**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Лабораторная работа №4

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Прозорова А. Д. |
| Преподаватель |  | Кринкин К. В. |

Санкт-Петербург

2016

**Цель работы**

Создать двунаправленный список музыкальных композиций MusicalComposition и **api** ( ***a****pplication****p****rogramming****i****nterface - в данном случае набор функций*) для работы со списком.

**Задание**

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

**Ход работы**

*1. Описание структуры MusicalComposition.*

struct MusicalComposition {

char name[80];

char author[80];

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prior;

};

typedef struct MusicalComposition MusicalComposition;

*2. Функция создания элемента списка.*

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

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

strcpy(head->name, name);

strcpy(head->author, author);

head->year = year;

head->prior=NULL;

head->next=NULL;

return head;

}

*3. Функция создания списка. На вход функция принимает массив имен, авторов и годов создания и кол-во элементов в списке. Возвращает указатель на первый элемент списка.*

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

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

MusicalComposition\* head = tmp;

int i;

for (i=1; i<n; i++)

{

tmp->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i])

tmp->next->prior = tmp;

tmp = tmp->next;

}

return head;

}

*4. Функция добавления элемента в конец списка.*

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

while(head->next){

head=head->next;

}

head->next=element;

element->prior=head;

}

*5. Функция удаления определенного элемента из списка.*

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

MusicalComposition \*tmp = head;

while (tmp->next)

{

if (strcmp(tmp->name, name\_for\_remove) == 0) {

if ((tmp->prior) == NULL) *//если это первый элемент*

{

MusicalComposition \*ntmp = tmp->next;

strcpy(tmp->name, ntmp->name);

strcpy(tmp->author, ntmp->author);

tmp->year = ntmp->year;

ntmp->next->prior = tmp;

tmp->next = ntmp->next;

free(ntmp);

}

else //*если это не первый элемент*

{

tmp->prior->next = tmp->next;

tmp->next->prior = tmp->prior;

free(tmp);

}

}

tmp = tmp->next;

}

if ((strcmp(tmp->name, name\_for\_remove) == 0) && (tmp->next == NULL)) *//если это последний элемент*

{

tmp->prior->next = NULL;

free(tmp);

}

}

*6. Функция подсчета количества элементов в списке.*

int count(MusicalComposition\* head){

int count=0;

while (head) {

count++;

head=head->next;

}

return count;

}

*7. Функция вывода названий композиций в списке.*

void print\_names(MusicalComposition\* head){

MusicalComposition \*tmp = head;

while (tmp->next){

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

tmp = tmp->next;

}

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

}

**Вывод**

В процессе работы были получены навыки создания и управления одно- и двунаправленными линейными списками. В результате работы был получен двунаправленный список музыкальных композиций MusicalComposition и  набор функций для работы со списком.

**ПРИЛОЖЕНИЕ**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

struct MusicalComposition {

char name[80];

char author[80];

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prior;

};

typedef struct MusicalComposition MusicalComposition;

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

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

strcpy(head->name, name);

strcpy(head->author, author);

head->year = year;

head->prior=NULL;

head->next=NULL;

return head;

}

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);

int i;

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

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

MusicalComposition\* head = tmp;

int i;

for (i=1; i<n; i++)

{

tmp->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

tmp->next->prior = tmp;

tmp = tmp->next;

}

return head;

}

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

while(head->next){

head=head->next;

}

head->next=element;

element->prior=head;

}

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

MusicalComposition \*tmp = head;

while (tmp->next)

{

if (strcmp(tmp->name, name\_for\_remove) == 0) {

if ((tmp->prior) == NULL)

{

MusicalComposition \*ntmp = tmp->next;

strcpy(tmp->name, ntmp->name);

strcpy(tmp->author, ntmp->author);

tmp->year = ntmp->year;

ntmp->next->prior = tmp;

tmp->next = ntmp->next;

free(ntmp);

}

else

{

tmp->prior->next = tmp->next;

tmp->next->prior = tmp->prior;

free(tmp);

}

}

tmp = tmp->next;

}

if ((strcmp(tmp->name, name\_for\_remove) == 0) && (tmp->next == NULL))

{

tmp->prior->next = NULL;

free(tmp);

}

}

int count(MusicalComposition\* head){

int count=0;

while (head) {

count++;

head=head->next;

}

return count;

}

void print\_names(MusicalComposition\* head){

MusicalComposition \*tmp = head;

while (tmp->next){

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

tmp = tmp->next;

}

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

}