**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

**Тема: Линейные списки**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Атоян М. А. |
| Преподаватель |  | Государкин Я. С. |

Санкт-Петербург

2024

## Цель работы

Написать программу на языке C с использованием двусвязного списка.

## 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.*

## Выполнение работы

Описание созданных структур:

Структура *struct MusicalComposition*:

* + *char\* name* – название композиции, до 80 символов.
  + *char\* author* – автор композиции, до 80 символов.
  + *int year* – год создания.
  + *struct MusicalComposition\* prev* – указатель на предыдущую музыкальную композицию списка.
  + *struct MusicalComposition\* next* – указатель на следующую музыкальную композицию списка.

Описание созданных функций:

* *main()* принимает на вход данные для создания и изменения списка и обрабатывает их.
* *createMusicalComposition(char\* name, char\* author, int year)* создает музыкальную композицию.
* *MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)* создает из поданных массивов данных список музыкальных композиций.
* *push(MusicalComposition\* head, MusicalComposition\* element)*: добавляет новую композицию в список.
* *void removeEl(MusicalComposition\* head, char\* name\_for\_remove)* удаляет композицию из списка, при совпадении имени композиции с удаляемым именем.
* *count(MusicalComposition\* head)* возвращает количество музыкальных композиций в списке.
* *print\_names(MusicalComposition\* head)* выводит названия композиций.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Комментарии |
| 1 | 7  Fields of Gold  Sting  1993  In the Army Now  Status Quo  1986  Mixed Emotions  The Rolling Stones  1989  Billie Jean  Michael Jackson  1983  Seek and Destroy  Metallica  1982  Wicked Game  Chris Isaak  1989  Points of Authority  Linkin Park  2000  Sonne  Rammstein  2001  Points of Authority | Fields of Gold Sting 1993  7  8  Fields of Gold  In the Army Now  Mixed Emotions  Billie Jean  Seek and Destroy  Wicked Game  Sonne  7 | Вывод соответствует ожиданиям. |

## Выводы

В ходе выполнения лабораторной работы была написана программа с использованием двусвязного списка, изучены особенности реализации списков в языке Си.

# Приложение А Исходный код программы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct MusicalComposition{

char\* name;

char\* author;

int year;

struct MusicalComposition\* prev;

struct MusicalComposition\* next;

} MusicalComposition;

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

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

composition->name = name;

composition->author = author;

composition->year = year;

return composition;

}

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

MusicalComposition\* arr = malloc(sizeof(MusicalComposition) \* n);

for (int i = 0; i < n; ++i) {

arr[i].name = array\_names[i];

arr[i].author = array\_authors[i];

arr[i].year = array\_years[i];

if (i != 0)

arr[i].prev = &arr[i-1];

if (i != (n-1))

arr[i].next = &arr[i+1];

}

arr[0].prev = NULL;

arr[n-1].next = NULL;

return arr;

}

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

MusicalComposition\* composition = head;

while (composition->next != NULL)

composition = composition->next;

composition->next = element;

element->prev = composition;

element->next = NULL;

}

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

MusicalComposition\* composition = head;

while (strcmp(composition->name, name\_for\_remove) != 0)

composition = composition->next;

if (composition->next != NULL)

composition->next->prev = composition->prev;

if (composition->prev != NULL)

composition->prev->next = composition->next;

}

int count(MusicalComposition\* head) {

MusicalComposition\* composition = head;

int count = 0;

while (composition != NULL) {

++count;

composition = composition->next;

}

return count;

}

void print\_names(MusicalComposition\* head) {

MusicalComposition\* composition = head;

while (composition != NULL) {

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

composition = composition->next;

}

}

int main(){

int length;

scanf("%d\n", &length);

char\*\* names = malloc(sizeof(char\*)\*length);

char\*\* authors = malloc(sizeof(char\*)\*length);

int\* years = 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] = malloc(sizeof(char\*) \* (strlen(name)+1));

authors[i] = 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);

for (int i = 0; i < length; ++i){

free(names[i]);

free(authors[i]);

}

free(names);

free(authors);

free(years);

return 0;

}