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. 1304 _____ Чернякова В.А.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2022

Цель работы.

Освоение работы с линейными списками: однонаправленными и двунаправленными – использование их при написании программ на языке С. Совершенствования навыков создания и применения структур.

Задание.

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface - 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.

Структура MusicalComposition:

Для того, чтобы в дальнейшем в программе каждый раз не использовать сочетание *struct MusicalComposition* используется средство, называемое *typedef*, которое позволяет давать типам данных новые имена. Объявляется: *typedef struct MusicalComposition{…}MusicalComposition*; - имя *MusicalComposition* становится синонимом для *struct MusicalComposition*.

Структура содержит 5 элементов. *Char* name* – указатель на строку, которая содержит название композиции. *Char* author* – указатель на строку, которая содержит имя автора композиции или название музыкальной группы. *Int year* - целое число, год создания. *Struct MusicalComposition* prev* – указатель на такую же структуру, в которой содержится информация о предыдущем элементе списка. *Struct MusicalComposition* next* – указатель на

такую же структуру, в которой содержится информация о следующем элементе списка.

Функции.

Функция для создания элемента списка:

Функция *createMusicalComposition* возвращает указатель на *struct MusicalComposition*. Она принимает на вход *char* name, char* author* - указатели на строки, содержащие название композиции и его исполнителя соответственно, *int year* – целое значение года выпуска композиции. Создается указатель *mus_comp* типа *MusicalComposition*, в котором будут храниться значения структуры. Для хранения происходит динамическое выделение памяти с помощью функции *malloc()* размер выделяемой памяти равен размеру структуры *MusicalComposition*. В соответствующие элементы структуры копируются значения, которые были приняты функцией на вход. Элементы *prev* и *next* инициализируются нулевыми значениями *NULL*. Функция возвращает указатель *mus_comp*.

Функция для работы со списком:

Функция *createMusicalCompositionList* возвращает указатель на *struct MusicalComposition.* Она принимает на вход *char** array_names, char** array_authors* – указатели на указатели на строки, содержащие названия введенных композиций и их исполнителей соответственно, *int* array_years* – указатель на массив, в котором хранятся целые значения, а именно года выпуска композиций, *int n* – целое значение количества хранимых данных. Далее создаются три переменные *(*head, *tmp, *first*) – указатели типа *MusicalComposition*. С помощью цикла *for* до значения *i* меньше *n*, происходит инициализация каждого элемента списка. Если *i==0*, то есть данные являются первыми, *first* присваивается указатель на элементы списка, создаваемые с помощью функции *createMusicalComposition(array_names[i], array_authors[i], array_years[i])*. Указатель *head* теперь соответствуют указателю на первый элемент *first* - головной элемент. Если условие *i==0* не выполняется, то *tmp* присваивается указатель на элементы списка,

создаваемые с помощью функции *createMusicalComposition(array_names[i], array_authors[i], array_years[i])*. Для *head* следующим элементом инициализируется (*head -> next*) *tmp*. Для *tmp* предыдущим элементом инициализируется (*tmp -> prev*) *head*. В конце каждой итерации *head* инициализируется *tmp*. Функция возвращает *first* - первый элемент.

Функция добавление элемента в конец списка:

Функция типа *void* называется *push* и принимает на вход указатель на первый элемент списка **head* типа *MusicalComposition* и указатель на элемент **element* типа *MusicalComposition*, который необходимо добавить в конец списка. Для того чтобы добавить элемент в конец списка, необходимо получить указатель на элемент, указатель на следующий от которого равен *NULL*, то есть значения в нем нет. С помощью цикла *while()*, который работает до тех пор пока следующий элемент *next* относительно текущего *head* (*head -> next*, обращение к элементу *next* используемой структуры) не равен нулевому значению *NULL*. На каждой итерации происходит переход к следующему элементу списка: *head = head -> next*. По завершению цикла в элемент структуры *next* относительно текущего элемента *head* (*head -> next*, обращение к элементу *next* используемой структуры) записывается указатель *element*. В элемент структуры *prev* относительно добавленного в конец элемента *element*(*element -> prev*, обращение к элементу *prev* используемой структуры) записывается указатель *head*, он становится предыдущим элементом для нового.

Функция удаления элемента из списка:

Функция типа *void* называется *removeEl* и принимает на вход указатель на первый элемент списка **head* типа *MusicalComposition* и указатель на строку **name_for_remove* типа *char*, с которой необходимо сравнивать элемент списка и удалить в случае совпадения. С помощью цикла *while*, который работает до тех пор, пока указатель *head* не равен *NULL*, происходит сравнение двух строк с помощью условного оператора *if* и применения функции *strcmp()* стандартной библиотеки языка: элемента структуры *name*,

которая определяется для элемента списка *head*(*head->name*) и *name_for_remove*. При совпадении строк функция возвращает значение равное *0*, происходит удаление элемента. Следующим элементом для предыдущего относительно *head* (*head->prev->next*) становится элемент, который на данный момент выполнения итерации являлся для *head* следующим (*head->next*). Предыдущим элементом для следующего относительно *head* (*head->next->prev*) становится элемент, который на данный момент выполнения итерации являлся для head предыдущим *(head->prev*). Выделенная память на уже удаленный элемент очищается с помощью функции *free()*. Для перехода к следующему элементу списка значению *head* присваивается указатель на следующий элемент списка *head -> next*.

Функция подсчета количество элементов списка:

Функция типа *int* называется *count* и принимает на вход указатель на первый элемент списка **head* типа *MusicalComposition*. В начале функции происходит инициализация переменной *cnt = 0* типа *int*, для подсчета количество элементов списка. С помощью цикла *while*, который работает до тех пор, пока указатель *head* не равен *NULL*, на каждой итерации значение *cnt++* увеличивается на единицу. Для перехода к следующему элементу списка значению *head* присваивается указатель на следующий элемент списка *head -> next*. Функция возвращает значение инициализируемой в начале переменной *return cnt*.

Функция вывода названия композиций:

Функция типа *void* называется *print_names* и принимает на вход указатель на первый элемент списка **head* типа *MusicalComposition*. Далее с помощью цикла *do-while* выполняется вывод значения *head->name* (строки, содержащей название музыкальной композиции) с помощью функции *puts()*. Для перехода к следующему элементу списка значению *head* присваивается указатель на следующий элемент списка *head -> next*. Работа цикла продолжается до тех пор, пока указатель *head* не равен *NULL*.

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	Программа работает корректно.
2.	2 All for us Labrinth 2018	All for us Labrinth 2018 2 3 All for us	Программа работает корректно.

	Enemy Imagine Dragons 2015 Shivers Ed Sheeran 2021 Enemy	Shivers 2	

Выводы.

В ходе лабораторной работы была освоена работа с линейными списками, а именно двунаправленными. Улучшены навыки работы со структурами. Написана программа, которая создает двунаправленный список и осуществляет работу с ним.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Chernyakova_Valeria_lb2/main.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* autor,int year);

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


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, *tmp, *first;
    for (int i = 0; i < n; i++){
            if (i == 0){
                first = createMusicalComposition(array_names[i], array_authors[i], array_years[i]);
                head = first;
            }
            else {
                tmp = createMusicalComposition(array_names[i], array_authors[i], array_years[i]);
```

```
                head -> next = tmp;
                tmp -> prev = head;
                head = tmp;
            }
    }
    return first;
}


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

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


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

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


int count(MusicalComposition* head);

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


void print_names(MusicalComposition* head);

void print_names(MusicalComposition* head){
    do {
```

```
            puts(head -> name);
            head = head -> next;
        }while(head != 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;

    }
```