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. 1382 _____Коренев Д.А.
Преподаватель _____Жангиров Т.Р.

Санкт-Петербург
2022

Цель работы.

Изучить структуру данных — линейные списки, научиться применять ее на практике.

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.

Выполнение работы.

Структура MusicalComposition имеет поля name — назвение композиции, authot — автор композиции/музыкальная группа, year — год создания, node — указатель на следующую структуру MusicalComposition — следующая композиция, prev — указатель на предыдущую структуру MusicalComposition — предыдущая композиция.

Функция createMusicalComposition принимает в качестве аргументов название композиции, автора композиции/музыкальную группу, год создания. Выделяет динамическую память, в которой создает структуру MusicalComposition и передает в поля соответствующие значения, возвращает ее.

Функция createMusicalCompositionList в качестве аргументов принимает массив названий композиций, авторов, года создания, количество композиций. Выделяет динамическую память для структуры MusicalComposition, переменная head хранит указатель на эту область памяти. При помощи цикла for создает необходимое количество композиций в линейный список, у элементов которого есть поля указывающие на предыдущую и следующую композиции. Возвращает указатель на первый элемент.

Функция push в качестве аргументов принимает указатель на первый элемент линейного списка, композицию. При помощи цикла while в переменную link передается указатель последнего элемента. Создается динамическая память, в которой будет хранится новая композиция, в поле node последнего элемента хранится указатель на добавленный элемент, а в поле prev добавленного элемента хранится указатель на последний элемент.

Функция removeEl в качестве аргументов принимает указатель на указатель первого элемента линейного списка, название композиции. Если первый элемент — тот, который нужно удалить, в переменную head

передается указатель на следующий элемент, удаляется композиция, в поле prev присваивается значение NULL. Если это не первый элемент, при помощи цикла do while перебираются элементы, и при нахождении нужного, он удаляется, полям node и prev соседних элементов присваиваются необходимые значения.

Функция count принимает в качестве аргументов указатель на первый элемент линейного списка, и прибавляет в переменную cnt единицу, пока указатель на следующую композицию не равен NULL. Возвращает количество композиций.

Функция print_names выводит название композиций пока указатель на следующую композицию не равен NULL.

Тестирование.

Тестирование представлено в табл. 1.

Таблица 1 – Тестирование

№ 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	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

	Linkin Park 2000 Sonne Rammstein 2001 Points of Authority		
2	5 The Night We met Lord Huron 2015 Minority Green Day 2000 Knee Socks Arctic Monkeys 2013 Young And Beatiful Lana Del Rey 2013 Magnolia Playboi Carti 2017 After Dark Mr.Kitty 2014 The Night We met	The Night We met Lord Huron 2015 5 6 Minority Knee Socks Young And Beatiful Magnolia After Dark 5	Программа работает верно

Вывод.

В ходе выполнения лабораторной работы было написан двунаправленный список музыкальных композиций и API для работы с ним. Я применил на практике знания связанные с такой структурой данных как линейный список.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


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


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

// Функции для работы со списком MusicalComposition
MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n){

    MusicalComposition* link = malloc(sizeof (MusicalComposition));
    MusicalComposition* head = link;

    for(int i = 0; i < n; i++){

        link->name = array_names[i];
        link->author = array_authors[i];
        link->year = array_years[i];
        MusicalComposition* this = link;

        if (i < n-1) {
            link->node = malloc(sizeof(MusicalComposition));
            link = link->node;
        } else {
            link->node = NULL;
        }
        link->prev = this;
    }

    return head;
}
```

```
// добавляет element  в конец списка musical_composition_list
void push(MusicalComposition* head, MusicalComposition* element){
    MusicalComposition* link = head;

    while (link->node){
        link = link->node;
    }


    link->node = malloc(sizeof (MusicalComposition));


    element->node = NULL;
    link->node = element;
    element->prev = link;
}



// удаляет элемент element списка, у которого значение name равно
значению  name_for_remove
void removeEl(MusicalComposition* *head, char* name_for_remove){

    MusicalComposition* link = (*head);

    //если это первый жлемент то head = следеющий элемент
    if (strcmp((*head)->name, name_for_remove) == 0){
        *head = (*head)->node;
        free((*head)->prev);
        (*head)->prev = NULL;
        printf("REMOVED\n");
    } else {
        //если это не первый то проверяем с указанием на следующий
элемент пока у следюущгео элемента node != NULL
        do{
            //проеверить name в следующем элементе
            if (strcmp(link->node->name, name_for_remove) == 0){
                //del link
                MusicalComposition* del_link = link->node;

                //change node link
                link->node = link->node->node;
                link->node->prev = link;

                //deliting
                free(del_link);
                break;
            } else {
                link = link->node;
            }
        }while(link->node);
    }
}


//возвращает количество элементов списка
int count(MusicalComposition* head){
```

```
    MusicalComposition* link = head;

    int cnt = 1;
    while(link->node != NULL){
        cnt++;
        link = link->node;
    }
    return cnt;
}


//Выводит названия композиций.
void print_names(MusicalComposition* head){
    MusicalComposition* cur = head;
    do{
        puts(cur->name);
        cur = cur->node;
    }while(cur != NULL);

}



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);


    //print_names_rev(head);


    for (int i=0;i<length;i++){
        free(names[i]);
        free(authors[i]);
    }
    free(names);
    free(authors);
    free(years);

    return 0;

}
```