**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. 3341 |  | Мокров И.О. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Цель работы заключается в том, чтобы овладеть навыками работы с линейными списками на языке программирования C. Это включает в себя изучение структур данных в языке C и их применение на практике через создание двунаправленного списка музыкальных композиций.

В процессе выполнения работы студент изучит создание структур данных, функции для управления списком (API), а также основные операции с ними, такие как добавление, удаление и перебор элементов списка.

Задание также требует использования динамического выделения памяти и работы с указателями. Результатом успешного выполнения работы будет освоение принципов работы с линейными структурами данных и их применение для решения конкретной задачи.

## Задание

1 вариант.

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.

## Основные теоретические положения

Линейные двунаправленные списки в Си представляют собой структуры данных, где каждый элемент содержит не только указатель на следующий элемент, но и на предыдущий. Такая двунаправленность позволяет обходить список как в прямом, так и в обратном направлении. Каждый элемент списка, помимо данных, содержит указатели на следующий и предыдущий элементы, а начало списка определяется указателем на первый элемент, а конец списка - на последний.

Операции над двунаправленными списками включают добавление и удаление элементов как в начало, так и в конец списка, а также поиск и обход элементов. При добавлении или удалении элементов обновляются указатели на следующий и предыдущий элементы, чтобы сохранить целостность списка. Такие списки обеспечивают быстрый доступ как к началу, так и к концу списка, что делает их эффективными для множества задач, таких как реализация очередей, двусторонних стеков и других структур данных.

## Выполнение работы

Для начала, была объявлена структура MusicalComposition, представляющая собой элемент списка. Эта структура содержит поля для хранения названия композиции (name), имени автора (author) и года создания (year). Особенностью данной структуры являются указатели на следующий и предыдущий элементы списка (next и prev соответственно), что позволяет реализовать двунаправленный список.

Далее была реализована функция createMusicalComposition, которая создает новый элемент списка на основе переданных ей параметров: названия, автора и года. Функция выделяет память под новый элемент и инициализирует его поля переданными значениями.

Функция createMusicalCompositionList создает двунаправленный список музыкальных композиций на основе переданных массивов с названиями, авторами и годами композиций. Она последовательно создает элементы списка, связывая их указателями next и prev таким образом, чтобы обеспечить двунаправленность списка.

Функция push добавляет новый элемент в конец списка. Она перемещается по списку до его последнего элемента и устанавливает указатель next последнего элемента на новый элемент, обновляя также указатель prev нового элемента на предыдущий.

Функция removeEl удаляет элемент списка с заданным названием. Она перебирает элементы списка, сравнивая названия, и при нахождении удаляемого элемента корректно обновляет указатели соседних элементов.

Функция count возвращает количество элементов в списке, просто перебирая его и подсчитывая элементы.

Наконец, функция print\_names выводит названия всех композиций в списке, последовательно проходя по элементам и печатая их названия.

Программа использует указатели для эффективной работы с динамической памятью. Указатели представляют собой переменные, которые содержат адреса памяти других переменных или объектов. В данной программе они играют ключевую роль в создании и управлении списком музыкальных композиций.

В структуре MusicalComposition, указатели next и prev связывают элементы списка, обеспечивая двунаправленность. next указывает на следующий элемент, а prev - на предыдущий. Это позволяет эффективно перемещаться по списку в обоих направлениях.

Функция createMusicalComposition возвращает указатель на созданный элемент списка. Здесь указатель используется для передачи адреса выделенной памяти в качестве возвращаемого значения.

Функция createMusicalCompositionList создает список музыкальных композиций на основе переданных массивов с данными о композициях. Она также возвращает указатель на первый элемент списка, что позволяет программе начать работу с созданным списком.

Функция push использует указатель для добавления нового элемента в конец списка. Она перемещается по списку до последнего элемента с помощью указателя next и добавляет новый элемент, устанавливая соответствующие указатели.

Функция removeEl также использует указатели для удаления элемента из списка. Она перемещается по списку, сравнивая имена композиций, и обновляет указатели соседних элементов, чтобы они обходили удаляемый элемент.

Все эти функции работают с указателями, чтобы обеспечить эффективное управление памятью и операциями над списком. Правильное использование указателей позволяет эффективно работать с динамической памятью и управлять списком, даже если его размер меняется во время выполнения программы.

## Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 4  Mixed Emotions  The Rolling Stones  1989  Billie Jean  Michael Jackson  1983  Wicked Game  Chris Isaak  1989  Points of Authority  Linkin Park  2000  Sonne  Rammstein  2001  Points of Authority | Mixed Emotions The Rolling Stones 1989  4  5  Mixed Emotions  Billie Jean  Wicked Game  Sonne  4 | Пример корректной  работы программы: добавление элементов, удаление элементов |
|  | 2  Fields of Gold  Sting  1993  Points of Authority  Linkin Park  2000  Sonne  Rammstein  2001  Points of Authority | Fields of Gold Sting  1993  2  3  Fields of Gold  Sonne  2 | Пример корректной работы программы:  добавление элементов, удаление элементов |

## Выводы

В этой лабораторной работе мы освоили работу с линейными списками на языке Си на примере использующей их программы. Так же мы ознакомились с реализацией линейных списков при помощи структур.

Результатом выполненной работы стала программа, которая при помощи линейных списков и структур обрабатывает текст, содержащий музыкальные композиции.

# 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\* next;

struct MusicalComposition\* prev;

}MusicalComposition;

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

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

composition->name = name;

composition->author = author;

composition->year = year;

return composition;

}

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

MusicalComposition\* list[n];

for(int i = 0; i<n; i++) {

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

}

list[0]->prev = NULL;

list[0]->next = list[1];

int last\_el = n-1;

list[last\_el]->next = NULL;

list[last\_el]->prev = list[n-2];

for(int i = 1; i<last\_el; i++) {

list[i]->prev=list[i-1];

list[i]->next=list[i+1];

}

return list[0];

}

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

while(head->next != NULL){

head = head->next;

}

head->next = element;

element->prev = head;

element->next = NULL;

}

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

MusicalComposition\* pointer = head;

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

head->next->prev = NULL;

head = head->next;

}

while(1) {

pointer = pointer->next;

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

if(pointer->next != NULL) pointer->next->prev = pointer->prev;

pointer->prev->next = pointer->next;

}

if(pointer->next == NULL) break;

}

}

int count(MusicalComposition\* head){

int count = 0;

MusicalComposition\* pointer = head;

while(1) {

count++;

if(pointer->next == NULL) break;

pointer = pointer->next;

}

return count;

}

void print\_names(MusicalComposition\* head){

MusicalComposition\* pointer = head;

while(1) {

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

if(pointer->next != NULL){

pointer = pointer->next;

} else break;

}

}

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;

}