МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Программирование»
Тема: Структуры данных, линейные списки.

Студентка гр. 0382 _____ Кривенцова Л.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Освоить и научиться создавать и редактировать структуры данных и линейные списки языка программирования Си.

Задание.

Вариант 1.

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*

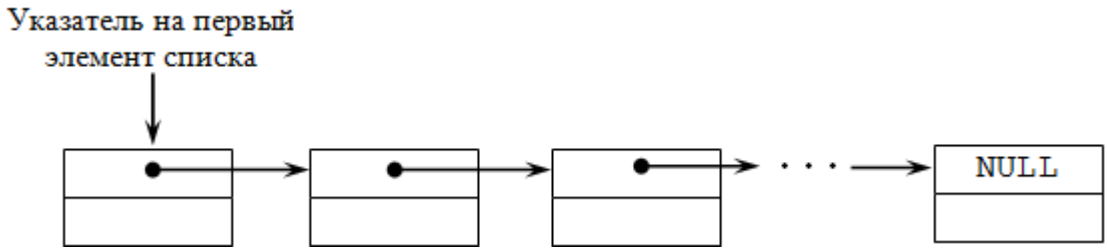
В функции main написана некоторая последовательность вызова команд для проверки работы вашего списка.

Функцию main менять не нужно.

Основные теоретические положения.

Список - некоторый упорядоченный набор элементов любой природы.

Линейный однонаправленный (односвязный) список - список, каждый элемент которого хранит помимо значения указатель на следующий элемент. В последнем элементе указатель на следующий элемент равен NULL (константа нулевого указателя).



! Чтобы использовать NULL, необходимо подключить #include <stddef.h>

Давайте сделаем из структуры Circle (урок 6.1 шаг 6) элемент списка Node:

```
struct Node{
int x;
int y;
float r;
struct Node* next; // указатель на следующий элемент
};
```

И проинициализируем два элемента списка в функции main():

```
 int main(){
struct Node * p1 = (struct Node*)malloc(sizeof(struct Node));
struct Node * p2 = (struct Node*)malloc(sizeof(struct Node));
  p1->x = 2; // используем -> поскольку p1 - указатель на структуру Node
   p1->y = 2;
   p1->r = 2.5;
   p2->x = 5;
   p2->y = 5;
   p2->r = 5.5;
   p1->next = p2;
   p2->next = NULL;
   free(p1);
   free(p2);
    return 0;
}
```

У нас получился линейный список из двух элементов: p1 и p2.

Выполнение работы.

Ход решения:

Используется стандартная библиотека языка си и её заголовочные файлы *stdlib.h, string.h*(для работы со строками) и *stddef.h* (для использование NULL).

С помощью ключевого слова typedef определяется тип данных (structure) MusicalComposition. Структура содержит 5 полей: 3 для хранения информации об элементе списка и 2 (prev и next), хранящие ссылки для перемещения по списку.

Переменные:

1. Функции MusicalComposition* createMusicalComposition(char* name, char* author, int year):

MusicalComposition cur*; - указатель на создаваемый в функции элемент списка.

2. Функции MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n):

int i; - целочисленная переменная, счётчик для цикла *for;*

MusicalComposition present;* - указатель на текущий элемент списка;

MusicalComposition past;* - указатель на текущий элемент списка;

MusicalComposition head;* -указатель на первый элемент списка.

3. Функции int count(MusicalComposition* head):

int counttt; - целочисленная переменная, счётчик элементов списка.

4. Функции void print_names(MusicalComposition* head):

*MusicalComposition *present;* - указатель на текущий элемент списка.

Функции:

1. MusicalComposition* createMusicalComposition(char* name, char* author, int year).

Функция для создания элемента списка (тип элемента MusicalComposition). Выделяет динамическую память для элемента списка и присваивает значения его полям.

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

Функция создает список музыкальных композиций MusicalCompositionList.

3. void push(MusicalComposition* head, MusicalComposition* element).

Функция добавляет element в конец списка musical_composition_list.

4. void removeEl (MusicalComposition* head, char* name_for_remove).

Функция удаляет элемент element списка, у которого значение name равно значению name_for_remove.

5. int count(MusicalComposition* head).

Функция считает и возвращает количество элементов списка.

6. void print_names(MusicalComposition* head).

Функция выводит названия композиций.

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	Программа выводит верный ответ.

Выводы.

Были освоены навыки создавать и редактировать структуры данных и линейные списки языка программирования Си.

Разработана программа, создающая двунаправленный список и **api** (***a****pplication* ***p****rogramming* ***i****nterface - в данном случае набор функций*) для работы со списком.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stddef.h>
typedef struct MusicalComposition
{
    char name[80];
    char author[80];
    int year;
    struct MusicalComposition *next;
    struct MusicalComposition *prev;
} MusicalComposition;
// Описание структуры MusicalComposition


// Создание структуры MusicalComposition

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

// Функции для работы со списком MusicalComposition

MusicalComposition* createMusicalCompositionList(char** array_names,
char** array_authors, int* array_years, int n){
    int i; MusicalComposition* present; MusicalComposition* head;
MusicalComposition* past;
    head = createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
    past = head;
    for (i = 1; i<n; i++)
    {
        present = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        if (i == n-1) present->next=NULL;
        past->next = present;
        present->prev = past;
        past = present;
    }
    return head;
}
```

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

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

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

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

```
}
```