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»

Студент гр. 9383 _____ Корсунов А.А.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2020

Цель работы.

Изучить двусвязные списки, а также реализовать двусвязный список на языке 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]).
- 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:

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

Данная функция создает элемент списка с веденными значениями и возвращает указатель на этот элемент;

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

Данная функция с помощью предыдущей функции создает список элементов с веденными значениями и возвращает указатель на первый элемент этого списка.

3) **push(MusicalComposition* head, MusicalComposition* element);**

Данная функция добавляет переданный элемент в конец списка.

4) **removeEl(MusicalComposition* head, char* name_for_remove);**

Данная функция удаляет из списка элемент с переданным значением при помощи изменения полей у соседних элементов списка.

5) **count(MusicalComposition* head);**

Данная функция возвращает количество элементов в списке.

6) **print_names(MusicalComposition* head);**

Данная функция выводит список на экран.

Тестирование.

№ п/п	Входные данные	Выходные данные
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	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

	Chris Isaak 1989 Points of Authority Linkin Park 2000 Sonne Rammstein 2001 Points of Authority	

Вывод.

В ходе проделанной работы были изучены двусвязные списки и работа с ними. Разработана требуемая по заданию программа.

Приложение А
Исходный код программ

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


typedef struct MusicalComposition{
   char* name;
   char* author;
   int year;
   struct MusicalComposition* next;
   struct MusicalComposition* prev;
} MusicalComposition;

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

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

```
}

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

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

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

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