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

Студентка гр. 3341 _____ Пчелкин Н.И.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург

2024

Цель работы

Целью работы является освоение работы с линейными списками.

Для достижения поставленной цели требуется решить следующие задачи:

- ознакомиться со структурой «список»;
- ознакомиться со списком операций используемых для списков;
- изучить способы реализации этих операций на языке С;
- написать программу, реализующую двусвязный линейный список и решающую задачу в соответствии с индивидуальным заданием.

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

Элемент списка типа *MusicalComposition* состоит из:

- указателя на *char name* - названия композиции
- указателя на *char author* - автора композиции
- целого числа *year* - года создания композиции
- указателя на тип *MusicalComposition prev* - указателя на предыдущую композицию
- указателя на тип *MusicalComposition next* - указателя на следующую композицию

Функции:

- *MusicalComposition* createMusicalComposition(char* name, char* author, int year)* принимает указатель на название композиции, автора композиции и год создания, с помощью функции *malloc* выделяет память для одного элемента типа *MusicalComposition*, полям *name, author, year* данного элемента присваивает соответствующие значения, полям *prev, next* присваивает *NULL*, возвращает указатель на созданный элемент.
- *MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)* принимает указатель на массив названий композиций, авторов композиций, лет создания и количество элементов в массивах; *head* присваивает нулевой элемент списка, созданный с помощью функции *createMusicalComposition*. Создает новый элемент списка и через ту же функцию заполняет в ней поля *name*, *author* и *year*. В поле *prev* при переходе на новый элемент списка вписывается предыдущий элемент. Перед переходом на новый элемент, в *next* записывается указатель на новый элемент.
- *void push(MusicalComposition* head, MusicalComposition* element)* принимает указатель на нулевой элемент списка и на композицию, которую нужно добавить в список; с помощью цикла *while* проходит по списку до последнего элемента, т.е. пока поле *next* текущего элемента не равно *NULL;* после цикла полю *next* текущей музыкальной композиции (последней)

присваивает *element*, полю *prev* добавленной музыкальной композиции присваивает *head.*

- *void removeEl (MusicalComposition* head, char* name_for_remove)* принимает указатель на нулевой элемент списка и на название композиции, которую нужно удалить из списка. Изначально осуществляется проверка на то, что элемент с именем равным *name_for_remove* не первый. Если это так, то у следующего элемента в поле *prev* записывается *NULL*. В противном случае программа проходит по всему списку и сравнивает имя следующего элемента с *name_for_remove.* При нахождении нужного элемента у текущего элемента меняется поле *next* на элемент, который идет через один от текущего. А у следующего элемента после того, который должен быть удален, (если он существует) заменяется поле *prev* на текущий элемент.
- *int count(MusicalComposition* head)* принимает указатель на нулевой элемент списка; количеству композиций *counter* присваивает 0, с помощью цикла *while* проходит по списку пока указатель на текущий элемент не равен *NULL* и добавляет 1 к *counting* на каждом шаге цикла*;* возвращает *counter.*
- void *print_names(MusicalComposition* head)* принимает указатель на нулевой элемент списка; с помощью цикла *while* проходит по списку пока указатель на текущий элемент не равен *NULL* и на каждом шаге цикла выводит поле *name* текущей композиции.
- *main* содержит некоторую последовательность вызова команд для проверки работы списка.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	3 Fields of Gold Sting 1993 In the Army Now Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Sonne Rammstein 2001 Mixed Emotions	Fields of Gold Sting 1993 3 4 Fields of Gold In the Army Now Sonne 3	*main* считывает в массив 3 названия композиций, авторов и лет; название, автора и год композиции, которую нужно будет добавить в список, затем название композиции, которую нужно удалить. Выводит название, автора и год первой композиции, количество элементов в списке (3), затем добавляет элемент и снова выводит количество элементов (4), удаляет элементы с заданным названием, после выводит названия композиций и количество элементов в списке.
2.	3 In the Army Now Sting 1993 Mixed Emotions	In the Army Now Sting 1993 3 4 In the Army	В данном примере удаляется не одна, а 3 композиции с заданным именем.

	Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Mixed Emotions Rammstein 2001 Mixed Emotions	Now 1	

Выводы

В ходе выполнения работы были изучены:

- основные принципы работы с линейными списками;
- структура списков и операции, применяемые к ним;
- способы реализации этих операций на языке С;
- написана программа, реализующая двусвязный линейный список и решающая задачу в соответствии с индивидуальным заданием.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: 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;
} MusicalComposition;

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

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* actual_composition = head;

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

void push(MusicalComposition* head, MusicalComposition* element){
    while(head->next != NULL){
        head = head->next;
    }
    head->next = element;
    head->next->prev = head;
}

void removeEl(MusicalComposition* head, char* name_for_remove){
    if(strcmp(head->name, name_for_remove) == 0){
        head = head->next;
```

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

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

void print_names(MusicalComposition* head){
    while(head != NULL){
        printf("%s\n", head->name);
        head = head->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;

}
```