**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Иванкова В.М. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

# **Цель:**

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

# **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.

# **Ход работы:**

Код программы:

**#include <stdlib.h>**

**#include <stdio.h>**

**#include <string.h>**

**#include <stddef.h>**

// *Описание структуры MusicalComposition*

**typedef struct MusicalComposition {**

**char name[80];**

**char author[80];**

**int year;**

**struct MusicalComposition\* prev;**

**struct MusicalComposition\* next;**

**} MusicalComposition;**

// *Создание структуры MusicalComposition*

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

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

**p->prev = NULL;**

**p->next = NULL;**

**strcpy(p->name, name);**

**strcpy(p->author, author);**

**p->year = year;**

**return p;**

**}**

// *Функции для работы со списком MusicalComposition*

**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);**

// *Функция создания плейлиста*

**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;**

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

**{**

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

**tmp->next->prev = tmp;**

**tmp = tmp->next;**

**}**

**return head;**

**}**

// *Функция добавления элемента в конец списка*

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

**while (head->next!=NULL)**

**head = head->next;**

**head->next = element;**

**element->prev = head;**

**}**

// *Функция подсчета количества элементов в списке*

**int count(MusicalComposition\* head){**

**int num = 0;**

**while (head!=NULL){**

**head = head->next;**

**num++;**

**}**

**return num;**

**}**

// *Функция вывода имен композиций на печать*

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

**while (head!=NULL){**

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

**head = head->next;**

**}**

**}**

// *Функция удаления элемента*

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

**while (strcmp(head->name, name\_for\_remove)!=0){**

**head = head->next;**

**}**

**while ((strcmp(name\_for\_remove, head->name)!=0) && (head!=NULL)){**

**head = head->next;**

**}**

**if ((head->prev!=NULL)&&(head->next!=NULL)){**

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

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

**}**

**else if (head->next == NULL){**

**head->prev->next = NULL;**

**}**

**else if(head->prev == NULL){**

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

**strcpy(head->name,head->next->name);**

**strcpy(head->author, head->next->author);**

**head->year = head->next->year;**

**}**

**head->prev->next = NULL;**

**head->prev = NULL;**

**}**

**}**

**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;**

**}**

# **Вывод:**

В ходе данной работы успешно создан двунаправленный список музыкальных композиций «MusicalComposition». Изучены основные способы работы со структурами: создание, описание и объединение в списки.