**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**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3344 |  | Охрименко Д. И. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Освоение создания и работы со списками на языке Си, посредством использования массивов и структур данных.

## Задание.

Создайте двунаправленный список музыкальных композиций MusicalComposition и api (application programming interface - 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

Для работы с двусвязным списком структур MusicalComposition, которые содержат информацию о музыкальных произведениях: название (name), автора (author) и год создания (year) – созданы следующие функции:

1. createMusicalComposition – возвращает ссылку на новоиспечённую структуру данных; память для элементов структуры выделяется вне функции.

2. createMusicalCompositionList – создаёт двусвязный список, содержащий данные из поступающего массива данных имён и названий произведений. При каждой новой итерации добавляется ссылка новый элемент списка, на его конец – tail.

3. push – функция идёт до конца списка, пока не встретит NULL, который должна заменить на новый элемент списка.

4. removeEl – перебирает элементы списка, пока не найдёт совпадение с названием автора произведения. Если совпадение найдено, удаляет этот элемент из списка, очищает память и связывает соседние элементы ссылками друг на друга.

5. count - функция подсчета количества элементов в списке путём перебора списка.

6. print\_names – форматированный вывод данных о произведении один за другим в стандартный поток выхода. Используем цикл «пока не NULL».

Таким образом, налажен полный доступ к каждому элементу списка, список можно расширять, уменьшать, выводить и измерять его длину, как на аналогичных списках из динамических языков программирования (без строгой типизации).

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 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 | Верный вывод информации |

## Выводы

Были приобретены все навыки, необходимые для построения структур данных и связи между ними, чтобы организовать список на языке программирования, предлагающий только использование массивов.

# Приложение А Исходный код программы

Название файла: lb2.c

#include <stddef.h>

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition{

char\* name;

char\* author;

int year;

struct MusicalComposition\* prev;

struct MusicalComposition\* next;

} MusicalComposition;

// Создание структуры MusicalComposition

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

{

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

MusComp->name = name;

MusComp->author = autor;

MusComp->year = year;

return MusComp;

}

// Функции для работы со списком MusicalComposition

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

{

MusicalComposition\* head = NULL;

MusicalComposition\* tail = NULL;

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

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

if(NULL == head){

head = MusComp;

tail = MusComp;

} else {

tail->next=MusComp;

MusComp->prev=tail;

tail = MusComp;

}

}

return head;

}

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

{

MusicalComposition\* pointer = head;

while(pointer->next != NULL){

pointer = pointer->next;

}

pointer->next = element;

element->next = NULL;

element->prev = pointer;

}

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

{

MusicalComposition\* pointer = head;

while(strcmp(pointer->name, name\_for\_remove)){

pointer = pointer->next;

}

pointer->prev->next = pointer->next;

pointer->next->prev = pointer->prev;

free(pointer);

}

int count(MusicalComposition\* head)

{

int counter = 0;

MusicalComposition\* pointer = head;

do counter++; while((pointer=pointer->next) != NULL);

return counter;

}

void print\_names(MusicalComposition\* head){

MusicalComposition\* pointer = head;

do printf("%s\n", pointer->name); while((pointer=pointer->next) != 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);

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

free(names[i]);

free(authors[i]);

}

free(names);

free(authors);

free(years);

return 0;

}