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. 3342 _____ Гончаров С.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Разработать программу на языке С, реализовать двухсвязный линейный список.

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.

Выполнение работы

Разработана структура узла двусвязного списка. Создана функция, создающая экземпляр узла списка. Создана функция для создания всего списка, которая внутри себя вызывает функцию создания узла. Чтобы оперировать созданным списком, был разработан API, включающий следующие методы:

1) Подсчёт количества узлов в списке:

   Данная функция проходит весь список, пока не достигнет конца (NULL). Счетчик увеличивается с каждым элементом и возвращается после завершения перебора.

2) Добавление нового элемента в список:

   Метод проходит весь список, пока не достигнет конца, и добавляет переданный в функцию новый элемент в поле “next” последнего элемента. Поле “prev” нового элемента указывает на предыдущий последний элемент. Метод не возвращает значений.

3) удаление элемента списка:

   Функция перебирает элементы списка и при совпадении поля name с переданным в функцию аргументом, удаляет найденный элемент. Функция ничего не возвращает

4) вывод названий всех композиций из списка:

   Функция проходит весь список, пока не дойдёт до конца и выводит поле name каждого пройденного элемента. Функция ничего не возвращает.

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 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

	2001 Points of Authority	

Выводы

Был реализован двусвязный список, разработаны функции для работы с ним на языке программирования C.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


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


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

    if (!composition) {
        return NULL;
    }

    strncpy(composition->name, name, 80);
    strncpy(composition->author, author, 80);
    composition->year = year;
    composition->prev = NULL;
    composition->next = NULL;
    return composition;
}


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* current = head;

    for (int i = 1; i < n; i++) {
        MusicalComposition* composition =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        current->next = composition;
        composition->prev = current;
        current = composition;
    }
    return head;
}
```

```
void push(MusicalComposition* head, MusicalComposition* element) {
    MusicalComposition* p = head;
    while (p != NULL) {
        if (p -> next == NULL) {
            p -> next = element;
            element -> prev = p;
            break;
        }
        else{
            p = p->next;
        }
    }
}


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


void print_names(MusicalComposition* head) {
    MusicalComposition* p = head;
    while (p != NULL){
        printf("%s\n", (*p).name);
        p = p->next;
    }
}


int count(MusicalComposition* head) {
    int counter = 0;
    MusicalComposition* p = head;
    while (p != NULL) {
        counter++;
        p = p->next;
    }
    return counter;
}
```

```
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;

}
```