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. 3341 _____ Анисимов Д.А.

Преподаватель _____ Глазунов С.А.

Санкт-Петербург
2024

Цель работы

Цель работы заключается в изучении линейных структур данных в языке программирования Си и методов их реализации.

Задание

1 вариант.

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

В языке программирования C структуры представляют собой пользовательские типы данных, объединяющие несколько переменных различных типов через ключевое слово struct. Это позволяет создавать более сложные объекты, упрощая управление и использование данных. Линейные списки абстрактная структура данных, состоящая из узлов с данными и указателями на следующие узлы. Однонаправленные списки имеют однонаправленные связи между узлами, в то время как двунаправленные списки позволяют двигаться по списку в обоих направлениях. Реализация этих структур включает создание соответствующих структур для узлов списка и функций для их работы, таких как добавление, удаление и обход элементов, что обеспечивает эффективное управление данными и выполнение операций, таких как поиск и сортировка.

Выполнение работы

Создаётся структура данных *MusicalComposition* — узел двусвязного списка. Он содержит информацию о музыкальной композиции (строку *char* name* — название композиции, строку *char* author* — имя автора, целое число *int year* — год создания) и два указателя *struct MusicalComposition** — на следующий и предыдущий узлы списка.

Далее, реализуется функция createMusicalComposition, которая принимает данные о названии, авторе и годе композиции и создает новый элемент списка на основе этих данных. Для этого функция выделяет память под новый элемент, копирует переданные строки и сохраняет год создания.

После этого создается функция createMusicalCompositionList, которая создает сам список музыкальных композиций на основе переданных массивов данных о названиях, авторах и годах. Функция создает первый элемент списка, а затем последовательно добавляет остальные элементы, устанавливая соответствующие указатели на предыдущие и следующие элементы.

Для работы с созданным списком реализуются функции push, removeEl, count и print_names.

Функция push осуществляет добавление нового элемента в конец списка. Путем последовательного перемещения по элементам списка до его последнего элемента она устанавливает указатель next последнего элемента на новый элемент, одновременно обновляя указатель prev нового элемента на предыдущий элемент списка.

Функция removeEl предназначена для удаления элемента списка с указанным названием. Она просматривает все элементы списка, сравнивая названия, и при обнаружении удаляемого элемента корректно обновляет указатели соседних элементов.

Функция count возвращает количество элементов в списке. Для этого она просто перебирает все элементы списка, подсчитывая их количество.

И, наконец, функция print_names выводит названия всех композиций, находящихся в списке. Она последовательно проходит по элементам списка и выводит на экран названия каждой композиции.

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	4 Mixed Emotions The Rolling Stones 1989 Billie Jean Michael Jackson 1983 Wicked Game Chris Isaak 1989 Points of Authority Linkin Park 2000 Sonne Rammstein 2001 Points of Authority	Mixed Emotions The Rolling Stones 1989 4 5 Mixed Emotions Billie Jean Wicked Game Sonne 4	Тест с e.moevm
2	3 Rakbladsvalsen I Hypothermia 2006 M/s salmonella Lifelover 2006 Mountains Made Of Steam Silver Mt. Zion 2012 So Special FFF	Rakbladsvalsen I Hypothermia 2006 3 4 Rakbladsvalsen I M/s salmonella So Special 3	Удаление последнего элемента.

	2021 Mountains Made Of Steam		

Выводы

В ходе данного исследования была поставлена цель освоения работы с линейными списками. Для достижения этой цели были выполнены следующие задачи:

1. Изучение структуры "список" как абстрактной структуры данных, позволяющей хранить и организовывать элементы в линейной последовательности.

2. Ознакомление с операциями, используемыми для работы со списками, такими как добавление элемента, удаление элемента, поиск элементов и т.д.

3. Изучение способов реализации этих операций на языке программирования C, включая работу с указателями и динамическим выделением памяти.

4. Разработка программы, которая реализует двусвязный линейный список и решает конкретную задачу в соответствии с индивидуальным заданием. Программа содержит функции для создания списка, добавления элементов, удаления элементов и вывода информации о списках.

Таким образом, выполнение поставленных задач позволило освоить работу с линейными списками и применить полученные знания при разработке программы на языке C.

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;
    char* author;
    int year;

    struct MusicalComposition* prev;
    struct MusicalComposition* next;
} MusicalComposition;


// С о з д а н и е  с т р у к т у р ы MusicalComposition

MusicalComposition*  createMusicalComposition(char*  name,  char*
autor,int year);

// 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);

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

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

int count(MusicalComposition* head);

void print_names(MusicalComposition* head);


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;
```

```
    }

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

    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*                    temp                    =
createMusicalComposition(array_names[1],              array_authors[1],
array_years[1]);
        head->next = temp;
        temp->prev = head;

        for (int i = 2; i < n; i++){
            temp->next    =    createMusicalComposition(array_names[i],
array_authors[i], array_years[i]);
            temp->next->prev = temp;
            temp = temp->next;
        }
        return head;
    }

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

        return counter;
    }

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

        head->next = element;
        element->prev = head;

        while(head->prev != NULL)
            head = head->prev;
    }
```

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

    while(head->prev != NULL)
        head = head->prev;
}

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