

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.

Студент гр. 0382

Санников В.А.

Преподаватель

Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Научиться создавать и редактировать структуры данных и линейные списки языка Си.

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]`).
 - Пол `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

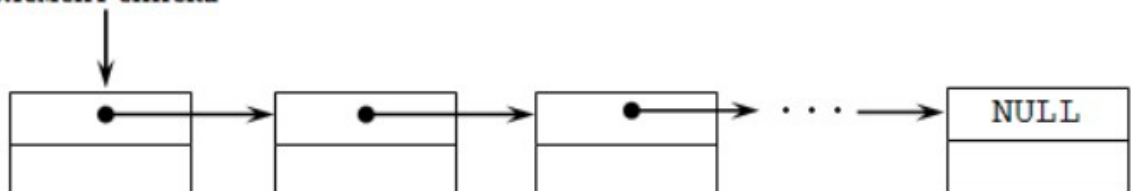
В функции `main` написана некоторая последовательность вызова команд для проверки работы вашего списка.

Функцию `main` менять не нужно.

Основные теоретические положения.

Список - некоторый упорядоченный набор элементов любой природы. Линейный однонаправленный (односвязный) список - список, каждый элемент которого хранит помимо значения указатель на следующий элемент. В последнем элементе указатель на следующий элемент равен `NULL` (константа нулевого указателя).

Указатель на первый
элемент списка



! Чтобы использовать NULL, необходимо подключить #include

Давайте сделаем из структуры Circle элемент списка Node:

```
struct Node{  
    int x;  
    int y;  
    float r;  
    struct Node* next; // указатель на следующий элемент  
};
```

И проинициализируем два элемента списка в функции main():

```
int main(){  
    struct Node * p1 = (struct Node*)malloc(sizeof(struct Node));  
    struct Node * p2 = (struct Node*)malloc(sizeof(struct Node));  
    p1->x = 2; // используем -> поскольку p1 - указатель на структуру Node  
    p1->y = 2;  
    p1->r = 2.5;  
    p2->x = 5;  
    p2->y = 5;  
    p2->r = 5.5;  
    p1->next = p2;  
    p2->next = NULL;  
    free(p1);  
    free(p2);  
    return 0;  
}
```

У нас получился линейный список из двух элементов: p1 и p2.

Выполнение работы.

Ход работы:

Была создана структура элемента списка struct MusicalComposition с именем типа MusicalComposition (через оператор typedef).

Структура состоит из полей `char* name` (название песни), `char* author` (автор), `int year` (год создания). Также добавлены поля `prev` и `next` – указатели на соответственно предыдущий и следующий элемент списка.

Функция `MusicalComposition* createMusicalComposition(char* name, char* author, int year)` является конструктором экземпляра `MusicalComposition`, принимающим данные о композиции, и возвращающим указатель на готовый экземпляр.

Функция `MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)` создаёт двунаправленный список из элементов `MusicalComposition`. Через поля `prev` (у первого элемента - `NULL`) и `next` (у последнего - `NULL`) создаётся направленная связь между элементами списка. Функция принимает массивы с именами, авторами и годами и возвращает указатель на первый элемент списка.

Функция `void push(MusicalComposition* head, MusicalComposition* element)` добавляет элемент `element` в конец списка, добавляя в поле `next` последнего элемента списка указатель на `element`.

Функция `void removeEl(MusicalComposition* head, char* name_for_remove)` удаляет элемент из списка по его названию. Сначала происходит поиск этого элемента, через цикл `while`. Далее, если это первый элемент списка, то происходит копирование данных из второго элемента в первый, затем второй элемент удаляется путём изменения полей `next` и `after` предыдущего и следующего элемента соответственно; если это последний элемент, то в предпоследнем элементе в качестве следующего элемента указывается `NULL`; в остальных случаях элемент удаляется так, как было описано ранее. В конце освобождается память под удалённым элементом.

Функция `int count(MusicalComposition* head)` и `void print_names(MusicalComposition* head)` выполняют подсчёт элементов в списке и их вывод соответственно через цикл `while`.

Разработанный программный код см. в приложении А.

Тестирование.

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	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 верно

	2000		
	Sonne		
	Rammstein		
	2001		
	Points of Authority		

Выводы.

Созданы и отредактированы структуры данных и линейные списки языка программирования Си.

Разработана программа, создающая двунаправленный список и api (application programming interface - в данном случае набор функций) для работы со списком.

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;
    char* author;
    int year;
    struct MusicalComposition* next;
    struct MusicalComposition* prev;
}MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char*
author, int year){
    MusicalComposition* cmp =
(MusicalComposition*)malloc(sizeof(struct MusicalComposition));
    cmp->name = name;
    cmp->author = author;
    cmp->year = year;
    cmp->next = NULL;
    cmp->prev = NULL;
    return cmp;
}

// Функции для работы со списком 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]);
    MusicalComposition* prev = head;
    for (int i = 1; i < n; i++){
        MusicalComposition* current =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        prev->next = current;
        current->prev = prev;
        prev = current;
    }
    return head;
}

void push(MusicalComposition* head, MusicalComposition* element){
    MusicalComposition* end = head;
    while (end->next != NULL){
```



```

        end = end->next;
    }
    end->next = element;
    element->prev = end;
}

void removeEl(MusicalComposition* head, char* name_for_remove){
    while (head->next != NULL) {
        if (!strcmp(head->name, name_for_remove)){
            head->prev->next = head->next;
            head->next->prev = head->prev;
            MusicalComposition* tmp = head;
            head = head->prev;
            free(tmp);
        }
        head = head->next;
    }
}

int count(MusicalComposition* head){
    int n = 1;
    while(head->next != NULL){
        n += 1;
        head = head->next;
    }
    return n;
}

void print_names(MusicalComposition* head){
    while(head){
        puts(head->name);
        head = head->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;

}

```