**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 |  | Попов С.С. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

# Цель работы

Научиться создавать двусвязные списки и работать 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.

*Функцию main менять не нужно.*

# Содержание

Main.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition

{

char name[80];

char author[80];

int year;

struct MusicalComposition\* next;

struct MusicalComposition\* prev;

} MusicalComposition;

// Создание структуры MusicalComposition

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

{

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

strcpy(newMusicalComposition->name, name);

strcpy(newMusicalComposition->author, author);

newMusicalComposition->year = year;

newMusicalComposition->next = NULL;

newMusicalComposition->prev = NULL;

return newMusicalComposition;

}

// Функции для работы со списком MusicalComposition

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

{

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

MusicalComposition\* PrevComp = HeadComp;

int i;

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

{

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

CurrentComp->prev = PrevComp;

PrevComp->next = CurrentComp;

CurrentComp->next = NULL;

PrevComp = CurrentComp;

}

return HeadComp;

}

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

{

if (head == NULL)

{

head = element;

return;

}

MusicalComposition\* CurrentComp = head;

while (CurrentComp->next != NULL)

{

CurrentComp = CurrentComp->next;

}

CurrentComp->next = element;

element->prev = CurrentComp;

}

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

{

//According to the requirements the function can't delete the head element

if (head == NULL)

{

return;

}

MusicalComposition\* CurrentComp = head;

do

{

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

{

MusicalComposition\* prev = CurrentComp->prev;

MusicalComposition\* next = CurrentComp->next;

if (prev != NULL)

{

prev->next = CurrentComp->next;

}

if (next != NULL)

{

next->prev = CurrentComp->prev;

}

MusicalComposition\* DelComp = CurrentComp;

CurrentComp = CurrentComp->next;

free(DelComp);

}

else

{

CurrentComp = CurrentComp->next;

}

} while (CurrentComp != NULL);

}

int count(MusicalComposition\* head)

{

int count = 0;

MusicalComposition\* CurrentComp = head;

while (CurrentComp != NULL)

{

count++;

CurrentComp = CurrentComp->next;

}

return count;

}

void print\_names(MusicalComposition\* head)

{

MusicalComposition\* CurrentComp = head;

while (CurrentComp != NULL)

{

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

CurrentComp = CurrentComp->next;

}

}

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;

}

# Вывод

Мы научились создавать двусвязные списки и работать с ними