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. Вариант 1

Студент гр. 3343 _____ Коршков А.А.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург
2024

Цель работы

Научиться работать со структурами, указателями на структуры, создавать линейные двунаправленные списки, писать функции для работы с ними.

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.

  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

Первое, что необходимо создать – структуру двунаправленного списка. Двунаправленный список содержит указатель как на следующий, так и на предыдущий элемент, а также информацию об имени, авторе и годе создания композиции.

Для создания элемента списка используется функция CreateMusicalComposition, на вход которой подаётся имя, автор и год создания музыкальной композиции. Создаём динамический массив для того чтобы сохранить получаемые данные.

Чтобы создать список музыкальных композиций используется createMusicalCompositionList, на вход подаются списки имён, авторов и годов и количество музыкальных композиций. Создаётся динамический массив из n элементов, в цикле добавляем элементы из списков и добавляем указатели на следующий и предыдущий элементы. Если элемент первый, то указатель на предыдущий элемент NULL, если элемент последний, то указатель на следующий элемент NULL.

В функции push ищется элемент списка, у которого нет указателя на следующий элемент (это последний элемент), ему присваивается указатель на добавляемый элемент, а у добавляемого элемента появляется указатель на последний элемент списка.

В функции removeEl чтобы удалить элемент, нужно у соседних элементов поменять указатели на следующий или предыдущий элемент, в зависимости, где эти соседние элементы стоят. Если элемент последний, то нужно у предпоследнего элемента сделать указатель на следующий элемент нулевым.

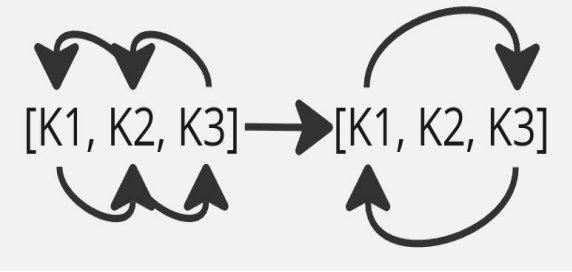

Рисунок 1 – Пример удаления элемента из списка

Функция count считает количество элементов в списке. Для этого мы считаем элементы, пока указатель на следующий элемент не ноль.

Функция print_names работает аналогично, только она не возвращает количество элементов, а выводит имена композиций, пока указатель не нулевой.

Выводы

Была создана программа, которая создаёт список музыкальных композиций, каждый элемент которой содержит информацию об имени, исполнителе и годе выхода композиции, а также указатель на следующий и предыдущий элемент. Были реализованы функции для добавления, удаления элементов, а также подсчёта количества композиций и выводе списка всех композиций.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

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

MusicalComposition *createMusicalComposition(char *name, char *autor, int
year)
{
    MusicalComposition *structure = malloc(sizeof(MusicalComposition));
    structure->name = name;
    structure->author = autor;
    structure->year = year;
    return structure;
}

MusicalComposition *createMusicalCompositionList(char **array_names, char
**array_authors, int *array_years, int n)
{
    MusicalComposition *root = malloc(sizeof(MusicalComposition) * n);
    for (int i = 0; i < n; i++)
    {
        root[i].name = array_names[i];
        root[i].author = array_authors[i];
        root[i].year = array_years[i];
        if (i == 0)
        {
            root[i].prev = NULL;
            root[i].next = &root[i + 1];
        }
        if (i == n - 1)
        {
            root[i].next = NULL;
            root[i].prev = &root[i - 1];
        }
        else
        {
            root[i].next = &root[i + 1];
            root[i].prev = &root[i - 1];
        }
    }
    return root;
}

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

```
{
    MusicalComposition *old_element = head;
    while (old_element->next != NULL)
    {
        old_element = old_element->next;
    }
    element->prev = old_element;
    old_element->next = element;
}

void removeEl(MusicalComposition *head, char *name_for_remove)
{
    MusicalComposition *element = head;
    while (element != NULL)
    {
        if ((strstr(name_for_remove, element->name) != NULL) &&
(strlen(name_for_remove) == strlen(element->name)))
        {
            if (element->next == NULL)
            {
                element->prev->next = NULL;
            }
            else
            {
                element->next->prev = element->prev;
                element->prev->next = element->next;
            }

        }
        element = element->next;
    }
}

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

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

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Тест № 1.

Входные данные:

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

Выходные данные:

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 № 2.

Входные данные:

4

Memories

Thutmose

2019

Way Up

Jaden

2019

Grenade

Bruno Mars

2010

Attention

Charlie Puth

2018

Bring Me To Life

Evanescence

2003

Attention

Выходные данные:

Memories Thutmose 2019

4

5

Memories

Way Up

Grenade

Bring Me To Life

4