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. 9381 Колованов Р.А.

Преподаватель Берленко Т.А.

Санкт-Петербург

2020

Цель работы.

Реализация двунаправленного динамического списка и набора функций для работы с ним; получение навыков работы со структурами и динамической памятью.

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:
    1. *n* - длина массивов *array_names, array_authors, array_years.*
    2. поле name первого элемента списка соответствует первому элементу списка array_names (array_names[0]).

3. поле author первого элемента списка соответствует первому элементу списка array_authors (array_authors[0]).
4. 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 nameравно 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 начала создается структура, которая будет содержать информацию о элементе двусвязного списка:

Листинг 1.

```
typedef struct MusicalComposition MusicalComposition;
```

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

В данной структуре предназначение полей следующее:

- *name* – имя композиции (массив символов).
- *author* – автор композиции (массив символов).
- *year* – год создания композиции (число).
- *next* – указатель на следующий элемент списка (если равен NULL, то данный элемент является хвостом списка).
- *previous* – указатель на предыдущий элемент списка (если равен NULL, то данный элемент является головой списка).

Далее были реализованы следующие функции:

- *MusicalComposition* createMusicalComposition(char* name, char* author, int year);*

  Функция создает экземпляр структуры *MusicalComposition* в динамической памяти и возвращает указатель на него. В качестве аргументов принимает *name*, *author* и *year*. Строки name и author копируются в выделенную для них динамическую память, указатели на копии присваиваются полям *name* и *author*, значение *year* также присваивается полю *year*. Полям next и previous присваивается значение NULL.
- *MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n);*

Функция создает список из экземпляров структуры *MusicalComposition* и возвращает указатель на начало списка. Экземпляры создаются при помощи функции *createMusicalComposition()*. Элементы списка связываются со своими соседями при помощи инициализации полей *next* и *previous*. Значения для элементов списка берутся из массивов names, authors и years (их размер равен n), которые в свою очередь передаются в качестве аргументов.

Далее были реализованы функции для работы со списком:

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

  Функция добавляет в конец списка элемент element, который передается в качестве аргумента. Также в качестве аргумента принимается *head* – указатель на первый элемент списка. Функция пробегается до последнего элемента списка (элемент, у которого указатель *next* равен NULL) и полю *next* этого элемента присваивает *element*, а полю *previous* элемента *element* присваивает указатель на последний элемент списка.
- *void removeElement(MusicalComposition* head, char* name_for_remove);*

  Функция удаляет элемент списка с полем name, равным строке *name_for_remove,* которая передается в качестве аргумента. Также в качестве аргумента принимается *head* – указатель на первый элемент списка. Функция пробегается по списку и ищет такой элемент, строка *name* которого равна строке *name_for_remove.* Если такой элемент не найден (*head == NULL*) или элементом является первый элемент списка (*head->previous*

== *NULL*), то функция завершает работу. Если элемент найден, то происходит очистка памяти, занимаемой элементом, а связи между соседями переходят друг на друга.

- *int count(MusicalComposition* head);*

  Функция возвращает количество элементов в списке. В качестве аргумента принимает *head* – указатель на первый элемент списка. Функция пробегается по элементам до тех пор, пока не встретит конец списка (элемент, у которого указатель *next* равен NULL).

- *void print_names(MusicalComposition* head);*

  Функция пробегается по элементам списка до конца и подряд печатает имена композиций (поле *name* каждого элемента). В качестве аргумента принимает *head* – указатель на первый элемент списка.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
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	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

	Metallica 1982 Wicked Game Chris Isaak 1989 Points of Authority Linkin Park 2000 Sonne Rammstein 2001 Points of Authority	

Выводы.

Было изучена работа со структурами и динамической памятью; получены навыки работы с ними. Была разобрана реализация динамического двусвязного списка.

Разработана программа, в которой реализован двусвязный динамический список музыкальных композиций и набор функций для работы с этим списком:

- Добавление элемента в конец списка
- Удаление элемента по имени композиции
- Вывод названий хранящихся композиций
- Количество композиций в списке

Для хранения элементов списка использовались функции выделения и освобождения динамической памяти *malloc()* и *free()*. В программе использовались операторы if-else, циклы for и while, указатели и структуры.

Приложение А

Исходный код программы

Название файла: main.c

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

typedef struct MusicalComposition MusicalComposition;

struct MusicalComposition {
      char name[80];
      char author[80];
      int year;
      MusicalComposition* next;
      MusicalComposition* previous;
};

MusicalComposition* createMusicalComposition(char* name, char*
author, int year) {
      MusicalComposition* element =
malloc(sizeof(MusicalComposition));

      strcpy(element->name, name);
      strcpy(element->author, author);

      element->year = year;
      element->next = NULL;
      element->previous = NULL;

      return element;
}

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* temp = head;

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

      return head;
}

void push(MusicalComposition* head, MusicalComposition* element) {
```

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

	while (head->next != NULL) {
	head = head->next;
	}

	head->next = element;
	element->previous = head;
}

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

	if (head == NULL || head->previous == NULL) {
	// Не удаляем первый элемент, так как не можем изменить
переменную head в функции main()
	return;
	}

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

int count(MusicalComposition* head) {
	int count = 0;

	while (head != NULL) {
	head = head->next;
	count++;
	}

	return count;
}

void print_names(MusicalComposition* head) {
	while (head != NULL) {
	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;

}
```