МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе

№4

по дисциплине «Программирование»

Тема: “Структуры данных, линейные списки”

Студент гр. 7381 Кортев Ю.В.

Преподаватель Берленко Т.А

Санкт-Петербург

2017

Цель.

Закрепить навыки работы со структурами. Освоить на практике работу с двунаправленными списками.

Задача.

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

Вывод

На практике освоен алгоритм работы с двунаправленным масивом, закреплены навыки работы со структурами.

Исходный код.

```
#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* back;
}MusicalComposition;

// Создание структуры MusicalComposition
void removeEl(MusicalComposition* head, char* name_for_remove);
int count(MusicalComposition* head);
void print_names(MusicalComposition* head);
MusicalComposition* createMusicalComposition(char* name, char* author,int
year);
MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years, int n);
void push(MusicalComposition* head, MusicalComposition* element);
MusicalComposition* createMusicalComposition(char* name, char* author,int
year)
{
   MusicalComposition*
make=(MusicalComposition*)malloc(sizeof(MusicalComposition));
        make->name = name;
```

```
        make->author = author;
        make->year = year;
        make->next = NULL;
        make->back = NULL;
        return make;

}


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


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

  MusicalComposition* tmp=head;

   while(tmp->next)
   {
      tmp=tmp->next;
   }
   element->next=NULL;
   element->back=tmp;
```

```
    tmp->next=element;
}


void removeEl(MusicalComposition* head, char* name_for_remove)
{
    MusicalComposition* tmp = head;

    while (tmp->next)
    {
        if(strcmp(tmp->name,name_for_remove)==0)
        {
            tmp->next->back=tmp->back;
            tmp->back->next=tmp->next;
            tmp->next=NULL;
            tmp->back=NULL;
            free(tmp);
     return;
        }
        tmp=tmp->next;
    }
}



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


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;

}
```