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. 1304 _____ Заика Т.П

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Создание двунаправленного линейного списка, работа с линейным списком.

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.

Линейные списки

Структуры и функции

Указатель на структуру

Выполнение работы.

В ходе работы для решения поставленной задачи было принято создать двунаправленный список музыкальный композиций MusicalComposition и api для работы со списком. Для этого была создана структура элемента списка MusicalComposition, в которой помимо указанных полей были реализованы поля next и prev типа указателя на структуру MusicakComposition для создания двунаправленности списка. Функция createMusicalCompositionList позволяет нам создать двунаправленный список, функция push позволяет добавить в список элемент в конец списка, функция removeEl реализует удаление элемента по имени композиции из списка, функция count помогает понять количество элементов в списке, а функция print_names вывести все имена композиций из списка. Выполняя реализацию перечисленных функций в соответствии с текстом задания, получилось создать программу, выполняющую функционал двунаправленного списка музыкальных композиций.

Далее будет приведено описание переменных и функций, реализующих работу двунаправленного списка. В функции main написана некоторая последовательность команд для проверки работы списка, поэтому её и её переменные описывать не будем.

Переменные:

char* name — строка неизвестной длины (не больше 80 символов), название композиции

char* author — строка неизвестной длины (не больше 80 символов), автор композиции/музыкальная группа

int year — целое число, год создания

struct MusicalComposition* next — указатель на следующую структуру списка при создании элемента списка

struct MusicalComposition* prev — указатель на предыдущую структуру списка при создании элемента списка

MusicalComposition* cur — указатель на текущую структуру списка

MusicalCompostion* prev — указатель на предыдущую структуру списка

MusicalComposition* out — указатель на первую стркутуру в списке (используется только в функции createMusicalCompositionList)

MusicalComposition* tmp — указатель на текущую структуру списка (используется в функциях count и print_names, выбрано такое имя т.к. в данных функциях не используется двунаправленность списка)

Функции:

MusicalComposition* createMusicalComposition(char* name, char* author, int year) — функция для создания элемента списка типа MusicalComposition

MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n) — создает список музыкальных композиций MusicalCompositionList, в котором n - длина передаваемых массивов, array_names, array_authors, array_years - массивы названий, авторов и годов соответственно

void push(MusicalComposition* head, MusicalComposition* element) — добавляет element в конец списка musical_composition_list

void removeEl(MusicalComposition* head, char* name_for_remove) — удалаяет элеменет 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

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.	1 0 3 4 5 2 0 3 4 17 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	Успешный тест

Выводы.

Было исследовано, изучено создание двунаправленного линейного списка, работа с линейным списком, а также структуры, функции и указатели.

Разработана программа, выполняющая реализацию двунаправленного списка музыкальных композиций, а также api, позволяющий работать со списком: инициализировать список элементами с названием, автором и годом выпуска, добавлять новые элементы в список, удалять ненужные элементы по названию композиции, подсчитавать текущее количество элементов, и выводить на экран названия композиций в списке.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

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

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

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

	MusicalComposition*                    out                    =
createMusicalComposition(array_names[0],            array_authors[0],
array_years[0]);
	prev = out;

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

	return out;
}

void push(MusicalComposition* head, MusicalComposition* element){
	MusicalComposition* cur = head;
	while(cur->next){
		cur = cur->next;
	}
```

```
      element->prev = cur;
      cur->next = element;
}

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

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

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;

}
```