указатель на первый элемент списка и определенная строка символов. Функция ничего не возвращает.

- **Count**

Функция считает количество элементов в списке. На вход подается указатель на первый элемент списка. Функция возвращает количество элементов.

- **Print_names**

Функция выводит на экран значение члена name всех элементов. На вход подается указатель на первый элемент списка. Функция ничего не возвращает.

Вывод:

В ходе выполнения лабораторной работы были освоены навыки работы со списками, также закреплены навыки выделения динамической памяти и ее освобождения.

Исходный код:

main.c

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

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_for_push =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
	element_for_push->name = (char*)malloc(sizeof(char)*81);
	element_for_push->author = (char*)malloc(sizeof(char)*81);

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

	return element_for_push;
}

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

```
	}

	c = head->next;

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

	element->next = NULL;
	element->prev = c;
	c->next = element;
	}

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

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

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

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

	MusicalComposition* c;//не выделяю память malloc

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

	return head;
```

```
}

void removeEl(MusicalComposition* head, char* name_for_remove) {
	MusicalComposition *c ;
	c = head;

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

int count(MusicalComposition* head) {
	int i=1;

	MusicalComposition* c = (MusicalComposition*)malloc(sizeof(MusicalComposition));
	c = head->next;
	while (c) {
		c = c->next;
		i++;
	}
	return i;
}

void print_names(MusicalComposition* head) {
	MusicalComposition* c;
	c = head->next;

	printf("%s\n", head->name);
	printf("%s\n", c->name);
	while (c->next) {
```

```
            if (c->next->year != -1)
                printf("%s\n"  , c->next->name );
            c = c->next;
        }
}

int main() {
        int length;
        scanf("%d\n", &length);

        char** names = (char**)malloc(length*sizeof(char*));
        char** authors = (char**)malloc(length*sizeof(char*));
        int* years = (int*)malloc(length*sizeof(int));

        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;
}
```