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. 3344 _____ Волков А.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Изучить структуру данных линейный список. Реализовать двунаправленный (двусвязный) линейный список и основные взаимодействия с ним на языке программирования Си.

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:

1) **n** - длина массивов **array_names, array_authors, array_years.**

2) поле **name** первого элемента списка соответствует первому элементу списка array_names (**array_names[0]**).

3) поле **author** первого элемента списка соответствует первому элементу списка array_authors **(array_authors[0**]).

4) 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

Подключив заголовочные файлы *<stdlib.h>*, *<stddef.h>*, *<stdio.h>*, *<string.h>*, определяем структуру *MusicalCompostion* и при помощи оператора *typedef* избавляемся от потребности постоянно писать *struct MusicalCompositon*.

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

Принимает на вход строку с названием композиции, строку с автором текста и год выпуска трека.

Возвращает указатель на созданную структуру с данными параметрами, поля *next* и *prev* устанавливаются в *NULL*.

Память под данную структуру выделяется с помощью *malloc*.

2) Функция *void push(MusicalComposition* head, MusicalComposition* element)*:

Получает указатель на начало списка и указатель на тот элемент, который необходимо добавить в конец списка.

Ничего не возвращает.

При помощи цикла *while* проходит до последнего элемента, у которого значение поля *next* заменяется на *element*, а поле *prev* уже у него самого на найденный последний элемент. Поле *next* становится *NULL*.

3) Функция *MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)*:

Принимает на вход массив строк с названиями треков, массив строк с авторами треков и массив чисел, года выпуска треков, а также число – общее кол-во треков.

Возвращает указатель на первый элемент списка.

При помощи функции *createMusicalComposition* создаётся первый элемент списка. Оставшиеся элементы создаются и «связываются» в теле цикла *for*. По окончании работы цикла, возвращается указатель на начало списка.

4) Функция *void removeEl(MusicalComposition* head, char* name_for_remove)*:

Принимает на вход указатель на начало списка и строку с названием трека, который нужно удалить из списка.

Ничего не возвращает.

При помощи цикла *while* и функции *strcmp* ищется нужный элемент, если он не находится, то функция ничего не изменяет, иначе «связываются» предыдущий и следующий за текущим элементы. Очищается память, выделенная под найденный элемент.

5) Функция *int count(MusicalComposition* head)*:

Принимает указатель на корень списка.

Возвращает кол-во элементов списка.

При помощи цикла *while* находится последний элемент. Счетчик увеличивается с каждой итерацией, на последнем элементе цикл заканчивается, счетчик возвращается в программу.

6) Функция void print_names(MusicalComposition* head):

Принимает указатель на начало списка.

Ничего не возвращает.

При помощи цикла *while* поочередно печатает название всех треков в списке.

7) Функция *main()*:

Создает сценарий для тестов созданного *API*.

Во-первых, считывает количество композиций, далее поочередно считываются композиции и данные об авторе и годе выпуска. После этого вызывается функция *createMusicalCompositionList*, куда передаются сформированные массивы данных.

Во-вторых, считываются данные о треке для использования функции *push*. В-третьих, считывается название трека, который нужно удалить из списка. После удачного считывания нужных данных на экран выводится:

1) Название, автор, год трека, который является началом.

2) Количество элементов до использования функции *push*.

3) Количество элементов после использования функции *push*.

4) Название всех треков после вызова функции *removeEl*.

5) Финальное количество элементов.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
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 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	Корректно выводится начальный элемент и подсчитывается изначальное количество элементов. После добавления трека верно пересчитывается их количество. После удаления правильно выводятся названия и итоговое количество элементов.

Выводы

Была изучена важная структура данных, линейный список. Реализованы функции для взаимодействия с двусвязным списком на языке программирования Си.

Основным преимуществом связных линейных списков является то, что они предоставляют механизм для хранения произвольного количества данных, очень подходят в случаях сегментированной памяти, ибо память для них необязательно выделяется последовательными блоками. Но при их использовании отсутствует произвольный доступ к объектам (по индексу).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb_2.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>


// Описание структуры MusicalComposition

typedef struct MusicalComposition
{
    char *name;
    char *author;
    unsigned int year;
    struct MusicalComposition *next;
    struct MusicalComposition *prev;
} MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* autor,int
year)
{
                  MusicalComposition    *ptr    =    (MusicalComposition
*)malloc(sizeof(MusicalComposition));
    ptr->name = name;
    ptr->author = autor;
    ptr->year = year;
    ptr->next = NULL;
    ptr->prev = NULL;

    return ptr;
}

// Функции для работы со списком MusicalComposition

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

    while (head->next != NULL) {
        head = head->next;
    }
    head->next = element;
    element->prev = head;
    element->next = NULL;
}

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]);
```

```
    head->prev = NULL;

    for (int i = 1; i < n; i++) {
                                        MusicalComposition    *element    =
createMusicalComposition(array_names[i],                array_authors[i],
array_years[i]);
        push(head, element);
    }

    return head;
}


void removeEl(MusicalComposition* head, char* name_for_remove)
{
    MusicalComposition *current = head;
    while (strcmp(current->name, name_for_remove) != 0) {
        current = current->next;
    }
    if (current->prev == NULL) {
        current->next->prev = NULL; // удаление первого элемента
    } else if (current->next == NULL) {
        current->prev->next = NULL; // удаление последнего элемента
    } else {
        current->next->prev = current->prev;
        current->prev->next = current->next;
    }

    free(current);
}

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

    return count;
}

void print_names(MusicalComposition* head)
{
    MusicalComposition *current = head;
    while (current->next != NULL) {
        puts(current->name);
        current = current->next;
    }

    puts(current->name);
}


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;

}
```