**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 |  | Тимофеев А.А. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

Оглавление

[Оглавление 2](#_Toc468657768)

[Цель 3](#_Toc468657769)

[Задание 3](#_Toc468657770)

[Содержание 4](#_Toc468657771)

[Вывод 8](#_Toc468657772)

# **Цель**

Написание программы с линейных списков и работа с ними

# **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]**).
* 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.

# **Содержание**

#include <stdlib.h>

|  |
| --- |
| #include <stdio.h> |
| #include <string.h> |
|  |
| struct MusicalComposition { |
| char name[80]; //Название композиции |
| char author[80]; //Автор композиции |
| int year; //Год написания |
| struct MusicalComposition\* next; //Указатель на следующий элемент |
| struct MusicalComposition\* prev; //Указатель на предыдущий элемент |
| }; |
|  |
| typedef struct MusicalComposition MusicalComposition; |
|  |
| MusicalComposition\* createMusicalComposition(char \*name, char\* author, int year) |
| { //Функция создания структуры |
| MusicalComposition\* new\_song = (MusicalComposition\*)malloc(sizeof(MusicalComposition)); //Выделение памяти |
| strcpy(new\_song->name, name); |
| strcpy(new\_song->author, author); |
| new\_song->year = year; |
| new\_song->next = NULL; |
| new\_song->prev = NULL; |
| return new\_song; |
| } |
|  |
| MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n) //Функция создания списка |
| { |
| MusicalComposition \*head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]); |
| MusicalComposition \*prev = head; |
| MusicalComposition \*current; |
| for (int i = 1; i < n; ++i) |
| { |
| current = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]); |
| current->prev = prev; |
| prev->next = current; |
| prev = current; |
| } |
| return head; |
| } |
|  |
| void push(struct MusicalComposition\* head, struct MusicalComposition\* element) |
| { //Добавление элемента в конец списка |
| while (head->next) |
| head = head->next; |
| head->next = element; |
| element->prev = head; |
| element->next = NULL; |
| } |
|  |
| void removeEl(struct MusicalComposition\*head, char\*name\_for\_remove) |
| { |
| MusicalComposition \*curr = head; |
| while (curr->next) |
| { |
| if (strcmp(curr->name, name\_for\_remove) == 0) |
| if ((curr->prev) == 0) |
| { |
| MusicalComposition \*curr\_2 = curr->next; |
| strcpy(curr->name, curr\_2->name); |
| strcpy(curr->author, curr\_2->author); |
| curr->year = curr\_2->year; |
| curr\_2->next->prev = curr; |
| curr->next = curr\_2->next; |
| free(curr\_2); |
| } |
| else |
| { |
| curr->prev->next = curr->next; |
| curr->next->prev = curr->prev; |
| free(curr); |
| } |
| curr = curr->next; |
| } |
| if ((strcmp(curr->name, name\_for\_remove) == 0) && (curr->next == 0)) |
| { |
| curr->prev->next = NULL; |
| free(curr); |
| } |
| } |
|  |
| int count(struct MusicalComposition\*head) //Функция подсчета элементов |
| { |
| int count = 0; |
| while (head) |
| { |
| count++; |
| head = head->next; |
| } |
| return count; |
| } |
|  |
| void print\_names(MusicalComposition \*head) //Функция печати имен |
| { |
| MusicalComposition \*curr = head; |
| while (curr->next) |
| { |
| printf("%s\n", curr->name); |
| curr = curr->next; |
| } |
| printf("%s\n", curr->name); |
| } |
|  |
| int main() //Основная функция main |
| { |
| int length; |
| scanf("%d\n", &length); |
|  |
| char \*\*names = (char\*\*)malloc(sizeof(char\*)\*length); |
| char \*\*authors = (char\*\*)malloc(sizeof(char\*)\*length); |
| int \*years = (int\*)malloc(sizeof(int)\*length); |
|  |
| for (int i = 0; i < length; ++i) |
| { |
| char name[80]; |
| char author[80]; |
|  |
| fgets(name, 80, stdin); |
| fgets(author, 80, stdin); |
| fscanf(stdin, "%d\n", &years[i]); |
|  |
| (\*strstr(name, "\n")) = 0; |
| (\*strstr(author, "\n")) = 0; |
|  |
| names[i] = (char\*)malloc(sizeof(char\*)\*(strlen(name) + 1)); |
| authors[i] = (char\*)malloc(sizeof(char\*)\*(strlen(author) + 1)); |
|  |
| strcpy(names[i], name); |
| strcpy(authors[i], author); |
| } |
|  |
| MusicalComposition \*head = createMusicalCompositionList(names, authors, years, length); |
| char name\_for\_push[80]; |
| char author\_for\_push[80]; |
| int year\_for\_push; |
|  |
| char name\_for\_remove[80]; |
|  |
| fgets(name\_for\_push, 80, stdin); |
| fgets(author\_for\_push, 80, stdin); |
| fscanf(stdin, "%d\n", &year\_for\_push); |
| (\*strstr(name\_for\_push, "\n")) = 0; |
| (\*strstr(author\_for\_push, "\n")) = 0; |
|  |
| MusicalComposition \*element\_for\_push = createMusicalComposition(name\_for\_push, author\_for\_push, year\_for\_push); |
|  |
| fgets(name\_for\_remove, 80, stdin); |
| (\*strstr(name\_for\_remove, "\n")) = 0; |
|  |
| printf("%s %s %d\n", head->name, head->author, head->year); |
| int k = count(head); |
|  |
| printf("%d\n", k); |
| push(head, element\_for\_push); |
|  |
| k = count(head); |
| printf("%d\n", k); |
|  |
| removeEl(head, name\_for\_remove); |
| print\_names(head); |
|  |
| k = count(head); |
| printf("%d\n", k); |
|  |
| return 0; |
| } |

# Вывод

Выполнив данную лабораторную работу, я освоил и закрепил 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.