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: Структуры данных, линейные списки

Студент гр. 0382 _____ Тихонов С. В.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Изучить новую структуру данных: двусвязные и односвязные списки. Освоить использование таких списков в практических задачах, а также их реализацию в 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. 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.

Выполнение работы.

Для удобства применения структуры Musical Composition в соответствии с заданием с помощью ключевого слова typedef был определен одноименный тип данных. Структура включает в себя: название композиции, имя автора, год написания, указатели на предыдущий и следующий элемент двунаправленного списка.

Далее была написана функция для создания элемента списка. Создается указатель на структуру, на который выделяется память с помощью функции malloc(). Потом в структуру копируется информация об авторе и названии песни, а также информация о годе выпуска. Переменные prev и next, которые являются указателями на предыдущий и следующий элементы списка принимают значение NULL. Возвращаемое значение - указатель на структуру.

Далее была реализована функция для создания списка заданной длины . Сначала создается указатель на структуру head, куда с помощью функции создания элемента списка заносятся значения первого элемента. Далее в цикле создаются новые элементы, которые связываются между собой с помощью указателей. Возвращаемое значение - указатель на структуру.

Функция push добавляет элемент типа MusicalComposition в конец списка.

Функция removeEl удаляет элемента с определенным названием композиции.

Функция count проходит по списку и считает его количество элементов и функция print_names, которая выводит названия композиций.

Тестирование

Результаты тестирования представлены в Таблице 1.

Таблица 1- результаты тестирования.

№ п/п	Входные данные	Результаты работы	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	Программа работает верно.

Вывод.

В ходе выполнения лабораторной работы были изучены особенности работы с двунаправленными списками на языке Си.

В результате была написана программа, обрабатывающая входные данные: создание, удаление, добавление, подсчет элементов двунаправленного списка композиций, их авторов и годов издания.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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


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


// Создание структуры MusicalComposition
MusicalComposition* createMusicalComposition(char* name, char* author, int year) {
    struct MusicalComposition* musical_comp =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    musical_comp->name = name;
    musical_comp->author = author;
    musical_comp->year = year;
    musical_comp->prev = NULL;
    musical_comp->next = NULL;
    return musical_comp;
}

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

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

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

void print_names(MusicalComposition* head) {
    while (head) {
        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;
}
```