**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Тимофеев А.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Тимофеев Андрей Александрович | | |
| Группа 6304 | | |
| Тема работы: Линейные списки | | |
| Содержание пояснительной записки:   * Содержание * Введение * Описание функций * Работа с репозиторием * Примеры работы программы * Заключение * Список использованных источников * Приложение: Исходный код программы | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 02.12.2016 | | |
| Дата сдачи реферата: 28.12.2016 | | |
| Дата защиты реферата: 28.12.2016 | | |
| Студент |  | Тимофеев А.А. |
| Преподаватель |  | Берленко Т.А. |

# **Аннотация**

В данной работе был создан проект на языке программирования C, который позволяет работать с набором функций, отвечающих за список музыкальных композиций. Для функционирования списка были созданы и описаны необходимые функции, позволяющие добавлять, удалять, сортировать и выводить элементы списка и их количество на консоль, а также описана структура элемента списка. Помимо этого, была проведена работа над оптимизацией исходного кода программы для ускорения ее быстродействия и оптимального использования памяти и ресурсов. Приведено полное описание исходного кода.

**Содержание**

[Аннотация 3](#_Toc470137644)

[Введение 5](#_Toc470137645)

[Цель работы 5](#_Toc470137646)

[Формулировка задачи 5](#_Toc470137647)

[Индивидуальное задание 6](#_Toc470137648)

[Решение задачи 6](#_Toc470137649)

[1. Создание структуры данных 6](#_Toc470137650)

[2. Функция создания нового экземпляра данного типа 6](#_Toc470137651)

[3. Функция создания списка 7](#_Toc470137652)

[4. Функция считывания значений списка 7](#_Toc470137653)

[5. Функция добавления элемента в конец списка 8](#_Toc470137654)

[6. Функция удаления элемента 9](#_Toc470137656)

[7. Функция подсчета элементов списка 11](#_Toc470137657)

[8. Функция вывода элементов списка 11](#_Toc470137658)

[9. Функция замены букв автора на прописные 11](#_Toc470137659)

[10. Функция добавления 3х элементов в конец списка 13](#_Toc470137661)

[11. Makefile 14](#_Toc470137664)

[Работа с репозиторием 14](#_Toc470137665)

[Заключение 15](#_Toc470137666)

[Список использованных источников 16](#_Toc470137667)

[Приложение 17](#_Toc470137668)

[Исходный код программы 17](#_Toc470137669)

# **Введение**

## Цель работы

Практика применения сложных типов (struct) в языке С. Использование их для реализации сложных структур данных. В частности, одно- и дву-направленных линейных списков. Создание API для работы с ними. Закрепление знаний об указателях (в том числе на сложные типы), динамической памяти, массивах, стандартном вводе-выводе и основных функций библиотек “stdio.h”, “stdlib.h”, “string.h”.

**Формулировка задачи**

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (application***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

## Индивидуальное задание

1. Сделать все буквы в имени автора прописными.
2. Добавить в конец списка еще 3 элемента (имя, автор, год - произвольные).

**Решение задачи**

## Создание структуры данных

С помощью типа *struct* объявляем новый сложный тип *struct MusicalComposition*, экземплярами которого и будем заполнять список.

/\* Структура "Музыкальная композиция \*/

struct MusicalComposition {

char name[80]; // Название композиции

char author[80]; // Автор композиции

int year; // Год написания композиции

struct MusicalComposition\* next; // Указатель на следующий элемент

struct MusicalComposition\* prev; // Указатель на предыдущий элемент

};

/\*Объявление типа данных MusicalComposition\*/

typedef struct MusicalComposition MusicalComposition;

## Функция создания нового экземпляра данного типа

Выделяем память для нового элемента и проинициализируем поля переданными в функцию параметрами с помощью указателя на элемент.

/\* Функция создания и инициализации нового экземпляра структуры "MusicalComposition" \*/

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

{

/\* Выделим память для нового экземпляра структуры "MusicalComposition" \*/

MusicalComposition\* rr = (MusicalComposition\*)malloc(sizeof(MusicalComposition));

/\* Проинициализируем поля параметрами, переданными в функцию \*/

strcpy(new\_song->name, name);

strcpy(new\_song ->author, author);

new\_song ->year = year;

new\_song ->next = NULL;

new\_song ->prev = NULL;

/\* Вернем указатель на новый экземпляр структуры "MusicalComposition" \*/

return new\_song;

}

## Функция создания списка

Создаем сначала один элемент и инициализируем его поля, а затем с помощью еще двух указателей в цикле создаем остальные элементы и связываем их. Возвращаем указатель на первый элемент в списке.

/\* Функция создания списка \*/

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

{

/\* Указатель на первый элемент, который вернет функция \*/

MusicalComposition \*head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);

/\* Указатели для движения по списку \*/

MusicalComposition \*prev = head;

MusicalComposition \*current;

int i = 1;

/\* Заполняем список данными, переданными массивами array\_names, array\_authors, array\_years \*/

for (i = 1; i < n; ++i)

{

current = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

/\* Двигаем указатели \*/

current->prev = prev;

prev->next = current;

prev = current;

}

/\* Возвращаем указатель на первый элемент списка - head \*/

return head;

}

## Функция считывания значений списка( находится в функции main)

Первоначально выполняем цикл пока пользователь не введет подходящее значение количества элементов - целое неотрицательное число (length). Получив значение переменной length, начинаем последовательно считывать названия композиций (name), авторов композиций(author) и года (&years[i]) и записываем их в массивы (names,authors,years) соответственно. Передаем функции createMusicalCompositionList() данные массивы для создания линейного списка.

|  |
| --- |
| Int main() |
|  | { |
|  |  |
|  | int length; |
|  |  |
|  | printf("Enter the length of the Musical list\n"); |
|  | //Вводим количество элементов списка |
|  | scanf("%d", &length); |
|  |  |
|  |  |
|  |  |
|  | int tmp; |
|  | // очищаем панель ввода |
|  | while((tmp = getchar()) != '\n'); |
|  |  |
|  |  |
|  | // освобождаем память под название композиции, имя автора и год издания |
|  | char \*\*names = (char\*\*)malloc(sizeof(char\*)\*length); |
|  |  |
|  | char \*\*authors = (char\*\*)malloc(sizeof(char\*)\*length); |
|  |  |
|  | int \*years = (int\*)malloc(sizeof(int)\*length); |
|  | // вводим название композиции, имя автора и год |
|  | printf("Enter the composition's name, author and year of origin:\n"); |
|  |  |
|  | for (int i = 0; i < length; ++i) |
|  |  |
|  | { |
|  | //создаем массивы символов, которые будут принимать вводимые значения |
|  | char name[80]; |
|  |  |
|  | char author[80]; |
|  |  |
|  | fgets(name, 80, stdin); |
|  |  |
|  | fgets(author, 80, stdin); |
|  |  |
|  | fscanf(stdin, "%d", &years[i]); |
|  | //очищаем ввод |
|  | while((tmp = getchar()) != '\n'); |
|  |  |
|  | printf("Enter the composition's name, author and year of origin:\n"); |
|  | //удаляем \n в строке и присваиваем этой ячейке ноль |
|  | (\*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)); |
|  |  |
|  |  |
|  | // копируем из строки, которая принимала, в строку, которая будет передавать |
|  | strcpy(names[i], name); |
|  |  |
|  | strcpy(authors[i], author); |
|  |  |
|  | } |

## Функция добавления элемента в конец списка

Функция принимает в качестве аргумента указатель на новый элемент (head). Считываем название композиции, автора и год написания, которые мы хотим добавить в список, и заполняем структуру (MusicalComposition \*element) полученными переменными. Циклом сместим указатель на последний элемент списка и добавим (MusicalComposition \*element) в список, используя указатели и связав его с последним элементом списка.

/\* Функция добавления элемента в конец списка\*/

void push(MusicalComposition\* head)

{

char name\_for\_push[80];

char author\_for\_push[80];

int year\_for\_push;

/\* Считываем имя, автора и год на добавление\*/

printf("Enter name, author and year (using spaces):\n");

scanf("%s %s %d", name\_for\_push, author\_for\_push, &year\_for\_push);

/\* Выделяем память под новый элемент\*/

MusicalComposition \*element = createMusicalComposition(name\_for\_push, author\_for\_push, year\_for\_push);

MusicalComposition \*current = head;

/\* Передвигаем указатель в конец списка \*/

while (current->next)

current = current->next;

/\* Добавляем элемент в конец списка и устанавливаем связь\*/

current->next = element;

element->prev = current;

element->next = NULL;

}

## Функция удаления элемента

Удаление происходит поиском совпадения между введенным значением имени и именами элементов списка. Если совпадение найдено, то выполняется следующее:

Для того чтобы выполнить удаление элемента, необходимо учитывать его положение в списке. Так как список двулинейный, то возможно 4 варианта.

* 1. Если элемент единственный (перед ним и после него отсутствуют другие элементы), тогда удаляем весь список и закрываем доступ ко все элементам меню (кроме выхода и создания списка) (\*CreatedList = 0;), так как при вызове функций с отсутствующим списком, ответ будет некорректным.
  2. Если элемент первый и после него идут другие элементы, тогда необходимо сместить первоначальное значение списка (head) на следующий элемент.
  3. Если элемент последний и перед ним идут элементы, тогда необходимо сместить указатели с последнего элемента на предпоследнее.
  4. Если элемент находится между другими элементами, тогда необходимо связать элемент, идущий до него с элементом, идущим после него.

/\* Функция удаления элемента \*/

void removeEl(MusicalComposition\* head, char\* name\_for\_remove, int\* CreatedList)

{

/\* Двигаем указатель до конца списка \*/

while (head)

{

/\* Если находим в списке находим в списке удовлетворяющую нас позицию, то выполняем следующее:\*/

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

{

/\*1-ый случай. Если элемент первый и после него идут другие элементы\*/

else if (head->prev == NULL)

{

/\*Смещаем вперед указатель на следующий элемент\*/

\*head = \*head->next;

head->prev = NULL;

return;

}

/\* 2-ой случай. Если элемент последний и перед ним идут другие элементы

else (head->next == NULL)

{

/\*Смещаем назад указатель на предыдущий элемент\*/

head = head->prev;

head->next = NULL;

return;

}

/\*3-ий случай. Если до и после элемента есть другие элементы\*/

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

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

return;

}

head = head->next;

}

}

## Функция подсчета элементов списка

Чтобы подсчитать количество элементов, необходимо использовать счетчик и смещать с помощью цикла указатель до последнего элемента.

/\* Функция подсчета элементов списка \*/

int count(MusicalComposition\*head)

{

MusicalComposition\* current = head;

int count = 0;

/\* Двигаем указатель пока существет список\*/

while (current)

{

count++; //Увеличиваем счетчик - количество элементов

current = current->next;

}

return count;

};

## Функция вывода элементов списка

Функция вывода элементов списка аналогична предыдущей функции, только вместо счетчика выполняется функция вывода названия композиции, автора и года выпуска.

/\* Функция вывода элементов списка\*/

void print\_names(MusicalComposition \*head)

{

MusicalComposition \*current = head;

int i = 0;

/\* Пока существует список выполняем цикл\*/

while (current)

{

/\* Выводим на экран имя, автора и год \*/

printf(" %d. %s %s %d\n", i++, current->name, current->author, current->year);

current = current->next; //Смещаем указатель вперед

}

}

## Функция замены букв автора на прописные

MusicalComposition\* Propis(MusicalComposition \*head)

{

int i=0;

/\* Создаем указатель на первый элемент списка \*/

MusicalComposition \*ptr = head;

/\* Пока элемент не последний \*/

while (ptr!= NULL)

/\* Меняем буквы на прописные \*/

{ for (i = 0; i < strlen(ptr->author); i++) {

if (islower(ptr->author[i]))

ptr->author[i]=toupper(ptr->author[i]);

}

/\* Переходим к следующему элементу \*/

ptr=ptr->next;

}

/\* Возвращаем указатель на первый элемент списка \*/

return head;

}

## Функция добавления 3х элементов в конец списка

void dobavlenie(struct MusicalComposition\* head)

{

int i=0;

/\* Идем в конец списка \*/

while(head->next)

head = head->next;

/\* Три раза добавляем элемент в конец \*/

for (int i = 0; i < 3; i++)

push(head, createMusicalComposition("Default", "Default", i));

}

## Makefile

Создадим поочерёдно объектные файлы функций и главной функции, после скомпилируем их вместе, также предусмотрена функция очистки после сборки проекта.

#Главная цель

main: idz.o kurs.o menu.o

gcc menu.o -o main idz.o kurs.o

rm \*.o

#Объектный файл для функции main

menu.o: menu.c idz.h kurs.h

gcc -c menu.c

#Объектный файл для функций, сделанных к 4-ой лабораторной

|  |
| --- |
|  |

kurs.o: kurs.c kurs.h

gcc -c -std=c99 kurs.c

#Объектный файл для функций для курсовой

idz.o: idz.c idz.h kurs.h

gcc -c idz.c

# **Работа с репозиторием**

1. Скопируем репозиторий группы 6304

*Git clone https://github.com/moevm/pr1-2016-6304.git*

1. От ветки master создаем локальную ветку с названием работы и именем:

*Git checkout –b “Timofeev\_kursework”*

1. В папку с клонированным репозиторием скопируем файлы с работой с помощью команды *cp –r*
2. Добавляем папку с файлом на ветку командой

*git add Kurswork*

1. Коммитим изменения:

*Git commit –m “Залил курсовую работу”*

1. Отправляем изменения в репозиторий командой

*Git push origin*

1. На сайте github создаем *pull request* в ветку *master.*

**примеры работы программы:**

|  |  |  |
| --- | --- | --- |
| № действия | ВВОд | вЫВОД |
| 1 | 4  Magic  Roy\_Woods  2016  Ain’t\_it\_funny  Danny\_Brown  2015  Not\_the\_one  Lil\_PEEP  2016  Rose\_Golden  Kid\_Cudi  2016 | Заносим в список 4 музыкальные композиции |
| 2 | 1  Blue\_Jay\_Session  Tory\_Lanez  2016 | Добавляем одну композицию в конец списка |
| 3 | 2  2  ULT Denzel\_Curry 2016  Kybalion GHOSTMANE  2016 | Добавляем 2 музыкальные композиции в конец списка |
| 4 | 3  Kybalion | Удаляем композицию |
| 5 | 4 | Выводим список композиций:  Magic  Ain’t\_it\_funny  Not\_the\_one  Rose\_Golden  Blue\_Jay\_Session  ULT |
| 6 | 5 | Выводим количество композиций:  6 |
| 7 | 6 | Заменяем буквы автора на прописные |
| 8 | 7 | Добавляем 3 произвольных элемента в конец списка |

# 

# **Заключение**

Выполнив данную курсовую работу, мы освоили и закрепили на практике написание программы с применением структур в качестве элементов сложных типов представления данных таких, как одно- и дву-направленные линейные списки, а также API для работы с ними, в частности, функции вставки, удаления, подсчета и вывода элементов списка. Повторили работу с проектом в целом: разбиение его на заголовки (header files) и файлы кода (source files), а также написание makefile для сборки программы, так же темы: указатели, динамическая память, массивы и функции стандартных библиотек.

# **Список использованных источников**

* 1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб: Издательство «Невский Диалект», 2001. 352 с.
  2. UNIX. Программное окружение / Керниган Б., Пайк Р. СПб: Символ Плюс, 2003. 416с.

# **Приложение**

# **Исходный код программы**

* 1. *Menu.c*

|  |
| --- |
| #include <stdlib.h> |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | #include <ctype.h> |
|  | #include "kurs.h" |
|  | #include "idz.h" |
|  | int main(){ |
|  | int length; int a = 1,year; |
|  | printf("Количество композиций в списке: "); |
|  | scanf("%d", &length); |
|  | char\*\* names = (char\*\*)malloc(sizeof(char\*)\*length); |
|  | char\*\* authors = (char\*\*)malloc(sizeof(char\*)\*length); |
|  | int\* years = (int\*)malloc(sizeof(int)\*length); |
|  | int i,t; |
|  | char name[80]; |
|  | char author[80]; |
|  | for (i=0;i<length;i++) |
|  | { |
|  | printf("%d) Введите название композиции, ее автора и год ее создания : ", i+1); |
|  | scanf("%s %s %d", name, author, &year); |
|  | printf("-------------------------------------------------\n"); |
|  | 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); |
|  | while (a!=0){ |
|  |  |
|  | printf("Выберите действие: \n"); |
|  | printf("0. Выйти\n1. Добавить одну композицию\n2. Добавить несколько композиций \n3. Удалить композицию\n4. Список композиций\n5. Количество композиций\n6. Перезаписать автора композиции с прописными буквами\n7. Добавить 3 композиции в конец списка\n -----------:"); |
|  | scanf("%d", &a); |
|  | printf("-------------------------------------------------\n"); |
|  | switch (a){ |
|  | case 0: break; |
|  |  |
|  | case 1: printf("Введите название композиции, ее автора и год ее создания: "); //просим ввести название, автора и год написания композиции |
|  | scanf("%s %s %d", name, author, &year);//записываем значения |
|  | push(head, createMusicalComposition(name, author, year));//обращаемся к функции push |
|  | break; |
|  |  |
|  | case 2: printf("Сколько композиций вы хотите добавить?\n"); |
|  | scanf("%d", &t);//записываем количество композиций для добавления |
|  | for(i=0;i<t;i++){ |
|  | printf("Введите название композиции, ее автора и год ее создания: ");//просим ввести название, автора и год написания композиции |
|  | scanf("%s %s %d", name, author, &year);//записываем значения |
|  | push(head, createMusicalComposition(name, author, year));//обращаемся к функции push |
|  | } |
|  | break; |
|  |  |
|  | case 3: printf("Введите название композиции: "); |
|  | scanf("%s", name); |
|  | removeEl(head, name); |
|  | break; |
|  |  |
|  | case 4: print\_names(head); |
|  | break; |
|  |  |
|  | case 5: printf("Количество композиций в списке %d\n", count(head)); |
|  | break; |
|  |  |
|  | case 6: Propis(head); |
|  | break; |
|  | case 7: dobavlenie(head); |
|  | break; |
|  |  |
|  | default: printf("Дейсвие не существует"); break; |
|  | } |
|  | printf("-------------------------------------------------\n"); |
|  | } |
|  | return 0; |
|  |  |
|  | } |

* 1. *Kurs.c*

|  |
| --- |
| #include <stdlib.h> |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  |  |
|  |  |
|  | struct MusicalComposition { |
|  | char name[80]; |
|  | char author[80]; |
|  | int year; |
|  | struct MusicalComposition\* next; |
|  | struct MusicalComposition\* prev; |
|  | }; |
|  |  |
|  | typedef struct MusicalComposition MusicalComposition; |
|  |  |
|  | MusicalComposition\* createMusicalComposition(char \*name, char\* author, int year) |
|  | { |
|  | MusicalComposition\* new\_song = (MusicalComposition\*)malloc(sizeof(MusicalComposition)); |
|  | strcpy(new\_song->name, name); |
|  | strcpy(new\_song->author, author); |
|  | new\_song->year = year; |
|  | new\_song->next = NULL; |
|  | new\_song->prev = NULL; |
|  | return new\_song; |
|  | } |
|  |  |
|  | 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 \*prev = head; |
|  | MusicalComposition \*current; |
|  | for (int i = 1; i < n; ++i) |
|  | { |
|  | current = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]); |
|  | current->prev = prev; |
|  | prev->next = current; |
|  | prev = current; |
|  | } |
|  | return head; |
|  | } |
|  |  |
|  | void push(MusicalComposition\* head) |
|  | { |
|  | char name\_for\_push[80]; |
|  | char author\_for\_push[80]; |
|  | int year\_for\_push; |
|  |  |
|  | printf("Enter name, author and year (using spaces):\n"); |
|  | scanf("%s %s %d", name\_for\_push, author\_for\_push, &year\_for\_push); |
|  |  |
|  |  |
|  | MusicalComposition \*element = createMusicalComposition(name\_for\_push, author\_for\_push, year\_for\_push); |
|  | MusicalComposition \*current = head; |
|  |  |
|  |  |
|  | while (current->next) |
|  | current = current->next; |
|  |  |
|  |  |
|  | current->next = element; |
|  | element->prev = current; |
|  | element->next = NULL; |
|  | } |
|  |  |
|  | void removeEl(MusicalComposition\*head, char\*name\_for\_remove) |
|  | { |
|  |  |
|  | while (head) |
|  | { |
|  | if (strcmp(head->name, name\_for\_remove) == 0) |
|  | { |
|  | if (head->next == NULL) |
|  | { |
|  | head->prev->next = NULL; |
|  | return; |
|  | } |
|  | else if (head->prev == NULL) |
|  | { |
|  | head->next->prev = NULL; |
|  | return; |
|  | } |
|  |  |
|  | head->prev->next = head->next; |
|  | head->next->prev = head->prev; |
|  |  |
|  | break; |
|  | } |
|  |  |
|  | head = head->next; |
|  | } |
|  |  |
|  | } |
|  | int count(MusicalComposition\*head) |
|  | { |
|  | int count = 0; |
|  | while (head) |
|  | { |
|  | count++; |
|  | head = head->next; |
|  | } |
|  | return count; |
|  | }; |
|  |  |
|  | void print\_names(MusicalComposition \*head) |
|  | { |
|  | while (head) |
|  | { |
|  | printf("%s\n", head->name); |
|  | head = head->next; |
|  | } |
|  | } |

* 1. *Kurs.h*

|  |
| --- |
| typedef struct MusicalComposition MusicalComposition; |
|  | MusicalComposition\* createMusicalComposition(char \*name, char\* author, int year); |
|  | 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); |

* 1. *Idz.c*

|  |
| --- |
| #include <stdlib.h> |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  | #include <ctype.h> |
|  | #include "kurs.h" |
|  | struct MusicalComposition { |
|  | char name[80]; |
|  | char author[80]; |
|  | int year; |
|  | struct MusicalComposition\* next; |
|  | struct MusicalComposition\* prev; |
|  | }; |
|  |  |
|  |  |
|  | typedef struct MusicalComposition MusicalComposition; |
|  |  |
|  | MusicalComposition\* Propis(MusicalComposition \*head) { |
|  | int i=0; |
|  | MusicalComposition \*ptr = head; |
|  | while (ptr!= NULL) |
|  | { for (i = 0; i < strlen(ptr->author); i++) { |
|  | if (islower(ptr->author[i])) |
|  | ptr->author[i]=toupper(ptr->author[i]); |
|  | } |
|  | ptr=ptr->next; |
|  |  |
|  | } |
|  | return head; |
|  | } |
|  |  |
|  | void dobavlenie(struct MusicalComposition\* head) |
|  | { |
|  |  |
|  | int i=0; |
|  | while(head->next) |
|  | head = head->next; |
|  | for ( i = 0; i<3; i++) |
|  | push(head, createMusicalComposition("Default","Default", i)); |
|  | } |

* 1. *Idz.h*

|  |
| --- |
| Struct MusicalComposition\* Propis(MusicalComposition \*head); |
| void dobavlenie(struct MusicalComposition \*head); |  |
|  |  |

* 1. *Makefile*

|  |  |
| --- | --- |
|  | main: idz.o kurs.o menu.o |
|  | gcc menu.o -o main idz.okurs.o |
|  | rm \*.o |
|  |  |
|  | menu.o: menu.c idz.h kurs.h |
|  | gcc -c menu.c |
|  |  |
|  | kurs.o: kurs.c kurs.h |
|  | gcc -c -std=c99 kurs.c |
|  |  |
|  | idz.o: idz.c idz.h kurs.h |
|  | gcc -c idz.c |