**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9304 |  | Каменская Е.К. |
| Преподаватель |  | Чайка К.В |

Санкт-Петербург

2020

## Цель работы.

Изучить действия со структурами языка Си и на их основе создать двунаправленный линейный список.

## 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.

Структура MusicalComposition:

* char name[81] – массив символов для названия композиции с ‘\0’ на конце;
* char author[81] – массив символов для автора композиции с ‘\0’ на конце;
* int year – целое число, год записи;
* struct MusicalComposition\* prev – указатель на предыдущий элемент;
* struct MusicalComposition\* next – указатель на следующий элемент;

Функции:

**createMusicalComposition** – принимает на вход массивы символов с названием и автором и год записи. С помощью функции malloc выделяет память под структуру MusicalComposition и заполняет ее поля переданными данными, при этом полям указателей присваивается значение NULL.

**createMusicalCompositionList** – получает на вход массив названий, авторов и лет и общее число композиций. Создает указатель на структуру MusicalComposition\* head, который приравнивается первому элементу списка. Далее в цикле создаются последующие элементы. Функция возвращает указатель head.

**push** – получает указатель на голову списка и на новый элемент. Добавляет новый элемент в конец списка.

**removeEl** – с помощью функции strcmp сравнивает названия композиций с переданным названием и удаляет элемент списка, где их значения совпадают.

**count** – итерируется по всему списку и возвращает количество элементов в нем.

**print\_names** – печатает все названия композиций из списка.

**clearAll** – освобождает память, выделенную под список.

Разработанный программный код см. в приложении А.

## Выводы.

В ходе выполнения работы были изучены способы работы со структурами, реализован двусвязный список и функции для его создания, изменения и удаления.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition{

char name[81];

char author[81];

int year;

struct MusicalComposition\* prev;

struct MusicalComposition\* next;

}MusicalComposition;

// Создание структуры MusicalComposition

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

MusicalComposition\* new\_mc = malloc(sizeof(MusicalComposition));

strcpy(new\_mc->name, name);

strcpy(new\_mc->author, author);

new\_mc->year = year;

new\_mc->prev = NULL;

new\_mc->next = NULL;

return new\_mc;

}

// Функции для работы со списком MusicalComposition

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

MusicalComposition\* head = NULL;

MusicalComposition\* prev = NULL;

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

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

if(i==0){ head = elem;

//head->prev = NULL;

prev = head;

}

else{

elem->prev = prev;

prev->next = elem;

prev = elem;

}

}

return head;

}

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

while(head->next){

head = head->next;

}

head->next = element;

element->prev = head;

}

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

MusicalComposition\* to\_rem;

while(head){

if(!strcmp(head->name, name\_for\_remove)){

if(head->prev){ head->prev->next = head->next;}

if(head->next){ head->next->prev = head->prev;}

to\_rem = head;

head = head->next;

free(to\_rem);

}

else{ head = head->next; }

}

}

int count(MusicalComposition\* head){

int count = 0;

while(head){

head = head->next;

count++;

}

return count;

}

void print\_names(MusicalComposition\* head){

while(head){

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

head = head->next;

}

}

void clearAll(MusicalComposition\* head){

while(head->next){

head = head->next;

free(head->prev);

}

free(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);

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);

clearAll(head);

return 0;

}