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 _____ Ягудин Д.Р.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Целью работы является освоение работы с линейными списками на языке Си на примере использующей их программы. Для освоения работы с линейными списками также необходимо изучить структуры в языке Си.

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

Структура MusicalComposition представляет собой элемент списка, включающий поля для хранения названия композиции (name), имени автора (author) и года создания (year). Особенность этой структуры заключается в наличии указателей на следующий и предыдущий элементы списка (соответственно, next и prev), что позволяет реализовать двунаправленный список.

Функция createMusicalComposition создает новый элемент списка на основе переданных параметров: названия, автора и года. Она выделяет память под новый элемент и инициализирует его поля переданными значениями.

Функция createMusicalCompositionList формирует 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, чтобы обеспечить двунаправленность списка.

Функция push добавляет новый элемент в конец списка. Она находит последний элемент списка, создает новый элемент и устанавливает указатель next последнего элемента на новый элемент, а также обновляет указатель prev нового элемента на предыдущий.

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

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

Функция print_names выводит названия всех композиций в списке, последовательно проходя по элементам и печатая их названия.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	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
2.	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

Выводы

Была освоена работа с созданием линейных списков, а так же работа с ними на языке Си.

Результатом 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>
#include <stddef.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*       result       =
(MusicalComposition*)malloc(sizeof(MusicalComposition));;

    result->name = name;
    result->author = author;
    result->year = year;
    result->prev = NULL;
    result->next = NULL;

    return result;
}

MusicalComposition*             createMusicalCompositionList(char**
array_names, char** array_authors, int* array_years, int n){
    MusicalComposition* head;
    MusicalComposition* current;
    MusicalComposition* prev;

                  head    =    createMusicalComposition(array_names[0],
array_authors[0], array_years[0]);
    prev = head;

    for(int i = 1; i < n; i ++){
                   current  =  createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        prev->next = current;
        current->prev = prev;
        prev = current;
    }

    return head;
}
```

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

    while(current->next != NULL){
        current = current->next;
    }

    current->next = element;
    element->prev = current;
}

void removeEl(MusicalComposition* head, char* name_for_remove){
    MusicalComposition* current;
    MusicalComposition* prev;
    MusicalComposition* next;
    current = head;

    while(strcmp(current->name, name_for_remove)){
        current = current->next;
    }

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

    next->prev = prev;
    prev->next = next;
}

int count(MusicalComposition* head){
    int counter = 1;

    MusicalComposition* current = head;

    while(current->next != NULL){
        counter ++;
        current = current->next;
    }

    return counter;
}

void print_names(MusicalComposition* head){
    MusicalComposition* current = head;

    while(current->next != NULL){
        printf("%s\n", current->name);
        current = current->next;
    }

    printf("%s\n", 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;

}
```