**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Программной инженерии**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Двунаправленные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Корытов П.В. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| **Студент** Корытов Павел Валерьевич | | |
| **Группа** 6304 | | |
| **Тема работы (проекта):** Двунаправленные списки | | |
| **Исходные данные:**  Поставлена задача создать двунаправленный список и обеспечить возможность его заполнения, добавления элементов, удаления и сортировки. | | |
| **Содержание пояснительной записки:**   * Содержание * Введение * Теоретические обоснования двунаправленных списков * Описание необходимых для работы со списком функций. * Описание работы с git и github * Заключение * Код программы | | |
| **Предполагаемый объем пояснительной записки:**  Не менее 17 страниц. | | |
| **Дата выдачи задания:** 08.12.2016 | | |
| **Дата сдачи реферата:** | | |
| **Дата защиты реферата:** | | |
| Студент |  | Корытов П.В. |
| Преподаватель |  | Берленко Т.А. |

**Аннотация**

Задача была создать двунаправленный список и реализовать API и интерфейс пользователя для взаимодействия со списком.

Во время разработки были использованы текстовый редактор vim, GCC, IDE Code::Blocks и Microsoft Word

Результатом было некоторое количество файлов: \*.c, \*h – программный код, исполняемый файл, makefile, а также данный отчёт

**Summary**

The task was to create bidirectional list of structures and realize API an UI for interacting with list.

During the development, vim text editor, GCC, Code::Blocks IDE and Microsoft Word were used.

The results were number of files: \*.c and \*.h format – program code, executable file, makefile, and also this report.

**СОДЕРЖАНИЕ**

[**введение** 5](#_Toc470117734)

[**1. Теория** 6](#_Toc470117735)

[**1.1.** **Двунаправленные списки** 6](#_Toc470117736)

[**1.2.** **Реализация двунаправленного списка структур на С** 6](#_Toc470117737)

[**2.** **Создание ветки** 6](#_Toc470117738)

[**3.** **методы работы** 7](#_Toc470117739)

[**3.1.** **Объявление структуры** 7](#_Toc470117740)

[**3.2.** **Создание списка** 7](#_Toc470117741)

[**3.3.** **Добавление элемента в конец списка** 8](#_Toc470117742)

[**3.4.** **Удаление элемента** 8](#_Toc470117743)

[**3.5.** **Подсчёт элементов и вывод** 9](#_Toc470117744)

[**3.6.** **Сортировка** 9](#_Toc470117745)

[**4.** **MAKEFILE** 11](#_Toc470117746)

[**5.** **ЗАГРУЗКА ФАЙЛОВ НА РЕПОЗИТОРИЙ** 11](#_Toc470117747)

[**приложение А. КОд программы** 14](#_Toc470117748)

[**1.** **Файл main.c** 14](#_Toc470117749)

[**2.** **Файл sort.c** 16](#_Toc470117750)

[**3.** **Файл structure.c** 17](#_Toc470117751)

[**4.** **Файлы прототипов – create.h, sort.h, structure.h** 17](#_Toc470117752)

[**a.** **structure.h** 17](#_Toc470117753)

[**b.** **sort.h.** 17](#_Toc470117754)

[**c.** **create.h** 17](#_Toc470117755)

# **введение**

**Цель работы:** создать двунаправленный список структур MusicalComposition, включающих в себя два текстовых и одно числовое значение. Реализовать возможности:

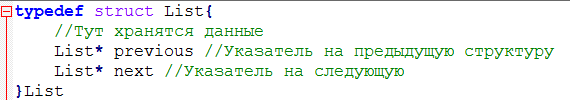
* Создания элемента типа MusicalComposition
* Создание двунаправленного списка структур типа MusicalComposition
* Добавления элементов в список
* Удаления элементов из списка по имени
* Выведения элементов списка на экран
* Сортировка списка по убывания по числовым или текстовым значениям

# **1. Теория**

* 1. **Двунаправленные списки**

Двунаправленный список есть набор структур, связанных между собой указателями. В каждой структуре есть указатель на следующую или предыдущую. У первой структуры указатель предыдущую – NULL, и у последней на следующую – NULL. Доступ к списку можно получить через первый элемент структуры – голову списка.

* 1. **Реализация двунаправленного списка структур на С**

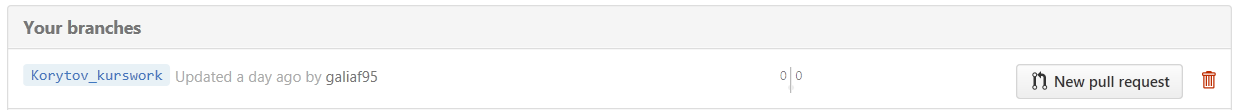


Для доступа к следующим элементам структуры необходимо совершить ряд присваивания, вида head = head->next, где head – голова списка.

1. **Создание ветки**

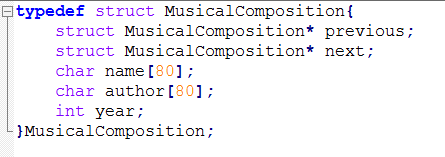
Открыт репозиторий [github.com/moevm/pr1-2016-2304.](file:///C:\Users\thexc\Desktop\github.com\moevm\pr1-2016-2304) На нём с помощью поля “Find or create branch” создана ветка Korytov\_kurswork





# **методы работы**

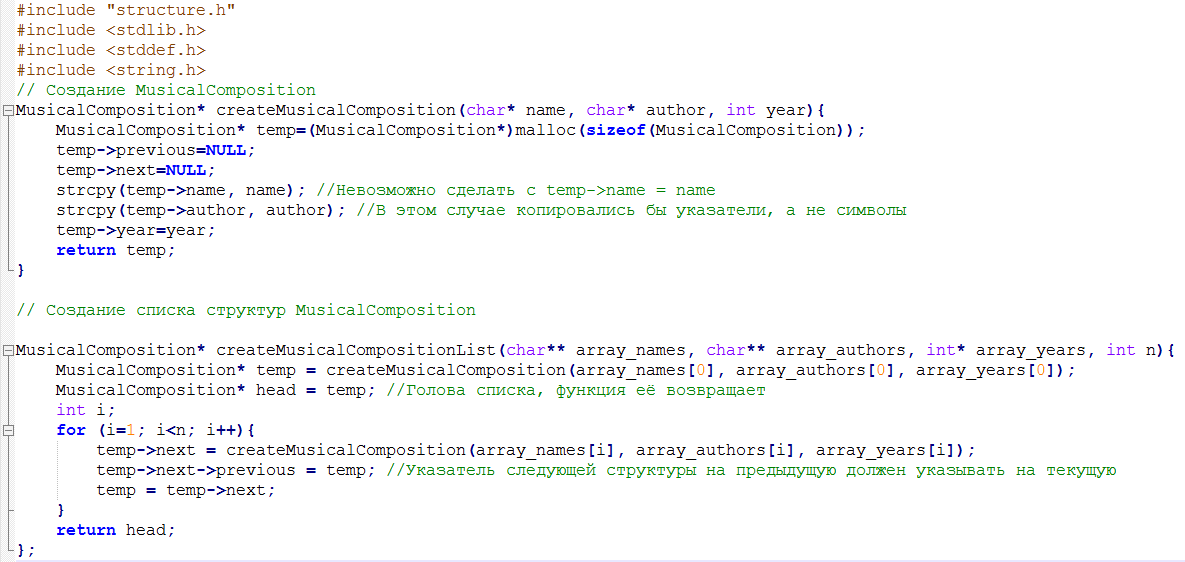
* 1. **Объявление структуры**



Наличие файла structure.h, описывающего структуру и необходимость его включения во все прочие файлы кода обусловлено тем, что gcc не может скомпилировать объектный файл, если какой-то тип не описан в этом конкретном файле. Если описать в каждом, то при соединении файлов будет ошибка – один тип описан несколько раз. Поэтому было принято решение создать отдельный файл для объявления структуры.

В данной структуре первые две переменные – указатели на предыдущий и на следующий элементы списка.

* 1. **Создание списка**

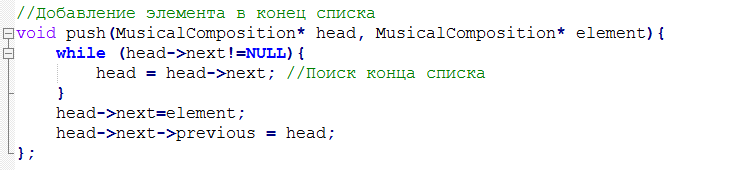


Для начала создаётся отдельный элемент списка из уже данной информации в функции createMusicalComposition. Пока у него указатели равны NULL.

Функция createMusicalCompositionList получает на вход массивы, которые записывает в структуры. Сначала создается голова списка и в неё записываются первые значения массивов, а потом в head->next создается следующая структура, после чего происходит head = head->next. Так продолжается, пока не кончатся массивы.

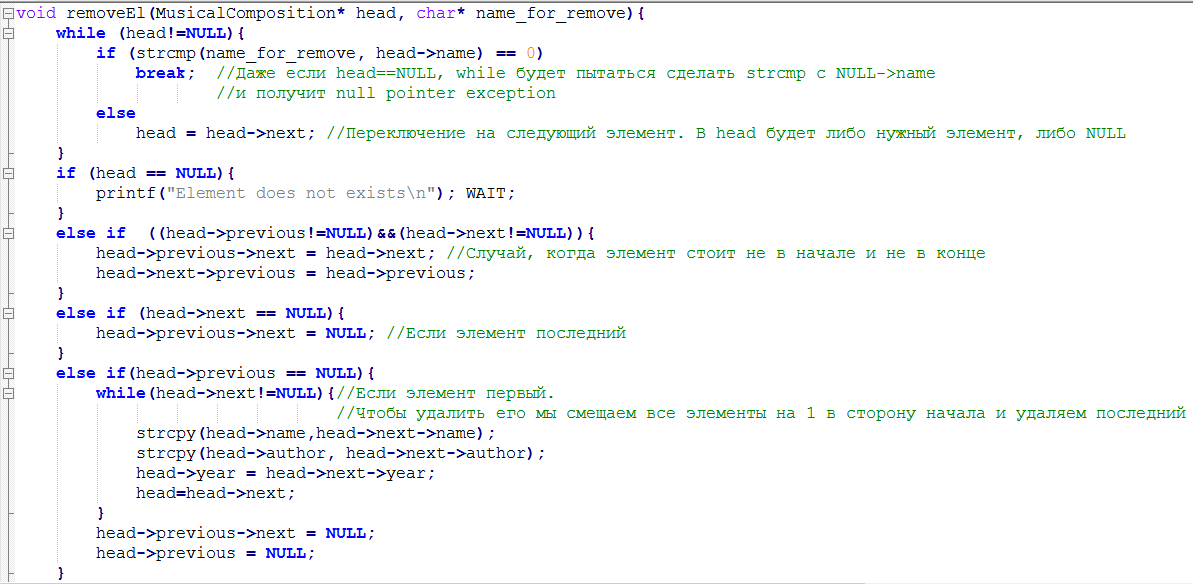
Функции создания списка вынесены в отдельный файл create.c

* 1. **Добавление элемента в конец списка**



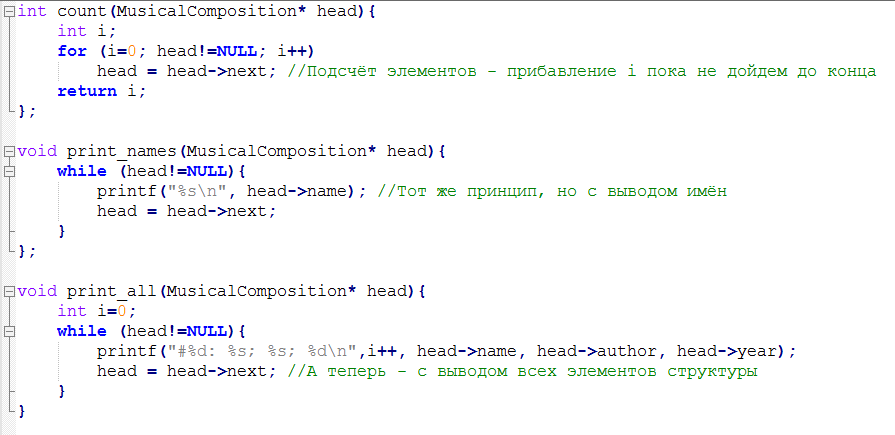
Происходит следующим образом – находится последний элемент списка, там применяется createMusicalComposition к head->next, после чего указателю previous в head->next присваивается указатель на текущий, теперь уже предпоследний элемент

* 1. **Удаление элемента**



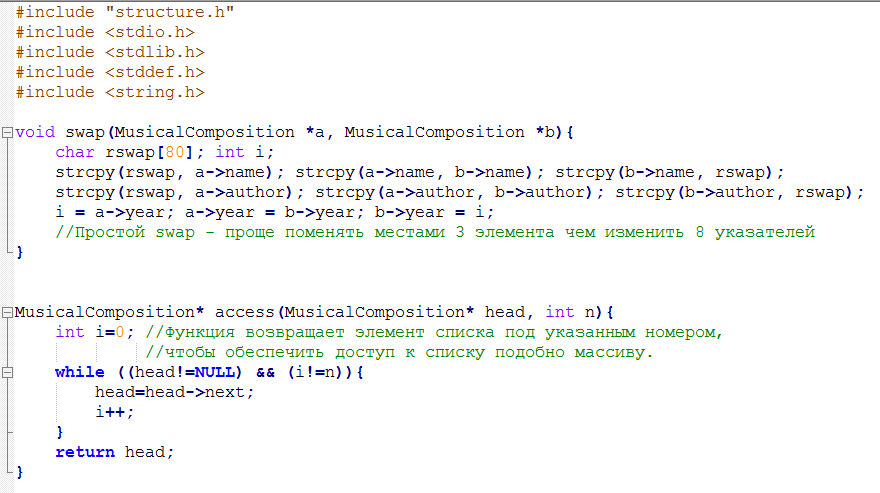
Удаление элемента, если он не стоит в начале списка, не составляет труда. В случае, если элемент не в конце, указатели предыдущего и следующего элементов «перебрасываются» через текущий, а если элемент в конце – указателю next в предпоследнем присваивается NULL. Если же элемент начале, то элементы сдвигаются на 1 в сторону начала, и задача сводится к предыдущей – к удалению последнего элемента.

* 1. **Подсчёт элементов и вывод**



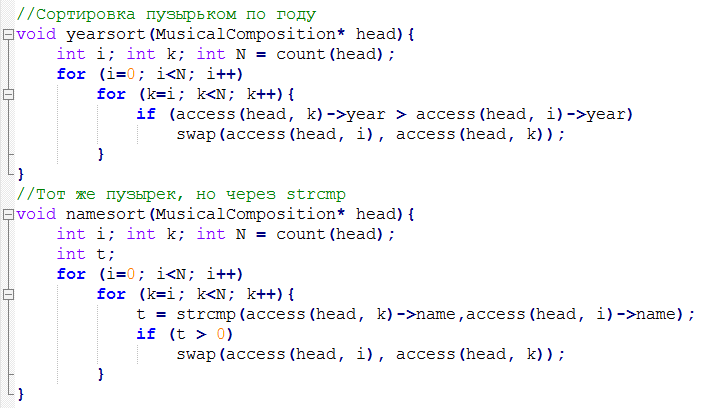
Подсчет элементов происходит прибавлением 1 к переменной и перемещением вперед по списку до достижения конца, т.е. NULL pointer. Если необходимо вывести элементы списка, то вместо прибавления 1 к переменной пишется printf и выводятся необходимые значения.

* 1. **Сортировка**



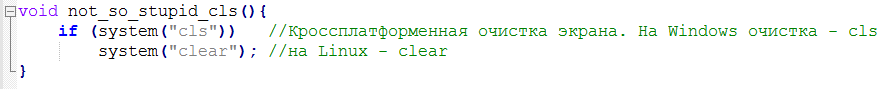
Для сортировки написан ряд вспомогательных процедур. Во – первых, это swap, который меняет местами значения в элементах списка. Меняются значения, так как изменение указателей, число которых доходит до 8, достаточно затруднительно, т.к. приходится разбирать множество исключительных случаев.

Во – вторых, функция access – возвращает элемент под заданным номером. Это сделано для облегчения сортировки списка.



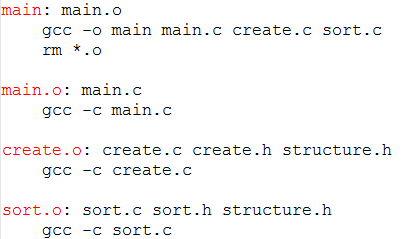
В результате задача по сортировке списка сводится к сортировке массивоподобной сущности, которую можно легко осуществить стандартным пузырьковым способом.

* 1. **Очистка экрана**



Очистка экрана реализована с помощью процедуры system, которая позволяет выполнять любые команды, доступные в терминале. Проблема в том, что в Linux и в Windows команды для очистки экрана разные. Для решения проблемы используется та особенность, что при ошибке выполнения команды system возвращает 1. В этом случае программа будет пытаться выполнить другую команду.

1. **MAKEFILE**

****

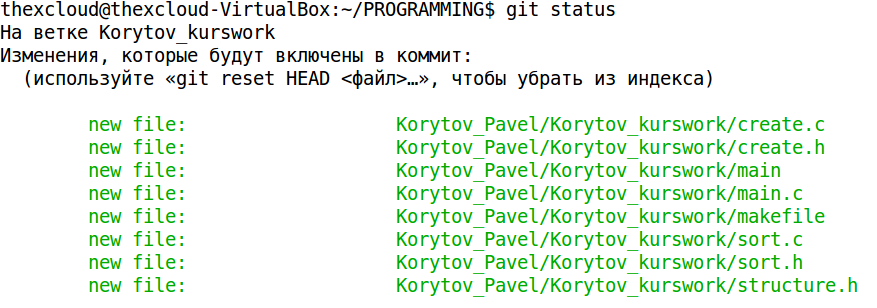
Сначала создаются объектные файлы – main.o, create.o и sort.o. Затем создается уже готовый исполняемый файл main, а объектные файлы удаляются.

1. **ЗАГРУЗКА ФАЙЛОВ НА РЕПОЗИТОРИЙ**

Сначала создана локальная ветка Korytov\_kurswork.



Затем в git добавлены все файлы курсовой работы



После чего сделан коммит и загрузка на github



**заключение**

Цель работы достигнута – создан двунаправленный список структур, реализована возможность добавления и удаления элементов, сделан подсчёт элементов и их сортировка.

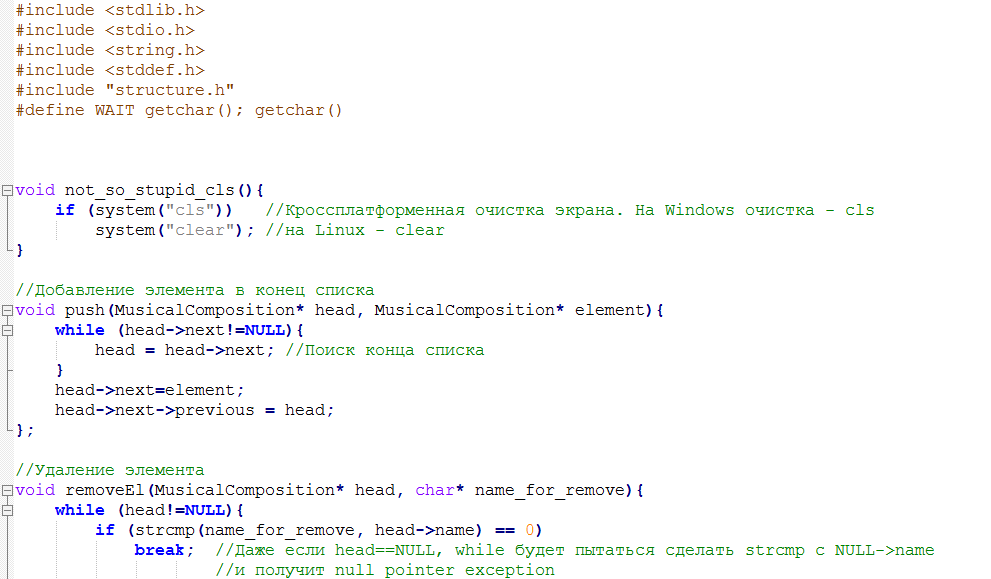
**список использованных источников**

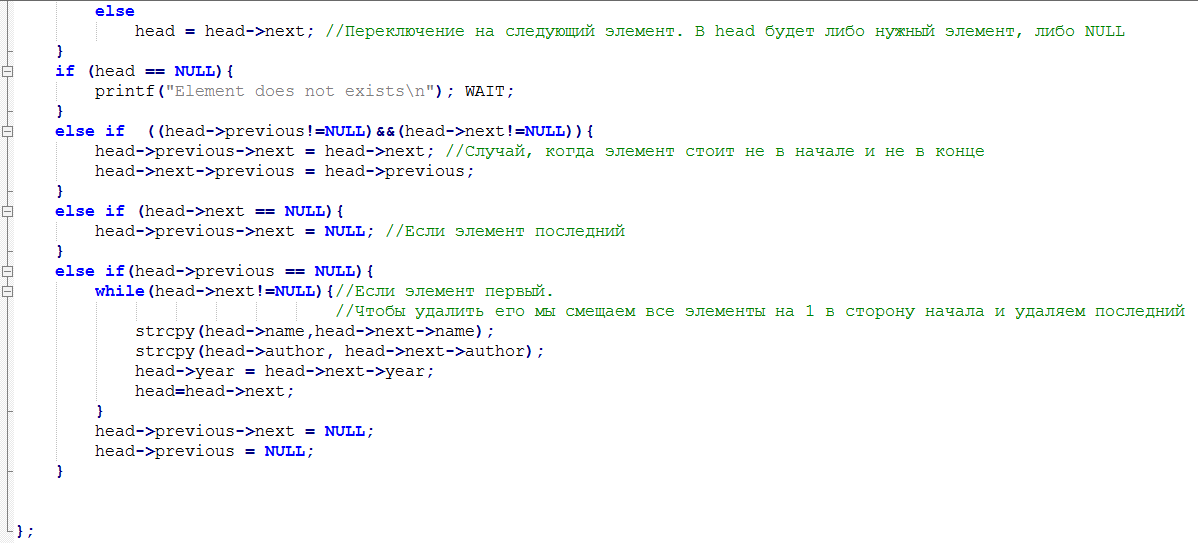
* Команда man в терминале Linux
* Керниган, Ритчи. Язык программирования Си.

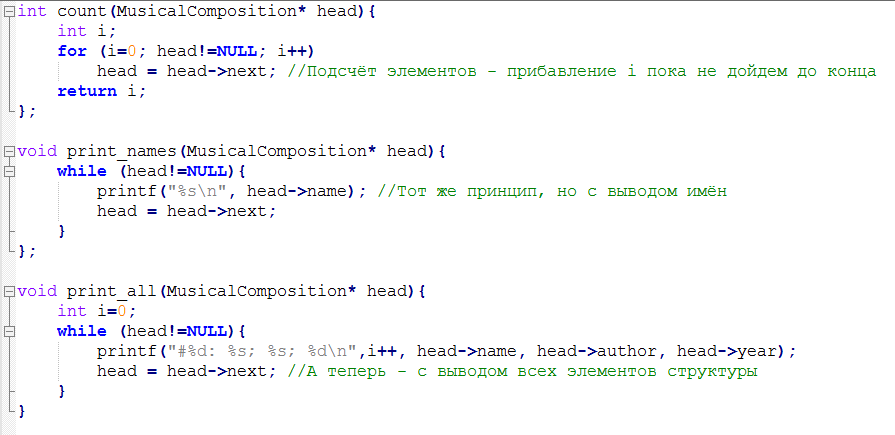
# **приложение А. КОд программы**

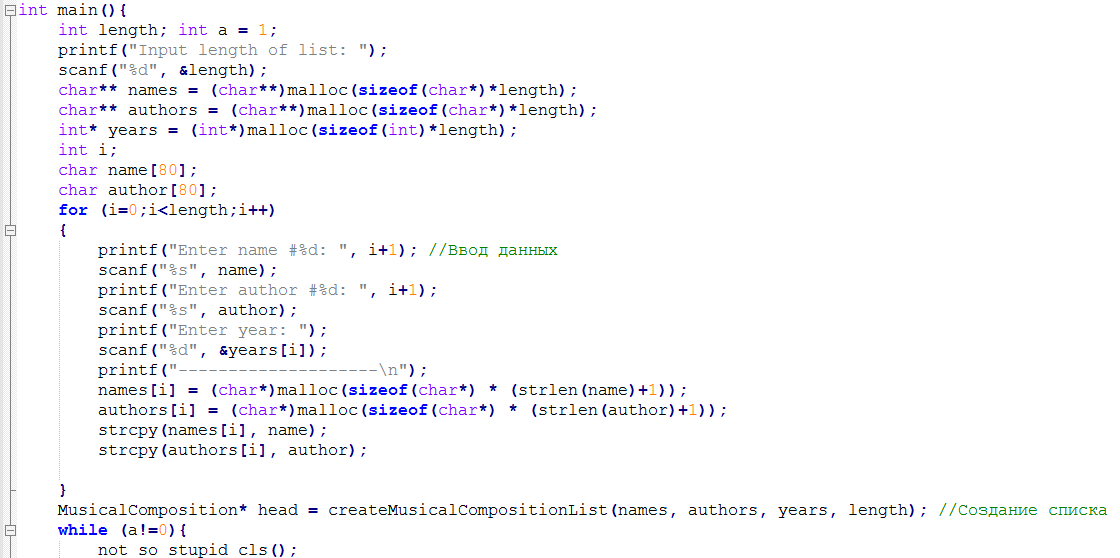
Для сохранения красоты форматирования кода было решено использовать скриншоты вместо копирования текста.

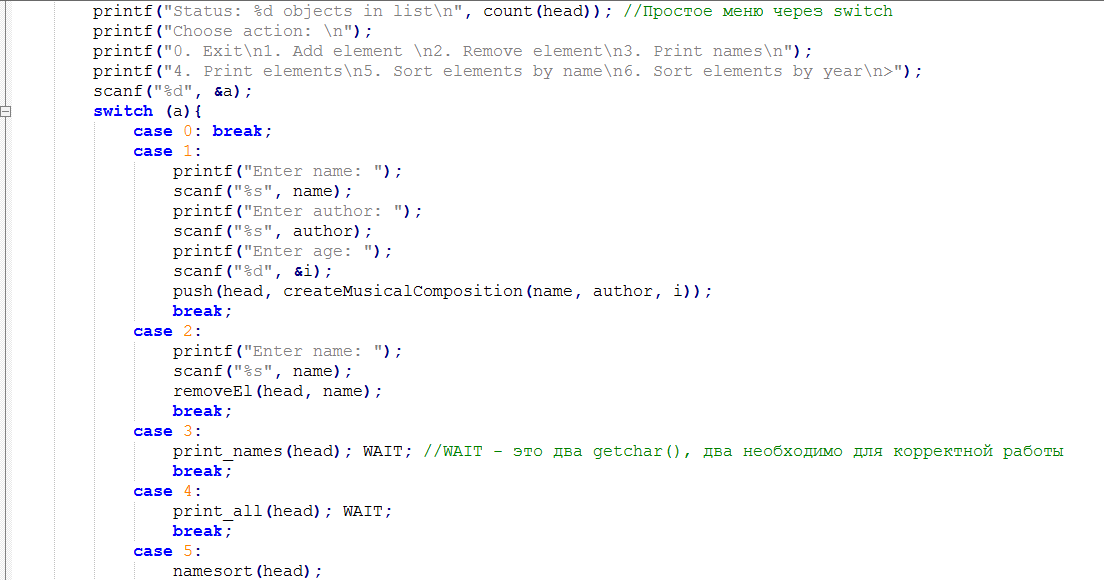
1. **Файл main.c**

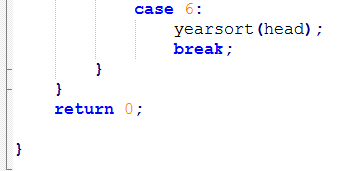


****

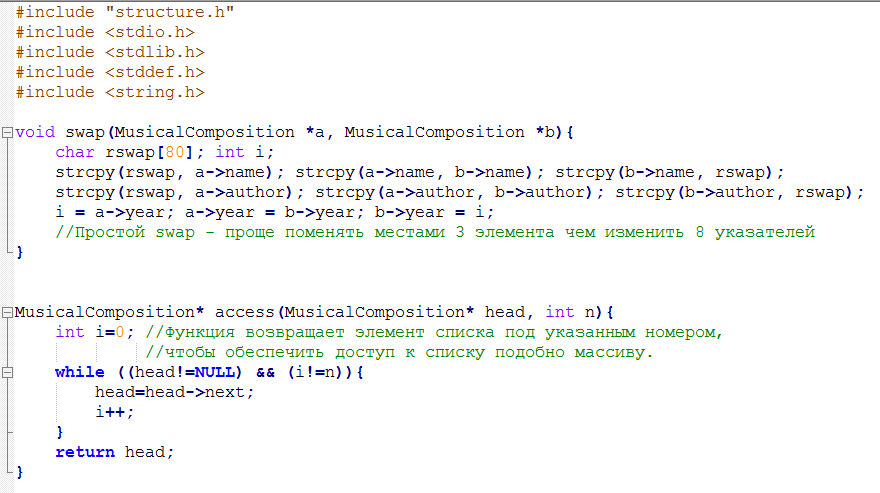
****

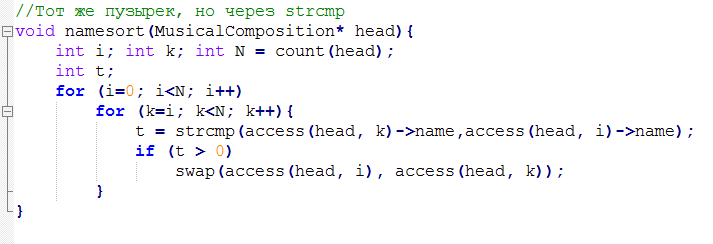
****

****

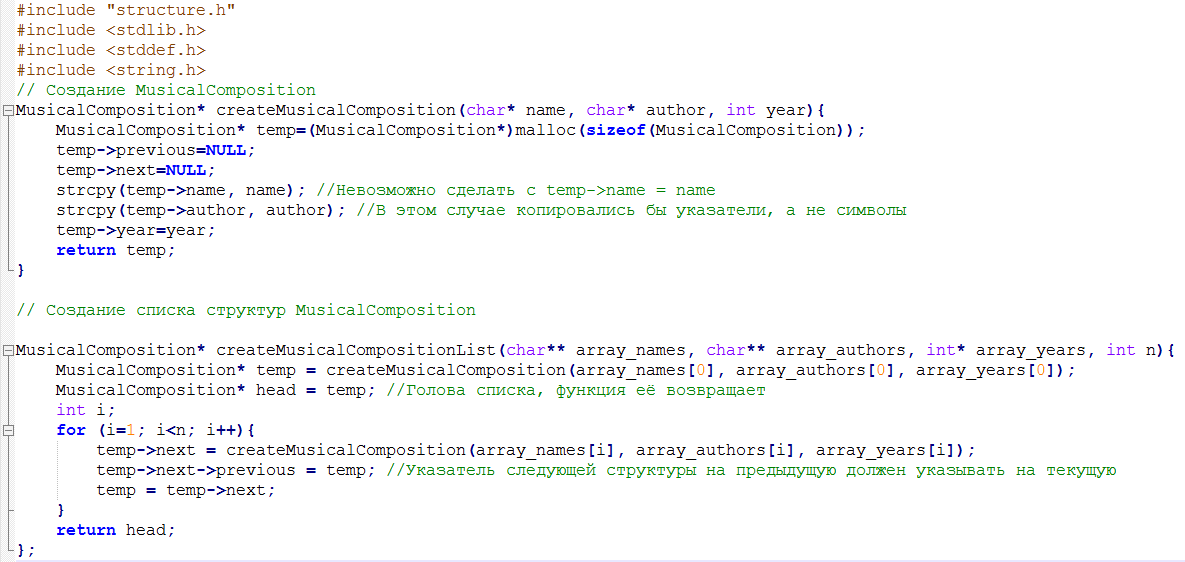
****

1. **Файл sort.c**

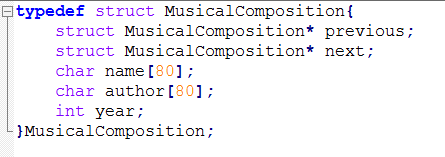
****

****

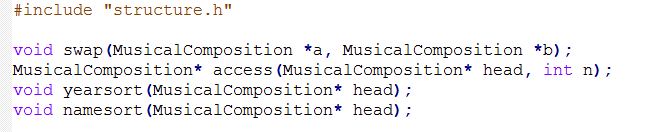
1. **Файл structure.c**

****

1. **Файлы прототипов – create.h, sort.h, structure.h**
   1. **structure.h**

****

* 1. **sort.h.**

****

* 1. **create.h**

****