**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  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 | Программа работает корректно. |

## Выводы

Был изучен принцип работы двусвязного (двунаправленного) списка. Создана программа на языке C, 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;

}