**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Куликов М.Л. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

Цель:создать двунаправленный список музыкальных композиций MusicalComposition и API (набор функций) для работы со списком.

Ход работы:

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.

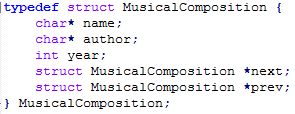
! Длина массивов array\_names, array\_authors, array\_years одинаковая и равна n, это проверять не требуется.

   Функция возвращает указатель на первый элемент списка.

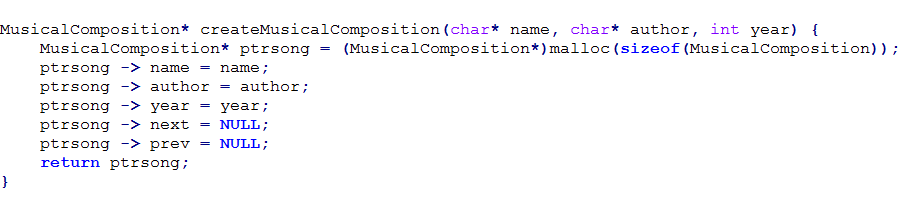
* void push(MusicalComposition\*  head, MusicalComposition\* element); // добавляет element  в конец списка musical\_composition\_list
* void removeEl (MusicalComposition\*  head, char\* name\_for\_remove); // удаляет элемент element списка, у которого значение name равно значению  name\_for\_remove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

содержание:

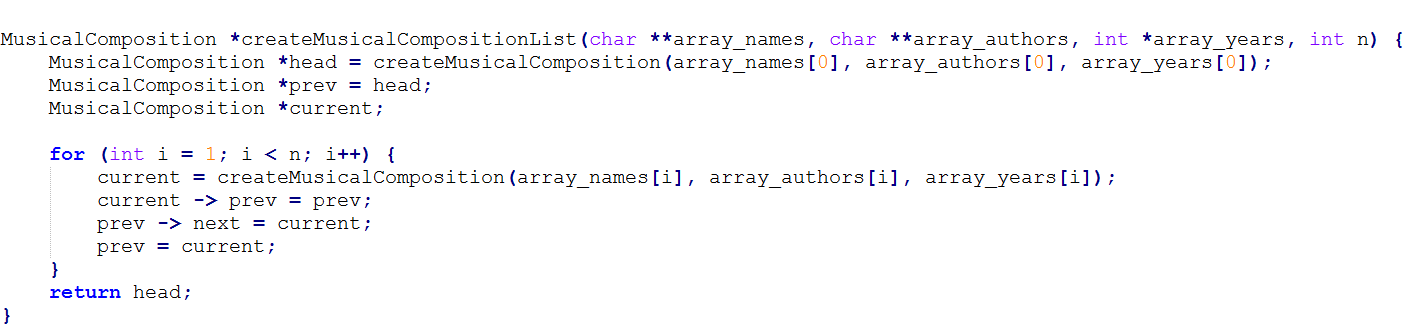
1. Структура элемента списка типa MusicalComposition.



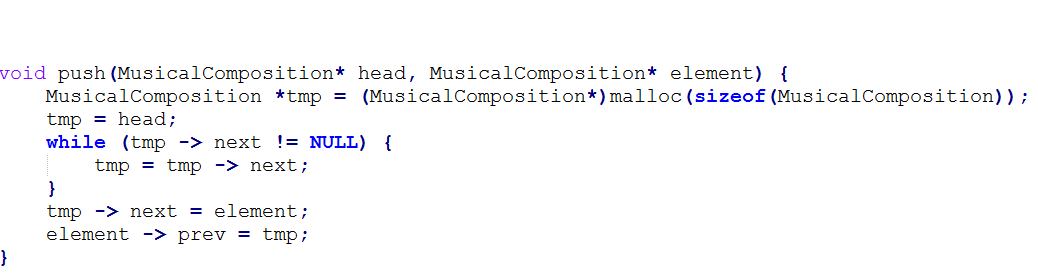
1. Функция для создания элемента списка типа MusicalComposition.



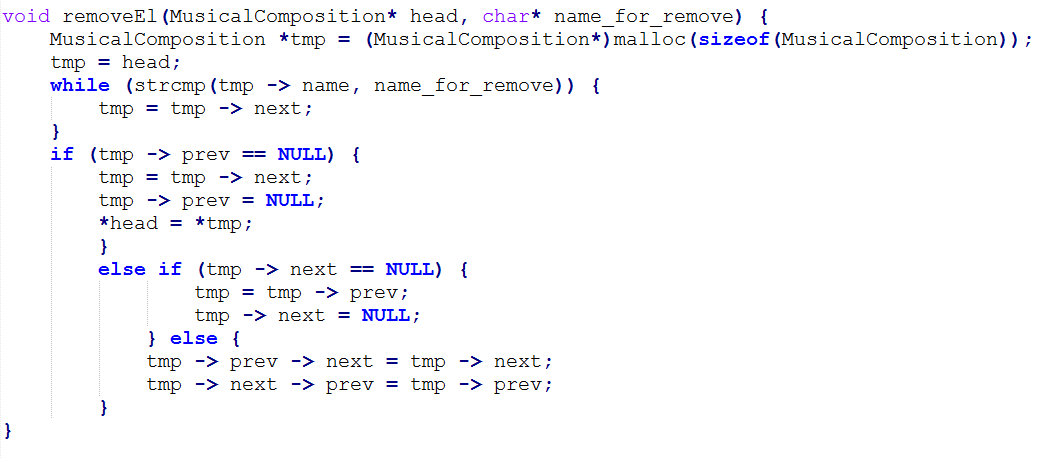
1. Функция для создания списка музыкальных композиций.



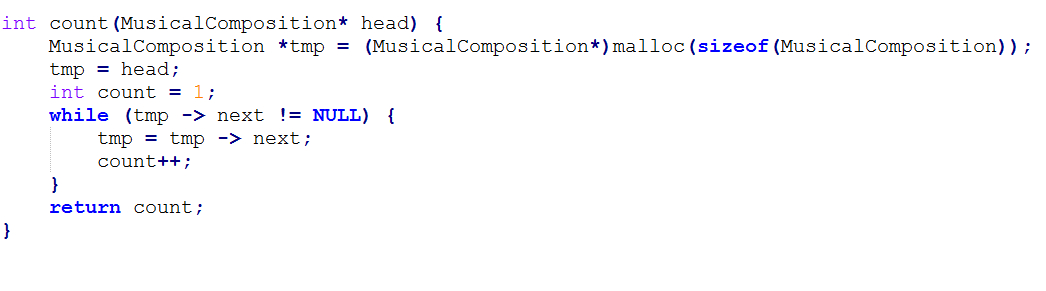
1. Функция для добавления нового элемента в конец списка.



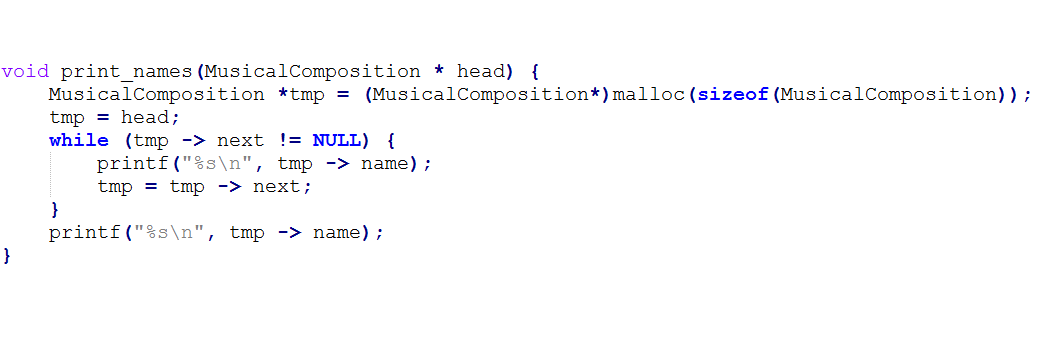
1. Функция для удаления элементa списка, с определённым названием композиции.



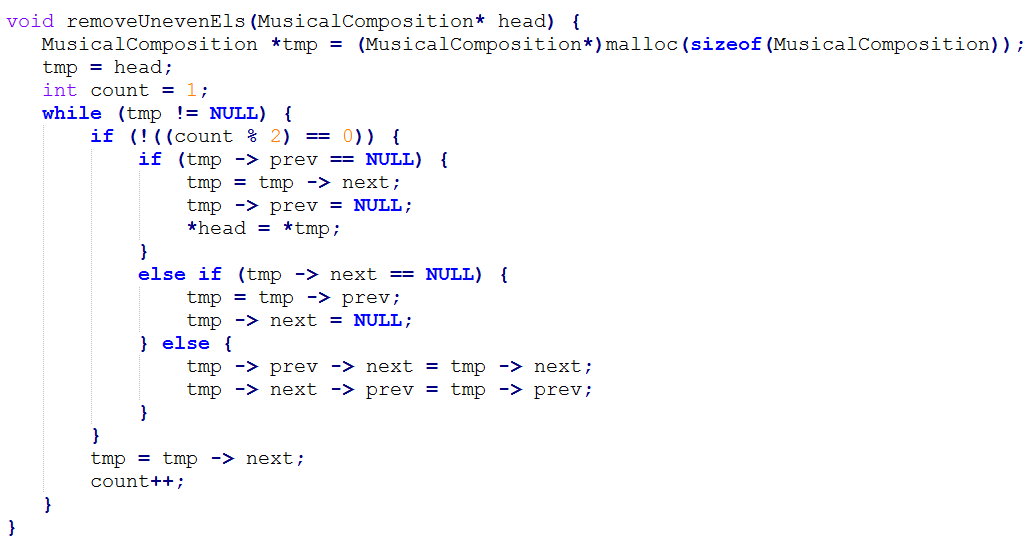
1. Функция для подсчёта количества элементов списка.



1. Функция для выведения названия композиций.



1. Функция удаления нечетных элементов списка.



ПРимеры работы программы:

вывод:

При выполнении данной лабораторной работы, были освоены и закреплены на практике навыки написания программы с применением структур в качестве элементов сложных типов представления данных таких, как двунаправленные линейные списки, а также API для работы с ними, в частности, функции вставки, удаления, подсчета и вывода элементов списка.