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. 3341 Чинаева М.Р.

Преподаватель Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью работы является освоение работы с линейными списками.

Для достижения поставленной цели требуется решить следующие задачи:

1. Ознакомиться со структурой данных «список».
2. Ознакомиться с операциями, используемыми для списков.
3. Изучить способы реализации этих операций на языке Си.
4. Написать программу, реализующую двусвязный линейный список и решающую задачу в соответствии с индивидуальным заданием.

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.

Основные теоретические положения

Двунаправленный список – это структура данных, которая состоит из узлов, каждый из которых содержит два указателя: один указывает на предыдущий узел, а другой – на следующий узел. Таким образом, двунаправленный список позволяет перемещаться как вперед, так и назад по списку.

Каждый узел двунаправленного списка содержит два поля: поле данных, которое хранит значение элемента списка, и два указателя: указатель на предыдущий узел и указатель на следующий узел.

При создании двунаправленного списка обычно создается специальный узел, называемый головным узлом или начальным узлом, который не содержит данных и используется для облегчения операций со списком.

Выполнение работы

Функции, представленные в работе:

1. MusicalComposition* createMusicalComposition(char* name, char* autor, int year)

Динамически выделяется память на структуру, далее каждому элементу структуры присваивается соответствующий элемент из входных данных. Указатель на следующий элемент и на предыдущий присваивается NULL.

Функция возвращает указатель на структуру MusicalComposition.

2. MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)

Создается указатель на первый элемент, ему присваивается NULL. Если количество элементов в массивах равно нулю, возвращается нулевой список, в противном случае создается элемент list и он становится первым элементом. Далее с помощью цикла while создается каждый элемент списка, при этом записывается прошлый элемент и происходит переход на следующий элемент.

Возвращается указатель на первый элемент

3. void push(MusicalComposition* head, MusicalComposition* element)

Если первый элемент – NULL, первым элементом списка становится элемент, который надо добавить. В обратном случае с помощью цикла while доходим до конца списка. Следующий элемент после последнего становится добавленным элементом.

4. void removeEl(MusicalComposition* head, char* name_for_remove)

С помощью цикла while проходимся по всему списку для поиска всех элементов, у которых имя совпадает с тем которое надо удалить. У предыдущего и последующего элементов заменяются соответствующие указатели, память из-под элемента освобождается.

5. int count(MusicalComposition* head)

Создается переменная count_elements для подсчета элементов в списке. Далее с помощью цикла while к ней прибавляется единица, если текущий элемент – не последний.

Функция возвращает количество элементов в списке.

6. void print_names(MusicalComposition* head)

С помощью цикла while выводит имена авторов музыкальных композиций каждого элемента списка, пока элемент ненулевой, то есть пока не дойдет до конца 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	Комментарии
1.	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	Тест с e.moevm

2.	2 Son Rammstein 2001 Son Rammstein 2001 Seek and Destroy Metallica 1982 Seek and Destroy	Son Rammstein 2001 2 3 Son Son 2	Удаляет только что добавленный элемент
3.	1 Sonne Rammstein 2001 Sonne Rammstein 2001 Points of Authority	Sonne Rammstein 2001 1 2 Sonne Sonne 2	Введено название элемента, которого нет в списке, следовательно ни один элемент не удаляется

Выводы

В ходе данного исследования была поставлена цель освоения работы с линейными списками. Для достижения этой цели были выполнены следующие задачи:

1. Изучение структуры "список", позволяющей хранить и организовывать элементы в линейной последовательности.

2. Ознакомление с операциями, используемыми для списков.

3. Изучение способов реализации этих операций на языке программирования C.

4. Написание программы, которая реализует двусвязный линейный список и решает конкретную задачу в соответствии с индивидуальным заданием.

Таким образом, выполнение поставленных задач позволило освоить работу с линейными списками и применить полученные знания при разработке программы на языке C.

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* autor,
int year) {
    MusicalComposition*                 new_music                 =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    new_music->name = name;
    new_music->author = autor;
    new_music->year = year;
    new_music->next = NULL;
    new_music->prev = NULL;
    return new_music;
}

MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n) {
    MusicalComposition* head = NULL;
    if (n == 0) {
        return head;
    }
    MusicalComposition*                   list                    =
createMusicalComposition(array_names[0],               array_authors[0],
array_years[0]);
    head = list;
    for (int i = 1; i < n; i++) {
        list->next    =    createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        list->next->prev = list;
        list = list->next;
    }
    return head;
}

void push(MusicalComposition* head, MusicalComposition* element) {
    MusicalComposition* list = head;
    if (head == NULL) {
        head = element;
        return;
    }
    while (list->next != NULL) {
```

```
        list = list->next;
    }
    list->next = element;
    list->next->prev = list;
};

void removeEl(MusicalComposition* head, char* name_for_remove) {
    MusicalComposition* list = head;
    while (list != NULL) {
        if (strcmp(name_for_remove, list->name) == 0) {
            if (list->prev != NULL) {
                list->prev->next = list->next;
            }
            if (list->next != NULL) {
                list->next->prev = list->prev;
            }
            free(list);
        }
        list=list->next;
    }
}


int count(MusicalComposition* head) {
    int count_elements = 0;
    MusicalComposition* list = head;
    while (list != NULL) {
        count_elements++;
        list = list->next;
    }
    return count_elements;
}

void print_names(MusicalComposition* head) {
    MusicalComposition* list = head;
    while (list != 0) {
        printf("%s\n", list->name);
        list = list->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;

    }
```