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

Студентка гр. 3341 _____ Романов А. К.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Целью работы является освоение работы с линейными списками.

Для достижения поставленной цели требуется решить следующие задачи:

- ознакомиться со структурой «список»;
- ознакомиться со списком операций используемых для списков;
- изучить способы реализации этих операций на языке С;
- написать программу, реализующую двусвязный линейный список и решающую задачу в соответствии с индивидуальным заданием.

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

Элемент списка типа *MusicalComposition* состоит из:

- указателя на *char name* - названия композиции
- указателя на *char author* - автора композиции
- целого числа *year* - года создания композиции
- указателя на тип *MusicalComposition prev* - указателя на предыдущую композицию
- указателя на тип *MusicalComposition next* - указателя на следующую композицию

Функции:

- *MusicalComposition* createMusicalComposition(char* name, char* author,int year)* – принимает на вход имя композиции и ее исполнителя, а также год записи. Создает новый объект типа MusicalComposition, присваивая соответствующим полям нужные значения.
- *MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)* – принимает на вход список названий композиций, список авторов, список годов записи и количество элементов в каждом из них. Функция создает первый элемент списка при помощи функции *createMusicalComposition,* а затем аналогично n – 1 оставшихся элементов. Возвращает указатель на первый элемент списка.
- *void push(MusicalComposition* head, MusicalComposition* element)* – принимает на вход указатель на первый элемент списка и новый объект типа MusicalComposition. Через указатели *next* функция доходит от головного элемента до последнего (чей указатель *next* ссылается на NULL). После чего добавляется полученный объект, т.е. его полю *prev* присваивается указатель на последний элеммент списка, а полю *next* последнего элемента присваивается указатель на новый элемент.
- *void removeEl(MusicalComposition** head, char* name_for_remove)* – принимает на вход указатель на первый элемент списка и имя композиции,

которую следует удалить. Функция проверяет имена всех композиций на соответствие name_for_remove. Если искомый элемент первый, то head устанавливается на второй элемент, при этом освобождается память, выделеннаяпод первый элемент. Иначе, когда найден требуемый элемент (предположим, что его номер x), то элементу x – 1 в поле next указыается адрес элемента x + 1, а элементу x + 1 в поле prev указывется адрес элемента x – 1. Память, выделенная под элемент x также освобождается.

- *void print_names(MusicalComposition* head)* — принимает указатель на первый элемент списка. Проходит по всем его элементам и выводит имена композиций.
- *int count(MusicalComposition* head)* — принимает указатель на первый элемент списка. Выводит количество всех элементов списка.
- *int main()* - вызывает описанные функции в установленном условием порядке.

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.	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	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	Moevm confirmed

	2001 Points of Authority		
2.	4 All of the lights Kanye West 2010 Plug in baby Muse 2001 Paranoid android Radiohead 1997 Let it happen Tame Impala 2015 MX Deftones 1997 All of the lights	All of the lights Kanye West 2010 4 5 Plug in baby Paranoid android Let it happen MX 4	Проверяется удаление первого элемента
3.	4 Poetic justice Kendrick Lamar 2012 About a girl Nirvana 1989 Party monster The Weeknd 2017	Poetic justice Kendrick Lamar 2012 4 5 Poetic justice Party monster Holiday	Удаление более одного элемета

	Holiday Green Day 2004 Lovefool The Cardigans 1996 Lovefool About a girl		

Выводы

В ходе выполнения работы были изучены:

- основные принципы работы с линейными списками;
- структура списков и операции, применяемые к ним;
- способы реализации этих операций на языке С;
- написана программа, реализующаю двусвязный линейный список и решающаю задачу в соответствии с индивидуальным заданием.

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[81];
    char author[81];
    int year;
    struct MusicalComposition *next;
    struct MusicalComposition *prev;
} MusicalComposition;

int count(MusicalComposition* head);
void print_names(MusicalComposition* head);
void removeEl(MusicalComposition** head, char* name_for_remove);
void push(MusicalComposition* head, MusicalComposition* element);
MusicalComposition* createMusicalComposition(char* name, char* author,int
year);
MusicalComposition*   createMusicalCompositionList(char**   array_names,
char** array_authors, int* array_years, int n);

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;

}

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

    song->year = year;
    strncpy(song->name, name, 81);
    strncpy(song->author, author, 81);
    return song;
```

```
}

MusicalComposition*   createMusicalCompositionList(char**   array_names,
char** array_authors, int* array_years, int n){
    MusicalComposition*                    head_song                    =
createMusicalComposition(array_names[0],               array_authors[0],
array_years[0]);
    head_song->prev = NULL;
    head_song->next = NULL;
    MusicalComposition* previous_ptr = head_song;
    for(int i = 1; i < n; i++){
        MusicalComposition*                    song                    =
createMusicalComposition(array_names[i],               array_authors[i],
array_years[i]);
        song->prev = previous_ptr;
        song->next = NULL;
        previous_ptr->next = song;
        previous_ptr = song;

    }
    return head_song;
}

void push(MusicalComposition* head, MusicalComposition* element){
    MusicalComposition* ptr = head->next;
    while (1)
    {
        if (ptr->next == NULL){
            break;
        }
        ptr = ptr->next;
    }
    element->prev = ptr;
    element->next = NULL;
    ptr->next = element;

}

int count(MusicalComposition* head){
    int counter = 1;
    MusicalComposition* ptr = head->next;
    while (1)
    {
        counter++;

        if (ptr->next == NULL){
            break;
        }
        ptr = ptr->next;

    }
    return counter;
}

void removeEl(MusicalComposition** head, char* name_for_remove) {
    MusicalComposition* ptr = (*head)->next;
```

```
    if (strstr((*head)->name, name_for_remove)) {
        (*head)->next->prev = NULL;
        MusicalComposition *temp = *head;
        *head = (*head)->next;
        free(temp);
    } else {
        while (ptr != NULL) {
            if (strstr(ptr->name, name_for_remove)) {
                ptr->prev->next = ptr->next;
                if (ptr->next != NULL) {
                    ptr->next->prev = ptr->prev;
                }
                free(ptr);
                break;
            }
            ptr = ptr->next;
        }
    }
}

void print_names(MusicalComposition* head){
    MusicalComposition* ptr = head->next;
    puts(head->name);
    while (1)
    {
        puts(ptr->name);
        if (ptr->next == NULL){
            break;
        }
        ptr = ptr->next;
    }
}
```