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. 0382 _____ Довченко М.К.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Создание двунаправленного списка и функций для работы с ним.

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.

Основные теоретические положения.

Двунаправленный список — структура данных, которая хранит в себе некоторые поля с данными , а также 2 поля типа *struct struct_name** ,хранящие в себе ссылку на прошлый элемент списка и на следующий элемент списка. Эти два поля позволяют последовательно обрабатывать элементы списка.

Выполнение работы.

Для выполнения данной задачи необходимо написать несколько функций и объявить структуры для работы с двунаправленными списками.

Структура *MusicalComposition* состоит из следующих переменных:

- *char* name* – переменная для хранения названия музыкальной композиции
- *char* author* – переменная для хранения имени автора музыкальной композиции
- *int year* – переменная для хранения года выпуска музыкальной композиции
- *struct MusicalComposition* prev* – указатель на следующий элемент двунаправленного списка

- *struct MusicalComposition* next* – указатель на предыдущий элемент двунаправленного списка

Функция *MusicalComposition *createMusicalComposition(char* name, char* author, int year)* создает элемент списка из переданных ей аргументов, не создавая ссылок на предыдущий и следующий элементы списка.

Функция *MusicalComposition* createMusicalCompositionList (char** array_names, char** array_authors, int* array_years, int n)* с помощью переданных ей аргументов (*int n* отвечает за размер списка) создает список и возвращает ссылку на первый элемент списка, используя циклы *for* и *while*.

Функция *void push(MusicalComposition* head, MusicalComposition* element)* перемещает передаваемый элемент списка в самый конец списка используя цикл *while* в котором текущему элементу списка присваивается следующий элемент списка пока. Цикл выполняется до тех пор пока значение *head->next* будет равно чему либо.

Функция *void removeEl(MusicalComposition* head, char* name_for_remove)* удаляет элемент из списка если название музыкальной композиции совпадает с передаваемой в функцию строкой *name_for_remove.*

Функция *int count(MusicalComposition *head)* возвращает количество элементов списка

Функция *void print_names(MusicalComposition *head)* выводит на экран название каждой композиции находящейся в списке.

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	Fields of Gold Sting 1993	Программа работает корректно

	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	7 8 Fields of Gold In the Army Now Mixed Emotions Billie Jean Seek and Destroy Wicked Game Sonne 7	

Выводы.

Была исследована работа двунаправленных списков.

Разработана программа, работающая с данными, используя двунаправленные списки и функции для работы с ними.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Сначала указываем имя файла, в котором код лежит в репозитории:

Название файла: lab2_2.c

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

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

// Создание структуры MusicalComposition

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

// 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], a
rray_authors[0], array_years[0]);
    for(int i = 1; i < n; i++){
        MusicalComposition* temp = NULL;
        temp = createMusicalComposition(array_names[i], array_authors[i],
 array_years[i]);
        temp->prev = head;
        head->next= temp;
        head = temp;
    }
    while(head->prev) {
        head = head->prev;
    }
    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){
    while (head != NULL) {
        if (!strcmp(head->name, name_for_remove)) {
            if (head->prev == NULL) {
                head->prev = head->next->prev;
                head = head->next;
            } else {
                if (head->next == NULL) {
                    head->prev->next = NULL;
                    head = head->prev;
                } else {
                    head->next->prev = head->prev;
                    head->prev->next = head->next;
                    head = head->next;
                }
            }
        } else {
            head = head->next;
        }
    }
}

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

void print_names(MusicalComposition* head){
    MusicalComposition *temp = head;
    while (temp) {
        printf("%s\n", temp->name);
        temp = temp->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, author
s, 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;
```

```
}
```