**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 |  | Кербель Д. А. |
| Преподаватель |  | Государкин Я. С. |

Санкт-Петербург

2024

## Цель работы

В ходе работы необходимо научиться применять двунаправленные линейные списки на Си для хранения данных полей структуры. Написать соответствующую программу, реализующую двунаправленный список.

## 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

Описание функций:

* createMusicalComposition: создает новый узел MusicalComposition с заданным именем, автором и годом.
* createMusicalCompositionList: создает связанный список узлов MusicalComposition из трех массивов значений имени, автора и года.
* push: добавляет новый узел MusicalComposition в конец связанного списка.
* RemoveEl: удаляет узел MusicalComposition с совпадающим именем из связанного списка.
* count: Возвращает количество узлов в связанном списке.
* print\_names: печатает имена всех узлов MusicalComposition в связанном списке.
* Функция main выполняет следующие действия:
  + Считывает из входных данных целочисленную длину, которая представляет количество музыкальных композиций для сохранения.
  + Выделяет память для трех массивов значений имени, автора и года и считывает значения из входных данных.
  + Создает связанный список узлов MusicalComposition с помощью функции createMusicalCompositionList.
  + Печатает имя первого узла, автора и значения года.
  + Подсчитывает количество узлов в связанном списке с помощью функции count и печатает результат.
  + Создает новый узел MusicalComposition с входными значениями и добавляет его в конец связанного списка с помощью функции push.
  + Снова подсчитывает количество узлов в связанном списке и печатает результат.
  + Удаляет узел с совпадающим именем из связанного списка с помощью функции RemoveEl.
  + Печатает имена всех оставшихся узлов в связанном списке с помощью функции print\_names.
  + Снова подсчитывает количество узлов в связанном списке и печатает результат.
  + Освобождает память, выделенную для массивов и узлов связанного списка.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
| 1. | 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 | Выходные данные соответствуют ожиданиям. |
| 2. | 1  Flag of Hate  Kreator  1985  One  Metallica  1989  Flag of Hate | Flag of Hate Kreator 1985  1  2  One  1 | Выходные данные соответствуют ожиданиям. |
| 3. | 2  Floods  Pantera  1996  Flag of Hate  Kreator  1985  One  Metallica  1989  Midnight Sun  Kreator  2022  Flag of Hate | Floods Pantera 1996  2  3  Floods  Flag of Hate  One  3 | Выходные данные соответствуют ожиданиям. |

## Выводы

В ходе выполнения лабораторной работы были освоены необходимые навыки для создания двунаправленных списков на языке Си.

# Приложение А Исходный код программы

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct MusicalComposition{

char\* author; char\* name; int year;

struct MusicalComposition\* next;

struct MusicalComposition\* prev;

} MusicalComposition;

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

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

musicalComposition->name = name; musicalComposition->author = author; musicalComposition->year = year;

return musicalComposition;

}

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

MusicalComposition\* list = (MusicalComposition\*)malloc(sizeof(MusicalComposition) \* n);

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

list[i].name = array\_names[i]; list[i].author = array\_authors[i]; list[i].year = array\_years[i];

if(i != 0) list[i].prev = &list[i - 1];

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

}

list[0].prev = NULL; list[n - 1].next = NULL;

return list;

}

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

MusicalComposition\* current = head;

while (current->next){

current = current->next;

}

current->next = element; current->next->prev = current; current->next->next = NULL;

}

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

MusicalComposition\* current = head;

while (current){

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

if (current->prev != NULL){

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

}

if (current->next != NULL){

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

}

current = NULL;

break;

}

current = current->next;

}

}

int count(MusicalComposition\* head){

int counter = 0;

MusicalComposition\* current = head;

while (current){

counter++;

current = current->next;

}

return counter;

}

void print\_names(MusicalComposition\* head){

MusicalComposition\* current = head;

while(current){

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

current = current->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;

}