МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
"ЛЭТИ" ИМ. В.И.УЛЬЯНОВА(ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №2

по дисциплине «Программирование»

Тема: Линейные списки.

Студент гр. 1304 _____ Мусаев А.И.

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.

Выполнение работы:

Создается структура MusicalComposition, которая имеет 5 полей:

1. char* name - название композиции
2. char* author - имя автора
3. int year - год выпуская композиции
4. struct MusicalComposition* next - указатель на следующий элемент списка
5. struct MusicalComposition* prev - указатель на предыдущий элемент списка

Далее создается функция MusicalComposition* createMusicalCompositionList, которая добавляет новую композицию в список. В ней происходит выделение памяти под лемент структуры и поля заполняются соответствующими им значенями посредством функций стандартной библиотеки. Стоит упомянуть, что поля next и prev мы приравниваем к NULL, так как при создании одиночного элемента он ни с чем не связан.

Далее идёт функция createMusicalCompositionList, которая создаст линейный список и заполнит его элементами массивов, поданных на вход. Через цикл сначала она с помощью флага создаст корневой элемент списка, потом постепенно создаст сам линейный список и вернет указатель на первый элемент.

После нее идёт функция push, которая добавляет элемент в конец списка. Принцип ее работы прост: она заменяет указатель на следующий последнего элемента, NULL, на элемент, поданный на вход.

После идёт функция removeEl, которая удаляет композицию с поданным названием. Она пробегает по списку, находит элемент с нужным названием, связывает его предыдущий элемент с последующим за ним элементом и отчищает память нынешнего элемента.

После идёт функция count, которая считает количество элементов списка. Она банально доходит через цикл до последнего элемента и попутно считает сколько прошла.

И, наконец, функция print_names, которая выводит все названия композиций. У нее тоже банальный принцип, она идёт до последнего и попутно выводит все названия.

Тестирование:

Test	Input	Result	Комментарий
№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 Point of Authority Linkin Park 2000 Sonne Rammstein 2001 Point of Authority	Fields of Gold Sting 1993 7 8 Fields of Gold In the Army Nowe Mixed emotions Billie Jean Seek adn Destroy Wicked Game Sonne 7	Верно

Вывод:

Была написана программа, которая создает двунаправленный список музыкальных композиций и api для работы с ним.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


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


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

MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n){
    MusicalComposition* cur;
    MusicalComposition* last;
    MusicalComposition* first;
    int flag=1;
    int i;
    for(i=0;i<n;i++){
        cur=createMusicalComposition(array_names[i],array_authors[i],
```

```
        array_years[i]);
        if (flag){
            first=cur;
            flag=0;
            last=first;
            continue;
        }
        cur->prev=last;
        last->next=cur;
        last=cur;
    }
    return first;
}

void push(MusicalComposition* head, MusicalComposition* element){
    MusicalComposition* tmp = head;
    while (1){
        if (tmp->next==NULL){
            element->prev = tmp;
            tmp->next = element;
            break;
        }
        tmp=tmp->next;
    }
}

void removeEl(MusicalComposition* head, char* name_for_remove){
    MusicalComposition* tmp = head;
    while (tmp){
        if (strcmp(tmp->name,name_for_remove)==0){
            tmp->next->prev=tmp->prev;
            tmp->prev->next = tmp->next;
            free(tmp);
        }
        tmp=tmp->next;
    }
}

int count(MusicalComposition* head){
```

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

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

}
```