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 _____ Гусева Е.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Изучение структур данных и линейных списков. Работа с линейными списками на примере лабораторной работы.

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

Была подключена стандартная библиотека для ввода и вывода *<stdio.h>*, стандартная библиотека *<stdlib.h>*, библиотека для работы со строками *<string.h>*.

Была создана структура *MusicalComposition* с полями названия *char *name,* автора *char *author,* года выпуска *int year,* указателями на предыдущий и следующий элемент двусвязного списка *MusicalComposition *prev* и *MusicalComposition *next.* Была создана функция *MusicalComposition *createMusicalCompositionList(char **array_names, char **array_authors, int *array_years, int n)* для заполнения линейного списка, которое происходит в цикле, где мы создаем для элемента структуру, в которой мы заполняем указатели на предыдущий и следующий элементы и другие поля с информацией о песнях. Была создана функция *void push(MusicalComposition *head, MusicalComposition *element)* для добавления нового элемента в конец списка. Циклом *while* доходили до последнего элемента и изменяли его указатель на следующий элемент, а также заполняли поля указатели нового элемента. Была реализована функция *void removeEl(MusicalComposition *head, char *name_for_remove),* которая удаляет элемент списка, у которого значение *name* равно значению *name_for_remove.* В функции циклом проходим по всем элементам списка, если значение *name* равно значению *name_for_remove*, то указатель предыдущего элемента для текущего становится указателем предыдущего для следующего для текущего, аналогично для указателя следующего элемента, также очищаем память для элемента. В функции *int count(MusicalComposition *head)* возвращаем количество элементов списка, проходим циклом *while* по всем элементам и увеличиваем переменную-счетчик каждую итерацию. В функции *void print_names(MusicalComposition *head)* происходит вывод названий всех композиций на экран, проходим циклом *while* по всем элементам и выводим название песни каждую итерацию.

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

Была освоена работа с линейными списками на языке Си на примере лабораторной работы, а также изучена работа со структурами.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_for_lb2.c

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


// О п и с а н и е  с т р у к т у р ы MusicalComposition
//typedef struct MusicalComposition MusicalComposition;

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

// С о з д а н и е  с т р у к т у р ы MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* autor,int
year);

// Ф у н к ц и и  д л я  р а б о т ы  с о  с п и с к о м MusicalComposition

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

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

void removeEl(MusicalComposition* head, char* name_for_remove);

int count(MusicalComposition* head);

void print_names(MusicalComposition* head);


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;
```

```
}

MusicalComposition* createMusicalComposition(char* name, char* autor,int
year){

    MusicalComposition*                              mus=(MusicalComposition
*)malloc(sizeof(MusicalComposition));

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

    mus->prev=NULL;
    mus->next=NULL;

    return mus;
}

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* elm = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        push(head, elm);
    }

    return head;
}

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

    MusicalComposition* coopy = head;

        while(coopy->next){
            coopy = coopy->next;
        }
        coopy->next = element;
        element->prev = coopy;
        element->next = NULL;
        return;
}

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

```
            free(temp);
            break;
        }
        temp = temp->next;
    }
    return;
}

int count(MusicalComposition* head){

    MusicalComposition* temp=head;

    int count=0;
    while(temp){
        temp=temp->next;
        count++;
    }
    return count;
}

void print_names(MusicalComposition* head){
    MusicalComposition* temp=head;
    while(temp){
            printf("%s\n",temp->name);
            temp = temp->next;
        }
        return;
}
```