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 _____ Спасов Д.В.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы.

Изучение структуры список,а также углубленное изучение струтур,и их взаимодействия друг с другом. Пркатика работы с односвязными, двусвязными списками.

Задание.

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface - в данном случае набор функций*) для работы со списком.

Структура элемента списка (тип - MusicalComposition):

- name - строка неизвестной длины (гарантируется, что длина неможет быть больше 80 символов), название композиции.
- author - строка неизвестной длины (гарантируется, что длина неможет 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 первомуэлементу списка array_names (**array_names[0]**).
- поле **author** первого элемента списка соответствуетпервому элементу списка array_authors (**array_authors[0]**).
- поле **year** первого элемента списка соответствует первомуэлементу 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 количествоэлементов списка
- void print_names(MusicalComposition* head); //Выводит названиякомпозиций.

В функции main написана некоторая последовательность вызова команд для проверки работы вашего списка.

Функцию main менять не нужно.

Выполнение работы.

Все функции реализованы по условию задания.

1)При создание нового узла заполняются все поля,кроме полей указателей.

2)При инициализации списка в переменных указателях на структуры a и b хранятся соответственно текущее и предыдущее значение узла, таким образом ,чтобы список был полностью проинициализирован.

3)При добавление элемента ищется самый последний элемент,а дальше в поле указателя на следующей элемент этому элементу.

4)При удаление элементу,как и в инициализации ,хранится предыдущей итекущей узел списка,обеспечивающий таким образом целостность списка.

5)Подсчет и распечатывание списка выполняется аналогичным образом.

Тестирование.

Здесь результаты тестирования, которые помещаются на одну страницу.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.		`Fields of Gold`	Пример из условия.

	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	Sting 1993 7 8 Fields of Gold In the Army Now Mixed Emotions Billie Jean Seek and Destroy Wicked Game Sonne 7	

	Points o f Authority Linkin Park 2000 Sonne Rammstein 2001 Points o f Authority		
2.	2 Help Beatles 1976 Yesterday Beatles 1975 Ulet PH 2000	Help Beatles 1976 2 3 Help Yesterday Ulet 3	-
3.	2 Atl Dance 2015 Slava Vladimir 2017 XACKu Panelka	Atl Dance 2015 2 3 Atl Slava XACKu 3	-

	2019		

Выводы.

Были изучены основы работы со структурами,а также изучена работа со списком. Проведена работа с двусвязными и односвязными списками.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#define _CRT_SECURE_NO_WARNINGS


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


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



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



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* song_1;
    MusicalComposition* song_2;
    song_1 = head;
    for (int i = 1; i < n; i++)
    {
        song_2 = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        song_1->next = song_2;
        song_2->previous = song_1;
        song_1 = song_1->next;
    }
    return head;
}


void push(MusicalComposition* head, MusicalComposition* element)
{
    if (!head)
        head = element;
    else
    {
        while (head->next)
            head = head->next;
        head->next = element;
        element->previous = head;
    }
}


void removeEl(MusicalComposition* head, char* name_for_remove)
{
```

```
	MusicalComposition* elem;
	if(head)
		while(head->next)
		{
			if (!strcmp(head->name, name_for_remove))
			{
				elem = head;
				head = head->next;
				head->previous = elem->previous;
				free(elem);
				elem = head->previous;
				elem->next = head;
			}
			else
				head = head->next;
		}
}


int count(MusicalComposition* head)
{
	int songs_counter = 0;
	if (!head)
		return 0;
	while (head->next)
	{
		head = head->next;
		songs_counter++;
	}
	return songs_counter + 1;
}


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;

}
```