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.

Студентка гр. 0382 _____ Охотникова Г.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Освоить такие динамические типы данных как линейные списки, научиться с ними работать.

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.

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).

Выполнение работы.

При выполнении данной лабораторной работы были созданы структура и функции. Для удобства дальнейшей работы с помощью оператора *typedef* был определен одноименный со структурой тип данных.

Структура *Musical Composition*:

char name* - строка с названием композиции;

char author* - строка с именем автора композиции;

int year - целое число с годом создания;

*struct MusicalComposition *prev* - указатель на предыдущий элемент списка;

*struct MusicalComposition *next* - указатель на следующий элемент списка.

Функция *MusicalComposition* createMusicalComposition(char* name, char* author, int year):

Данная функция принимает в качестве аргументов название композиции, имя автора и год создания. Для элемента структуры память выделяется динамически. Полям присваиваются соответствующие значения, которые были преданы в функцию. Возвращаемое значение – элемент списка.

Функция *MusicalComposition* createMusicalCompositionList(char array_names, char** array_authors, int* array_years, int n):**

Данная функция принимает в качестве аргументов двумерные массивы с названиями композиций и их авторами, а также одномерный массив с годами создания композиций и количество элементов списка. Для начала головному элементу с помощью вызова функции для создания элемента списка присваиваются значения нулевых элементов массивов. Затем в цикле происходит создание остальных элементов списка. Затем в цикле происходит возвращение к первому элементу списка, чтобы вернуть указатель на него.

Функция void *push(MusicalComposition* head, MusicalComposition* element)*:

Данная функция в качестве аргументов принимает указатель на первый элемент списка и элемент, который нужно добавить в конец. В цикле происходит перемещение для последнего на данный момент элемента, а затем его полю next присваивается адрес на новый элемент, полю prev которого присваивается адрес на тот элемент, который был последним до этого. Функция ничего не возвращает.

Функция *void removeEl(MusicalComposition* head, char* name_for_remove):

Данная функция в качестве аргументов принимает указатель на первый элемент списка и имя элемента, который требуется удалить. В цикле происходит поиск нужного элемента с помощью функции strcmp, которая позволяет сравнить две строки. Когда нужный элемент найден, учитывается существование указателей на предыдущий и следующий момент, на основе этого происходит переприсвоение полей, после очищается память, которая была выделена на удаленный элемент. Функция ничего не возвращает.

Функция *int count(MusicalComposition* head):*

Данная функция принимает в качестве аргумента указатель на первый элемент списка. Далее в цикле при перемещении последовательно по элементам списка, счетчик каждый раз увеличивается на единицу. Функция возвращает полученное значение.

Функция *void print_names(MusicalComposition* head):*

Данная функция принимает в качестве аргумента указатель на первый элемент списка. Затем в цикле печатаются названия всех композиций. Функция ничего не возвращает.

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	Fields of Gold Sting 1993 7 8 Fields of Gold In the Army Now Mixed Emotions Billie Jean Seek and Destroy Wicked Game Sonne	Программа работает верно.

	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	

Выводы.

Были исследованы методы работы с линейными списками.

Разработана программа, в которой происходит создание двунаправленного списка и различных функций для работы с ним.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.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, int year) {
    struct 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], array_authors[0],
array_years[0]);
    MusicalComposition* elem;
```

```
        for (int i = 1; i < n; i++) {
            elem       =       createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
            elem->prev = head;
            head->next = elem;
            head = elem;
        }
        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(strcmp(head->name, name_for_remove)) {
            head = head->next;
        }
        if (head->prev) {
            head->prev->next = head->next;
        }
        if (head->next) {
            head->next->prev = head->prev;
        }
        free(head);
    }

    int count(MusicalComposition* head) {
        int k = 0;
        while(head) {
            head = head->next;
```

```
            k++;
        }
        return k;
    }

    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;

}
```