**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 |  | Григорьев И.С. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

Оглавление

[ЦЕЛЬ: 3](#_Toc468537147)

[ЗАДАНИЕ: 3](#_Toc468537148)

[Структура элемента списка (тип – struct MusicalComposition) 3](#_Toc468537149)

[Функция для создания элемента списка (тип элемента struct MusicalComposition) 3](#_Toc468537150)

[Функции для работы со списком: 3](#_Toc468537151)

[ХОД РАБОТЫ: 4](#_Toc468537152)

[Структура "Музыкальная композиция 4](#_Toc468537153)

[Функция создания и инициализации нового экземпляра структуры "MusicalComposition" 4](#_Toc468537154)

[Функция создания списка 4](#_Toc468537155)

[Добавление элемента в конец списка 5](#_Toc468537156)

[Функция удаления элемента 5](#_Toc468537157)

[Подсчет элементов списка 6](#_Toc468537158)

[Печать имен 6](#_Toc468537159)

[ВЫВОД: 6](#_Toc468537160)

Цель:

Написание программного кода для работы с линейным двунаправленными списками (вставка, удаление элементов)

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 – struct 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 struct MusicalComposition)

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

Функции для работы со списком:

* Struct 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]**).

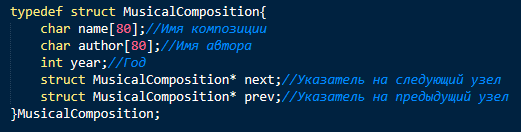
Функция возвращает указатель на первый элемент списка.

* void push (struct MusicalComposition\* head, struct MusicalComposition\* element); добавляет **element** в конец списка **musical\_composition\_list**
* void removeEl (struct MusicalComposition\* head, char\* name\_for\_remove); // удаляет элемент **element** списка, у которого значение **name** равно значению **name\_for\_remove**
* int count(struct MusicalComposition\* head); //возвращает количество элементов списка
* void print\_names(struct MusicalComposition\* head); //Выводит названия композиций

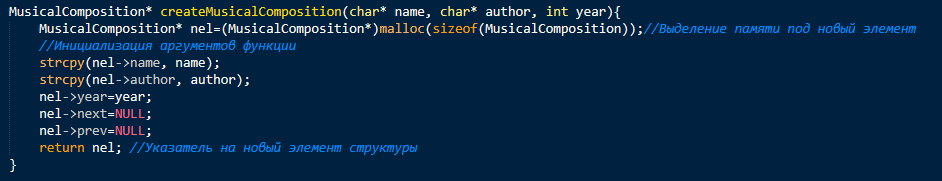
Содержание:

Структура "Музыкальная композиция

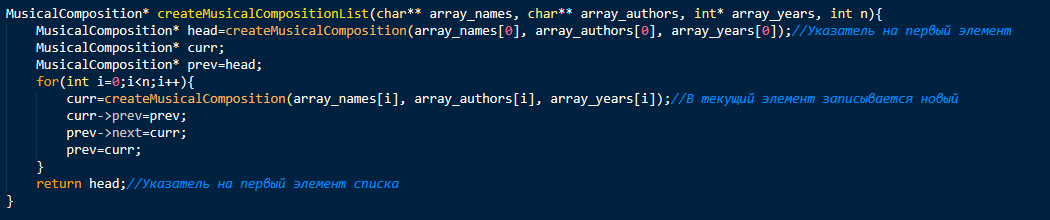
Чтобы не писать каждый раз "struct MusicalComposition", воспользуемся оператором **typedef**.



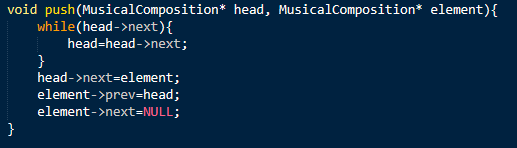
Функция создания и инициализации нового экземпляра структуры "MusicalComposition"



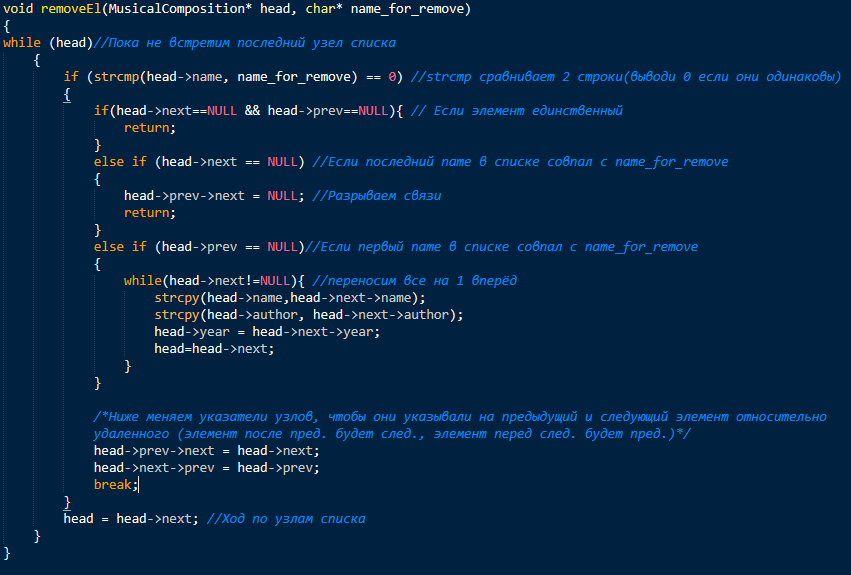
Функция создания списка



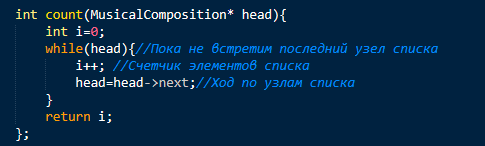
Добавление элемента в конец списка



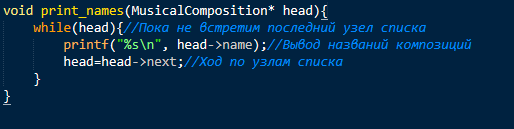
Функция удаления элемента



Подсчет элементов списка



Печать имен



**Вывод:**

Выполнив данную лабораторную работу, мы освоили и закрепили на практике написание программы с применением структур в качестве элементов сложных типов представления данных таких, как одно- и дву- направленные линейные списки, а также API для работы с ними, в частности, функции вставки, удаления, подсчета и вывода элементов списка.