**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. 3342 |  | Епонишникова А.И |
| Преподаватель |  | Глазунов С.А |

Санкт-Петербург

2024

## Цель работы

Целью работы является на практике изучить работу с линейными списками и их использование в языке программирования Си

## 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

Структура MusicalComposition:

Включает в себя поля: name, author, year, next и prev. next и prev – указатели на последующий и предыдущий элемент списка.

createMusicalComposition:

Создание музыкальной композиции (название, автор, год). Возвращает указатель на композицию.

createMusicalCompositionList:

Создание головного элемента, заполняет его поля, затем через цикл for заполняет список количеством элементов, которое подается в качестве аргумента. Функция возвращает указатель на головной элемент списка.

Push:

Функции в качестве аргументов подаются указатель на головной элемент списка и элемент, который надо вставить в конце. Текущему элементу подается указатель на головной элемент списка. Сначала происходит проверка первый элемент NULL или нет. Если нет, то с помощью while переместиться к концу списка, затем изменение поля на значение указателя, вставляемого элемента.

RemoveEl:

Функции в качестве аргументов подаются указатель на головной элемент списка и name\_for\_remove. Используя while, нужно пройтись по всему списку, пока не найдется name, которое надо удалить. Когда элемент найден, переустанавливаются указатели. Если элемент был последним, то предыдущему от текущего устанавливается указатель на следующий, который равен NULL.  
Если элемент был первым, то последующему от текущего устанавливается указатель на предыдущий, который будет равен NULL. Если элемент в средине списка, то предшествующему элементу устанавливается указатель на следующий элемент после удаляемогу, а для следующего – на предыдущий от удаляемого. Далее элемент очищается.

Count:

Создается переменная count, который и будет считать количество элементов в списке. Текущий элемент устанавливается на головной элемент списка. С помощью while происходит движение по списку, каждый раз счетчик увеличивается, пока указатель на следующий элемент не будет равен NULL

Print\_names:

Текущий элемент устанавливается на головной элемент списка. С помощью while выводятся поля name элементов списка.

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  200  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

На практике научились работать с линейными списки, а также поняли различия между списками и массивами. Была реализована программа с использованиями двусвязного линейного списка.

# Приложение А Исходный код программы

Название файла: lab1.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

typedef struct MusicalComposition {

char\* name;

char\* author;

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

}MusicalComposition;

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

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

if(composition == NULL){

printf("Memory error");

exit(0);

}

composition->name = name;

composition->author = author;

composition->year = year;

composition->next = NULL;

composition->prev = NULL;

return composition;

}

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 \*curr = head;

MusicalComposition \*next;

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

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

curr->next = next;

next->prev = curr;

curr = next;

}

return head;

}

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

MusicalComposition \*curr = head;

if(head == NULL){

return;

}

else{

while(curr->next!=NULL){

curr = curr->next;

}

curr->next = element;

element->prev = curr;

}

}

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

MusicalComposition\* curr = head;

while(curr!=NULL){

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

if(curr->next == NULL){

curr->prev->next = NULL;

}

else if(curr->prev == NULL){

curr->next->prev = NULL;

}

else{

curr->prev->next=curr->next;

curr->next->prev = curr->prev;

}

free(curr);

}

curr = curr->next;

}

}

int count(MusicalComposition\* head){

int count = 0;

MusicalComposition \*curr = head;

while(curr!= NULL){

count++;

curr = curr->next;

}

return count;

}

void print\_names(MusicalComposition\* head){

MusicalComposition \*curr = head;

while(curr!= NULL){

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

curr = curr->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);

if(names == NULL || authors == NULL || years == NULL){

printf("Memory error");

exit(0);

}

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));

if(names[i] == NULL || authors[i] == NULL){

printf("Memory error");

exit(0);

}

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;

}