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 _____ Кардаш Я.Е.

Преподаватель _____ Чайка К.В

Санкт-Петербург

2022

Цель работы.

Изучить линейные списки, научиться их создавать и применять.

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.

Ход работы.

Для выполнения задания сначала была объявлена структура списка. В ней хранились поля с данными, а также указатели на предыдущий и следующий элементы списка. Далее были написаны функции создания элемента списка и самого списка. Функция createMusicalComposition записывала в объявленную структуру необходимые данные, а createMusicalCompositionList связывала их в единый список. Далее были созданы следующие функции для работы со списком:

push – добавляет в конец списка еще один элемент

removEl – удаляет элемент с заданным именем

count – возвращет количество элементов списка

print_names – печатает все имена (одно из полей структуры)

Экспериментальные результаты.

№ Теста	**Вход**	**Выход**
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 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	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

	2001 Points of Authority	

Выводы.

В ходе работы были изучены основные методы работы со списками, созданы необходимые структуры и функции для объявления списка а также функции для работы с ним.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#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* PREV;
} MUSICALCOMPOSITION;

// СОЗДАНИЕ СТРУКТУРЫ MUSICALCOMPOSITION

MUSICALCOMPOSITION*    CREATEMUSICALCOMPOSITION(CHAR*    NAME,    CHAR*
AUTHOR,INT YEAR);

// ФУНКЦИИ ДЛЯ РАБОТЫ СО СПИСКОМ MUSICALCOMPOSITION

MUSICALCOMPOSITION*   CREATEMUSICALCOMPOSITIONLIST(CHAR**   ARRAY_NAMES,
CHAR** ARRAY_AUTHORS, INT* ARRAY_YEARS, INT N);

VOID PUSH(MUSICALCOMPOSITION* HEAD, MUSICALCOMPOSITION* ELEMENT);

VOID REMOVEEL(MUSICALCOMPOSITION* HEAD, CHAR* NAME_FOR_REMOVE);

INT COUNT(MUSICALCOMPOSITION* HEAD);
```

```
VOID PRINT_NAMES(MUSICALCOMPOSITION* HEAD);



MUSICALCOMPOSITION*    CREATEMUSICALCOMPOSITION(CHAR*    NAME,    CHAR*
AUTHOR,INT YEAR){
    MUSICALCOMPOSITION * CUR= MALLOC(SIZEOF(MUSICALCOMPOSITION));
    CUR->NAME = NAME;
    CUR->AUTHOR = AUTHOR;
    CUR->YEAR =YEAR;
    CUR->PREV=NULL;
    CUR->NEXT=NULL;
    RETURN CUR;
}

MUSICALCOMPOSITION*   CREATEMUSICALCOMPOSITIONLIST(CHAR**   ARRAY_NAMES,
CHAR** ARRAY_AUTHORS, INT* ARRAY_YEARS, INT N){
    MUSICALCOMPOSITION* CUR;
    MUSICALCOMPOSITION* HEAD;
    MUSICALCOMPOSITION* LAST;
    FOR(INT I=0;I<N;I++){
        CUR                                                              =
CREATEMUSICALCOMPOSITION(ARRAY_NAMES[I],ARRAY_AUTHORS[I],ARRAY_YEARS[I]);
        IF(I==0){
            HEAD=CUR;
            LAST=HEAD;
            CONTINUE;}
        CUR->PREV=LAST;
        LAST->NEXT=CUR;
        LAST=CUR;
        }
    RETURN HEAD;
}

VOID PUSH(MUSICALCOMPOSITION* HEAD, MUSICALCOMPOSITION* ELEMENT){
    MUSICALCOMPOSITION* CUR = HEAD;
    WHILE(CUR->NEXT){
        CUR = CUR->NEXT;}
    CUR->NEXT = ELEMENT;
    ELEMENT->PREV = CUR;
    }
```

```
VOID REMOVEEL(MUSICALCOMPOSITION* HEAD, CHAR* NAME_FOR_REMOVE){
    MUSICALCOMPOSITION* CUR =HEAD;
    WHILE(CUR){
        IF(STRCMP(CUR->NAME,NAME_FOR_REMOVE)==0){
            CUR->PREV->NEXT=CUR->NEXT;
            CUR->NEXT->PREV=CUR->PREV;
            FREE(CUR);}
        CUR=CUR->NEXT;
        }
    }

INT COUNT(MUSICALCOMPOSITION* HEAD){
    INT K=0;
    MUSICALCOMPOSITION* CUR = HEAD;
    WHILE(CUR->NEXT){
        K+=1;
        CUR=CUR->NEXT;}
    K+=1;
    RETURN K;

}

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;

}
```