МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЕТ по лабораторной работе №4

по дисциплине «Программирование»

Тема: “Структуры данных, линейные списки”

Студент гр. 7381 Аженилок В.А.

Преподаватель Берленко Т.А

Санкт-Петербург

Цель:

Научиться работать с двунаправленным списком. Закрепить имеющиеся знания по выделению и очистке динамической памяти. Написать функции для работы со списком: добавлять, удалять, создаваться и выводить элементы списка и их количество.

Задание:

Создать 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]**).
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

Основные теоретические положения:

Создана структура с именем MusicalComposition с членами типов char*,char*,int , и 2 члена с типом указателя на саму структуру, эти члены содержат адреса предыдущих и последующих элементов.

Далее создается ряд функция для работы с двухсвязным списком:

createMusicalComposition.

Создана функция createMusicalComposition. Эта функция создает элемент списка. На вход она получает 3 параметра с именем, автором и годом создания. В ней используются такие функции как malloc(динамическое выделение памяти) и strcpy(копирование символов).Функция возвращает указатель на первый элемент списка.

Push

Создана функция push. Эта функция добавляет элемент списка в конец списка. На вход получает указатель на первый элемент и элемент который нужно добавить в список. С помощью цикла while определяется местоположение последнего элемента. Функция ничего не возвращает.

createMusicalCompositionList

Создана функция **createMusicalCompositionList.** Эта функция создает целый список. На вход получает массив имен, авторов, лет издания и количество элементов которые будут в списке. Если количество элементов равняется нулю функция возвращает NULL. Иначе список заполняет при помощи цикла for. В нѐм вызываются описанные функции push и createMusicalComposition. Функция возвращает указатель на первый элемент списка.

removeEl

Создана функция removeEl. Эта функция удаляет элемент из списка, если значение члена name совпало с полученной функцией строкой. На вход поступает указатель на первый элемент и определенная строка символов. Функция ничего не возвращает.

Count

Создана функция count. Эта функция считает количество элементов в списке. На вход поступает указатель на первый элемент. В цикле while перебираются все элементы и счетчик с каждым шагом увеличивается на единицу. Функция возвращает количество элементов.

Print_names

Создана функция print_names. Функция выводит на экран значение члена name всех элементов. На вход получает указатель на первый элемент списка. Функция ничего не возвращает.

Вывод

В данной лабораторной работе создан двунаправленный список с информацией о музыкальных композициях. В ходе лабораторной работы получены знания по нахождения утечек памяти и закреплены знания по её

выделению. Освоены алгоритмы написания функций для работы с двусвязным списком.

Код программы:

main.c

```c
#include "mc.h"

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

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;
}
```

Makefile

```
obj = main.o mc.o
exe = mc

all: $(obj)
    gcc -o $(exe) $(obj)

main.o: main.c mc.h
```

```
    gcc -c main.c

mc.o: mc.c mc.h
    gcc -c mc.c

clean:
    rm $(exe) $(obj)
```

mc.c

```
#include "mc.h"

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

MusicalComposition* createMusicalComposition(char* name, char* author, int year)
{
    MusicalComposition* mc =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
    mc->name = (char*)malloc(81 * sizeof(char));
    strcpy(mc->name, name);
    mc->author = (char*)malloc(81 * sizeof(char));
    strcpy(mc->author, author);
    mc->year = year;
    mc->next = NULL;
    mc->prev = NULL;
    return mc;
}

MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years, int n)
{
    if (n <= 0)
        return NULL;
    MusicalComposition *mc = createMusicalComposition(array_names[0],
array_authors[0], array_years[0]);
    MusicalComposition *iter = mc;
    for (int i = 1; i < n; ++i)
    {
        iter->next = createMusicalComposition(array_names[i], array_authors[i],
array_years[i]);
        iter->next->prev = iter;
        iter = iter->next;
    }
    return mc;
}

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

```
{
    if (*head == NULL) {
        *head = element;
        return;
    }
    MusicalComposition* iter;
    for (iter = *head; iter->next; iter = iter->next);
    element->prev = iter;
    iter->next = element;
}

void removeEl(MusicalComposition** head, char* name_for_remove)
{
    if (*head == NULL)
        return;
    MusicalComposition* iter = *head;
    MusicalComposition* temp;
    while (iter != NULL) {
        temp = iter;
        iter = iter->next;
        if (strcmp(temp->name, name_for_remove) == 0)
        {
            if (temp == *head)
                *head = iter;
            if (temp->prev != NULL)
                temp->prev->next = temp->next;
            if (temp->next != NULL)
                temp->next->prev = temp->prev;
            free(temp->name);
            free(temp->author);
            free(temp);
        }
    }
}

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

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

```
        iter = iter->next;
    }
}
```

mc.h

```
#ifndef __MC_H__
#define __MC_H__

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

MusicalComposition* createMusicalComposition(char*, char*, int);

MusicalComposition* createMusicalCompositionList(char**, char**, int*, int);

void push(MusicalComposition**, MusicalComposition*);

void removeEl(MusicalComposition**, char*);

int count(MusicalComposition*);

void print_names(MusicalComposition*);

#endif // __MC_H__
```