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 _____ Хайруллов Д.Л.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью данной лабораторной работы является освоение работы с линейными списками и структурами в языке программирования C.

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.

Функцию main менять не нужно.

Выполнение работы

Для выполнения работы необходимо было реализовать структуру MusicalComposition и функции для работы с элементами типа этой структуры.

Функция createMusicalComposition: данная функция создает новую структуру MusicalComposition и заполняет ее поля переданными аргументами. Выделяется память под новую структуру, копируются переданные строки в поля name и author, устанавливается значение поля year, указатели next и prev инициализируются значением NULL. Возвращается указатель на новую структуру.

Функция createMusicalCompositionList: данная функция создает список MusicalComposition на основе переданных массивов строк и массива с годами. Создается первая структура с помощью createMusicalComposition, затем в цикле создаются и добавляются в список новые структуры, связывая их указателями next и prev. Возвращается указатель на первую структуру в списке.

Функция push: функция добавляет новый элемент в конец списка MusicalComposition. Происходит перемещение указателя текущего элемента до конца списка, затем устанавливается связь с новым элементом и обновляются указатели next и prev.

Функция removeEl: функция удаляет элемент из списка MusicalComposition по заданному имени. Поиск элемента происходит по имени, затем в зависимости от положения элемента в списке выполняется изменение связей и освобождение памяти.

Функция count: функция подсчитывает количество элементов в списке MusicalComposition. Происходит циклический обход списка и увеличение счетчика на каждой итерации. Возвращается общее количество элементов.

Функция print_names: функция печатает названия музыкальных произведений из списка MusicalComposition. Происходит циклический обход списка и вывод названия каждого элемента на экран.списка MusicalComposition.

Происходит циклический обход списка и вывод названия каждого элемента на экран.

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

Были изучены основы использования линейных списков и структур в программах, написанных на языке С. Разработана программа, в которой описываются новая структура для хранения информации о музыкальных композициях и функции для работы с элементами типа этой структуры.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define MEMORY_ERROR "Error: error reallocating memory\n"

// Описание структуры MusicalComposition
typedef struct  MusicalComposition{
    char name[80];
    char author[80];
    int year;
    struct MusicalComposition* next;
    struct MusicalComposition* prev;
}MusicalComposition;

// Создание структуры MusicalComposition
MusicalComposition* createMusicalComposition(char* name, char* author,int
year){
    MusicalComposition*                 new_composition                 =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    if (new_composition == NULL){
        printf(MEMORY_ERROR);
        exit(1);
    }
    strcpy(new_composition->name, name);
    strcpy(new_composition->author, author);
    new_composition->year = year;
    new_composition->next = NULL;
    new_composition->prev = NULL;
    return new_composition;
}

// Функции для работы со списком MusicalComposition
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]);
    MusicalComposition* previous = head;
    MusicalComposition* current = head;
    for(int i = 1; i < n; i++){
        current        =        createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        previous->next = current;
        current->prev = previous;
        previous = current;
    }
    return head;
}

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

```
    MusicalComposition* 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 = head;
    while(strcmp(current->name, name_for_remove) != 0){
        current = current->next;
    }
    if(current->next != NULL && current->prev != NULL){
        current->prev->next = current->next;
        current->next->prev = current->prev;
        free(current);
    }
    else if(current->next == NULL && current->prev != NULL){
        current->prev->next = NULL;
        free(current);
    }
    else if(current->next != NULL && current->prev == NULL){
        current->next->prev = NULL;
        free(current);
    }
    else{
        free(current);
    }
}

int count(MusicalComposition* head){
    MusicalComposition* current = head;
    if(head == NULL){
        return 0;
    }
    int counter = 1;
    while(current->next != NULL){
        counter++;
        current = current->next;
    }
    return counter;
}

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

    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;

}
```