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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Программирование»

Тема: Линейные списки

Студент гр. 0382 _____ Сергеев Д.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2021

Цель работы.

Изучение линейных списков на языке Си.

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.

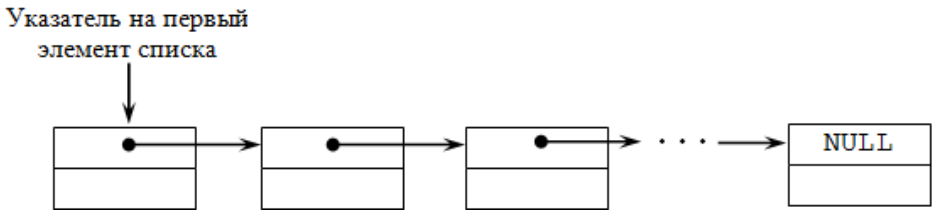
- void push(MusicalComposition* head, MusicalComposition* element); // добавляет element в конец списка musical_composition_list
- void removeEl (MusicalComposition* head, char* name_for_remove); // удаляет элемент element списка, у которого значение name равно значению name_for_remove
- int count(MusicalComposition* 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.

- Список - некоторый упорядоченный набор элементов любой природы.
- Линейный однонаправленный (односвязный) список - список, каждый элемент которого хранит помимо значения указатель на следующий элемент. В **последнем** элементе **указатель** на следующий элемент равен NULL (константа нулевого указателя).



Чтобы использовать NULL, необходимо подключить #include <stddef.h>

Выполнение работы.

- Структура *MusicalComposition*

  Создана структура *MusicalComposition* с полями:

  - *Char* name* – хранит название композиции;
  - *Char* author* – хранит наименование автора композиции;
  - *Int year* – хранит год выпуска композиции;
  - *Struct MusicalComposition* next* – хранит указатель на следующий элемент списка;
  - *Struct MusicalComposition* prev* – хранит указатель на предыдущий элемент списка;

С помощью оператора *typedef* для удобства *Struct MusicalComposition* был определен как *MusicalComposition*.

- Функция *createMusicalComposition (char* name, char* autor, int year)*

  Функция создает указатель на объект структуры *MusicalComposition*, а также определяет его поля name, author и year в соответствии с переданными в функцию значениями. Память под объект структуры и поля выделяется с помощью функции *calloc()*. Функция возвращает указатель на объект структуры *MusicalComposition*.

- Функция *MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)*

  Функция создает указатель на объект структуры *MusicalComposition*, выделяя под него память с помощью функции *calloc()*, этот указатель является головным в будущем списке и называется *head*. У *head* определяются поля *author, name* и *year*, а также *prev*, в g которое помещается значение *NULL*, далее создаются две вспомогательные переменные *cur*(текущий) и *tm*р(временный) имеющие

тип *MusicalComposition**. Далее в цикле *for*, создаются элементы списка с помощью функции *createMusicalComposition()*, их помещают в переменную *cur*, полям *author, name и year* присваиваются соответствующие значения, а также задают предыдущему элементы указатель на следующий, а нынешнему на предыдущий. После цикла последнему элементы в поле *next* присваивается *NULL*. Функция возвращает головной указатель.

- Функция *push(MusicalComposition* head, MusicalComposition* element)*

  Функция идет до конца текущего списка с помощью цикла *while()*, после чего в поле *next* последнего элементы списка ставит указатель на переданный в функцию элемент, также переопределяет поля *prev* и *next* переданного элемента.

- Функция *removeEl(MusicalComposition* head, char* name_for_remove)*

  Функция идет по списку, пока не встретит элемент поле name которого совпадает с переданной строчкой, далее этот элемент удаляется из списка, это делается так: поле *next* предыдущего элемента изменяют с указателя на текущий элемент на указатель на следующий, а поле *prev* следующего элемента меняют на указатель на предыдущий элемент.

- Функция *count(MusicalComposition* head)*

  Функция идет по списку и подсчитывает количество элементов списка, функция возвращает количество элементов в списке.

- Функция *print_names(MusicalComposition* head)*

  Функция идет по списку и выводит на консоль поле *name* каждого элемента.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	7 Fields of Gold Sting 1993 In the Army Now Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Billie Jean Michael Jackson 1983 Seek and Destroy Metallica 1982 Wicked Game Chris Isaak 1989 Points of Authority Linkin Park 2000 Sonne Rammstein 2001 Points of Authority	Fields of Gold Sting 1993 7 8 Fields of Gold In the Army Now Mixed Emotions Billie Jean Seek and Destroy Wicked Game Sonne 7	Ответ верный.

Выводы.

Были изучены линейные списки на языке Си. Разработана программа, которая содержит двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface* - *в данном случае набор функций*) для работы со списком.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.c

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


struct MusicalComposition{
   char* name;
   char* author;
   int year;
   struct MusicalComposition* next;
   struct MusicalComposition* prev;
};
typedef struct MusicalComposition MusicalComposition;


MusicalComposition* createMusicalComposition(char* name, char* autor,int year)
{
   MusicalComposition* my=calloc(1,sizeof(MusicalComposition));
   my->name=calloc(100,sizeof(char));
   my->name=name;
   my->author=calloc(100,sizeof(char));
   my->author=autor;
   my->year=year;
   return my;
}

MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors,
int* array_years, int n)
{
   MusicalComposition* head=calloc(1,sizeof(struct MusicalComposition));
   head->author=array_authors[0];
   head->name=array_names[0];
   head->year=array_years[0];
   head->prev=NULL;
   MusicalComposition* cur;
   MusicalComposition* tmp=head;
   for (int i=1;i<n;i++)
   {
      cur=createMusicalComposition(array_names[i],array_authors[i],array_years[i]);
      tmp->next=cur;
      cur->prev=tmp;
      tmp=cur;
   }
   tmp->next=NULL;
```

```
    return head;
}

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

void removeEl(MusicalComposition* head, char* name_for_remove)
{
    while(strcmp(head->name,name_for_remove)!=0)
    {
        head=head->next;
    }
    head->prev->next=head->next;
    head->next->prev=head->prev;
}

int count(MusicalComposition* head)
{
    int k=0;
    while(head!=NULL)
    {
        k++;
        head=head->next;
    }
    return k;
}

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

}
```