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. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Изучение линейных списков на языке Си.

Задание.

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface - 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**

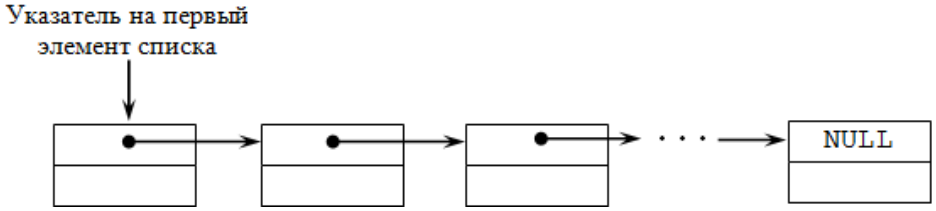
- void removeEl (MusicalComposition* head, char* name_for_remove); // удаляет элемент **element** списка, у которого значение **name** равно нию **name_for_remove**
- int count(MusicalComposition* head); //возвращает количество элементов списка
- void print_names(MusicalComposition* head); //Выводит названия композиций

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

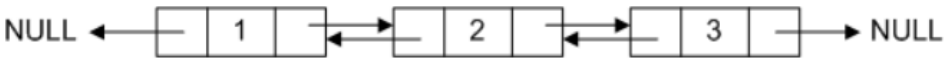
Функцию main менять не нужно.

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).



- Двунаправленны список - это структура данных, которая состоит из узлов, которые хранят данные, указатели на предыдущий узел и следующий узел.



Чтобы использовать NULL, необходимо подключить #include <stddef.h>

Выполнение работы.

Структура элемента списка:

Структура *MusicalComposition* имеет 5 полей: *name* (типа char, название композиции), *author* (типа char, автор композиции/музыкальная группа), *year* (типа int, год создания), *next* (указатель на следующий элемент списка) и *prev* (указатель на предыдущий элемент списка).

При помощи оператора *typedef* был определён тип данных одноимённой структуры.

Функция для создания элемента списка:

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

Функция принимает аргументы *name*, *author* и *year*. Посредством функции *malloc* выделяется динамическая память для элементов заданного ранее типа, указатель передаётся в переменную *node*. Затем каждое поле структуры *MusicalComposition* заполняется соответствующими значениями, полученными в качестве аргументов. Также в поля *next* и *prev* записывается NULL. Функция возвращает значение *node*.

Функции для работы со списком:

- ***MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n);***

  — Создаёт список музыкальных композиций:

  Функция получает массив названий композиций, массив авторов, массив лет и длину этих массивов. Посредством функции *createMusicalComposition* создаётся первый элемент списка и записывается в *head*, указатель на него храниться в переменной *old*. Затем, с помощью цикла *for()*, той же функцией *createMusicalComposition* создаются остальные элементы. Новые элементы хранят указатели на предыдущий и следующий элементы списка в соответствующих полях.

Функция возвращает значение *head* — указатель на первый элемент списка.

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

  — Добавляет новый элемент в конец списка:

  Функция получает указатель на первый элемент списка и значение элемента, который необходимо добавить. При помощи цикла *while()* (здесь и далее выходом из цикла является встреча элемента, поле *next* которого имеет значение NULL - окончание списка, переменной *head* передаётся указатель на следующий элемент списка на каждой итерации) определяется крайний элемент списка и далее его полю *next* присваивается указатель на новый элемент. Затем полю нового элемента *prev* присваивается указатель на последний элемент изначального списка, в поле *next* нового элемента записывается значение NULL.

- ***void removeEl (MusicalComposition* head, char* name_for_remove);***

  — Удаляет элемент, поле name которого совпадает со значением, переданным в функцию:

  Функция получает указатель на первый элемент списка и “название композиции для сравнения”. В цикле *while()* посредством функции *strcmp()* происходит сравнение поля *name* каждого элемента и переданного значения — если значения совпали, то данный элемент удаляется из списка: элементу списка, идущему до данного передаётся адрес элемента следующего за данным, аналогично и в обратную сторону.

- ***int count(MusicalComposition* head);***

  — Возвращает количество элементов списка:

  Функция получает указатель на первый элемент списка. В цикле *while()* идёт подсчёт — в переменную *count*, элементов списка. Функция

возвращает значение переменой *count,* увеличенное на один (для того, что бы последний элемент списка тоже был учтён при счёте).

- ***void print_names(MusicalComposition* head);***

  — Выводит названия композиций:

  Функция получает указатель на первый элемент списка. Внутри функции посредством цикла *while* печатаются все названия композиций. После выхода из цикла печатается название, хранящееся в последнем элементе.

Функция main():

Дана в задаче изначально.

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	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	Ответ верный.

	Linkin Park 2000 Sonne Rammstein 2001 Points of Authority		

Выводы.

Были изучены линейные списки на языке Си.

Разработана программа, которая содержит двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface - в данном случае набор функций*) для работы со списком.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.c

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include <string.h>


// Описание структуры MusicalComposition
typedef struct MusicalComposition{
   char name[80];
   char author[80];
   int year;
   struct MusicalComposition* next;
   struct MusicalComposition* prev;
}MusicalComposition;

// Создание структуры MusicalComposition

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

// 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 *old = head;
   for (int i=1; i<n; i++){
      struct MusicalComposition *node = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
      old->next = node;
      node->prev = old;
      old = node;
   }
   return head;
}
```

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

void removeEl(MusicalComposition* head, char* name_for_remove){
   while(head->next != NULL){
      if (strcmp(head->name, name_for_remove) == 0){
         head->prev->next = head->next;
         head->next->prev = head->prev;
      }
      head = head->next;
   }
}

int count(MusicalComposition* head){
   int count = 0;
   while (head->next != NULL) {
      count=count+1;
      head=head->next;
   }
   return count+1;
}

void print_names(MusicalComposition* head) {
   while (head->next!=NULL) {
      printf("%s\n", head->name);
      head = head->next;
   }
   printf("%s\n", head->name);
}

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;

}
```