**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Зубов К.А |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

## **Цель**

Написание программы с использованием линейных двунаправленных списков, используя функции для работы 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]**).

*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

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{ //описание структуры

char name[80]; //название композиции

char author[80]; //имя автора

int year; //год

struct MusicalComposition \*next; //указатель на следующий элемент списка

struct MusicalComposition \*prev; // указатель на предыдущий элемент

};

typedef struct MusicalComposition MusicalComposition; //изменение названия типа

//прототипы функций

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 print\_names(MusicalComposition\* head);

int 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;

}

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

MusicalComposition\* soundtrack = (MusicalComposition\*)malloc(sizeof(MusicalComposition)); //Выделение памяти

//инициализация полей

strcpy(soundtrack->name, name);

strcpy(soundtrack->author, author);

soundtrack->year = year;

soundtrack->next = NULL;

soundtrack->prev = NULL;

return soundtrack;

}

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n) //Функция для создания списка композиций

{

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

MusicalComposition \*prev = head; // элемент для движения по списку

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(MusicalComposition\* head, MusicalComposition\* element){ //добавление элемента

while(head->next)

head = head->next;

head->next = element;

element->prev = head;

element->next = NULL;

}

void removeEl(MusicalComposition\* head, char\* name\_for\_remove){ //удаление элемента

while(head){

if(strcmp(head->name, name\_for\_remove) == 0){ //сравнение поля имени с заданным именем

if(head->prev == 0){ //если элемент в начале

head->next->prev = NULL;

return;

}

else if (head->next == NULL){ //если элемент в конце

head->prev->next = NULL;

return;

}

head->prev->next = head->next;

head->next->prev = head->prev;

break;

}

head = head->next;

}

}

int count(MusicalComposition\*head) // функиця подсчета элементов

{

int count = 0;

while (head)

{

count++;

head = head->next;

}

return count;

};

void print\_names(MusicalComposition\* head){ //вывод имен

while(head){

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

head = head->next;

}

}

**Вывод**

Выполнив данную лабораторную работу, было освоено и закреплено на практике написание программы с применением структур в качестве элементов сложных типов представления данных таких, как одно- и двунаправленные линейные списки.