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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Программирование»

Тема: «Линейные списки»

Студент гр. 7381 _____ Вологдин М.Д.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2017

Цель работы

Познакомиться со списками и структурами на языке СИ, научиться их создавать и производить операции над ними.

Задание

Создать 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

Заголовочные файлы, необходимые для создания проекта:

1. <stdlib.h> – содержит прототипы функций для выделения памяти “void * malloc(size_t sizemem);” “void * realloc(void * ptrmem, size_t size);” и “void free(void * ptrmem);”

2. <stdio.h> – содержит прототипы функций "int printf(const char* format [, argument]...);" и "int fscanf(FILE *fp, const char * форматная_строка, ...);", которые используются для ввода из потока ввода и вывода в поток вывода.

3. <string.h> – содержит прототипы функций для работы со строками “char * strstr(char * string1, const char * string2);” “size_t strlen(const char * string);” “char * strcpy(char * destptr, const char * srcptr);” “int strcmp(const char * string1, const char * string2);”

4. <stddef.h> - содержит макрос нулевого указателя NULL.

Вывод

В результате работы были освоены структуры данных линейные списки, а также был создан и освоен набор функций для работы с ними.

Исходный код проекта

Файл “**Makefile**”

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

Файл "**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* create = (MusicalComposition*)malloc(sizeof(MusicalComposition));
```

```
    create->name = name;
    create->author = author;
    create->year=year;
    create->next=NULL;
    create->prev=NULL;
    return create;
}


// Функции для работы со списком MusicalComposition
void push(MusicalComposition* head, MusicalComposition* element)
{
    MusicalComposition* temp = (MusicalComposition*)malloc(sizeof(MusicalComposition));
        if ( head->next == NULL )
        {
            element->next = NULL;
            element->prev = head;
            head->next = element;
            return;
        }

        temp = head->next;

        while (temp->next)
        {
            temp = temp->next;
        }

        element->next = NULL;
        element->prev = temp;
        temp->next = element;
        return;
}
MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int*
array_years, int n)
{
    if (n==0)
        return NULL;
    MusicalComposition *head = createMusicalComposition(array_names[0],array_authors[0],array_years[0]);
    MusicalComposition *temp = (MusicalComposition*)malloc(sizeof(MusicalComposition));
    for (int i=1;i<n;i++)
    {
        temp=createMusicalComposition(array_names[i],array_authors[i],array_years[i]);
        push(head,temp);
    }
    return head;
}



void removeEl(MusicalComposition* head, char* name_for_remove)
{
    if (head==NULL)
        return;
    MusicalComposition *temp = head;

    if (strcmp(name_for_remove, temp->name) == 0)
    {
```

```
        head=head->next;
        free(temp);
        return;
    }
    while(temp && (strcmp(temp->name,name_for_remove) != 0))
    {
        temp=temp->next;
    }
    if (temp->next)
    {
        temp->next->prev = temp->prev;

    }
    temp->prev->next = temp->next;
    return;
}

int count(MusicalComposition* head)
{
    if (head == NULL)
        return 0;
    int i=0;
    MusicalComposition* temp = head;
    while(temp)
    {
        temp = temp->next;
        i++;

    }
    return i;
}

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

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;

}
```