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. 3341 _____ Бойцов В.А.

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:

1. Ознакомиться со структурой данных «список»;
2. Ознакомиться с операциями, используемыми для списков;
3. Изучить способы реализации этих операций на языке Си;
4. 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

Линейный список состоит из элементов, каждый из которых представляет из себя структуру *MusicalComposition*, которая содержит в себе:

- строку *char* name* – название композиции;
- строку *char* author* – информация об авторе или группе;
- целое число *int year* – год написания композиции;
- указатель на предыдущий элемент списка *MusicalComposition* previous*;
- указатель на следующий элемент списка *MusicalComposition* next*.

Далее был написан набор функций для работы со списком.

MusicalComposition createMusicalComposition(char* name, char* author,int year)* – функция для создания отдельной структуры *MusicalComposition*. Функция принимает на вход строки *char* name* и *char* author* и число *int year*. Создаётся указатель на структуру *MusicalComposition* newComposition*, для неё выделяется память с помощью функции *malloc()*. Далее с помощью функции *calloc()* для полей *newComposition->name* и *newComposition->author* выделяется достаточно памяти для копирования в них строк *name* и *author* соответственно. Это необходимо, т.к. при обычном присваивании указателя изменение строк вне списка повлечет за собой изменение строк в списке, чего быть не должно.

Необходимо отдельно отметить, что выделенную память необходимо очищать отдельно, а значит, требуется функция для очистки памяти для всего списка и в том числе для очистки выделенной памяти для полей его элементов, однако в задании не сказано о необходимости реализации этой функции, а её непосредственное применение в работе невозможно из-за запрета на изменение функции *main()*.

Далее с помощью функции *strcpy*() в поля структуры копируются строки *name* и *author*, заполняется поле year, а указателям *newComposition->next* и *newComposition->previous* присваивается значение *NULL*. Функция возвращает указатель *newComposition*.

Функция *MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)* принимает на вход массивы имён композиций *char** array_names*, информации об их авторах *char** array_authors* и годах *int* array_years*, а также число элементов в массивах *int n*. Создаётся указатель на голову списка *MusicalComposition* MusicalCompositionList*, первый элемент списка задаётся функцией *createMusicalComposition()*. Также создаётся указатель на последний элемент списка *currentComposition*. Далее в цикле *for()*, пробегающем все значения от *1* до *n-1*, с помощью *createMusicalComposition()* создаётся указатель на новый элемент списка *MusicalComposition* tempComposition*, полю *tempComposition->previous* задаётся значение *currentComposition*, а *currentComposition->next – tempComposition*. Затем указателю *currentComposition* передаётся значение *tempComposition*. Функция возвращает *MusicalCompositionList*.

Функция *void push(MusicalComposition* head, MusicalComposition* element)* принимает указатель на первый элемент списка *MusicalComposition* head*, в конец которого необходимо вставить *MusicalComposition* element*. В функции создаётся указатель *MusicalComposition* tail*, который с помощью цикла *while()* находит хвост списка, затем поля *next* и *previous* настраиваются аналогично предыдущей функции.

Функция *void removeEl(MusicalComposition* head, char* name_for_remove)* принимает указатель на первый элемент списка *MusicalComposition* head*, из которого необходимо удалить элемент с именем композиции *name_for_remove*. с помощью цикла *while()* указатель *MusicalComposition* node* находит необходимый элемент списка или конец списка, если нужного элемента в списке нет. Затем, если элемент для удаления нашелся (проверяется с помощью функции *strcmp()*), очищаются поля *name*, *author*, указатели предыдущего и последующего элементов списка редактируются, а затем память для *node* также очищается.

Функция *int count(MusicalComposition* head)* принимает указатель на первый элемент списка *MusicalComposition* head*. Создаётся переменная *int*

num, равная *0*. Если *head* не равен *NULL*, в цикле *while()* значение *num* увеличивается, пока не найдётся поле *next*, равное *NULL*, что означает конец списка. Функция возвращает *num*.

Функция *void print_names(MusicalComposition* head)* принимает указатель на первый элемент списка *MusicalComposition* head*. Если *head* не равен *NULL*, в цикле *do {} while* на экран выводятся имена композиций списка с помощью функции *puts()*, пока не найдется поле *next*, равное *NULL*.

Функция *int main()* задана условием задания и содержит в себе небольшой код для тестирования написанного *api*.

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	Результаты работы данного кода из задания
2.	7 Fields of Gold Sting 1993 In the Army Now Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Billie Jean Michael Jackson	Fields of Gold Sting 1993 7 8 Fields of Gold Mixed Emotions Sonne 3	Удалено большее количество элементов списка

	1983 Seek and Destroy Metallica 1982 Wicked Game Chris Isaak 1989 Points of Authority Linkin Park 2000 Sonne Rammstein 2001 Points of Authority		

Выводы

В результате выполнения работы были освоены основные принципы работы с линейными списками.

Была изучена структура данных «список», необходимые для обращения с ней операции, а также была написана программа, реализующая двусвязный линейный список.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


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


MusicalComposition*   createMusicalComposition(char*   name,   char*
author,int year)
{
                          MusicalComposition*     newComposition     =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
                    newComposition->name=(char*)calloc(strlen(name)+1,
sizeof(char));
                newComposition->author=(char*)calloc(strlen(author)+1,
sizeof(char));
    strcpy(newComposition->name, name);
    strcpy(newComposition->author, author);
    newComposition->year=year;
    newComposition->next = NULL;
    newComposition->previous = NULL;
    return newComposition;
}


MusicalComposition*             createMusicalCompositionList(char**
array_names, char** array_authors, int* array_years, int n)
{
                    MusicalComposition*    MusicalCompositionList    =
createMusicalComposition(array_names[0],               array_authors[0],
array_years[0]);
                      MusicalComposition*     currentComposition     =
MusicalCompositionList;
    for (int i=1; i<n; i++)
    {
                           MusicalComposition*   tempComposition   =
createMusicalComposition(array_names[i],               array_authors[i],
array_years[i]);
        tempComposition->previous = currentComposition;
        currentComposition->next=tempComposition;
        currentComposition=tempComposition;
    }
    return MusicalCompositionList;
}
```

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

void removeEl(MusicalComposition* head, char* name_for_remove)
{
    MusicalComposition* node = head;
     while(node->next!=NULL && strcmp(node->name, name_for_remove)!
=0)
        node=node->next;
    if (strcmp(node->name, name_for_remove)==0)
    {
        free(node->name);
        free(node->author);
        node->previous->next=node->next;
        node->next->previous=node->previous;
        free(node);
    }
}

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

void print_names(MusicalComposition* head)
{
    if (head!=NULL)
    {
        MusicalComposition* current = head;
        do
        {
            puts(current->name);
            current=current->next;
        }while(current!=NULL);
    }
}


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;

}
```