**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Эвергрин П.С. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**для курсовой работы**

|  |  |  |
| --- | --- | --- |
| Студент: Эвергрин П.С. | | |
| Группа: 6303 | | |
| Тема работы: Линейные списки | | |
|  | | |
| Содержание пояснительной записки:   * Содержание * Введение * Описание функций, необходимых для работы с двунаправленным линейным списком * Описание функции добавления элементов после нечетного элемента списка * Примеры работы программы * Заключение * Список использованных источников * Приложение А. Исходный код программы | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 24.12.2016 | | |
| Дата сдачи курсовой работы: | | |
| Дата защиты курсовой работы: | | |
| Студент |  | Эвергрин П.С. |
| Преподаватель |  | Берленко Т.А. |

**Аннотация**

В данной работе была разработана программа на языке программирования C, позволяющая работать с набором функций, отвечающих за созданиесписка музыкальных композиций. Для работы со списком были созданы и описаны необходимые функции, позволяющие добавлять элементы в конец списка и после каждого нечетного элемента списка, удалять, выводить элементы списка и их количество на консоль, а также описана структура элемента списка. Помимо этого, была проведена работа над оптимизацией исходного кода. Приведено полное описание исходного кода.

Содержание

[Введение 5](#_Toc470196914)

[1. Функции для работы с двунаправленным линейным списком. 6](#_Toc470196915)

[1.1. Создание элемента 6](#_Toc470196916)

[1.2. Создание двунаправленного списка 6](#_Toc470196917)

[1.3. Добавление элемента в конец списка 7](#_Toc470196918)

[1.4. Удаление элементов списка 7](#_Toc470196919)

[1.5. Подсчет количества элементов в списке 8](#_Toc470196920)

[1.6. Вывод элементов на консоль 8](#_Toc470196921)

[2. Функция добавления трёх элементов после каждого нечетного элемента списка 9](#_Toc470196922)

[2.1. Создание функции 9](#_Toc470196923)

[Примеры работы программы 11](#_Toc470196924)

[Заключение 13](#_Toc470196925)

[Список использованных источников 14](#_Toc470196926)

[Приложение А 15](#_Toc470196927)

[Исходный код программы 15](#_Toc470196928)

### 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.

*typedef struct MusicalComposition*

*{*

*char\* name;*

*char\* author;*

*int year;*

*struct MusicalComposition \*next;*

*struct MusicalComposition \*prev;*

*} MusicalComposition*;

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 в консоль

## Функции для работы с двунаправленным линейным списком.

## Создание элемента

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

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

*current->name = name;*

*current->author = author;*

*current->year = year;*

*current->next = NULL;*

*current->prev = NULL;*

*return current;*

*}*

В качестве аргументов функция принимает два указателя: на название композиции (char\* name) и на ее автора (char\* author), и год создания (int year). Необходимо выделить память для структуры типа MusicalComposition, а также присвоить значения ее переменным. Переменная status будет необходима при удалении элементов из списка.

## Создание двунаправленного списка

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

*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 \*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;*

*}*

Функция принимает в качестве аргументов указатель на указатель массива названий композиций (char\*\* array\_names), указатель на указатель массива композиторов песен (char\*\* array\_authors) и указатель на массив, содержащий даты создания композиций (int\* array\_years), размер массива (int n). Создаем «голову» списка. Заполняем ее данными. Далее те же действия происходят со всеми

(n-1) элементами массива.

## Добавление элемента в конец списка

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

*MusicalComposition \*current = head;*

*while ((current->next)!=NULL) {*

*current = current->next;*

*}*

*current->next = element;*

*element->prev = current;*

*}*

Аргументами функции являются указатель на «голову» списка (MusicalComposition\* head) и указатель на элемент (MusicalComposition\* element), добавляемый в конец списка. В теле функции происходит поиск последнего элемента, затем в конец списка добавляется новый элемент.

## Удаление 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;*

*}*

*}*

*}*

На вход функции подаются указатель на «голову» списка (MusicalComposition\* head, char\* name\_for\_remove) и указатель на названия композиций, которые нужно удалить (char\* name\_for\_remove). С помощью strcmp выполняется сравнивание имени элемента списка с именем элемента, который необходимо удалить (name\_for\_remove). (head->status = 0) осуществляется, если в списке удалены все элементы, кроме последнего. Т.к. по умолчанию status = 1, то при удалении всех элементов списка переменная не будет меняться.

## Подсчет количества элементов в списке

*int count(MusicalComposition\* head)*

*{*

*MusicalComposition\* current = head;*

*int count = 0;*

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

*{*

*count++;*

*current = current->next;*

*}*

*return count;*

*}*

Функция принимает на вход указатель на «голову» списка (MusicalComposition\* head). Пока существует следующий элемент, будет происходить подсчет элементов.

## Вывод элементов на консоль

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

*{*

*MusicalComposition\* current = head;*

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

*{*

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

*current = current->next;*

*}*

*}*

В качестве аргумента функция принимает указатель на «голову» списка (MusicalComposition\* head).

Пока существует следующий элемент списка, на консоль будет выводиться название композиции.

## Функция добавления трёх элементов после каждого нечетного элемента списка

## Создание функции

*void addtolist(MusicalComposition\* head)*

*{*

*MusicalComposition\* current=head;*

*MusicalComposition\* element\_for\_push1;*

*int k=count(head);*

*for (int i = 0; i < k; i++) {*

*if (i%2==0){*

*for (int j = 0; j < 3; j++) {*

*element\_for\_push1=createMusicalComposition("One-Dance", "Drake", 2016);*

*element\_for\_push1->next=current->next;*

*element\_for\_push1->prev=current;*

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

*current->next->prev=element\_for\_push1;*

*current->next=element\_for\_push1;*

*current=current->next;*

*}*

*}*

*else{*

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

*}*

*}*

*}*

На вход функции подаются указатели на структуры (MusicalComposition\* head). Текущему элементу присваивается «голова» списка (MusicalComposition\* current = head). Также создадим временный указатель, которому тоже присвоим значение «головы» списка и указатель на следующий элемент (MusicalComposition \*next). Пока существует следующий элемент списка, проверяется условие: если элемент четный, то после него добавить 2 произвольных элемента. Если условие не выполняется для элемента (он нечетный), то переходим к следующему, увеличивая счетчик на единицу. Когда встречается последний элемент списка, он также проверяется условием, и если он четный, то после него добавляются 2 элемента.

### Примеры работы программы

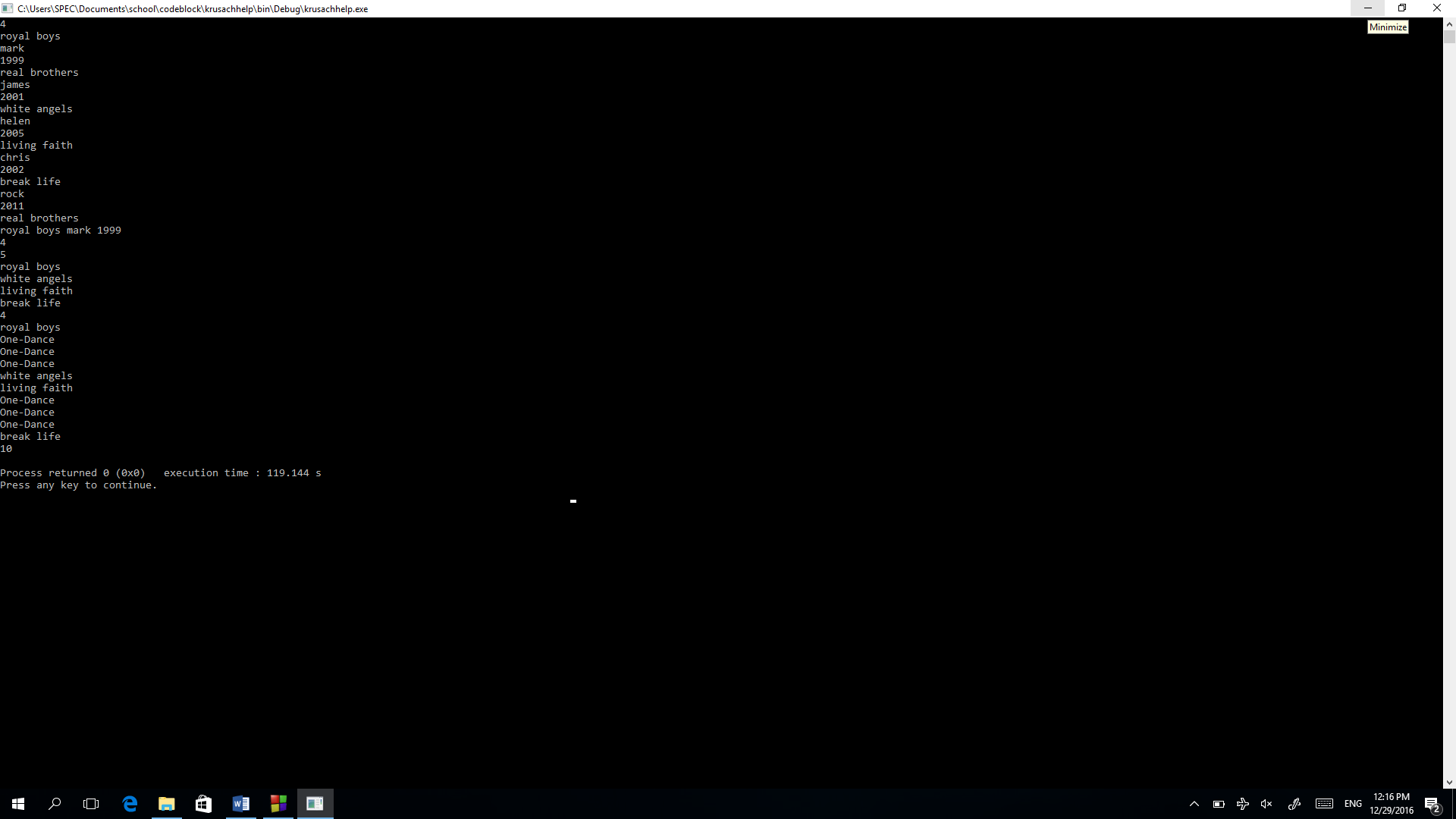
Для демонстрации всех возможностей программы используется следующий формат ввода:

* Количество элементов в списке
* Элементы списка
* Элемент для добавления в список
* Название композиции, которую нужно удалить

И вывода:

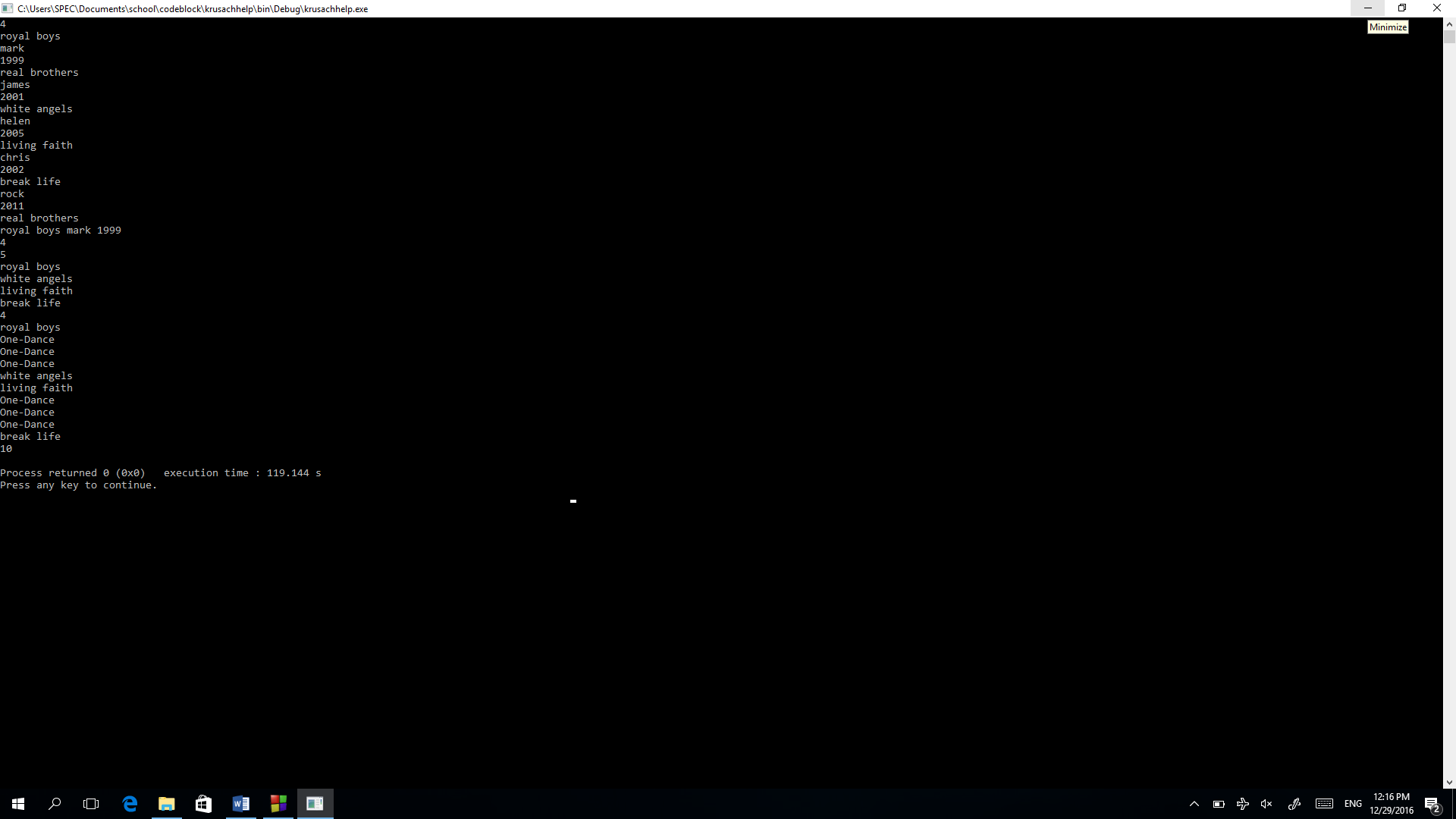
* Название, автор и год написания первого элемента списка
* Количество элементов до добавления нового
* Количество элементов после добавления
* Названия всех композиций после удаления
* Количество элементов после удаления
* Названия всех композиций после выполнения функции добавления элементов.
* Количество элементов после добавления.

ВВОД



На вход подается 4 значения, которые добавляются в список, затем подается значение, которое добавляется 5м элементом, и после него вводится название композиции, которую необходимо удалить.

ВЫВОД



После выполнения программы выводится следующее: первый элемент списка, число композиций, число композиций после добавления нового элемента, список элементов после добавления, число элементов после удаления, список и число композиций, после выполнения функции push\_double.

### Заключение

Был создан и описан двунаправленный список музыкальных композиций MusicalComposition и api для работы со списком. Также были созданы и описаны все необходимые функции и структуры для работы программы.

### 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.
3. Программирование на языке Си / Подбельский В.В., Фомин С.С. / М.: Финансы и статистика, 2004. 600 с.

## Приложение А

## Исходный код программы

*//#include <stdafx.h>*

*#include <stdlib.h>*

*#include <stdio.h>*

*#include <string.h>*

*typedef struct MusicalComposition*

*{*

*char\* name;*

*char\* author;*

*int year;*

*struct MusicalComposition \*next;*

*struct MusicalComposition \*prev;*

*} MusicalComposition;*

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

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

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

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

*int count(MusicalComposition\* head);*

*void addtolist(MusicalComposition\* head);*

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

*int main(){*

*int length;*

*printf("Length of List: ");*

*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("Name of Head-Group: ");*

*printf("%s %s %d\n", head->name, head->author, head->year);*

*int k = count(head);*

*printf("Number of Groups at first: %d\n", k);*

*push(head, element\_for\_push);*

*k = count(head);*

*printf("Number of Groups on addition: %d\n", k);*

*removeEl(head, name\_for\_remove);*

*printf("Names of groups Left:\n");*

*print\_names(head);*

*k = count(head);*

*printf("Number of Groups Left: %d\n", k);*

*addtolist(head);*

*k = count(head);*

*print\_names(head);*

*return 0;*

*}*

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

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

*current->name = name;*

*current->author = author;*

*current->year = year;*

*current->next = NULL;*

*current->prev = NULL;*

*return current;*

*}*

*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;*

*}*

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

*MusicalComposition \*current = head;*

*while (current->next) {*

*current = current->next;*

*}*

*current->next = element;*

*element->prev = current;*

*}*

*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;*

*}*

*}*

*}*

*int count(MusicalComposition\* head)*

*{*

*MusicalComposition\* current = head;*

*int count = 0;*

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

*{*

*count++;*

*current = current->next;*

*}*

*return count;*

*}*

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

*{*

*MusicalComposition\* current = head;*

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

*{*

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

*current = current->next;*

*}*

*}*

*void addtolist(MusicalComposition\* head)*

*{*

*MusicalComposition\* current=head;*

*MusicalComposition\* element\_for\_push1;*

*int k=count(head);*

*for (int i = 0; i < k; i++) {*

*if (i%2==0){*

*for (int j = 0; j < 3; j++)*

*{*

*element\_for\_push1=createMusicalComposition("One-Dance", "Drake", 2016);*

*element\_for\_push1->next=current->next;*

*element\_for\_push1->prev=current;*

*if (current->next)*

*current->next->prev=element\_for\_push1;*

*current->next=element\_for\_push1;*

*current=current->next;*

*}*

*}*

*else{*

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

*}*

*}*

*}*