**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. 3344 |  | Коршунов П.И. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Освоение работы с линейными списками на языке Си на примере использующей их программы.

## 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

Была подключена стандартная библиотека для ввода и вывода *<stdio.h>*, стандартная библиотека *<stdlib.h>*, библиотека для работы со строками *<string.h>.* Была создана структура *MusicalComposition* с полями названия *char \*name,* автора *char \*author,* года выпуска *int year,* указателями на предыдущий и следующий элемент двусвязного списка *MUSCOMP \*prev* и *MUSCOMP \*next.* Была создана функция *MUSCOMP \*createMusicalCompositionList(char \*\*array\_names, char \*\*array\_authors, int \*array\_years, int n)* для заполнения линейного списка, которое происходит в цикле, где мы создаем для элемента структуру, в которой мы заполняем указатели на предыдущий и следующий элементы и другие поля с информацией о песнях. Была создана функция *void push(MUSCOMP \*head, MUSCOMP \*element)* для добавления нового элемента в конец списка. Циклом *while* доходили до последнего элемента и изменяли его указатель на следующий элемент, а также заполняли поля указатели нового элемента. Была реализована функция *void removeEl(MUSCOMP \*head, char \*name\_for\_remove),* которая удаляет элемент списка, у которого значение *name* равно значению *name\_for\_remove.* В функции циклом проходим по всем элементам списка, если значение *name* равно значению *name\_for\_remove*, то указатель предыдущего элемента для текущего становится указателем предыдущего для следующего для текущего, аналогично для указателя следующего элемента, также очищаем память для элемента. В функции *int count(MUSCOMP \*head)* возвращаем количество элементов списка, проходим циклом *while* по всем элементам и увеличиваем переменную-счетчик каждую итерацию. В функции *void print\_names(MUSCOMP \*head)* происходит вывод названий всех композиций на экран, проходим циклом *while* по всем элементам и выводим название песни каждую итерацию.

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 |
|  | 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 | - |

## Выводы

Была освоена работа с линейными списками на языке Си на примере использующей их программы.

# Приложение А Исходный код программы

Название файла: Korshunov\_Petr\_lb2.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

typedef struct MusicalComposition MUSCOMP;

// Описание структуры MusicalComposition

struct MusicalComposition

{

char \*name;

char \*author;

int year;

MUSCOMP \*prev;

MUSCOMP \*next;

};

// Создание структуры MusicalComposition

MUSCOMP \*createMusicalComposition(char \*name, char \*autor, int year);

// Функции для работы со списком MusicalComposition

MUSCOMP \*createMusicalCompositionList(char \*\*array\_names, char \*\*array\_authors, int \*array\_years, int n);

void push(MUSCOMP \*head, MUSCOMP \*element);

void removeEl(MUSCOMP \*head, char \*name\_for\_remove);

int count(MUSCOMP \*head);

void print\_names(MUSCOMP \*head);

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);

}

MUSCOMP \*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;

MUSCOMP \*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;

}

MUSCOMP \*createMusicalComposition(char \*name, char \*autor, int year)

{

MUSCOMP \*mus = (MUSCOMP \*)malloc(sizeof(MUSCOMP));

if (!mus)

{

printf("Error: failed to allocate memory");

exit(0);

}

mus->name = name;

mus->author = autor;

mus->year = year;

return mus;

}

MUSCOMP \*createMusicalCompositionList(char \*\*array\_names, char \*\*array\_authors, int \*array\_years, int n)

{

MUSCOMP \*head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);

head->prev = NULL;

MUSCOMP \*prev = head;

for (int i = 1; i < n; i++)

{

MUSCOMP \*temp = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

prev->next = temp;

temp->prev = prev;

prev = temp;

}

prev->next = NULL;

return head;

}

void push(MUSCOMP \*head, MUSCOMP \*element)

{

MUSCOMP \*temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = element;

element->prev = temp;

element->next = NULL;

return;

}

void removeEl(MUSCOMP \*head, char \*name\_for\_remove)

{

MUSCOMP \*temp = head;

while (temp->next != NULL)

{

if (strcmp(temp->name, name\_for\_remove) == 0)

{

MUSCOMP \*before = temp->prev;

MUSCOMP \*after = temp->next;

before->next = after;

after->prev = before;

free(temp);

temp = after;

}

else

{

temp = temp->next;

}

}

}

int count(MUSCOMP \*head)

{

MUSCOMP \*temp = head;

int c = 1;

while (temp->next != NULL)

{

++c;

temp = temp->next;

}

return c;

}

void print\_names(MUSCOMP \*head)

{

MUSCOMP \*temp = head;

while (temp->next != NULL)

{

printf("%s\n", temp->name);

temp = temp->next;

}

printf("%s\n", temp->name);

}