**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

**Тема: Лабораторная работа № 1. Линейные списки.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3343 |  | Синицкая Д.В. |
| Преподаватель |  | Государкин Я. С. |

Санкт-Петербург

2024

## Цель работы

Изучение и применение двунаправленных линейных списков в языке программирования Cи для хранения структур данных. Получение умений создания двунаправленных линейных списков в Cи и успешное применение функций взаимодействия с ними (добавления элементов, удаления элементов и т.п.).

## Задание

Вариант 1. 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

Создание структуры MusicalComposition: структура содержит поля для имени композиции, имени автора и года создания.

Создание функции createMusicalComposition(char\* name, char\* author, int year): функция выделяет память под новую музыкальную композицию, копирует переданные данные и возвращает указатель на композицию.

Создание функции createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n): функция создает список композиций на основе переданных данных и возвращает указатель на начало списка.

Метод добавления нового элемента в конец списка push(MusicalComposition\* head, MusicalComposition\* element): функция перемещает указатель в конец списка и добавляет новый элемент.

Метод удаления элемента из списка по названию removeEl(MusicalComposition\* head, char\* name\_for\_remove): функция перебирает элементы списка, находит элемент по названию и освобождает память, занимаемую элементом.

Метод подсчета количества элементов в списке count(MusicalComposition\* head): функция перебирает элементы списка и возвращает количество элементов.

Метод вывода названий композиций из списка print\_names(MusicalComposition\* head): функция выводит названия композиций из списка.

В функции main: считываются данные о композициях и заполняются массивы. Создается список музыкальных композиций. Добавляется новая композиция в список. Удаляется композиция по названию и выводятся результаты операций. Освобождается выделенная память.

Разработанный программный код см. в приложении А.

## Выводы

В процессе выполнения лабораторной работы мной были освоены навыки необходимые для создания двунаправленных линейных списков на языке Cи, а также умения взаимодействовать с ними (удалять и добавлять элементы, возвращать количество элементов списка, выводить назания элементов) в соответствии с требованиями. Были изучены необходимые языковые конструкции и особенности написания двунаправленных линейных списков на языке Cи.

# Приложение А Исходный код программы

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

//описание структуры MusicalComposition

typedef struct MusicalComposition {

char name[80];

char author[80];

int year;

struct MusicalComposition\* next; //следующий элемент

struct MusicalComposition\* prev; //предыдущий элемент

} MusicalComposition;

//создание структуры MusicalComposition

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

//выделение памяти под новую музыкальную композицию

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

//копирование переданных значений имени и автора

strcpy(newComposition->name, name);

strcpy(newComposition->author, author);

//присвоение года созданной композиции

newComposition->year = year;

//указатель на след. элемент устанавл. в NULL так как элемент еще не добавлен в список

newComposition->next = NULL;

//указатель на пред. элемент устанавл. в NULL так как элемент еще не добавлен в список

newComposition->prev = NULL;

//возвращение указателя на новую музыкальную композицию

return newComposition;

}

//функция создания списка MusicalComposition

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

//указатель на начало списка

MusicalComposition\* head = NULL;

//указатель для перемещения по списку

MusicalComposition\* current = NULL;

//создание списка композиций на основе переданных данных

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

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

//если список пустой, устанавливаем новую композицию как начальный элемент списка

if (head == NULL) {

head = newComposition;

current = head;

} else {

//если список не пустой, добавляем новую композицию в конец списка

current->next = newComposition; //указание на следующую композицию

newComposition->prev = current; //создание ссылки на предыдущую композицию

current = newComposition; //перемещение указателя на новую композицию

}

}

//возвращение указателя на начало списка музыкальных композиций

return head;

}

//добавление нового элемента в конец списка

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

MusicalComposition\* current = head;

//перемещение в конец списка

while (current->next != NULL) {

current = current->next;

}

//добавление нового элемента

current->next = element;

element->prev = current;

}

//удаление элемента из списка по названию

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

MusicalComposition\* current = head;

//перебор элементов списка

while (current != NULL) {

//поиск элемента по названию и удаление его из списка

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

if (current->prev != NULL) {

current->prev->next = current->next;

}

if (current->next != NULL) {

current->next->prev = current->prev;

}

//освобождение памяти, занимаемой удаляемым элементом

free(current);

break;

}

//переход к следующему элементу

current = current->next;

}

}

//подсчет количества элементов в списке

int count(MusicalComposition\* head) {

int count = 0; //инициализация счётчика количества элементов

MusicalComposition\* current = head;

//перебор элементов списка

while (current != NULL) {

count++;

current = current->next;

}

return count;

}

//вывод названий композиций из списка

void print\_names(MusicalComposition\* head) {

MusicalComposition\* current = head;

//перебор элементов списка

while (current != NULL) {

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

current = current->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);

//считывание и заполнение массивов данных

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);

}

//создание списка музыкальных композиций

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];

//добавление новой композиции

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);

//удаление композиции по названию и вывод результатов

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);

//освобождение памяти

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

free(names[i]);

free(authors[i]);

}

free(names);

free(authors);

free(years);

return 0;

}