**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. 3342 |  | Корниенко А.Е. |
| Преподаватель |  | Глазунов С. А. |

Санкт-Петербург

2024

## Цель работы

Изучить композицию двунаправленного списка, и реализовать его при помощи структур на языке Си. Добавить несколько базовых методов для работы с экземплярами структуры двунаправленного списка.

## 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:

1) 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]).

2)void push(MusicalComposition\* head, MusicalComposition\* element); // добавляет element в конец списка musical\_composition\_list

3)void removeEl (MusicalComposition\* head, char\* name\_for\_remove); // удаляет элемент element списка, у которого значение name равно значению name\_for\_remove

4)int count(MusicalComposition\* head); //возвращает количество элементов списка

5)void print\_names(MusicalComposition\* head); //Выводит названия композиций.

## Выполнение работы

Необходимо было создать структуру двунаправленного списка, содержащего следующие поля: char\* name(название композиции), char\* author(автор композиции), int year(год создания), struct MusicalComposition\* next(указатель на следующую композицию), struct MusicalComposition\* prev(указатель на предыдущую композицию).

Далее идет описание функций для работы с этой структурой:

1) MusicalComposition\* createMusicalComposition(char\* name, char\* author,int year) – создание переменной типа нашей структуры. Для этого выделяется память при помощи malloc(), затем присваиваются необходимые значения для полей.

2) MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n) - создает список музыкальных композиций. Сначала выделяется память для head, затем в цикле присваиваются значения полям, выделяется память для следующего элемента списка, описывается зависимость элементов.

3) void push(MusicalComposition\* head, MusicalComposition\* element) – добавления элемента в конец. Создаётся current, который при помощи цикла указывает на последний элемент( next == NULL ).

4) void removeEl(MusicalComposition\* head, char\* name\_for\_remove) - удаление элемента по названию композиции. При помощи цикла находим совпадение, меняем зависимости в списке, очищаем память удалённого элемента, а также память удаляется для полей структуры, для которых также выделялась память.

5) int count(MusicalComposition\* head) – подсчет количества элементов в списке. Заводится переменная-счётчик и при помощи цикла проходим по всем его элементам.

6) void print\_names(MusicalComposition\* head) – вывод всех композиций при помощи цикла проходим по всем элементам списка, и выводи имена.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | 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 |

## Выводы

Была разработана программа, создающая двунаправленный список из музыкальных композиций и выполняющая с ним определенные функции. Изучена работа с линейными списками, со структурами и реализация их на языке Си.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition

{

struct MusicalComposition\* pNext;

struct MusicalComposition\* pPrev;

char\* name;

char\* author;

int year;

}MusicalComposition;

// Создание структуры MusicalComposition

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

{

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

if(element != NULL){

element->name = malloc((strlen(name) + 1) \* sizeof(char));

element->author = malloc((strlen(author) + 1) \* sizeof(char));

if(element->name != NULL && element->author != NULL){

strcpy(element->name,name);

strcpy(element->author,author);

}

element->year = year;

element->pNext = NULL;

element->pPrev = NULL;

}

return element;

}

// Функции для работы со списком MusicalComposition

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

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

MusicalComposition\* current = NULL;

if(head != NULL)

current = head;

MusicalComposition\* prev = NULL;

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

if (current != NULL){

current->name = malloc((strlen(array\_names[i]) + 1) \* sizeof(char));

current->author = malloc((strlen(array\_authors[i]) + 1) \* sizeof(char));

}

if (current->name != NULL && current->author != NULL){

strcpy(current->name, array\_names[i]);

strcpy(current->author, array\_authors[i]);

}

current->year = array\_years[i];

if(i != n - 1){

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

current->pPrev = prev;

if(next != NULL){

current->pNext = next;

}

}

else{

if (current != NULL){

current->pNext = NULL;

current->pPrev = prev;

}

}

if (current != NULL){

prev = current;

current = current->pNext;

}

}

return head;

}

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

{

MusicalComposition\* last = head;

while(last->pNext != NULL){

last = last->pNext;

}

last->pNext = element;

element->pPrev = last;

element->pNext = NULL;

}

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

{

MusicalComposition\* now = head;

if (strcmp(now->name, name\_for\_remove) == 0){

head = now->pNext;

head->pPrev = NULL;

free(now);

}

else{

while(now != NULL){

if(strcmp(now->name, name\_for\_remove) == 0 && now->pNext != NULL){

MusicalComposition\* current = now;

current->pPrev->pNext = current->pNext;

current->pNext->pPrev = current->pPrev;

now = now->pNext;

free(current->author);

free(current->name);

free(current);

}

else if(strcmp(now->name, name\_for\_remove) == 0)

{

MusicalComposition\* current = now;

MusicalComposition\* Prev = now->pPrev;

free(current->name);

free(current->author);

Prev->pNext = NULL;

free(current);

break;

}

now = now->pNext;

}

}

}

int count(MusicalComposition\* head)

{

MusicalComposition\* current = head;

int count = 0;

while(current){

count ++;

current = current->pNext;

}

return count;

}

void print\_names(MusicalComposition\* head)

{

MusicalComposition\* current = head;

while(current){

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

current = current->pNext;

}

}

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;

}