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. 3342 _____ Антипина В.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Целью работы является освоение работы с линейными списками.

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.

Основные теоретические положения

В языке Си можно определять указатели как на объекты стандартных типов, так и на структуры. Например, указатель на структуру Node выглядит так:

```
struct Node* nodeElement;
```

Для того чтобы обращаться к полям структуры, если имеется указатель на структуру, используется оператор “→”.

Узел – один элемент из списка, который связан с другими элементами;

Голова списка (head) – первый элемент в списке;

Хвост (tail) – все элементы списка, идущие после головы.

Линейный односвязный список – это структура данных, представляющая собой последовательность узлов, каждый из которых хранит какие-то полезные данные и указатель на следующий элемент. Важно, что в памяти элементы не находятся последовательно, в отличие от массивов.

Двусвязный список – это список с возможностью идти в обе стороны, в отличие от односвязного. Каждый элемент двусвязного списка имеет указатель на предыдущий элемент и на следующий.

Добавление узла в двусвязный список включает в себя следующие этапы:

- создание узла добавляемого элемента и заполнение его поля данных;
- переустановку указателя "next" узла, предшествующего добавляемому, на добавляемый узел;
- переустановка указателя "prev" узла, следующего за добавляемым, на добавляемый узел;
- установка указателя "next" добавляемого узла на следующий узел (тот, на который указывал предшествующий узел);
- установка указателя "prev" добавляемого узла на узел, предшествующий добавляемому (узел, переданный в функцию).

Выполнение работы

Были подключены библиотеки stdio.h, stdlib.h, string.h и была определена константа END_OF_STRING — символ конца строки. Была описана структура MusicalComposition с помощью typedef struct (typedef был использован, чтобы не приходилось каждый раз писать struct MusicalComposition). Структура состоит из массивов символов name и author, в которых будут храниться название композиции и её автор, переменной типа int, в которую будет записан год выпуска трека, двух указателей на структуру, в которых будут храниться указатели на предыдущий и следующий элементы списка.

Далее была написана функция, возвращающая указатель на описанную структуру, которая создаёт элемент списка. На вход функции createMusicalComposition подаются массивы символов, содержащие название композиции и автора, а также целое число — год выпуска. В функции динамически выделяется память на структуру, в её поля name, author и year записываются значения аргументов, подаваемых на вход, а next и prev присваивается NULL по умолчанию.

Функция createMusicalCompositionList принимает на вход двумерные массивы символов с названиями композиций и их авторами, а также массив чисел (где хранятся даты выпуска треков), создаёт двусвязный список структур, возвращает указатель на первый элемент этого списка (head). Для этого создаётся структура list, указатель на которую сохраняется в head. Далее в цикле for создаётся структура, указатель на которую записывается в поле next первой структуры, создаётся указатель на структуру для хранения адреса первой созданной структуры. List присваивается значение указателя на вторую структуру (таким образом список «сдвигается» на один элемент), в его поле prev записывается сохранённый адрес. Так как пока новых элементов нет, в поле next записывается NULL. Таким образом, по завершении цикла каждый элемент будет ссылаться на следующий и предыдущий, а последний и первый элементы списка в соответствующих полях будут хранить указатель на ноль.

В функции push реализуется добавление нового элемента списка в конец. Если список пустой, то в него добавляется единственный элемент. В противном случае создаётся указатель на структуру, в который записывается адрес первого элемента списка, затем в цикле while осуществляется поиск конца списка путём последовательных переходов к следующему элементу (cur — указатель на текущий элемент. Пока в поле next не окажется NULL (что будет означать, что текущий элемент — последний), cur присваивается значение адреса следующего элемента). После этого в поле prev нового элемента записывается адрес последнего элемента, а в поле next последнего записывается адрес нового элемента. Теперь добавленный элемент — последний в списке, а, значит, он должен ссылаться на NULL в поле next.

Функция count возвращает количество элементов в списке. Создаётся счётчик, который увеличивается на единицу в цикле while, где осуществляется переход к следующему элементу, пока он не окажется последним. В цикле не учитывается последний элемент, поэтому возвращаемое значение на единицу больше.

Функция print_names выводит на экран названия композиций в списке. Если в списке нет треков, выводится сообщение об этом. В противном случае выводится название каждого элемента до последнего в цикле (переход от одного элемента к следующему был описан ранее). После завершения цикла выводится последний элемент 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.	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	Корректно

Выводы

В ходе работы были изучены линейные списки, способы реализации операций для них на языке Си. Была написана программа, реализующая двусвязный линейный список и решающая задачу в соответствии с индивидуальным заданием (а именно — создание списка структур, который содержит информацию о музыкальных композициях, и реализацию функций над ним — добавление нового элемента в конец, удаление элемента по названию композиции, подсчёт количества элементов списка, вывод названий треков на экран).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Antipina_Veronika_lb2.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#define END_OF_STRING '\0'

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


// Создание структуры MusicalComposition

//MusicalComposition* createMusicalComposition(char* name, char*
autor,int year);
MusicalComposition* createMusicalComposition(char* name, char*
author, int year){
          MusicalComposition*              track              =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
       if(!track){
            perror("No memory");
       }

          track->name = name;
          track->author = author;
          track->year = year;
          track->next = NULL;
          track->prev = NULL;
          return track;
}

// Функции для работы со списком MusicalComposition

//MusicalComposition*         createMusicalCompositionList(char**
array_names, char** array_authors, int* array_years, int n);
MusicalComposition*          createMusicalCompositionList(char**
array_names, char** array_authors, int* array_years,int n){
          MusicalComposition* head = NULL;
          MusicalComposition*               list               =
createMusicalComposition(array_names[0],array_authors[0],array_years[0
]);
          list->next = NULL;
          list->prev = NULL;
          head = list;

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

```
                    list->next                                          =
createMusicalComposition(array_names[i],array_authors[i],array_years[i
]);
                    MusicalComposition* address_to_save = list;

                    list = list->next;
                    list->next = NULL;
                    list->prev = address_to_save;
            }
            return head;
    }

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

            if(head==NULL){
                    head = element;
                    return;
            }else{
                MusicalComposition* cur = head;
                while(cur->next){
                        cur = cur->next;
                }
                element->prev = cur;
                cur->next = element;
                element->next = NULL;
            }
    }

    //void removeEl(MusicalComposition* head, char* name_for_remove);
    void removeEl(MusicalComposition* head, char* name_for_remove){
            if(head==NULL)
                    return;

            MusicalComposition* cur = head;
              if(strcmp(head->name, name_for_remove)==0){
                    return;
               }
            MusicalComposition* save;
            while(cur->next!=NULL){
                    if(strcmp(cur->name,name_for_remove)==0){
                        cur->next->prev = cur->prev;
                        cur->prev->next = cur->next;
                        save = cur->next;
                        free(cur);
                        cur = save;
                    }else{
                           // if(cur->next!=NULL)
                                    cur = cur->next;
                    }
            }
    }

    //int count(MusicalComposition* head);
    int count(MusicalComposition* head){
            int counter = 0;
```

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

//void print_names(MusicalComposition* head);
void print_names(MusicalComposition* head){
        MusicalComposition* cur = head;
        if(head==NULL)
                printf("No music yet:(\n");
        else{
                while(cur->next!=NULL){
                        printf("%s\n",cur->name);
                        cur = cur->next;
                }
                printf("%s\n",cur->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;

    }
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица Б.2 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
2.	5 Stressed Out twenty one pilots 2015 El Manana Gorillaz 2005 The Earth Song Michael Jackson 1995 Help! The Beatles 1965 Gangstas Paradise Coolio 1995 Le Trublion Mikelangelo Loconte 2009 El Manana	Stressed Out twenty one pilots 2015 5 6 Stressed Out The Earth Song Help! Gangstas Paradise Le Trublion 5	