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 _____ Ситченко К.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Изучить и освоить возможности стандартной библиотеки языка 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 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.

Выполнение работы.

С помощью ключевого слова *typedef* был определен тип данных *MusicalComposition*. Структура включает в себя: *name* (название композиции), *author* (имя автора), *year* (год написания), а также указатели на предыдущий и следующий элемент двунаправленного списка (*next* и *prev*).

Функция *createMusicalComposition(char* name, char* autor, int year)* используется для создания элемента списка. Создается указатель на структуру (*elem*), на который с помощью функции *malloc()* выделяется динамическая память. Полям элемента присваиваются значения (название песни, ее автор и год выпуска), переданные в функцию. Переменные *prev* (указатель на предыдущий элемент) и *next* (указатель на следующий элемент) принимают значение NULL. Функция возвращает указатель на структуру.

Функция для создания списка заданной длины – *createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)*. Создается указатель на структуру *head*, куда с помощью

функции *createMusicalComposition* заносятся значения первого элемента. Далее в цикле создаются новые элементы, связанные между собой с помощью указателей. Функция возвращает указатель на структуру.

Функция *push(MusicalComposition* head, MusicalComposition* element)* добавляет элемент в конец списка.

Функция *removeEl(MusicalComposition* head, char* name_for_remove)* удаляет элемента списка с определенным названием.

Функция *count(MusicalComposition* head)* считает количество элементов в списке.

Функция *print_names(MusicalComposition* head)* выводит названия композиций.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в Таблице 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	Программа сработала верно

Выводы.

В ходе выполнения лабораторной работы были изучены особенности работы с двунаправленными списками на языке Си.

Разработана программа, которая имеет функции для создания, удаления, добавления и подсчета элементов двунаправленного списка композиций, их авторов и года издания.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

1. Название файла: main.c

```
#include <stdlib.h>
#include <stdio.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*
autor, int year) {
    MusicalComposition                  *elem                  =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    elem->next = elem->prev = NULL;
    strcpy(elem->name, name);
    strcpy(elem->author, autor);
    elem->year = year;
    return elem;
}

// 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              *elem              =
createMusicalComposition(array_names[i],       array_authors[i],
array_years[i]);
           prev->next = elem;
           elem->prev = prev;
           prev = elem;
    }
    return head;
}

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

```
}

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

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

void print_names(MusicalComposition* head) {
    while (head) {
          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;

}
```