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 _____ Гребнев Е.Д.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург
2024

Цель работы

Цель лабораторной работы заключается в изучении связанных списков и их применении в программах на языке Си.

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.

Выполнение работы

Программа создает структуру данных двусвязного списка для хранения музыкальных композиций. В структуре `MusicalComposition` хранятся данные о названии композиции, ее авторе и годе создания, а также указатели на предыдущий и следующий узлы в списке.

Функция `createMusicalComposition` создает и инициализирует новую музыкальную композицию.

Функция `createMusicalCompositionList` создает и инициализирует список музыкальных композиций из массивов названий, авторов и годов.

Функция `push` добавляет новый узел в конец списка.

Функция `removeEl` удаляет узел из списка по названию композиции.

Функция `count` подсчитывает количество узлов в списке.

Функция `print_names` выводит названия композиций из списка.

В функции `main` программа считывает данные о музыкальных композициях с клавиатуры, создает список, добавляет новую композицию, удаляет выбранную композицию и выводит названия оставшихся композиций.

В конце программы освобождаются выделенные ресурсы.

Разработанный программный код см. в приложении А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
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
2.	8 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	Fields of Gold Sting 1993 8 9 Fields of Gold In the Army Now Mixed Emotions Billie Jean Seek and Destroy Wicked Game Points of Authority Sonne Points of Authority 9

	Points of Authority Linkin Park 2000 Sonne Rammstein 2001 Points of Authority Fields of Gold Sting 1993	

Выводы

Были изучены линейные двунаправленные списки и работа с ними на языке Си. Реализована программа, которая добавляет элементы в список и удаляет выбранные пользователем. Алгоритм работает эффективнее за счет использования связанных списков.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


// О п и с а н и е  с т р у к т у р ы MusicalComposition

typedef struct MusicalComposition {
    char name[80];
    char author[80];
    int year;
    struct MusicalComposition *parent;
    struct MusicalComposition *child;
} MusicalComposition;


// С о з д а н и е  с т р у к т у р ы MusicalComposition
MusicalComposition *createMusicalComposition(const char *name, const
char *author, int year) {
    MusicalComposition            *musicalComposition            =
malloc(sizeof(MusicalComposition));

    strcpy(musicalComposition->name, name);
    strcpy(musicalComposition->author, author);
    musicalComposition->year = year;

    return musicalComposition;
}

// Ф у н к ц и и  д л я  р а б о т ы  с о  с п и с к о м MusicalComposition

MusicalComposition *createMusicalCompositionList(char **array_names,
char **array_authors, int *array_years, int n) {
    MusicalComposition               *compositions               =
malloc(sizeof(MusicalComposition) * n);

    for (int i = 0; i < n; ++i) {
        strcpy(compositions[i].name, array_names[i]);
        strcpy(compositions[i].author, array_authors[i]);
        compositions[i].year = array_years[i];
        compositions[i].parent = NULL;
        compositions[i].child = NULL;
        if (i != 0) {
            compositions[i].parent = &compositions[i - 1];
            compositions[i - 1].child = &compositions[i];
        }
    }

    return compositions;
}
```

```
void push(MusicalComposition *head, MusicalComposition *element) {
    MusicalComposition *current = head;
    while (current->child != NULL) {
        current = current->child;
    }
    current->child = element;
    element->parent = current;
}

void removeEl(MusicalComposition *head, char *name_for_remove) {
    MusicalComposition *current = head;
    while (current != NULL) {
        if (strcmp(name_for_remove, current->name) == 0) {
            current->parent->child = current->child;
            current->child->parent = current->parent;
        };
        current = current->child;
    }
}

int count(MusicalComposition *head) {
    MusicalComposition *current = head;
    int count = 0;
    while (current != NULL) {
        count++;
        current = current->child;
    }

    return count;
}

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

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;

    }
```