**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. 3341 |  | Романов А. К. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является освоение работы с линейными списками.

Для достижения поставленной цели требуется решить следующие задачи:

* ознакомиться со структурой «список»;
* ознакомиться со списком операций используемых для списков;
* изучить способы реализации этих операций на языке C;
* написать программу, реализующую двусвязный линейный список и решающую задачу в соответствии с индивидуальным заданием.

## 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

Элемент списка типа *MusicalComposition* состоит из:

* указателяна *char name -* названия композиции
* указателяна *char author -* автора композиции
* целого числа *year -* года создания композиции
* указателяна тип *MusicalComposition prev -* указателя на предыдущую композицию
* указателяна тип *MusicalComposition next -* указателя на следующую композицию

Функции:

* *MusicalComposition\* createMusicalComposition(char\* name, char\* author,int year)* – принимает на вход имя композиции и ее исполнителя, а также год записи. Создает новый объект типа MusicalComposition, присваивая соответствующим полям нужные значения.
* *MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n) –* принимает на вход список названий композиций, список авторов, список годов записи и количество элементов в каждом из них. Функция создает первый элемент списка при помощи функции *createMusicalComposition,* а затем аналогично n – 1 оставшихся элементов. Возвращает указатель на первый элемент списка.
* *void push(MusicalComposition\* head, MusicalComposition\* element) –* принимает на вход указатель на первый элемент списка и новый объект типа MusicalComposition. Через указатели *next* функция доходит от головного элемента до последнего (чей указатель *next* ссылается на NULL). После чего добавляется полученный объект, т.е. его полю *prev* присваивается указатель на последний элеммент списка, а полю *next* последнего элемента присваивается указатель на новый элемент.
* *void removeEl(MusicalComposition\*\* head, char\* name\_for\_remove)* – принимает на вход указатель на первый элемент списка и имя композиции, которую следует удалить. Функция проверяет имена всех композиций на соответствие name\_for\_remove. Если искомый элемент первый, то head устанавливается на второй элемент, при этом освобождается память, выделеннаяпод первый элемент. Иначе, когда найден требуемый элемент (предположим, что его номер x), то элементу х – 1 в поле next указыается адрес элемента x + 1, а элементу x + 1 в поле prev указывется адрес элемента x – 1. Память, выделенная под элемент х также освобождается.
* *void print\_names(MusicalComposition\* head) —* принимает указатель на первый элемент списка. Проходит по всем его элементам и выводит имена композиций.
* *int count(MusicalComposition\* head) —* принимает указатель на первый элемент списка. Выводит количество всех элементов списка.
* *int main()* - вызывает описанные функции в установленном условием порядке.

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 |
|  | 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 | Moevm confirmed |
|  | 4  All of the lights  Kanye West  2010  Plug in baby  Muse  2001  Paranoid android  Radiohead  1997  Let it happen  Tame Impala  2015  MX  Deftones  1997  All of the lights | All of the lights Kanye West 2010  4  5  Plug in baby  Paranoid android  Let it happen  MX  4 | Проверяется удаление первого элемента |
|  | 4  Poetic justice  Kendrick Lamar  2012  About a girl  Nirvana  1989  Party monster  The Weeknd  2017  Holiday  Green Day  2004  Lovefool  The Cardigans  1996  Lovefool  About a girl | Poetic justice Kendrick Lamar 2012  4  5  Poetic justice  Party monster  Holiday | Удаление более одного элемета |

## Выводы

В ходе выполнения работы были изучены:

* основные принципы работы с линейными списками;
* структура списков и операции, применяемые к ним;
* способы реализации этих операций на языке C;
* написана программа, реализующаю двусвязный линейный список и решающаю задачу в соответствии с индивидуальным заданием.

# 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 \*next;

struct MusicalComposition \*prev;

} MusicalComposition;

int count(MusicalComposition\* head);

void print\_names(MusicalComposition\* head);

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

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

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

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

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;

}

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

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

song->year = year;

strncpy(song->name, name, 81);

strncpy(song->author, author, 81);

return song;

}

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

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

head\_song->prev = NULL;

head\_song->next = NULL;

MusicalComposition\* previous\_ptr = head\_song;

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

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

song->prev = previous\_ptr;

song->next = NULL;

previous\_ptr->next = song;

previous\_ptr = song;

}

return head\_song;

}

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

MusicalComposition\* ptr = head->next;

while (1)

{

if (ptr->next == NULL){

break;

}

ptr = ptr->next;

}

element->prev = ptr;

element->next = NULL;

ptr->next = element;

}

int count(MusicalComposition\* head){

int counter = 1;

MusicalComposition\* ptr = head->next;

while (1)

{

counter++;

if (ptr->next == NULL){

break;

}

ptr = ptr->next;

}

return counter;

}

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

MusicalComposition\* ptr = (\*head)->next;

if (strstr((\*head)->name, name\_for\_remove)) {

(\*head)->next->prev = NULL;

MusicalComposition \*temp = \*head;

\*head = (\*head)->next;

free(temp);

} else {

while (ptr != NULL) {

if (strstr(ptr->name, name\_for\_remove)) {

ptr->prev->next = ptr->next;

if (ptr->next != NULL) {

ptr->next->prev = ptr->prev;

}

free(ptr);

break;

}

ptr = ptr->next;

}

}

}

void print\_names(MusicalComposition\* head){

MusicalComposition\* ptr = head->next;

puts(head->name);

while (1)

{

puts(ptr->name);

if (ptr->next == NULL){

break;

}

ptr = ptr->next;

}

}