**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 |  | Эквас В.И. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**Аннотация**

В данной работе была создана программа на языке программирования C, которая позволяет работать с набором функций, отвечающих засписок музыкальных композиций. Для функционирования списка были созданы и описаны необходимые функции, позволяющие добавлять, удалять, сортировать и выводить элементы списка и их количество в консоль, также описана структура элемента списка. Помимо этого, была проведена работа над оптимизацией исходного кода программы для ускорения ее быстродействия и оптимального использования памяти и ресурсов клиента. Приведены примеры работы программы, а также полное описание исходного кода.

Оглавление

[Введение 1](#_Toc470220036)

[Функции для работы с двунаправленным линейным списком 2](#_Toc470220037)

[1.1. Функция для создания элемента списка. 2](#_Toc470220038)

[1.2. Функция для создания списка музыкальных композиций. 2](#_Toc470220039)

[1.3. Функция для добавления нового элемента в конец списка. 2](#_Toc470220040)

[1.4. Функция для удаления элементов списка 3](#_Toc470220041)

[1.5. Функция для подсчёта количества элементов списка. 4](#_Toc470220042)

[1.6. Функция для выведения названия композиций. 4](#_Toc470220043)

[1.7. Функция для удаления элементов, год которых меньше n. 4](#_Toc470220044)

[Примеры работы программы 5](#_Toc470220045)

[Заключение 7](#_Toc470220046)

[Список использованных источников 8](#_Toc470220047)

[Приложение А 9](#_Toc470220048)

# Введение

Необходимо 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.

Функции, необходимые для работы программы

* Создание элемента типа MusicalComposition
* Создание двунаправленного списка связанных элементов типа MusicalComposition
* Добавление элемента в конец списка
* Удаление определенного элемента (или элементов) из списка
* Подсчет количества элементов в списке
* Вывод элементов списка
* Удаление элементов, год которых меньше n

# Функции для работы с двунаправленным линейным списком

## Функция для создания элемента списка.

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

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

composition**->**name **=** name**;**

composition**->**author **=** author**;**

composition**->**year **=** year**;**

composition**->**next **=** NULL**;**

composition**->**prev **=** NULL**;**

**return** composition**;**

**}**

Данной функции передают три параметра: название (char\* name), имя автора (char\* author) и год создания композиции (int year); которые записываются в элемент после выделения памяти для структуры MusicalComposition. Функция возвращает указатель на созданный элемент списка.

## Функция для создания списка музыкальных композиций.

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

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

MusicalComposition **\***prev **=** head**;**

MusicalComposition **\***composition**;**

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

composition **=** createMusicalComposition**(**array\_names**[**i**],** array\_authors**[**i**],** array\_years**[**i**]);**

composition**->**prev **=** prev**;**

prev**->**next **=** composition**;**

prev **=** composition**;**

**}**

**return** head**;**

**}**

Функция получает количество существующих композиций и указатели на три массива: массив названий (char\*\* array\_names), массив авторов (char\*\* array\_authors) и массив годов созданий (int\* array\_years); и формирует линейный двунаправленный список.

## Функция для добавления нового элемента в конец списка.

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

MusicalComposition **\***composition **=** head**;**

**while** **(**composition**->**next**)** **{**

composition **=** composition**->**next**;**

**}**

composition**->**next **=** element**;**

element**->**prev **=** composition**;**

**}**

Данной функции передается указатель на начало линейного списка (MusicalComposition\* head) и элемент, который необходимо добавить (MusicalComposition\* element). Функция перемещается по списку, пока не доходит до последнего элемента, а затем соединяет последний и новый элементы списка посредством указателей.

## Функция для удаления элементов списка

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

**{**

MusicalComposition**\*** current **=** head**;**

**while** **(**current**)**

**{**

**if** **(**strcmp**(**current**->**name**,** name\_for\_remove**)** **==** **0)**

**{**

**if** **(**current**->**next **!=** NULL **&&** current**->**prev **!=** NULL**)**

**{**

current**->**prev**->**next **=** current**->**next**;**

current**->**next**->**prev **=** current**->**prev**;**

current **=** current**->**next**;**

**}**

**else** **if** **(**current**->**next **==** NULL **&&** current**->**prev**!=**NULL**)**

**{**

current**->**prev**->**next **=** NULL**;**

current **=** NULL**;**

**}**

**else** **if** **(**current**->**next**!=**NULL **&&** current**->**prev**==**NULL**)**

**{**

**\***head **=** **\***head**->**next**;**

head**->**prev **=** NULL**;**

current **=** head**;**

**}**

**else** **if** **(**current**->**next **==** NULL **&&** current**->**prev **==** NULL**)**

**{**

current**->**year**=-1;**

current **=** NULL**;**

**}**

**}else{**

current **=** current**->**next**;**

**}**

**}**

Функция получает указатель на начало списка (MusicalComposition\* head) и название композиции (char\* name\_for\_remove), которую нужно удалить. Перемещаясь по списку, она сверяет название композиции с переданной строкой. При совпадении значений функция связывает предыдущий и следующий элементы посредством указателей, тем самым удаляя найденный элемент из списка.

## Функция для подсчёта количества элементов списка.

**int** count**(**MusicalComposition**\*** head**)**

**{**

MusicalComposition**\*** composition **=** head**;**

**int** count **=** **0;**

**while** **(**composition **&&** composition**->**year **!=** **-1)**

**{**

count**++;**

composition **=** composition**->**next**;**

**}**

**return** count**;**

**}**

Функция получает указатель на первый элемент списка (MusicalComposition\* head), и пробегая весь список, увеличивает счётчик counter на каждом элементе списка.

## Функция для выведения названия композиций.

**void** print\_names**(**MusicalComposition**\*** head**)**

**{**

MusicalComposition**\*** composition **=** head**;**

**while** **(**composition **&&** composition**->**year**!=-1)**

**{**

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

composition **=** composition**->**next**;**

**}**

**}**

Данной функции передается указатель на первый элемент списка (MusicalComposition\* head). Перебирая все элементы, функция печатает названия каждой композиции.

## Функция для удаления элементов, год которых меньше n.

**void** del\_year**(**MusicalComposition**\*** head**,** **int** year\_for\_remove**)** **{**

MusicalComposition**\*** current **=** head**;**

**while** **(**current**)**

**{**

**if** **(**current**->**year**<**year\_for\_remove**)**

**{**

**if** **(**current**->**next **!=** NULL **&&** current**->**prev **!=** NULL**)**

**{**

current**->**prev**->**next **=** current**->**next**;**

current**->**next**->**prev **=** current**->**prev**;**

current **=** current**->**next**;**

**}**

**else** **if** **(**current**->**next **==** NULL **&&** current**->**prev **!=** NULL**)**

**{**

current**->**prev**->**next **=** NULL**;**

current **=** NULL**;**

**}**

**else** **if** **(**current**->**next **!=** NULL **&&** current**->**prev **==** NULL**)**

**{**

**\***head **=** **\***head**->**next**;**

head**->**prev **=** NULL**;**

current **=** head**;**

**}**

**else** **if** **(**current**->**next **==** NULL **&&** current**->**prev **==** NULL**)**

**{**

current**->**year **=** **-1;**

current **=** NULL**;**

**}**

**}**

**else** **{**

current **=** current**->**next**;**

**}**

**}**

**}**

Функция получается указатель на первый элемент массива (MusicalComposition\* head) и, пользуясь методом пузырька, сортирует список композиций по убыванию года создания, при этом меняя местами не элементы списка, а лишь их содержимое.

# Примеры работы программы

* Пример работы функции для добавления элемента в конец списка

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 //Кол-во элементов  Song1 //Элементы  Artist1  2000 Song2 Artist2 2001  Song3  Artist3  2002 Song4 //Элемент для пуша  Artist4  2003 | Song1 Artist1 2000 //Первый элемент  3 //Кол-во элементов до пуша  4 //Кол-во элементов после пуша  Song1 //Список после пуша  Song2  Song3  Song4  4 //Конечное кол-во элементов |

* Пример работы функции для удаления элементов с определенным именем.

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 //Кол-во элементов  Song1 //Элементы  Artist1  2000 Song2 Artist2 2001  Song3  Artist3  2002  Song3  Artist4  2003 Song3 //Элемент для удаления | Song1 Artist1 2000  4 // Кол-во элементов до удаления  Song1  Song2  2 //Кол-во элементов после удаления |

* Пример работы функции для удаления элементов, год которых меньше n.

|  |  |
| --- | --- |
| Входные данные | Выходные данный |
| 4 //Кол-во элементов  Song1 //Элементы  Artist1  2000 Song2 Artist2 2001  Song3  Artist3  2002  Song4  Artist4  2003 2002 //Минимальный год | 4 // Кол-во элементов до удаления  Song1 //Список до удаления  Song2  Song3  Song4  2 // Кол-во элементов после удаления  Song3 //Список после удаления  Song4 |

# Заключение

В ходе работы был создан двунаправленный линейный список музыкальных композиций MusicalComposition и api для работы со с ним. Также для работы программы были созданы и описаны все необходимые функции и структуры.

# Список использованных источников

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство "Невский Диалект", 2001. 352 с.

2. UNIX. Программное окружение / Керниган Б., Пайк Р. СПб.: Символ Плюс, 2003. 416 с.

# Приложение А

**Исходный код программы**

1. *#include <stdlib.h>*
2. *#include <stdio.h>*
3. *#include <string.h>*
4. **typedef** **struct** MusicalComposition
5. **{**
6. **char\*** name**;**
7. **char\*** author**;**
8. **int** year**;**
9. **struct** MusicalComposition **\***next**;**
10. **struct** MusicalComposition **\***prev**;**
11. **}** MusicalComposition**;**
12. MusicalComposition**\*** createMusicalComposition**(char\*** name**,** **char\*** author**,** **int** year**);**
13. MusicalComposition**\*** createMusicalCompositionList**(char\*\*** array\_names**,** **char\*\*** array\_authors**,** **int\*** array\_years**,** **int** n**);**
14. **void** push**(**MusicalComposition**\*** head**,** MusicalComposition**\*** element**);**
15. **void** removeEl**(**MusicalComposition**\*** head**,** **char\*** name\_for\_remove**);**
16. **int** count**(**MusicalComposition**\*** head**);**
17. **void** del\_year**(**MusicalComposition**\*** head**,** **int** year\_for\_remove**);**
18. **void** print\_names**(**MusicalComposition**\*** head**);**
19. **int** main**(){**
20. **int** length**;**
21. scanf**(**"%d\n"**,** **&**length**);**
23. **char\*\*** names **=** **(char\*\*)**malloc**(sizeof(char\*)\***length**);**
24. **char\*\*** authors **=** **(char\*\*)**malloc**(sizeof(char\*)\***length**);**
25. **int\*** years **=** **(int\*)**malloc**(sizeof(int)\***length**);**
27. **for** **(int** i **=** **0;** i**<**length**;** i**++)**
28. **{**
29. **char** name**[80];**
30. **char** author**[80];**
32. fgets**(**name**,** **80,** stdin**);**
33. fgets**(**author**,** **80,** stdin**);**
34. fscanf**(**stdin**,** "%d\n"**,** **&**years**[**i**]);**
36. **(\***strstr**(**name**,** "\n"**))** **=** **0;**
37. **(\***strstr**(**author**,** "\n"**))** **=** **0;**
39. names**[**i**]** **=** **(char\*)**malloc**(sizeof(char\*)** **\*** **(**strlen**(**name**)** **+** **1));**
40. authors**[**i**]** **=** **(char\*)**malloc**(sizeof(char\*)** **\*** **(**strlen**(**author**)** **+** **1));**
42. strcpy**(**names**[**i**],** name**);**
43. strcpy**(**authors**[**i**],** author**);**
45. **}**
46. MusicalComposition**\*** head **=** createMusicalCompositionList**(**names**,** authors**,** years**,** length**);**
47. **char** name\_for\_push**[80];**
48. **char** author\_for\_push**[80];**
49. **int** year\_for\_push**;**
50. **int** year\_for\_remove**;**
51. **char** name\_for\_remove**[80];**
53. fgets**(**name\_for\_push**,** **80,** stdin**);**
54. fgets**(**author\_for\_push**,** **80,** stdin**);**
55. fscanf**(**stdin**,** "%d\n"**,** **&**year\_for\_push**);**
56. **(\***strstr**(**name\_for\_push**,** "\n"**))** **=** **0;**
57. **(\***strstr**(**author\_for\_push**,** "\n"**))** **=** **0;**
59. MusicalComposition**\*** element\_for\_push **=** createMusicalComposition**(**name\_for\_push**,** author\_for\_push**,** year\_for\_push**);**
61. fgets**(**name\_for\_remove**,** **80,** stdin**);**
62. **(\***strstr**(**name\_for\_remove**,** "\n"**))** **=** **0;**
64. printf**(**"%s %s %d\n"**,** head**->**name**,** head**->**author**,** head**->**year**);**
65. **int** k **=** count**(**head**);**
67. printf**(**"%d\n"**,** k**);**
68. push**(**head**,** element\_for\_push**);**
70. k **=** count**(**head**);**
71. printf**(**"%d\n"**,** k**);**
73. removeEl**(**head**,** name\_for\_remove**);**
74. fscanf**(**stdin**,** "%d\n"**,** **&**year\_for\_remove**);**
75. del\_year**(**head**,** year\_for\_remove**);**
76. print\_names**(**head**);**
78. k **=** count**(**head**);**
79. printf**(**"%d\n"**,** k**);**
81. **return** **0;**
83. **}**
84. MusicalComposition**\*** createMusicalComposition**(char\*** name**,** **char\*** author**,** **int** year**)** **{**
85. MusicalComposition**\*** composition **=** **(**MusicalComposition**\*)**malloc**(sizeof(**MusicalComposition**));**
86. composition**->**name **=** name**;**
87. composition**->**author **=** author**;**
88. composition**->**year **=** year**;**
89. composition**->**next **=** NULL**;**
90. composition**->**prev **=** NULL**;**
91. **return** composition**;**
92. **}**
93. MusicalComposition**\*** createMusicalCompositionList**(char\*\*** array\_names**,** **char\*\*** array\_authors**,** **int\*** array\_years**,** **int** n**)** **{**
94. MusicalComposition **\***head **=** createMusicalComposition**(**array\_names**[0],** array\_authors**[0],** array\_years**[0]);**
95. MusicalComposition **\***prev **=** head**;**
96. MusicalComposition **\***composition**;**
97. **for** **(int** i **=** **1;** i **<** n**;** i**++)** **{**
98. composition **=** createMusicalComposition**(**array\_names**[**i**],** array\_authors**[**i**],** array\_years**[**i**]);**
99. composition**->**prev **=** prev**;**
100. prev**->**next **=** composition**;**
101. prev **=** composition**;**
102. **}**
103. **return** head**;**
104. **}**
105. **void** push**(**MusicalComposition**\*** head**,** MusicalComposition**\*** element**)** **{**
106. MusicalComposition **\***composition **=** head**;**
107. **while** **(**composition**->**next**)** **{**
108. composition **=** composition**->**next**;**
109. **}**
110. composition**->**next **=** element**;**
111. element**->**prev **=** composition**;**
112. **}**
113. **void** removeEl**(**MusicalComposition**\*** head**,** **char\*** name\_for\_remove**)**
114. **{**
115. MusicalComposition**\*** current **=** head**;**
116. **while** **(**current**)**
117. **{**
118. **if** **(**strcmp**(**current**->**name**,** name\_for\_remove**)** **==** **0)**
119. **{**
120. **if** **(**current**->**next **!=** NULL **&&** current**->**prev **!=** NULL**)**
121. **{**
122. current**->**prev**->**next **=** current**->**next**;**
123. current**->**next**->**prev **=** current**->**prev**;**
124. current **=** current**->**next**;**
125. **}**
126. **else** **if** **(**current**->**next **==** NULL **&&** current**->**prev**!=**NULL**)**
127. **{**
128. current**->**prev**->**next **=** NULL**;**
129. current **=** NULL**;**
130. **}**
131. **else** **if** **(**current**->**next**!=**NULL **&&** current**->**prev**==**NULL**)**
132. **{**
133. **\***head **=** **\***head**->**next**;**
134. head**->**prev **=** NULL**;**
135. current **=** head**;**
136. **}**
137. **else** **if** **(**current**->**next **==** NULL **&&** current**->**prev **==** NULL**)**
138. **{**
139. current**->**year**=-1;**
140. current **=** NULL**;**
141. **}**
142. **}else{**
143. current **=** current**->**next**;**
144. **}**
145. **}**
147. **}**
148. **int** count**(**MusicalComposition**\*** head**)**
149. **{**
150. MusicalComposition**\*** composition **=** head**;**
151. **int** count **=** **0;**
152. **while** **(**composition **&&** composition**->**year **!=** **-1)**
153. **{**
154. count**++;**
155. composition **=** composition**->**next**;**
156. **}**
157. **return** count**;**
158. **}**
159. **void** print\_names**(**MusicalComposition**\*** head**)**
160. **{**
161. MusicalComposition**\*** composition **=** head**;**
162. **while** **(**composition **&&** composition**->**year**!=-1)**
163. **{**
164. printf**(**"%s\n"**,** composition**->**name**);**
165. composition **=** composition**->**next**;**
166. **}**
167. **}**
168. **void** del\_year**(**MusicalComposition**\*** head**,** **int** year\_for\_remove**)** **{**
169. MusicalComposition**\*** current **=** head**;**
170. **while** **(**current**)**
171. **{**
172. **if** **(**current**->**year**<**year\_for\_remove**)**
173. **{**
174. **if** **(**current**->**next **!=** NULL **&&** current**->**prev **!=** NULL**)**
175. **{**
176. current**->**prev**->**next **=** current**->**next**;**
177. current**->**next**->**prev **=** current**->**prev**;**
178. current **=** current**->**next**;**
179. **}**
180. **else** **if** **(**current**->**next **==** NULL **&&** current**->**prev **!=** NULL**)**
181. **{**
182. current**->**prev**->**next **=** NULL**;**
183. current **=** NULL**;**
184. **}**
185. **else** **if** **(**current**->**next **!=** NULL **&&** current**->**prev **==** NULL**)**
186. **{**
187. **\***head **=** **\***head**->**next**;**
188. head**->**prev **=** NULL**;**
189. current **=** head**;**
190. **}**
191. **else** **if** **(**current**->**next **==** NULL **&&** current**->**prev **==** NULL**)**
192. **{**
193. current**->**year **=** **-1;**
194. current **=** NULL**;**
195. **}**
196. **}**
197. **else** **{**
198. current **=** current**->**next**;**
199. **}**
200. **}**
202. **}**