**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. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

**Цель:**

Создать двунаправленный список музыкальных композиций MusicalComposition и **api** ( ***a****pplication****p****rogramming****i****nterface - в данном случае набор функций*) для работы со списком.

**Задание:**

Создайте 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.

**Ход работы:**

* 1. **Код программы**

**// Подключение библиотек**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

**// Описание структуры MusicalComposition**

typedef struct MusicalComposition{

char name[80];

char author[80];

int year;

struct MusicalComposition\* next;

struct MusicalComposition\* prev;

}MusicalComposition;

**// Создание структуры MusicalComposition**

MusicalComposition\* createMusicalComposition(char\* name, char\* author, int year){

MusicalComposition\* song =(MusicalComposition\*)malloc(sizeof(MusicalComposition));

strcpy(song->name, name);

strcpy(song->author, author);

song->year=year;

MusicalComposition\* next=NULL;

MusicalComposition\* prev=NULL;

return song;

}

**//Функция создания плейлиста**

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)

{

MusicalComposition\* t1=createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);

MusicalComposition\* head=t1;

for(int i=1;i<n;i++){

t1->next=createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

t1->next->prev=t1;

t1=t1->next;

}

return head;

}

**//Функция добавления элемента в конец списка**

void push(MusicalComposition\* head, MusicalComposition\* element)

{

while(head->next){

head=head->next;

}

head->next=element;

element->prev=head;

}

**//Функция удаления элемента**

void removeEl(MusicalComposition\* head, char\* name\_for\_remove)

{

while((strcmp(name\_for\_remove,head->name)) && (head!=NULL))

{

head=head->next;

}

if(head->next==NULL){

head->prev->next=NULL;

}

if((head->next!=NULL) && (head->prev!=NULL)){

head->prev->next=head->next;

head->next->prev=head->prev;

}

if(head->prev==NULL){

while(head->next){

strcpy(head->name, head->next->name);

strcpy(head->author, head->next->author);

head->year=head->next->year;

}

head->prev->next=NULL;

head->prev=NULL;

}

}

**//Функция подсчета количества элементов в списке**

int count(MusicalComposition\* head)

{

int i=0;

while(head){

head=head->next;

i++;

}

return i;

}

**// функция вывода имен композиций на печать**

void print\_names(MusicalComposition\* head)

{

while(head){

printf("%s\n", head->name);

head=head->next;

}

}

* 1. **Вывод:**

В ходе данной работы успешно создан двунаправленный список музыкальных композиций «MusicalComposition». Изучены основные способы работы со структурами: создание, описание и объединение в списки.