**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: **Двунаправленный список**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Васильев А.А. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

**Цель работы**

Создайте 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:

* 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]**).
  + 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.

Функция “main” для проверки работоспособности функции уже написана.

**Ход работы**

1. **struct.h**

Создадим заголовочный файл с объявлением нового типа:

struct MusicalComposition{ //Объявляем структуру

char name[80];

char author[80];

int year;

struct MusicalComposition \*next, \* prev; //Указатели на

}; //связанные элементы.

typedef struct MusicalComposition MusicalComposition;

//Присваиваем, для удобства, новое имя типу-структуре

1. **createMC.c**

Создадим файл, который содержит описание функции для создания нового элемента:

#include <string.h>

#include <stdlib.h>

#include "struct.h" //Подключаем файл с объявлением структуры

MusicalComposition\* createMusicalComposition(char\* name, char\* author, int year)

{

MusicalComposition\* tmp=NULL; //Создаем указатель на структуру.

tmp =(MusicalComposition\*)malloc(sizeof(MusicalComposition)); /\*Выделяем под структуру память.\*/

strcpy(tmp->name, name); //Скопируем переданные в функцию

strcpy(tmp->author, author); //значения в поля созданной структуры

tmp->year = year; return tmp; //Возвращаем указатель на структуру

}

1. **createMCList.c**

Создадим файл с описанием функции для создания связанного списка. Содержание данного файла в Приложении 1.

1. **push.c**

Создадим исходник с описанием функции, которая добавляет один элемент в конец списка:

#include <stdlib.h>

#include "struct.h" //Подключаем файл с объявлением структуры

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

{

MusicalComposition\* tmp = head; //Сохраняем указатель на "head".

while(head->next) //Пока указатель на следующий элемент

{ //не будет равен нулю будем передвигать

head = head->next; //указатель на следующий.

} //После цикла "head" указывает на последний элемент

head->next = element; //Вместо указателя на "NULL" присваиваем ему

element->prev = head; //указатель на "element". Предыдущий указатель

element->next = NULL; // пермещаем на"head",а указатель на следующий на

head = tmp; //"NULL". Возвращаем сохраненное значение.

}

1. **removeEl.c**

Создадим файл с описанием функции, которая удаляет элемент с определенным названием композиции. (см. Приложение 2).

1. **count.c**

Создадим файл с функцией, подсчитывающей количество элементов в списке:

#include "struct.h"

int count(MusicalComposition\* head)

{ //Создаем указатель и присваиваем

MusicalComposition\* hold = head; //ему значение передаваемой структ.

int count=0; //Создаем счетчик со значением 0.

while(head) //Цикл "while" будет выполнятся пока

{ //указатель не примет значение "NULL".

count++; //При каждом выполнение цикла увеличиваем

head = head->next; //счетчик на единицу.

}

head = hold; //Присваиваем "head" исходное значение

return count; //Возвращаем счетчик

}

1. **print\_names.c**

Создадим исходник с описанием функции, которая выводит композиции из списка:

#include <stdio.h>

#include "struct.h" //Подключаем заголовочный файл с объявлением структуры

void print\_names(MusicalComposition\* head)

{

MusicalComposition\* hold = head; //Сохраняем значение "head"

while(head) //Пока указатель не равен "NULL"

{ //выводим текущее значение "name"

printf("%s\n", head->name);

head = head->next; //Делаем текущим указатель на след.

} //структуру

head = hold; //Возвращаем исходное значение указателю

};

1. **func.h**

Создадим заголовочный файл, в который поместим прототипы функций:

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

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

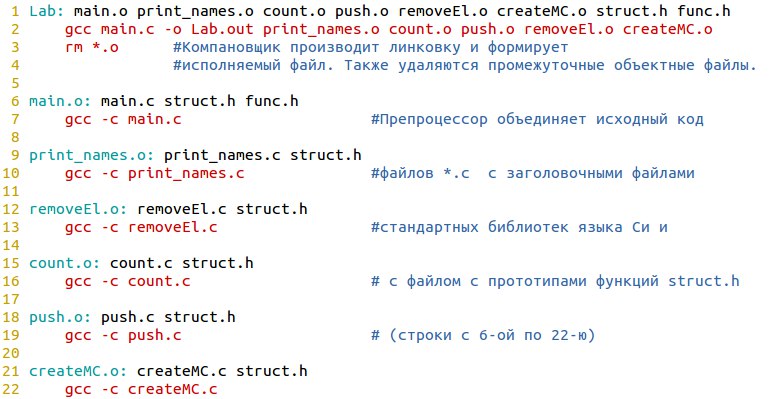
void push(MusicalComposition\*, MusicalComposition\*);

void removeEl(MusicalComposition\*\*, char\*);

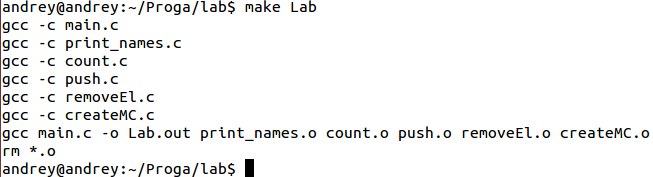
int count(MusicalComposition\*);

void print\_names(MusicalComposition\*);

1. Создадим makefile



1. Программа собирается без ошибок



**Вывод**

Выполнив данную лабораторную работу, мы закрепили на практике знания о структурах и двунаправленных списках. Также мы научились создавать API для работы со списками.

**Приложение**

**ПРИЛОЖЕНИЕ 1**

**createMCList.c**

#include <stdlib.h>

#include "struct.h" //Подключаем заголовочный файл с объявлением структуры

#include "func.h" //и файл с прототипом функции "createMusicalComposition"

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)

{

MusicalComposition\* hold = NULL; //Создаем указатели на вспомогательные структуры

MusicalComposition\* temp = NULL; // и присваиваем им нулевые указатели

MusicalComposition\* start=NULL;

start = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]); //Создаем первый элемент списка

hold = start; //и присваиваем значение "start" временной переменной, чтобы

int i; //вернуть значение в конце

start->prev = NULL;

for (i = 1; i<=n; i++) /\*Цикл начинается с единицы и выполняется, пока счетчик "i" не будет равен количеству элементов\*/

{

if(i<n){

//При условии: счетчик меньше "n" - присваиваем временной переменной

//"temp" текущее значение. Далее создаем следующий элемент и делаем его текущим.

temp = start;

start->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

start = start->next;

//Теперь элемент "temp" является предыдущим для текущего элемента "start"

start->prev = temp;

} else start->next = NULL; /\*При достижении последнего элемента присваиваем следующему элементу нулевой указатеь\*/

}

start = hold; //Возвращаем "start"у исходное значение, сохраненное раннее.

return start; //Возвращаем указатель на "start"

}

**ПРИЛОЖЕНИЕ 2**

**removeEl.c**

#include <stdlib.h>

#include "struct.h" //Подключаем файл с объявлением структуры

void removeEl(MusicalComposition\*\* TrueHead, char\* name\_for\_remove)

{

MusicalComposition \*head = \*TrueHead; //Создаем переменную и

//сохраняем в ней "head".

while(head) //Цикл выполняется пока указатель

{ //станет равен "NULL".

if (strcmp(head->name, name\_for\_remove)==0) //Сравниваем строки

{

if (head->next == NULL){ //Если элемент последний, то текущим

head->prev->next = NULL; //становится предыдущий элемент, а

free(head); //его указатель на след. становится NULL break;

} else if (head->prev == NULL){ //Если удаляемый элемент первый, то

head->next->prev = NULL; //"TrueHead" присваивается следующий

\*TrueHead = head->next; //элемент списка, т.е.новое начало списка.

free(head);

break;

} else {

//В случае, когда элемент является не крайним, перемещаем указатели данным образом

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

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

free(head);

}

}

head = head->next; /\*Если строки не совпали, делаем текущим указатель на след. структуру \*/

}

}