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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: Линейные списки.

Студентка гр. 7381 _____ Мартьянова Н. М.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2017

Цель работы:

Познакомиться с линейными списками и структурами, научиться создавать их и проводить над ними операции.

Задание

Создать двунаправленный список музыкальных композиций MusicalComposition и api для работы со списком.

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.

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

Основные теоретические положения:

Заголовочные файлы, необходимые для создания проекта:

<stddef.h> содержит макрос нулевого указател NULL

<string.h> содержит прототипы функций для работы со строками:

strcmp(const char* string1, const char* string2).

<stdio.h> содержит прототипы функций для ввода-вывода:

fgets, printf(const char* format [argument]…).

<stdlib.h> содержит прототипы функций для выделения памяти:

malloc(size_t sizemem).

Вывод:

В ходе выполнения лабораторной работы были освоены линейные списки, структуры данных.

Исходный код проекта

Файл main.c:

```
#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <stddef.h>
```

```
typedef struct MusicalComposition{
    struct MusicalComposition* next;
     struct MusicalComposition* prev;
    char* name;
    char* author;
    int year;
  }MusicalComposition;// Описание структуры MusicalComposition


// Создание структуры MusicalComposition


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


// Функции  для работы со списком MusicalComposition
MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors,
int* array_years, int n){
int i = 1;
```

```
    MusicalComposition *head = createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
    MusicalComposition *list = head;
    for (; i < n; i++)
    {
        list->next = createMusicalComposition(array_names[i], array_authors[i], array_years[i]);
        list->next->prev=list; list=list->next;
    }
    return head;
}
void push(MusicalComposition* head, MusicalComposition* element){
    if(!head) return;
            while(head->next){
             head = head->next;
            }
            element->prev = head;
            element->next = NULL;
            head->next = element;
}
void removeEl(MusicalComposition* head, char* name_for_remove){
    for(;strcmp(head->name,name_for_remove);)
    {
        head = head->next;
    }
    head->prev->next = head->next;
    head->next->prev = head->prev;
}
```

```
int count(MusicalComposition* head){
  int i = 0;
            while(head){
             i++;
             head = head->next;
            }
            return i;
}
void print_names(MusicalComposition* head){
 for (MusicalComposition* list = head; list != NULL; list=list->next)
        printf("%s\n", list->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);


    return 0;
```

}

Makefile

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