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

ОТЧЕТ

по лабораторной работе №4
по дисциплине
«Программирование» Тема:
«Структуры данных. Линейные списки»

Студент гр. 7381 ____ Дорох С.В.

Преподаватель ____ Берленко Т.А.

Санкт-
Петербург
2017

Цель работы:

Познакомится и научиться создавать структуры данных в языке Си. Научиться создавать и работать с двунаправленными линейными списками в Си. Закрепить знания в работе с динамической памятью.

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:

Была создана структура MusicalComposition хранящая в себе 2 переменных-указателя char*, int и 2 указателя на структуру, содержащие адресса следующего и предыдущего элементов.

Создана функция для создания элемента списка- createMusicalComposition. Данная функция получает на вход три параметра: имя, автор, и год издания композиции. В ней выделяется память malloc`ом под новый элемент списка и используется функция strncpy для копирования данных о композиции в новый элемент списка. Функция возвращает первый элемент.

Создаётся функция push, которая добавляет элемент в конец списка музыкальных композиций. На вход функции подаётся 2 параметра: указатель на элемент, который надо добавить, и указатель на первый элемент. Функция ничего не возвращает.

Функция removeEl удаляет элемент списка, у которого имя композиции совпадает с именем композиции для удаления. На вход программе подаются: указатель на первый элемент и указатель на имя элемента для удаления. При помощи функции strcmp проверяется список на совпадение имя композиции с именем композиции для удаления. Функция ничего не возвращает.

Функция count возвращает количество элементов списка. На вход подаётся указатель на первый эл-т списка. При помощи цикла for подсчитывается количество элементов списка.

Функция print_names выводит названия композиций. На вход программе подаётся указатель на первый элемент списка. При помощи цикла for и функции printf выводятся названия композиций. Функция ничего не возвращает.

Вывод: в данной лабораторной работе были изучены создание двунаправленного линейного списока и работа с ним. Закреплены знания по работе со структурами данных и по работе с динамической памятью.

Код программы:

Main.c:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.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){
    MusicalComposition*
newMC=(MusicalComposition*)malloc(sizeof(MusicalComposition));
    newMC->name = (char*)malloc(81*sizeof(char));
    strncpy(newMC->name, name, 81);
    newMC->author = (char*)malloc(81*sizeof(char));
    strncpy(newMC->author, author, 81);
    newMC->year = year;
    newMC->next = NULL;
    newMC->prev = NULL;
```

```
    return newMC;
}

// Функции для работы со списком MusicalComposition

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

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


    MusicalComposition* tmp = head;

    while(tmp->next != NULL)
        tmp=tmp->next;

    tmp->next = element;
    element->prev = tmp;
}

void removeEl(MusicalComposition* head, char* name_for_remove){
    MusicalComposition* tmp;
    for (tmp = head; tmp != NULL; tmp = tmp->next)
        if (strcmp(tmp->name, name_for_remove) == 0) {
            tmp->prev->next = tmp->next;
            tmp->next->prev = tmp->prev;
            free(tmp);
            free(tmp->name);
            free(tmp->author);
            return;
        }
```

```
}

int count(MusicalComposition* head){
    MusicalComposition* tmp = head;
    int size;
    for( size = 0; tmp != NULL ; size++)
        tmp = tmp->next;
    return size;
}

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


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);
    free(name_for_push);
    free(author_for_push);
    free(year_for_push);

    return 0;

}
```

Makefile:

```
all: main.o
    gcc main.o
main.o: main.c
    gcc -c main.c
```