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. 0304 Мажуга Д.Р

Преподаватель Чайка К.В

Санкт-Петербург

2021

Цель работы.

Изучить линейные списки и научиться с ними работать, создать основные функции для работы с линейными списками.

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.

Были использованы заголовочные файлы stdio.h, stdlib.h, string.h. Для работы с памятию применялись функции malloc и free. Для сравнения строк между собой — функция MusicalComposition, которая strcmp. Была представляет создана собой структура жлемент struct списка, содержащий ссылки на предыдущий элемент — previous, и на следующий — next.

Выполнение работы.

1) Подключаем необходимые заголовки, объявляем структуру struct MusicalComposition.

2)Создние функции MusicalComposition* createMusicalComposition(char* name, char* author, int year), которая динамически выделяет память под элемент списка и заполняет его данными.

3)Объявление Composition* createMusicalComposition(char** array_names, char** array_authors, int* array_years, int n), которая принимает на вход массивы с названиями, авторами и годами, а так же длины массивов. Функция создает двусвязный список, элементами которого являются данные из массивов.

4) Объявление функции void push(MusicalComposition* head, MusicalComposition* element), которая добавляет element в конец списка, т.к. первый элемент списка хранит указатель на последний, то перебора всех элементов не происходит.

5) Объявление функции void removeEl(MusicalComposition* head, char* name_for_remove), которая удаляет из двусвязного списка элемент с названием, содержащимся в строке name_for_remove. Т.к. в функции main хранится указатель на первый элемент списка, то его удаление осуществляется отдельно. Если удаляется не первый элемент, то указатели previous и next соседних элементов обновляются, а память освобождается.

6) Объявление функции int count(MusicalComposition* head), которя возвращает количество элементов в списке.

7) Объявление функции void print_names(MusicalComposition* head), которая выводит на экран названия композиций с новой строки каждое. Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования табл.1

№ 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	Ok

Выводы.

Были изучены двусвязные списки. Разработана программа, обрабатывающая композиции. Программа принимает на вход названия композиций, авторов, и года содания. Из этих данных создаётся список, с которым можно проводить такие операции, как вставка, удаление, подсчёт количества элементов, вывод всех названий на экран.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.c

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


// Описание структуры MusicalComposition
typedef struct MusicalComposition
{
   char* name;
   char* author;
   int year;

   struct MusicalComposition* next;
   struct MusicalComposition* previous;
} MusicalComposition;

// Создание структуры MusicalComposition

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

// Функции для работы со списком MusicalComposition

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

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

void removeEl(MusicalComposition* head, char* name_for_remove);

int count(MusicalComposition* head);

void print_names(MusicalComposition* head);


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;

}

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* buffer = head;

    for(int i = 1; i < n; i++)
    {
        buffer->next = createMusicalComposition(array_names[i], array_authors[i], array_years[i]);
        buffer->next->previous = buffer;
        buffer = buffer->next;
    }
    head->previous = buffer;

    return head;
}

void push(MusicalComposition* head, MusicalComposition* element)
{
    MusicalComposition* buffer = head->previous;
    buffer->next = element;
    element->next = NULL;
    element->previous = buffer;
    head->previous = element;
}

void removeEl(MusicalComposition* head, char* name_for_remove)
{
    if(!strcmp(head->name, name_for_remove)){
        if(head->next != NULL){
            MusicalComposition* buffer = head->previous;
            strncpy(head->name, buffer->name, 80);
            strncpy(head->author, buffer->author, 80);
            head->year = buffer->year;
            buffer->next->previous = NULL;
            head->previous = buffer->previous;
            free(buffer);
        }
        else
            free(head);

        return;
    }

    MusicalComposition* buffer = head->next;
    while(strcmp(name_for_remove, buffer->name) && buffer->next != NULL)
        buffer = buffer->next;
```

```
    if(head == NULL)
        return;

    buffer->previous->next = buffer->next;
    buffer->next->previous = buffer->previous;
    free(buffer);
}

int count(MusicalComposition* head){
    MusicalComposition* buffer = head;
    int cnt;

    for(cnt = 0; buffer != NULL; cnt++)
        buffer = buffer->next;

    return cnt;
}

void print_names(MusicalComposition* head){
    MusicalComposition* buffer = head;

    while(buffer != NULL)
    {
        printf("%s\n", buffer->name);
        buffer = buffer->next;
    }
}
```