МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Программирование»
Тема: Линейные списки

Студентка гр. 3341 Мильхерт А.С.

Преподаватель Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Цель данной лабораторной работы заключается в создании двунаправленного списка музыкальных композиций с использованием структуры MusicalComposition и разработке набора функций (API) для управления этим списком. Для работы со списком в данной лабораторной работе необходимо реализовать ряд функций, таких как создание узла списка для новой музыкальной композиции с помощью функции createMusicalComposition, создание списка целого набора музыкальных композиций с помощью функции createMusicalCompositionList, добавление нового узла в конец списка с помощью функции push, удаление определенного узла по названию композиции с помощью функции removeEl, подсчет количества элементов в списке с помощью функции count и вывод на экран названий всех композиций в списке с помощью функции print_names.

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

Ниже представлен ход выполнения лабораторной работы.

1. Определение структуры данных MusicalComposition для хранения информации о музыкальной композиции, включая название, автора и год создания.

2. Создание функции createMusicalComposition для создания нового узла списка с данными о музыкальной композиции.

3. Создание функции createMusicalCompositionList для создания пустого двунаправленного списка музыкальных композиций, возвращающей указатель на его первый элемент.

4. Создание функции push для добавления нового узла с данными о музыкальной композиции в конец списка.

5. Создание функции removeEl для удаления узла с заданным названием музыкальной композиции из списка.

6. Создание функции count для подсчета количества элементов в списке музыкальных композиций.

7. Создание функции print_names для вывода на экран названий всех музыкальных композиций в списке.

Код программы – см. Приложение А.

Тестирование

Результаты тестирования представлены в табл. 1

Табл. 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 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	Тест e.moevm

Выводы

В результате выполнения данной лабораторной работы был разработан код для работы с двунаправленным списком музыкальных композиций.

1. Была создана структура данных MusicalComposition для хранения информации о музыкальной композиции, такой как название, автор и год создания.

2. Реализованы функции createMusicalComposition, createMusicalCompositionList, push для создания списка музыкальных композиций, добавления нового узла в список и создания новой композиции соответственно.

3. Также были написаны функции removeEl для удаления композиции по названию, count для подсчета количества элементов в списке и print_names для вывода названий композиций на экран.

Эта лабораторная работа позволила углубить понимание принципов работы с двунаправленным списком.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


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

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

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

}


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]);
    for (int i = 1; i < n; i++) {
        MusicalComposition*                element                 =
createMusicalComposition(array_names[i],          array_authors[i],
array_years[i]);
        push(head, element);
    }
    return head;
}

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

```
    element->prev = current;
}

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

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

void print_names(MusicalComposition* head) {
    MusicalComposition* current = head;
    while (current != NULL) {
        printf("%s\n", current->name);
        current = current->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;
```

```
}
```