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»

Студенты гр. 7381 _____ Судакова П. С.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург
2017

Цель:

Реализовать двунаправленный список MusicalComposition и основные функции для работы с ним.

Задание: Реализовать структуру MusicalComposition в которой будут присутствовать поля: `name` – название композиции, `autor` – автор композиции, `year` – год создания. Также указатели на предыдущий и следующий элементы. Функции, которые нуждно реализовать для работы с данным списком: `createMusicalComposition` - создание одного элемента списка, `createMusicalCompositionList` – сознание списка из n элементов, push – добавление элемента вконец списка, `removeEi` – удаление элементаиз списка по заданному автору, `count` – количество элементов в списке, `print_names` – напечатать все композиции списка.

Описание функции, их реализация в Приложении А:

- `MusicalComposition* createMusicalComposition(char *name, char* autor, int year)`
  Входные параметры:
  1) `char *name` – указатель типа `char` на название композиции
  2) `char* autor` – указатель типа `char` на автора композиции
  3) `int year` – переменная типа `int` в которой хранит год создания композиции

  Выходные параметры:
  1) `MusicalComposition*` - указатель на голову списка
- `MusicalComposition* createMusicalCompositionList(char **arrayNames, char** arrayAutors, int* arrayYears,int n)`
  Входные параметры:
  1) `char **arrayNames` – указатель на массив указательей типа `char` названия композиций
  2) `char **arrayAutors` – указатель на массив указательей типа `char` авторов композиций
  3) `int* arrayYears` – переменная типа `int` в которой хранятся года создания композиций

  Выходные параметры:
  2) `MusicalComposition*` - указатель на голову списка
- `void push(MusicalComposition* head, MusicalComposition* element)`
  Входные параметры:
  1) `MusicalComposition* head` – указатель на голову списка в который нужно вставить элемент
  2) `MusicalComposition* element` – указатель на вставляемый элемент

- `void removeEi(MusicalComposition* head, char* nameForRemove)`
  Входные параметры:
  1) `MusicalComposition* head` – указатель на голову списка в который нужно вставить элемент
  2) `char* nameForRemove` – указатель типа `char` на имя автора, композицию которого нужно удалить
- `int count(MusicalComposition* head)`
  Входные параметры:
  1) `MusicalComposition* head` – указатель на голову списка в который нужно вставить элемент
  Выходные параметры:
  1) `Int` – количество элементов списка
- `void print_names(MusicalComposition* head)`
  Входные параметры:
  1) `MusicalComposition* head` – указатель на голову списка в который нужно вставить элемент

Тестирование

1) Создание одного элемента и вывод его на экран и количество элементов в данном списке.

```
1
fsahsd
```

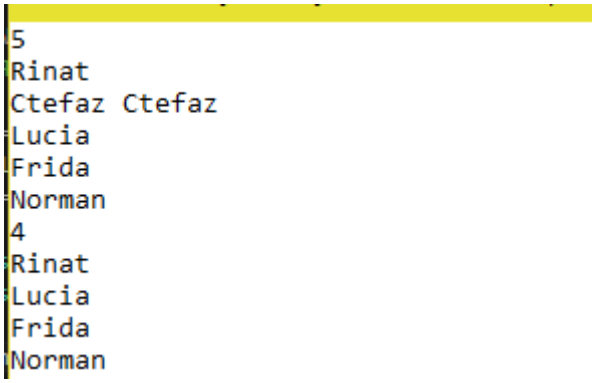
2) Создание списка элементов и вывод списка на экран и количество элементов в данном списке.

```
5
Rinat
Ctefaz Ctefaz
Lucia
Frida
Norman
```

3) Создание списка элементов и добавление одного элемента в конец. Вывод списка на экран и количество элементов в данном списке.

```
5
Rinat
Ctefaz Ctefaz
Lucia
Frida
Norman
6
Rinat
Ctefaz Ctefaz
Lucia
Frida
Norman
fsahsd
```

4) Создание списка элементов и удаление одного элемента. Вывод списка на экран и количество элементов в данном списке.



Вывод:

В результате выполнения лабораторной работы были на практике изучены и самостоятельно реализованы двунаправленные списки. Были учтены их слабые места и их преимущества. Был реализовано функционал для работы со списком.

Исходный код:

- // MusicalComposition.cpp:

```
#include <stdio.h>
#include "metodFromLr4.h"
#include <stdlib.h>

int main(){
	char **autor=malloc(5 * sizeof(char*)); autor[0] = "Rinat"; autor[1] = "Ctefaz Ctefaz";	autor[2] = "Lucia"; autor[3] = "Frida"; autor[4] = "Norman";
	int year[]={1945, 1965, 1915, 2015, 1945};
	int * years = year;
	char **names = malloc(5 * sizeof(char*)); names[0] = "Rinat"; names[1] = "Ctefaz Ctefaz";		names[2] = "Lucia"; names[3] = "Frida"; names[4] = "Norman";
	MusicalComposition *temp= createMusicalComposition("fsahsd","dhshjgf",1234);
	MusicalComposition *head = createMusicalCompositionList(names, autor, years,5);
	printf("%d",count(head));
	print_names(head);
	removeEi(head, "Ctefaz Ctefaz");
	printf("%d",count(head));
	print_names(head);
	return 0;
	}
```

- #include "metodFromLr4.h"

```
#include "string.h"
#include "stdlib.h"

MusicalComposition* createMusicalComposition(char *name, char* autor, int year){
	MusicalComposition *temp=malloc(1 * sizeof(MusicalComposition));;
	temp->autor = autor;
	temp->name = name;
	temp->year = year;
	temp->next=NULL;
	temp->prev=NULL;
	return temp;
}
MusicalComposition* createMusicalCompositionList(char **arrayNames, char** arrayAutor, int* arrayYear, int n){
	if (n >= 0){
		if (n==1) return createMusicalComposition(arrayNames[0], arrayAutor[0], arrayYear[0]);
		MusicalComposition *head = createMusicalComposition(arrayNames[0], arrayAutor[0], arrayYear[0]) ,
			*temp = createMusicalComposition(arrayNames[1], arrayAutor[1], arrayYear[1]), *current = head, *prevCurent=NULL;
		for (int i = 1; i <= n; i++){
			current->prev = prevCurent;
			if (i == n) break;
			current->next = temp;
			prevCurent = current;
			current = temp;
			temp = createMusicalComposition(arrayNames[i+1], arrayAutor[i+1], arrayYear[i+1]);

		}
		return head;

	}
	return NULL;
```

```
}
void push(MusicalComposition* head,MusicalComposition* element){
	if (head != NULL){
		for (; head->next != NULL; head = head->next);
		head->next = element;
		element->prev = head;
	}
	else{
		head = element;
	}
}
void removeEi(MusicalComposition* head, char* nameForRemove){
	struct MusicalComposition* current = head,*prev,*next;
	for (; current != NULL; current=current->next){
		prev = current->prev;
		next = current->next;
		if (strcmp(current->autor, nameForRemove) == 0){
			if (current != head){
				prev->next = next;
				next->prev = prev;
			}
			else{
				head = current->next;
			}
			free(current);

			break;

		}
	}
}
int count(MusicalComposition* head){
	int i;
	for (i = 0; head!= NULL; head = head->next, i++);
	return i;
}
void print_names(MusicalComposition* head){
	for (; head != NULL; head = head->next){
		printf("%s\n",head->name);
	}
	}
```

- ```
  #include "structComposition.h"
  MusicalComposition* createMusicalComposition(char *name, char* autor, int year);
  MusicalComposition* createMusicalCompositionList(char **arrayNames,char** arrayAutors,int*
  arrayYears,int n);
  void push(MusicalComposition* head,MusicalComposition* element);
  void removeEi(MusicalComposition* head, char* nameForRemove);
  int count(MusicalComposition* head);
  void print_names(MusicalComposition* head);
  ```

- ```
  typedef struct MusicalComposition{
  	char *name;
  	char *autor;
  	int year;
  	struct MusicalComposition* next;
  	struct MusicalComposition* prev;
  }MusicalComposition;
  ```

- MakeFile

```
all:
	gcc lab4.c metodFromLr4.h metodFromLr4.c structComposition.h -o lab4
```