МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Программирование»
Тема: «Линейные списки».

Студент гр. 1304 _____ Сенников К.Д.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы

Научиться работать с линейными списками с помощью языка 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

Написание структуры *MusicalComposition*:

Структура включает в себя 5 полей: *name* – записывается название композиции, *author* – записывается имя автора, *year* – записывается год

выпуска, *next* – хранящий в себе указатель на структуру *MusicalComposition*, указывающий на следующий элемент списка, и *previous* – хранящий в себе указатель на структуру *MusicalComposition*, указывающий на предыдущий элемент списка.

Функция *createMusicalComposition*:

Функция принимает на вход указатель на строки name и author, а также число year, возвращает указатель на структуру *MusicalComposition*. Сначала выделяется помять под хранение структуры, в поля структуры записываются соответствующие входные данные, а полям *previous* и *next* присваивается значение *NULL*.

Функция *createMusicalCompositionList*:

Функция принимает на вход массив названий композиций, имена авторов и годы выпусков – *array_names, array_authors, array_years* соответственно. Также на вход поступает *n* – число элементов в каждом массиве. Функция возвращает указатель на структуру, являющуюся первым элементом списка.

Функция объявляет два указателя на структуру *MusicalComposition*: *head* и *next*. С помощью цикла *for* и условных операторов *if, else* присваивается значение для head, для элемента, следующего за head, а затем для всех элементов. Все присвоения происходят с помощью функции *createMusicalComposition*, которая принимает на вход *i-ый* элемент массивов *array_names, array_authors, array_years*. В блоках if для i большего 1 в поля *next* и *previous* записываются соответствующие указатели на элемент списка.

Функция push:

Функция принимает на вход два указателя на структуру *MusicalComposition head* и *element*. Затем в функции создается указатель на структуру *MusicalComposition temp*, в который записывается значение *head*. С помощью цикла *while* находит указатель на последний элемент в списке и затем в поле *next* последнего элемента записывается указатель *element*, а в поле

element->previous записывается указатель *temp*. Функция ничего не возвращает.

Функция count:

Функция принимает на вход указатель на начальный элемент списка, а также проверяет не указывает ли он на *NULL*. Создается указатель на структуру *MusicalComposition temp*, которой присваивается значение *head*. Создается переменная *count*, которой присваивается значение 0; с помощью цикла while считается количество элементов и записывается в переменную *count*. Функция возвращает количество элементов списка.

Функция print_names:

Функция принимает на вход указатель на начальный элемент списка, а также проверяет не указывает ли он на *NULL*. Создается указатель на структуру *MusicalComposition temp*, которой присваивается значение *head*. С помощью цикла *while* и *printf* выводятся значение *name* текущего элемента списка(*temp*). Функция ничего не возвращает.

Функция removeEl:

Функция принимает на вход указатель на начальный элемент списка и указатель на первый элемент строки, которая хранит название композиции, подлежащей удалению. Создается указатель на структуру *MusicalComposition temp*, которой присваивается значение *head*. С помощью цикла *while* функция проходит по списку сравнивая поле *name* текущего элемента с переданным в функцию значением и как только находит проверяет текущий элемент на то, является ли он первым, последним или единственным в списке. В соответствии с определенным условием происходит переадресация предыдущего элемента в списке на следующий и следующего на предыдущий. Затем отчищается память текущего элемента и происходит выход из цикла с помощью оператора *break*. Функция ничего не возвращает.

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	
2.	4 Fields of Gold Sting 1993 In the Army Now Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Billie Jean Michael Jackson 1983 Sonne Rammstein 2001 Billie Jean	Fields of Gold Sting 1993 4 5 Fields of Gold In the Army Now Mixed Emotions Sonne 4	

Выводы

Изучена работа со списками с помощью языка СИ. Была разработана программа, оперирующая с двунаправленным списком и создающая список музыкальных композиций, каждый элемент которого включает в себя несколько полей.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Sennikov_Kirill_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* next;
    struct MusicalComposition* previous;
}MusicalComposition;


// Создание структуры MusicalComposition

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

// Функции для работы со списком MusicalComposition

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

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

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

int count(MusicalComposition* head);

void print_names(MusicalComposition* head);


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;
```

```
}

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

MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n){
    MusicalComposition * head;
    MusicalComposition * next;
    for (int i = 0; i < n; i++){
        if(i == 0){
            head       =       createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        }
        else{
            if(i == 1){
                next     =     createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
                head -> next = next;
                next -> previous = head;
            }
            else{
                next->next  =  createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
                next->next->previous = next;
                next = next->next;
            }
        }
    }
    return head;
}

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

void removeEl(MusicalComposition* head, char* name_for_remove){
    MusicalComposition* temp = head;
    while (temp != NULL){
        if(strcmp(temp->name, name_for_remove) == 0){
            if(temp->previous != NULL && temp->next != NULL){
                temp->previous->next = temp->next;
```

```
                temp->next->previous = temp->previous;
                free(temp);
                break;
            }
            else{
                if(temp->previous == NULL && temp->next != NULL){
                    head = temp->next;
                    temp->next->previous = NULL;
                    free(temp);
                    break ;
                }
                else{
                    if(temp->next == NULL && temp->previous != NULL){
                        temp->previous->next = NULL;
                        free(temp);
                        break;
                    }
                    else{
                        head = NULL;
                        free(temp);
                        break;
                    }
                }

            }
        }
        temp = temp->next;
    }
}

int count(MusicalComposition* head){
    MusicalComposition* temp = head;
    int count = 0;
    while(temp != NULL){
        count+= 1;
        temp = temp->next;
    }
    return count;
}

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