**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. 3342 |  | Колесниченко М.А. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является изучение принципов работы с линейными списками и реализация их в программе на языке C.

## 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.

## Выполнение работы

Была описана структура узла двусвязного списка. Далее разработана функция, выделяющая память и создающая экземпляр узла списка. Также была реализована функция для создания всего списка, которая внутри себя вызывает функцию создания узла.

Для работы с созданным списком был реализован API, в котором предусмотрены следующие функции:

1. подсчёт количества узлов в списке:

Функция проходит весь список до момента, пока элемент не станет равным NULL (конечный). Счётчик увеличивается с каждым элементом и возвращается после завершения перебора элементов.

1. функция для добавления нового элемента списка:

Функция проходит весь список, пока не дойдет до конечного элемента и присваивает в его поле next переданный в функцию новый элемент. У нового элемента поле prev = предыдущему конечному элементу. Функция ничего не возвращает.

1. функция для удаления элемента списка:

Функция перебирает элементы списка и при совпадении поля name с переданным в функцию аргументом, удаляет найденный элемент. Функция ничего не возвращает

1. функция, выводящая названия всех композиций из списка:

Функция проходит весь список, пока не дойдёт до конца и выводит поле name каждого пройденного элемента. Функция ничего не возвращает.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные 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 | 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

Был реализован двусвязный список, разработаны функции для работы с ним на языке программирования С.

# Приложение А Исходный код программы

Название файла: main.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition {

char name[81];

char author[81];

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

} MusicalComposition;

// Создание структуры MusicalComposition

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

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

if (composition==NULL){

exit(1);

}

strncpy(composition->name, name, 80);

composition->name[80] = '\0';

strncpy(composition->author, author, 80);

composition->author[80] = '\0';

composition->year = year;

composition->prev = NULL;

composition->next = NULL;

return composition;

}

// Функции для работы со списком MusicalComposition

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

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

MusicalComposition\* current = head;

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

MusicalComposition\* composition = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

current->next = composition;

composition->prev = current;

current = composition;

}

return head;

}

int count(MusicalComposition\* head){

int i = 0;

MusicalComposition\* p = head;

while (p != NULL){

i++;

p = p->next;

}

return i;

}

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

MusicalComposition\* p = head;

while (p != NULL){

if (p -> next == NULL){

p -> next = element;

element -> prev = p;

break;

}

else{

p = p->next;

}

}

}

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

MusicalComposition\* p = head;

while (p != NULL){

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

if (p == head) {

head = p->next;

if (head != NULL) {

(head)->prev = NULL;

}

} else {

p->prev->next = p->next;

if (p->next != NULL) {

p->next->prev = p->prev;

}

}

free(p);

break;

}

else{

p = p->next;

}

}

}

void print\_names(MusicalComposition\* head){

MusicalComposition\* p = head;

while (p != NULL){

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

p = p->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;

}