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. 3343 Пухов А.Д.

Преподаватель Государкин Я.С.

Санкт-Петербург
2024

Цель работы.

Изучение и применение на практике двунаправленных списков в языке Си.

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.

Выполнение работы.

Структура MusicalComposition:

- **char *name** — название композиции
- **char *author** — автор композиции
- **int year** — год создания
- **struct MusicalComposition *prev** — указатель на предыдущий элемент списка
- **struct MusicalComposition *next** — указатель на следующий элемент списка

Функции:

MusicalComposition *createMusicalComposition(char *name, char *autor, int year) — создаёт новый элемент списка.

MusicalComposition *createMusicalCompositionList(char **array_names, **char **array_authors, int *array_years, int n)** — создаёт список музыкальных композиций.

void push(MusicalComposition *head, MusicalComposition *element) — добавляет новый элемент в конец списка.

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 название композиций.

Разработанную программу смотрите 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 AuthorityFields of Gold Sting 1993 7 8 Fields of Gold In the Army Now Mixed Emotions Billie Jean Seek and Destroy Wicked Game Sonne 7	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	OK

Выводы.

В данной лабораторной работе было изучено и применено на практике написание двунаправленных списков в языке Си.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: app-001.c

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

// Описание структуры MusicalComposition
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 *createMusicalCompositionList(char **array_names, char
**array_authors, int *array_years, int n);
void push(MusicalComposition *head, MusicalComposition *element);
void removeEl(MusicalComposition *head, char *name_for_remove);
int count(MusicalComposition *head);
void print_names(MusicalComposition *head);

// Создание структуры MusicalComposition

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

// Функции для работы со списком MusicalComposition

MusicalComposition *createMusicalCompositionList(char **array_names, char
**array_authors, int *array_years, int n)
{
```

```
    if (n <= 0)
    {
        return NULL;
    }
    MusicalComposition *head = createMusicalComposition(array_names[0],
array_authors[0], array_years[0]);

    for (int i = 1; i < n; i++)
    {
        MusicalComposition *element = createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
        push(head, element);
    }
    return head;
}

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

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

```
            }
            else if ((tmp->prev != NULL) && (tmp->next == NULL))
            {
                tmp->prev->next = NULL;
            }
            else
            {
                tmp->prev->next = tmp->next;
                tmp->next->prev = tmp->prev;
            }
            free(tmp);
            tmp = NULL;
            break;
        }
    }
}

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

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