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: Лабораторная работа № 2: Линейные списки

Студент гр. 3343 _____ Пименов П.В.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург
2024

Цель работы

Изучить принцип работы двусвязного (двунаправленного) списка. Создать программу на языке C, реализующую двусвязный список музыкальных композиций.

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

Описание структур:

- MusicalComposition
  - char *name – название композиции, строка произвольной длины
  - char *author – имя автора, строка произвольной длины
  - int year – год создания композиции, целое число
  - struct MusicalComposition *left – указатель на прошлый элемент двусвязного списка
  - struct MusicalComposition *right – указатель на следующий элемент двусвязного списка

Описание функций для работы со списком:

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

- Функция принимает на вход название композиции, имя автора, год создания композиции, создает экземпляр структуры, возвращает указатель на него.

- MusicalComposition *createMusicalCompositionList(char **array_names, char **array_authors, int *array_years, int n)
  - Функция принимает на вход массивы названий композиций, имен авторов, годов создания, и размер этих массивов. Создает двусвязный список, возвращает указатель на первый элемент списка.
- void push(MusicalComposition *head, MusicalComposition *element)
  - Функция принимает на вход указатель на первый элемент двусвязного списка и указатель на экземпляр структуры MusicalComposition. Добавляет этот экземпляр в конец списка.
- void removeEl(MusicalComposition *head, char *name_for_remove)
  - Функция принимает на вход указатель на первый элемент двусвязного списка и название композиции, которую надо удалить из списка. Функция ищет элемент в списке с соответствующим названием и удаляет его из списка.
- int count(MusicalComposition *head)
  - Функция принимает на вход указатель на первый элемент двусвязного списка. Возвращает количество элементов в списке.
- void print_names(MusicalComposition *head)
  - Функция принимает на вход указатель на первый элемент двусвязного списка. Выводит в поток вывода названия композиций из списка, разделенные символом перевода строки.

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	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	Программа работает корректно.

	Sonne Rammstein 2001 Points of Authority		

Выводы

Был изучен принцип работы двусвязного (двунаправленного) списка. Создана программа на языке С, реализующую двусвязный список музыкальных композиций.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

typedef struct MusicalComposition
{
    char *name;
    char *author;
    int year;
    struct MusicalComposition *left;
    struct MusicalComposition *right;
} MusicalComposition;

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

MusicalComposition *createMusicalComposition(char *name, char *author,
int year)
{
    MusicalComposition *composition = (MusicalComposition
*)malloc(sizeof(MusicalComposition));
    composition->name = name;
    composition->author = author;
    composition->year = year;
    composition->left = NULL;
    composition->right = NULL;
    return composition;
}

MusicalComposition *createMusicalCompositionList(char **array_names,
char **array_authors, int *array_years, int n)
{
    if (n <= 0)
    {
        return NULL;
    }
    MusicalComposition *head =
createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
    for (int i = 1; i < n; ++i)
    {
```

```
        MusicalComposition *new =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        push(head, new);
    }
    return head;
}

void push(MusicalComposition *head, MusicalComposition *element)
{
    MusicalComposition *temp = head;
    if (temp != NULL)
    {
        while (temp->right != NULL)
        {
            temp = temp->right;
        }
        temp->right = element;
        element->left = temp;
    }
}

void removeEl(MusicalComposition *head, char *name_for_remove)
{
    MusicalComposition *temp = head;
    if (!strcmp(temp->name, name_for_remove)) {
        return;
    }
    while (temp != NULL)
    {
        if (!strcmp(temp->name, name_for_remove))
        {
            if (temp->left != NULL)
            {
                temp->left->right = temp->right;
            }
            if (temp->right != NULL)
            {
                temp->right->left = temp->left;
            }
            free(temp);
            temp = NULL;
            break;
        }
        temp = temp->right;
    }
}

int count(MusicalComposition *head)
{
    MusicalComposition *temp = head;
    int count = 0;
    while (temp != NULL)
    {
        count += 1;
        temp = temp->right;
```

```
    }
    return count;
}

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

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;
}
```