МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Структуры данных, линейные списки

Студент гр. 7381 Аженилок В.А

Преподаватель Берленко Т.А.

Санкт-Петербург

2017

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Аженилок В.А.

Группа 7381

Тема работы: Структуры данных, линейные списки

Исходные данные:

- 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]**).

- 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. *Фу*нкция 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.

Создать функцию для разделения списка на две части(индекс центрального элемента округлить в большую сторону),затем поменять их местами и соединить в один список.

Создать удобный интерфейс для работы программы.

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 23.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студент Аженилок В.А.

_____

Преподаватель Берленко Т.А.

_____

АННОТАЦИЯ

В результате выполнения курсовой работы была создана программа на языке СИ, которая позволяет работать со списком и с функциями для него, такими как: удаление элемента, добавление элемента в конец списка, подсчет количества элементов, удаление всех элементов, год которых меньше данного, а также вывод самого списка. Создан удобный интерфейс для пользования программой.

SUMMARY

As a result of the course work, a C program was created that allows you to work with the list and with functions for it, such as: deleting an item, adding an item to the end of the list, counting the number of items, deleting all elements in which the year is less than this, and also list output. A convenient interface for using the program was created.

СОДЕРЖАНИЕ

Введение

Целью работы является закрепить имеющиеся знания по выделению и очистке динамической памяти, научиться работать с двунаправленными списками, уметь создавать функции для работы с ними (добавлять и удалять элементы, подсчитывать их количество).

Функции программы

1. ФУНКЦИЯ MAIN.

Пользователю предлагают многократный выбор дальнейших действий

№1 – Добавить элемент в список.

№2 – Удалить элемент из списка.

№3 – Вывести количество элементов списка.

№4 – Вывести список композиций.

№5 – Разделение списка.

№6 Завершение программы.

2. ФУНКЦИИ ДЛЯ РАБОТЫ СО СПИСКОМ

1.MusicalComposition* createMusicalComposition(char* name, char* author,int year)

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.

2. MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)

Функция получает количество существующих композиций и указатели на три массива: массив названий (char** array_names), массив авторов (char** array_authors) и массив годов созданий (int* array_years); и формирует линейный двунаправленный список. Функция возвращает указатель на первый элемент списка.

3.void removeEl(MusicalComposition* list, char* name_for_remove)

Функция получает указатель на начало списка (MusicalComposition* head) и название композиции (char* name_for_remove), которую нужно удалить. Перемещаясь по списку,

она сверяет название композиции с переданной строкой. При совпадении значений функция связывает предыдущий и следующий элементы посредством указателей, а затем очищает удаленный элемент. Функция ничего не возвращает.

4. int count(MusicalComposition* list)

Функция получает указатель на первый элемент списка (MusicalComposition* head), и, пробегая весь список, подсчитывает количество элементов в нем. Функция возвращает количество элементов списка.

5. void print_names(MusicalComposition* list)

Данной функции передается указатель на первый элемент списка (MusicalComposition* list). Перебирая все элементы, функция печатает названия каждой композиции.

6. void push(MusicalComposition** head, MusicalComposition *element)

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

7. void swaphalfs(MusicalComposition** mcl)

Функция для разделения списка на две части(индекс центрального элемента округлить в большую сторону),затем обмена их местами и соединения в один список.

Заключение

В ходе выполнения данной курсовой работе создан двунаправленный линейный список, функции для работы с ним и удобный интерфейс для пользования. Были получены знания по работе со списком, закреплены знания по выделению и очищению динамической памяти . Была создана функция для разделения списка на две части(индекс центрального элемента округлить в большую сторону),затем обмена их местами и соединения в один список.

Список использованных источников

- Б.Керниган Д.Ритчи “язык прогаммирование Си”
- Подбельский В.С. Фомин С.С. “Курс программирования на Си: учебник “

Приложение А. Исходный код программы.

Makefile

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

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

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

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

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

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

main.c

```
#include "mc.h"
#include "mcc.h"


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

int main() {
    char command;

    MusicalComposition* list = NULL;

    char* name = (char*)malloc(81 * sizeof(char));
    char* author = (char*)malloc(81 * sizeof(char));
    int year;

    int n;

    printf("Введите команду:\n1: Добавить элемент в список.\n2: Удалить
элемент из списка.\n3: Вывести количество элементов списка.\n4: Вывести
```

```
список композиций.\n5: Разделить список на 2 части\n6: Завершение
программы.\n");
    do {
        printf("-> ");
        scanf(" %c", &command);
        switch (command) {
        case '1':
            printf("Введите название композиции:\t");
            scanf("%s", name);
            printf("Введите автора композиции:\t");
            scanf("%s", author);
            printf("Введите год издания:\t\t");
            scanf("%d", &year);
            push(&list, createMusicalComposition(name, author, year));
            break;
        case '2':
            printf("Введите название композиции:\t");
            scanf("%s", name);
            removeEl(&list, name);
            break;
        case '3':
            printf("Элементов в списке: %d\n", count(list));
            break;
        case '4':
            print_names(list);
            break;
        case '5':
            printf("Magic\n");
            swaphalfs(&list);
            break;
        case '6':
            printf("Освобождение памяти\n");
            break;
        default:
            printf("Неизвестная команда\n");
            break;
        }
    } while (command != '6');

    free(name);
    free(author);
    mclfree(&list);

    return 0;
}
```

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__
```

mcc.c.

```
#include "mcc.h"
#include "mc.h"

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

void swaphalfs(MusicalComposition** mcl) {
    if (!(*mcl) || !((*mcl)->next))
```

```
        return;
    MusicalComposition* head = *mcl;
    int mclsize = count(head);
    int mclhalf = mclsize / 2;
    int i;
    MusicalComposition* midd = head;
    for (i = 0; i < mclhalf - 1; ++i)
        midd = midd->next;
    MusicalComposition* tail = midd;
    for (i = mclhalf; i < mclsize; ++i)
        tail = tail->next;
    MusicalComposition* temp = head->next;
    *mcl = midd->next;
    tail->next = head;
    head->prev = midd;
    midd->next->prev = NULL;
    midd->next = NULL;

}

void mcfree(MusicalComposition* mc) {
    if (!mc)
        return;
    free(mc->name);
    free(mc->author);
    free(mc);
}

void mclfree(MusicalComposition** mcl) {
    MusicalComposition* iter = *mcl;
    MusicalComposition* temp;
    while (iter) {
        temp = iter;
        iter = iter->next;
        mcfree(temp);
    }
}
```

mcc.h.

```
#ifndef __MCC_H__
#define __MCC_H__

#include "mc.h"
```

```
void swaphalfs(MusicalComposition** mcl);

void mcfree(MusicalComposition*);

void mclfree(MusicalComposition**);

#endif // __MCC_H__
```