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 _____ Трушников А.П.

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.

Создайте двунаправленный список музыкальных композиций MusicalComposition и api (application programming interface - 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.

Фаил **“Makefile”**

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

Файл **"main.c"**

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

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

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* autor,int year){
    MusicalComposition* mus=(MusicalComposition*)malloc(sizeof(MusicalComposition));
    int i;
    for(i=0;i<80;i++){
        mus->name[i]=name[i];
```

```
        mus->author[i]=autor[i];
    }
    mus->year=year;
    mus->next=NULL;
    mus->prev=NULL;
    return mus;
}

// Функции для работы со списком MusicalComposition

void push(MusicalComposition* head, MusicalComposition* element){
    if(head->next==NULL){
        head->next=element;
        element->prev=head;
        return;
    }
    push(head->next,element);
}

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]);
    int i;
    for(i=1;i<n;i++){
        MusicalComposition* mus=createMusicalComposition(array_names[i],array_authors[i],array_years[i]);
        push(head,mus);
    }
    return head;
```

```
}

void removeEl(MusicalComposition* head, char* name_for_remove){
    int i;
    int equal=1;
    if(head==NULL){
        return;
    }

    for(i=0;i<80;i++){
        if(head->name[i]=='\0')
            break;
        if(head->name[i]!=name_for_remove[i]){
            equal=0;
        }
    }
    if(equal==1){
        head->prev->next=head->next;
        head->next->prev=head->prev;
        free(head);
        return;
    }
    if(head->next==NULL)
        return;
    removeEl(head->next,name_for_remove);
}

int count(MusicalComposition* head){
    int i=0;
    MusicalComposition*cur=head;
```

```
    while(cur!=NULL){
        i++;
        cur=cur->next;
    }
    return i;
}

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

}
```