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. 3342 _____ Галеев А.Д.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Цель работы состоит в изучении основных принципов организации и операций с линейными списками, понять различные способы реализации линейных списков, включая массивы, связанные списки, и их сравнительные преимущества и недостатки, а также их применения в различных областях программирования.

Задание

Вариант №1

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** 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]**).

Основные теоретические положения

Для решения задач в программе использовались функции стандартной библиотеки языка си. Так же была использована библиотека string.h, которая предоставляет набор функций для работы со строками.

Выполнение работы

Определение структуры MusicalComposition:

Она содержит поля для хранения имени композиции (name), имени автора (author), года создания (year) и указателя на следующий элемент списка (next).

Функция createMusicalComposition:

Выделяет память для новой структуры MusicalComposition и инициализирует ее переданными значениями имени, автора и года.

Функция createMusicalCompositionList:

Создает список MusicalComposition из массивов имен, авторов и годов.

Для каждого элемента массива создается новая структура MusicalComposition, которая добавляется в конец списка.

Функция push:

Добавляет новый элемент element в конец списка, начиная с головы списка head.

Функция removeEl:

Удаляет элемент списка с заданным именем name_for_remove.

Функция count:

Возвращает количество элементов в списке.

Функция print_names:

Печатает имена композиций, начиная с головы списка head.

В функции main:

Сначала считывается количество композиций.

Затем происходит выделение памяти и считывание данных для каждой композиции.

Создается список head, содержащий эти композиции.

Далее считываются данные для новой композиции (name_for_push, author_for_push, year_for_push) и добавляются в список с помощью push.

Также считывается имя композиции, которую нужно удалить (name_for_remove), и эта композиция удаляется из списка.

Тестирование

Результаты тестирования представлены в табл. 1

Таблица 1 – Результаты тестирования

№ проверки	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

Выводы

Было изучено понятие линейных списков.

Разработана программа создающая двунаправленный список музыкальных композиций.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main_lb2

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

// Описание структуры MusicalComposition

typedef struct MusicalComposition {
    char name[80];
    char author[80];
    int year;
    struct MusicalComposition* next;
    struct MusicalComposition* prev;
} MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* author,
int year) {
    MusicalComposition* composition =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    if (composition == NULL) {
        exit(1);
    }

    strncpy(composition->name, name, 80);
    composition->name[80] = '\0';
    strncpy(composition->author, author, 80);
    composition->author[80] = '\0';
    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) {
    MusicalComposition* head = createMusicalComposition(array_names[0],
array_authors[0], array_years[0]);
    MusicalComposition* current = head;

    for (int i = 1; i < n; i++) {
        MusicalComposition* composition =
createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        current->next = composition;
        composition->prev = current;
        current = composition;
```

```
    }
    return head;
}

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

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

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

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

}
```