**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3343 |  | Лобова Е. И. |
| Преподаватель |  | Государкин Я.С. |

Санкт-Петербург

2024

## Цель работы

Целью работы является освоение работы с линейными списками.

Для достижения поставленной цели требуется решить следующие задачи:

1. Ознакомиться со структурой данных «список».

2. Ознакомиться с операциями, используемыми для списков.

3. Изучить способы реализации этих операций на языке Cи.

4. Написать программу, реализующую двусвязный линейный список и решающую задачу в соответствии с индивидуальным заданием.

## 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

Для решения задания лабораторной работы была создана структура MusicalComposition с полями char name[80], char author[80], int year и с полями, указывающими на предыдущий и последующий элементы.

Функции, реализованные в программе:

* Функция *MusicalComposition\* createMusicalComposition(char\* name, char\* author, int year)* выделяет память под структуру, заполняет поля данными поданными в функцию и возвращает указатель на созданную структуру;
* Функция *MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)* принимает на вход три массива с данными для соответствующих полей структур и длину этих массивов, создает голову списка и далее при помощи цикла заполняет хвост списка, связывая каждый элемент с предыдущим и последующим. Возвращает указатель на голову списка.
* Функция *void push(MusicalComposition\* head, MusicalComposition\* element)* принимает на вход указатель на голову списка и элемент, который нужно добавить в конец списка. Делается проверка на ненулевой указатель головы (если нулевой, то элемент становится головой) и с помощью цикла while проходится до последнего элемента, который связывается с переданным вторым аргументом в функцию.
* Функция *void removeEl(MusicalComposition\* head, char\* name\_for\_remove)* принимает на вход указатель на голову списка и название элемента, который нужно удалить. Делается проверка, что голова не заданный для удаления элемент и с помощью цикла while делается проход до конца списка. Если найден заданный элемент, то поля previous и next соседних элементов списка меняются, а память выделенная под удаленный элемент очищается.
* Функция *int count(MusicalComposition\* head)* принимает на вход указатель на голову списка, создаёт переменную count, которую впоследствии и возвращает. При помощи цикла while переменная для подсчета количества элементов увеличивается, пока текущий указатель на элемент ненулевой.
* Функция *void print\_names(MusicalComposition\* head)* принимает на вход указатель на голову списка и при помощи цикла while делается проход по списку, а имя каждого элемента выводится.

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 | Выходные данные корректны. |

## Выводы

Были изучены такие моменты, как структура данных «список» и операции, используемые для них, а также их реализация на языке Си. Была написана программа, реализующая двусвязный линейный список.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

typedef struct MusicalComposition{

char name[80];

char author[80];

int year;

struct MusicalComposition\* next;

struct MusicalComposition\* previous;

}MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition\*

createMusicalComposition(char\* name, char\* author, int year){

MusicalComposition \*cur = (MusicalComposition\*)malloc(sizeof(MusicalComposition));

strcpy(cur->name,name);

strcpy(cur->author,author);

cur->year = year;

cur->next = NULL;

cur->previous = NULL;

return cur;

}

// Функции для работы со списком MusicalComposition

MusicalComposition\*

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

MusicalComposition\* head = NULL;

MusicalComposition\* list = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);

list->next = NULL;

list->previous = NULL;

head = list;

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

list->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

MusicalComposition\* previous = list;

list = list->next;

list->next = NULL;

list->previous = previous;

}

return head;

}

void

push(MusicalComposition\* head, MusicalComposition\* element){

MusicalComposition\* cur = head;

if (head==NULL){

head = element;

return;

}

while(cur->next!=NULL){

cur = cur->next;

}

cur->next = element;

element->next = NULL;

}

void

removeEl(MusicalComposition\* head, char\* name\_for\_remove){

MusicalComposition\* cur = head;

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

return;

}

while (cur!=NULL){

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

MusicalComposition\* previous = cur->previous;

if (cur->previous!=NULL){

cur->previous->next = cur->next;

}

if (cur->next!=NULL){

cur->next->previous = previous;

}

free(cur);

cur = NULL;

break;

}

cur = cur->next;

}

}

int

count(MusicalComposition\* head){

MusicalComposition\* cur = head;

int count = 0;

while(cur!=NULL){

count++;

cur = cur->next;

}

return count;

}

void

print\_names(MusicalComposition\* head){

MusicalComposition\* cur = head;

while(cur!=NULL){

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

cur = cur->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;

}