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 _____ Песчатский С. Д.

Преподаватель _____ Глазунов С. А.

Санкт-Петербург
2024

Цель работы

Изучить композицию двунаправленного списка, и реализовать его при помощи структур на языке Си. Добавить несколько базовых методов для работы с экземплярами структуры двунаправленного списка.

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:

1) 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]).

2)void push(MusicalComposition* head, MusicalComposition* element); // добавляет element в конец списка musical_composition_list

3)void removeEl (MusicalComposition* head, char* name_for_remove); // удаляет элемент element списка, у которого значение name равно значению name_for_remove

4)int count(MusicalComposition* head); //возвращает количество элементов списка

5)void print_names(MusicalComposition* head); //Выводит названия композиций.

Выполнение работы

Необходимо организовать структуру двусвязанного списка, содержащую данные: название композиции (char* name), автор композиции (char* author), год создания (int year), указатель на следующую композицию (struct MusicalComposition* next), указатель на предыдущую композицию (struct MusicalComposition* prev).

После этого необходимо описать функции для работы с этой структурой:

1) Создать музыкальную композицию (MusicalComposition* createMusicalComposition(char* name, char* author, int year)) - создание переменной данной структуры. Выделяется память с помощью malloc(), затем присваиваются значения полям.

2) Создать список музыкальных композиций (MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)) - создает список музыкальных композиций. Сначала память выделяется для головы списка, затем в цикле присваиваются значения полям элементов списка, выделяется память для следующего элемента списка, задаются связи между элементами.

3) Добавить элемент в конец списка (void push(MusicalComposition* head, MusicalComposition* element)) - создается текущий элемент, который с помощью цикла находит последний элемент списка (где next == NULL).

4) Удалить элемент из списка (void removeEl(MusicalComposition* head, char* name_for_remove)) - удаляет элемент по названию композиции. С помощью цикла находим совпадение, изменяем связи в списке и освобождаем память удаленного элемента.

5) Подсчет количества элементов в списке (int count(MusicalComposition* head)) - заводится переменная-счетчик, которая с помощью цикла подсчитывает количество элементов.

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

Выводы

Была разработана программа, создающая двунаправленный список из музыкальных композиций и выполняющая с ним определенные функции. Изучена работа с линейными списками, со структурами и реализация их на языке Си.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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


// Описание структуры MusicalComposition
typedef struct MusicalComposition{
     char* name;
     char* author;
     int year;
     struct MusicalComposition* next;
     struct MusicalComposition* previous;
} MusicalComposition;

// Создание структуры MusicalComposition
MusicalComposition* createMusicalComposition(char* name, char* au-
thor,int year){
     MusicalComposition      *music=malloc(sizeof(MusicalComposi-
tion));
     music->name=name;
     music->author=author;
     music->year=year;
    music->next=NULL;
    music->previous=NULL;
     return music;
}

// Функции для работы со списком MusicalComposition
void push(MusicalComposition* head, MusicalComposition* element){
     while(head->next!=NULL) head=head->next;
     element->previous=head;
     head->next=element;
}

MusicalComposition*    createMusicalCompositionList(char**    ar-
ray_names, char** array_authors, int* array_years, int n){
     MusicalComposition*        head=createMusicalComposition(ar-
ray_names[0],array_authors[0],array_years[0]);
     MusicalComposition* tmp;
     for(int i=1;i<n;i++){
          tmp=createMusicalComposition(array_names[i],array_au-
thors[i],array_years[i]);
          push(head,tmp);
     }
     return head;
}

void removeEl(MusicalComposition *head, char *name_for_remove){
     while(1){
          if(strcmp(head->next->name,name_for_remove)==0){
               if(head->next->next!=NULL) {
               MusicalComposition* tmp;
```

```
                    tmp = head->next->next;
                    free(head->next);
                    head->next=tmp;
                    }

                    break;
               }
               head=head->next;
          }
     }

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

     void print_names(MusicalComposition* head){
          while(head!=NULL){
               printf("%s\n",head->name);
               head=head->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(au-
thor)+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 = createMusicalComposi-
tion(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;

    }
```