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. 1304 Поршнев Р. А.

Преподаватель Чайка К. В.

Санкт-Петербург
2022

Цель работы.

Целью данной лабораторной работы является изучение линейных списков и структур данных.

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.

Функцию main менять не нужно.

Выполнение работы.

Для удобства объявления переменных типа структура MusicalComposition используем оператор typedef. У структуры MusicalComposition объявим поля name и author типа указатель на char, поле year типа int, поле next и prev типа указатель на структуру MusicalComposition. В поле name будет хранится название композиции, в поле author - имя автора композиции, в поле year – год издания композиции. Поле next является указателем на следующий экземпляр структуры MusicalComposition, а поле prev – указателем на предыдущий элемент структуры.

```
typedef struct MusicalComposition{

    char* name;
    char* author;
    int year;
    struct MusicalComposition* next;
    struct MusicalComposition* prev;
```

```
}MusicalComposition;
```

Функция createMusicalComposition возвращает тип указатель на структуру MusicalComposition и она нужна для инициализации полей экземпляра данной структуры. Для инициализации полей данная структура принимает название композиции, имя автора и год издания.

```
    MusicalComposition* createMusicalComposition(char* name, char*
author,int year){

        MusicalComposition*      comp      =      malloc(sizeof(struct
MusicalComposition));
        comp->name = name;
        comp->author = author;
        comp->year = year;
        comp->next = NULL;
        comp->prev = NULL;
        return comp;
}
```

Функция createMusicalCompositionList на вход получает указатель на указатели array_names и array_authors и указатель array_years на область памяти, начиная с которой хранятся года изданий композиций, а также количество элементов списка. В array_names хранятся названия композиций, в array_authors - их авторы. Далее происходит инициализация переменной comp типа указатель на структуру MusicalComposition с помощью функции createMusicalComposition. Переменная comp будет первым экземпляром структуры. Также инициализируем переменную now типа указатель на структуру, в которой будет хранится текущий экземпляр структуры. Далее происходит инициализация переменной tmp экземпляром comp. Далее запускается цикл, в котором инициализируется каждый элемент структуры, начиная со второго по порядку. Поле prev экземпляра now инициализируется tmp, а поле next экземпляра tmp инициализируется now. Таким образом создаётся двунаправленность списка. В конце каждой итерации переменная tmp инициализируется comp. После завершения цикла функции возвращается головной элемент структуры.

Функция push на вход получает экземпляр head, который является головным элементом, и новый экземпляр element. С помощью цикла while

происходит движение вглубь списка. После достижения конца списка осуществляется добавление нового элемента списка.

```
void push(MusicalComposition* head, MusicalComposition* element){

    while (head->next != NULL){
        head = head->next;
    }
    head->next = element;
    element->prev = head;
}
```

Функция removeEl получает на вход экземпляр head и название композиции, которую нужно удалить. Пока head не равен NULL происходит проверка: если поле name экземпляра head совпадает с названием композиции, которую нужно удалить, то поле next предыдущего экземпляра присваиваем следующему экземпляру относительно head. Также для следующего элемента относительно head поле prev присваиваем предыдущему элементу относительно head. Элемент, выпавший из списка, очищаем. После проверки элемент head приравниваем полю next экземпляра head.

```
void removeEl(MusicalComposition* head, char* name_for_remove){

    while(head != NULL){
        if (strcmp(head->name, name_for_remove) == 0){
            head->prev->next = head->next;
            head->next->prev = head->prev;
            free(head);
        }
        head = head->next;
    }
}
```

Функция count принимает экземпляр head. С помощью цикла while происходит движение вглубь списка, на каждой итерации увеличиваем счётчик n на 1. После достижения конца списка функции count возвращается количество элементов в списке n.

```
int count(MusicalComposition* head){

    int n = 0;
    while (head != NULL){
        head = head->next;
        n++;
    }
    return n;
}
```

Функция print_names принимает на вход головной элемент head. С помощью цикла while происходит движение вглубь списка и выводится поле name для текущего экземпляра.

```
void print_names(MusicalComposition* head){

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

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 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	Correct

2.	3 Osnova Kunteynir 2015 Onion scally Milano 2021 Bones Offline 2020 lil pump lil pump 2016 Onion	Osnova Kunteynir 2015 3 4 Osnova Bones lil pump 3	Correct
3.	3 red light bladee 2018 Kyoto yung lean 2013 Alone night lovell 2020 Peroxide ecco2k 2021 Alone	red light bladee 2018 3 4 red light Kyoto Peroxide 3	Correct

Выводы.

Была написана программа, которая создает двунаправленный список и производит над ним действия с помощью функций.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#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* comp = malloc(sizeof(struct MusicalComposition));

    comp->name = name;

    comp->author = author;

    comp->year = year;

    comp->next = NULL;

    comp->prev = NULL;

    return comp;

}

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

    int i;
```

```
        MusicalComposition*                    comp                    =
createMusicalComposition(array_names[0],              array_authors[0],
array_years[0]);

        MusicalComposition* now;

        MusicalComposition* tmp = comp;

        for(i = 1; i < n; i++){

            now       =       createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);

            now->prev = tmp;

            tmp->next = now;

            tmp = now;

        }

        return comp;

    }

    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 != NULL){

            if (strcmp(head->name, name_for_remove) == 0){

                head->prev->next = head->next;

                head->next->prev = head->prev;

                free(head);
```

```
        }
        head = head->next;
    }
}

int count(MusicalComposition* head){

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

void print_names(MusicalComposition* head){

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


int main(){
    int length;
    char c;
```

```
        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", &years[i]);
            fscanf(stdin, "%c", &c);//
            (*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", &year_for_push);
        fscanf(stdin, "%c", &c);//
        (*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;

}
```