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 _____ Деткова А.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Научиться работать со списками в языке 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.

Функцию *main* менять не нужно.

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).

Линейный двунаправленный список — список, каждый элемент которого, помимо значения, хранит указатель на следующий и предыдущий элементы списка. В первом элементе указатель на предыдущий элемент равен *NULL*, в последнем элементе указатель на следующий элемент равен *NULL*.

Выполнение работы.

Struct MusicalComposition — структура, описывает элемент списка, задает музыкальную композицию, ее имя, автора и год создания. Поля:

- *Struct MusicalComposition* prev* — указатель на предыдущий элемент списка.
- *Char name[80]* — название музыкальной композиции, массив символов длины 80.
- *Char author[80]* — исполнитель, массив символов длины 80.
- *int year* — год создания композиции, число целого типа.

typedef struct MusicalComposition MusicalComposition — называет тип данных, используется для простоты в дальнейшем, обозначает *struct MusicalComposition* как *MusicalComposition.*

Функции:

MusicalComposition createMusicalComposition (char* name, char* author, int year)* — создает элемент списка, инициализируя поля переданными значениями, и возвращает указатель на него. Внутри функции создается указатель на элемент списка *res,* выделяется память под него (динамически). Переданные в функцию значения присваиваются полям элемента списка *res*, поля в виде массивов символов инициализируются с помощью функции *strcpy*, поле year присваиванием.

MusicalComposition createMusicalCompositionList (char** array_names, char** array_authors, int* array_years, int n)* — создает список из 3 массивов, и возвращает указатель на первый элемент списка. Создается указатель на первый элемент списка *head*, для него динамически выделяется память, также создаются *cur* и prev, текущий и предыдущий элементы, далее, проходя по массивам имен, авторов и годов, заполняются поля элементов списка и заполняются ссылки, возвращается указатель на первый элемент списка.

void push (MusicalComposition head, MusicalComposition* element)* — вставляет элемент *element* в конец списка *head* (указатель на первый элемент

списка). Ищется последний элемент списка (указатель на следующий элемнт у такого равен *NULL*) и связывая ссылки с элементами.

void removeEl (MusicalComposition head, char* name_for_remove*) — удаляет элемент, имя композиции которого совпадает с *name_for_remove*. В списке находится элемент с нужным именем, этот элемент удаляется из списка новой связкой ссылок.

int count (MusicalComposition head)* — считает количество элементов списка, возвращает это значение.

void print_names (MusicalComposition head)* — выводит все значения полей *name* в списке.

Функция *main* осуществляет автоматическую проверку правильности написанной программы.

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 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	Корректная работа программы

Выводы.

Были изучены списки, способы работы с ними, преимущества и недостатки в сравнении с другими структурами данных, ранее известных.

Разработана программа, которая задает список и также несколько функций для работы с ним.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: list.c

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

struct MusicalComposition{
    struct MusicalComposition* prev;
    char name[80];
    char author[80];
    int year;
    struct MusicalComposition* next;
};

typedef struct MusicalComposition MusicalComposition;

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

MusicalComposition* createMusicalCompositionList (char** array_names,
char** array_authors, int* array_years, int n){
    MusicalComposition* head = malloc(sizeof(MusicalComposition));
    MusicalComposition* cur = head;
    MusicalComposition* prev = NULL;
    for (int i=0; i<n; i++){
        cur->prev = prev;
        strcpy(cur->name, array_names[i]);
        strcpy(cur->author, array_authors[i]);
        cur->year = array_years[i];
        prev = cur;
        if (i != n-1) {
            cur = malloc(sizeof(MusicalComposition));
            prev->next = cur;
        }else{
            prev->next = NULL;
        }
    }
    return head;
}

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

```
    }
    cur->next = element;
    element->prev = cur;
    element->next = NULL;
}

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

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

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