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 Романов Е.А.

Преподаватель Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Изучение линейных списков, как структуры данных, получение навыков создания двунаправленного списка и его практического применения.

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.

Выполнение работы

Программа состоит из 6 функций, представляющих собой интерфейс для работы с двунаправленным связанным списком, реализованном в виде структуры MusicalComposition.

Первая функция createMusicalCompositionList создаёт головной элемент списка, заполняет его поля – ссылки на другие элементы списка, а затем, при помощи цикла for заполняет список количеством элементов, которое подаётся функции в качестве аргумента. В результате работы функция возвращает указатель на головной элемент списка.

Вторая функция createMusicalComposition выделяет память размером структуры элемента списка, заполняет поля структуры, значения которых передаются функции в качестве аргументов, и возвращает указатель на выделенную память.

Третья функция push принимает в качестве аргументов указатель на головной элемент списка и указатель на структуру, которую необходимо добавить в список. При помощи цикла while функция находит указатель на последний элемент списка, изменяет его поле, указывающее на следующий элемент, на значение указателя, вставляемого элемента.

Четвёртая функция removeEl принимает в качестве аргументов указатель на головной элемент списка и строку. При помощи цикла while функция движется по списку и проверяет соответствие поля name элемента списка с полученной в качестве аргумента строкой. Проверка осуществляется с помощью функции strcmp стандартной библиотеки. Найдя указатель на элемент списка, который необходимо удалить, функция меняет указатели элементов, расположенных до и после удаляемой структуры и освобождает память, занимаемую удалённым элементом.

Пятая функция count принимает в качестве аргумента указатель на головной элемент списка, и при помощи цикла while движется по списку, подсчитывая количество полученных указателей на следующий элемент. Функция возвращает целое число – количество найденных элементов.

Шестая функция print_names принимает в качестве аргумента указатель на головной элемент и в цикле for выводит поля name всех элементов списка.

Переменные, используемые в программе:

-new_composition указатель на структуру область памяти, выделяемой для новой структуры MusicalComposition, локальная переменная функции creatMusicalComposition

-head указатель на структуру MusicalComposition, которая является головным элементом создаваемого двусвязного списка, локальная переменная функции creatMusicalCompositionList

-tmp указатель на структуру MusicalComposition, переменная используется в цикле в качестве указателя на элементы, задействованные в организации двусвязного списка, локальная переменная функции creatMusicalCompositionList

-el указатель на структуру MusicalComposition, локальная переменная функции removeEl, указывающая на следующий за головным элемент списка. Локальная переменная функций push, count и print_names, указывающая на головной элемент списка.

-amount целочисленная переменная, хранящая количество элементов списка, локальная переменная функции count

Функции стандартной библиотеки, используемые в программе:

-malloc выделяет память заданного размера

-free освобождает выделенную память

-printf выводит значения через стандартный поток вывода

-fprintf функция форматного вывода, использованная для вывода в поток вывода ошибок

-exit немедленно завершает выполнение программы

-strcmp сравнивает строки в лексографическом порядке

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	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	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

	2001 Points of Authority	

Выводы

В ходе выполнения лабораторной работы была освоена структура данных линейные списки. А также реализована программа на языке С, в которой на практике применяются полученные навыки.

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*              new_composition              =
(MusicalComposition*)malloc(sizeof(MusicalComposition));

    if (new_composition == NULL){
        fprintf(stderr, "Memory allocation error");
        exit(1);
    }

    *new_composition = (MusicalComposition){name,author,year};

    return new_composition;
}


MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n){
    if (n==0) return NULL;

    MusicalComposition                    *head                    =
createMusicalComposition(array_names[0],               array_authors[0],
array_years[0]);
    head->prev = NULL;

    if (n==1) return head;

    MusicalComposition                     *tmp                     =
createMusicalComposition(array_names[1], array_authors[1],array_years[1]);

    head->next = tmp;
    tmp->prev = head;

    for(size_t i = 2; i < n; i++){
```

```
            tmp->next     =     createMusicalComposition(array_names[i],
array_authors[i],array_years[i]);
            tmp->next->next = NULL;
            tmp->next->prev = tmp;
            tmp = tmp->next;
        }
        return head;
    }


    void push(MusicalComposition* head, MusicalComposition* element){
        MusicalComposition *el = head;

        while (el->next != NULL) el = el->next;

        el->next = element;
        element->prev = el;
        element->next = NULL;
    }

    void removeEl(MusicalComposition* head, char* name_for_remove){
        MusicalComposition *el = head->next;

        while(strcmp(el->name, name_for_remove)!= 0)el = el->next;

        if (el->prev == NULL) el->next->prev = NULL;

        else if (el->next == NULL) el->prev->next = NULL;

        else {
            el->prev->next = el->next;
            el->next->prev = el->prev;
        }
        free(el);
    }

    int count(MusicalComposition* head){
        int amount=1;
        MusicalComposition *el = head;
        while(el->next != NULL) {
            el = el->next;
            amount++;
        }
        return amount;
    }

    void print_names(MusicalComposition* head){
        MusicalComposition *el = head;
        for (int i=0;i<count(head);i++){
            printf("%s\n", el->name);
            el = el->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;

}
```