**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: Структуры данных, линейные списки

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3383 |  | Логинова А.Ю. |
| Преподаватель |  | Гаврилов А.В. |

Санкт-Петербург

2024

## Цель работы

Написать программу, реализовывающую двунаправленные линейные списки и функции, позволяющие с ними работать, такие как: добавление и удаление элементов, подсчет количества элементов в списке.

## 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

Двунаправленный линейный список был реализован с помощью структуры *musical\_composition* с полями: *name, author, year, next, prev*. Первые три поля — информация о самом объекте структуры, последние два — указатели на следующий и предыдущий объект в списке.

Создание объектов списка реализовано в функции *createMusicalComposition(),* которая принимает на вход данные для полей *name, author, year*. По умолчанию указатели на предыдущий и следующий элемент равны *NULL.* Если выделить память для элемента не удалось, функция сообщает об ошибке. Далее поля инициализируются.

Функция добавления элемента *push()* добавляет элемент в конец, обновляя указатель на следующий элемент у предыдущего элемента списка. Если нет начала списка или элемента, функция сообщает об ошибке.

Функция создания двунаправленного списка реализована с помощью цикла *for,* который в себе вызывает функцию *push()* для добавления элемента.

Удаление элемента реализовано в четырех функциях: *find\_el\_by\_name(),* *remove\_first(),* *remove\_last(), removeEl()*. Первая функция находит элемент, который необходимо удалить с помощью цикла *for* и функции сравнения строк *strcmp()*. Вторая функция заменяет все поля первого элемента на все поля второго, второй элемент удаляется с помощью функции *free*(). Третья функция изменяет указатель *next* предыдущего элемента на указатель *next* следующего элемента. Текущий элемент удаляется. *RemoveEl()* в себе вызывает две эти функции, проверяя, какой элемент нужно удалить. Если не первый и не последний, тогда происходит изменение указателей, где указатель предыдущего элемента next приравнивается к указателю следующего элемента next и таким же образом изменяется указатель prev.

Подсчитывание количества элементов в списке реализовано в функции *count()* с помощью цикла *for.*

Функция *print\_names()* выводит значение поля *name* всех элементов списка. Была введена статическая переменная *need\_lf* для того, чтобы перенос строки печатался во всех итерациях, кроме первой.

Результаты тестирования программы представлены в таблице 1.

## Тестирование.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| 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 | Программа работает корректно. |

## Выводы

Была написана программа, позволяющая работать с двусвязными списками. Были изучены возможности структур и реализованы стандартные методы работы с контейнерами.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

// Описание структуры MusicalComposition

typedef struct musical\_composition {

char \*name, \*author;

unsigned long year;

struct musical\_composition \*next, \*prev;

} MusicalComposition;

// Создание структуры MusicalComposition

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

// Функции для работы со списком MusicalComposition

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

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

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

int count(MusicalComposition\* head);

void print\_names(MusicalComposition\* head);

void remove\_first(MusicalComposition \*current);

void remove\_last(MusicalComposition \*current);

MusicalComposition\* find\_el\_by\_name(MusicalComposition \*head, char \*name);

void print\_error(char \*func\_name, char \*msg);

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

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

if(!musicalComposition) {

print\_error("createMusicalComposition", "memory allocation");

exit(errno);

}

musicalComposition->name = name;

musicalComposition->author = author;

musicalComposition->year = year;

musicalComposition->next = NULL;

musicalComposition->prev = NULL;

return musicalComposition;

}

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

MusicalComposition \*head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]),

\*current = head;

for (int i = 1; i < n; i++)

push(current, createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]));

return head;

}

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

if (!head)

return print\_error("push", "head is invalid");

if (!element)

return print\_error("push", "element for push is invalid");

for (; head->next; head = head->next);

head->next = element;

element->prev = head;

}

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

if(!head)

return print\_error("removeEl", "head is invalid");

if(!name\_for\_remove)

return print\_error("removeEl", "name for remove is invalid");

MusicalComposition \*el\_to\_remove = find\_el\_by\_name(head, name\_for\_remove);

if(!el\_to\_remove)

return;

if (!el\_to\_remove->prev)

return remove\_first(el\_to\_remove);

if (!head->next)

return remove\_last(el\_to\_remove);

el\_to\_remove->prev->next = el\_to\_remove->next;

el\_to\_remove->next->prev = el\_to\_remove->prev;

free(el\_to\_remove);

}

int count(MusicalComposition\* head) {

int count = 0;

for(; head; head = head->next, count++);

return count;

}

void print\_names(MusicalComposition\* head) {

static int need\_lf = 0;

for(; head; head = head->next) {

if (need\_lf)

printf("\n");

else

++need\_lf;

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

}

}

void remove\_first(MusicalComposition \*current) {

current->name = current->next->name;

current->author = current->next->author;

current->year = current->next->year;

current->next = current->next->next;

free(current->next->prev);

current->next->prev = current;

}

void remove\_last(MusicalComposition \*current) {

current->prev->next = current->next;

free(current);

}

MusicalComposition\* find\_el\_by\_name(MusicalComposition \*head, char \*name) {

for(; head && name; head = head->next)

if(strcmp(head->name, name) == 0)

return head;

return NULL;

}

void print\_error(char \*func\_name, char \*msg) {

if(func\_name && msg)

printf("[%s()] error: %s\n", func\_name, msg);

}

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;

}