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 _____ Кузнецов Р.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Получение навыков работы с линейными списками на языке программирования Си.

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

Были подключены такие библиотеки, *<stdio.h>*, *<stdlib.h>*, *<string.h>*. *С*оздана структура *MusicalComposition,* которая определена как *PROD.* В структуре созданы поля *char *name, char *author, int year* и указатели на следующий *MusicalComposition* next* и предыдущий MusicalComposition* prev элемент линейного двусвязного списка. *С*озданы 6 функций для работы с песнями: *PROD* createMusicalComposition(char* name, char* autor, int year);*, которая создает новый элемент для списка т.е. принимает имя композиции, имя автора и год, выделяет память под новый элемент и копирует данные в новый элемент; *PROD* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n);*, которая создает список композиций на основе количества данных в массивах для каждого элемента и также настраивает указатели для двусвязного списка; *void push(PROD* head, PROD* element);*, которая добавляет *element* в конец списка посредством, цикла *while*, который перемещается до конца двусвязного списка; *void removeEl(PROD* head, char* name_for_remove);*, которая перебирает список и ищет элемент с необходимым именем. Если такой элемент найден, то устанавливает указатели *next* и *prev* таким образом, чтобы они указывали на те списки, которые огибают искомый; *int count(PROD* head);*, которая считает количество элементов списка, посредством цикла, который пробегается по спискам и увеличивает счетчик; *void print_names(PROD* head);*, выводит на экран названия композиций посредством цикла *while*, который проходит по всем элементам списка.

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	-

Выводы

Были получены навыки работы с линейными списками на языке программирования Си.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Kuznetsov_Roman_lb2.c

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

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

PROD* createMusicalComposition(char* name, char* autor, int year);
PROD*     createMusicalCompositionList(char**     array_names,     char**
array_authors, int* array_years, int n);
void push(PROD* head, PROD* element);
void removeEl(PROD* head, char* name_for_remove);
int count(PROD* head);
void print_names(PROD* head);

PROD* createMusicalComposition(char* name, char* autor, int year) {
    PROD* curr = (PROD*)malloc(sizeof(PROD));

    curr->name = name;
    curr->author = autor;
    curr->year = year;

    return curr;
}

PROD*     createMusicalCompositionList(char**     array_names,     char**
array_authors, int* array_years, int n) {
    if  (n  <=  0  ||  array_names  ==  NULL  ||  array_authors  ==  NULL  ||
array_years == NULL) {
        return NULL;
    }
    PROD* top = createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
    top->prev = NULL;// т.е. начало списка, предыдущих элементов нет
    PROD* temp_prev = top; // просто копия адреса первого

    for (int i = 1; i < n; i++) {

        PROD*     incom     =     createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        temp_prev->next = incom;// указывает адрес на след структуру
        incom->prev = temp_prev;// указывает адрес из текущей структуры
на предыдущюю
        temp_prev = incom; // обновление адреса структуры (текущее)
    }
```

```
    temp_prev->next = NULL;// последний элемент(temp т.к. обновлялся в
for)

    return top;
}

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

    PROD* curr = head;//  копия

    while (curr->next != NULL) curr = curr->next;;// дальше по списку

    curr->next = element;//указатель на след элемент списка
    element->prev = curr;//соединение с предыдущим
    element->next = NULL;// показывает что это конец

    return;
}

void removeEl(PROD* head, char* name_for_remove) {

    PROD* curr = head;
    size_t a;

    while (curr->next != NULL) {
        a = strcmp(curr->name, name_for_remove);
        if (!a) {

                PROD* beyond = curr->next;
                PROD* before = curr->prev;

                before->next = beyond;
                beyond->prev = before;

                free(curr);

                curr = beyond;
        }
        else curr = curr->next;
}}

int count(PROD* head) {
    PROD* curr = head;
    int count = 1;

    while (curr->next != NULL)
        {count++;
        curr = curr->next;}

    return count;
}

void print_names(PROD* head) {
    PROD* curr = head;

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

```
        curr = curr->next;;
    }

    printf("%s\n", curr->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);

    }
    PROD*  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;

    PROD*  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;
}
```