**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

**Тема: Лабораторная работа № 2: Линейные списки**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Никишин С.А. |
| Преподаватель |  | Государкин Я. С. |

Санкт-Петербург

2024

## Цель работы

Реализовать двунаправленный список музыкальных композиций MusicalComposition и API для работы с ним.

## 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

**main():**

* Главная функция программы.
* Считывает количество композиций (length).
* Считывает данные (название, автор, год) для каждой композиции.
* Создает список композиций.
* Добавляет новую композицию.
* Удаляет композицию по имени.
* Печатает информацию о композициях.

**getSentence():**

* Считывает текст с консоли посимвольно.

**createMusicalComposition():**

* Создает экземпляр MusicalComposition.

**createMusicalCompositionList():**

* Создает связный список экземпляров MusicalComposition из предоставленных массивов.

**push():**

* Добавляет новый элемент в конец списка.

**removeEl():**

* Удаляет элемент из списка по имени.

**count():**

* Возвращает количество элементов в списке.

**print\_names():**

* Печатает имена всех элементов в списке.

Разработанный программный код см. в приложении А.

## Выводы

В ходе выполнения лабораторной работы была поставлена задача реализовать двунаправленный список музыкальных композиций MusicalComposition и API для работы с ним.

# Приложение А Исходный код программы

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition{

char\* name;

char\* author;

int year;

struct MusicalComposition \*previous;

struct MusicalComposition \*next;

}MusicalComposition;

// Создание структуры MusicalComposition

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

MusicalComposition \*musical\_composition=malloc(sizeof(MusicalComposition));

(\*musical\_composition).name=name;

(\*musical\_composition).author=autor;

(\*musical\_composition).year=year;

return musical\_composition;

}

// Функции для работы со списком MusicalComposition

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

MusicalComposition \*linear\_list=malloc(sizeof(MusicalComposition)\*n);

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

{

linear\_list[i].name=array\_names[i];

linear\_list[i].author=array\_authors[i];

linear\_list[i].year=array\_years[i];

}

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

if (i != n - 1) linear\_list[i].next = &linear\_list[i + 1];

if (i != 0) linear\_list[i].previous = &linear\_list[i - 1];

}

linear\_list[0].previous=NULL;

linear\_list[n-1].next=NULL;

return linear\_list;

}

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

MusicalComposition \*composition=head;

while((\*composition).next!=NULL)

{

composition=(\*composition).next;

}

(\*composition).next=element;

(\*element).previous=composition;

}

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

MusicalComposition\* composition = head;

while (composition != NULL) {

if (strcmp((\*composition).name, name\_for\_remove) == 0) {

composition->next->previous = composition->previous;

composition->previous->next = composition->next;

}

composition = composition->next;

}

}

int count(MusicalComposition\* head)

{

int i=0;

MusicalComposition \*composition=head;

while(composition != NULL)

{

composition=(\*composition).next;

i++;

}

return i;

}

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 = (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);

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

free(names[i]);

free(authors[i]);

}

free(names);

free(authors);

free(years);

return 0;

}