**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Двунаправленные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Запевалов А.И. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| **Студент** Запевалов Алексей Ильич | | |
| **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

В результате были созданы файлы \*.c, \*h – программный код, исполняемый файл, makefile, отчёт.

**СОДЕРЖАНИЕ**

[введение 5](#_Toc470346083)

[1. Теория 6](#_Toc470346084)

[1.1. Двунаправленные списки 6](#_Toc470346085)

[1.2. Реализация двунаправленного списка структур на С 6](#_Toc470346086)

[2. Создание ветки 6](#_Toc470346087)

[3. методы работы 7](#_Toc470346088)

[3.1. Объявление структуры 7](#_Toc470346089)

[3.2. Создание списка 7](#_Toc470346090)

[3.3. Добавление элемента/элементов в конец списка 8](#_Toc470346091)

[3.4. Удаление элемента 8](#_Toc470346092)

[3.5. Подсчёт элементов и вывод 9](#_Toc470346093)

[3.6 Смена мест половин списка. 9](#_Toc470346094)

[3.7 Смена элементов относительно одного из них. 10](#_Toc470346095)

[4. MAKEFILE 10](#_Toc470346096)

[5. ЗАГРУЗКА ФАЙЛОВ НА РЕПОЗИТОРИЙ 11](#_Toc470346097)

[приложение А. КОд программы 14](#_Toc470346098)

[1. Файл main.c 14](#_Toc470346099)

[2. Файл sort.c 16](#_Toc470346100)

[3. Файл structure.c 17](#_Toc470346101)

[4. Файлы прототипов – create.h, sort.h, structure.h 17](#_Toc470346102)

[a. structure.h 17](#_Toc470346103)

[b. sort.h. 17](#_Toc470346104)

[c. create.h 17](#_Toc470346105)

# **введение**

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

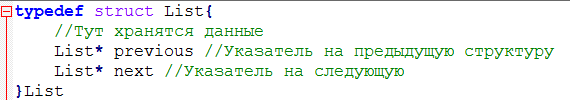
* Создания элемента типа MusicalComposition
* Создание двунаправленного списка структур типа MusicalComposition
* Добавления элементов в список
* Удаления элементов из списка по имени
* Выведения элементов списка на экран
* Сортировка списка по убывания по числовым или текстовым значениям

# **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**

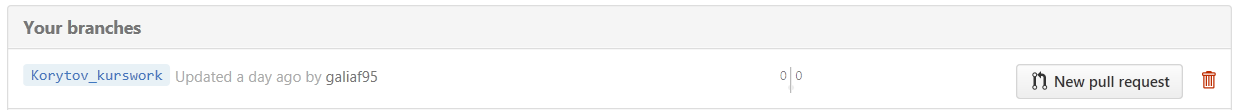


Для доступа к следующим элементам структуры необходимо совершить ряд присваивания, вида head = head->next, где head – голова списка.

1. **Создание ветки**

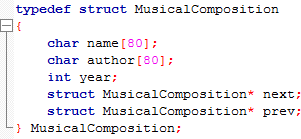
Открыт репозиторий [github.com/moevm/pr1-2016-2304.](file:///C:\Users\thexc\Desktop\github.com\moevm\pr1-2016-2304) На нём с помощью поля “Find or create branch” создана ветка Alexey\_Zapevalov\_kurswork



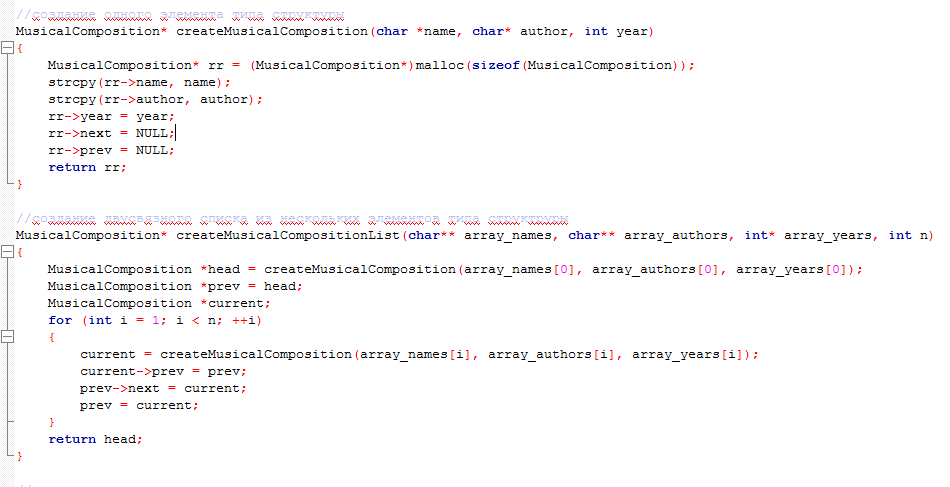


# **методы работы**

* 1. **Объявление структуры**

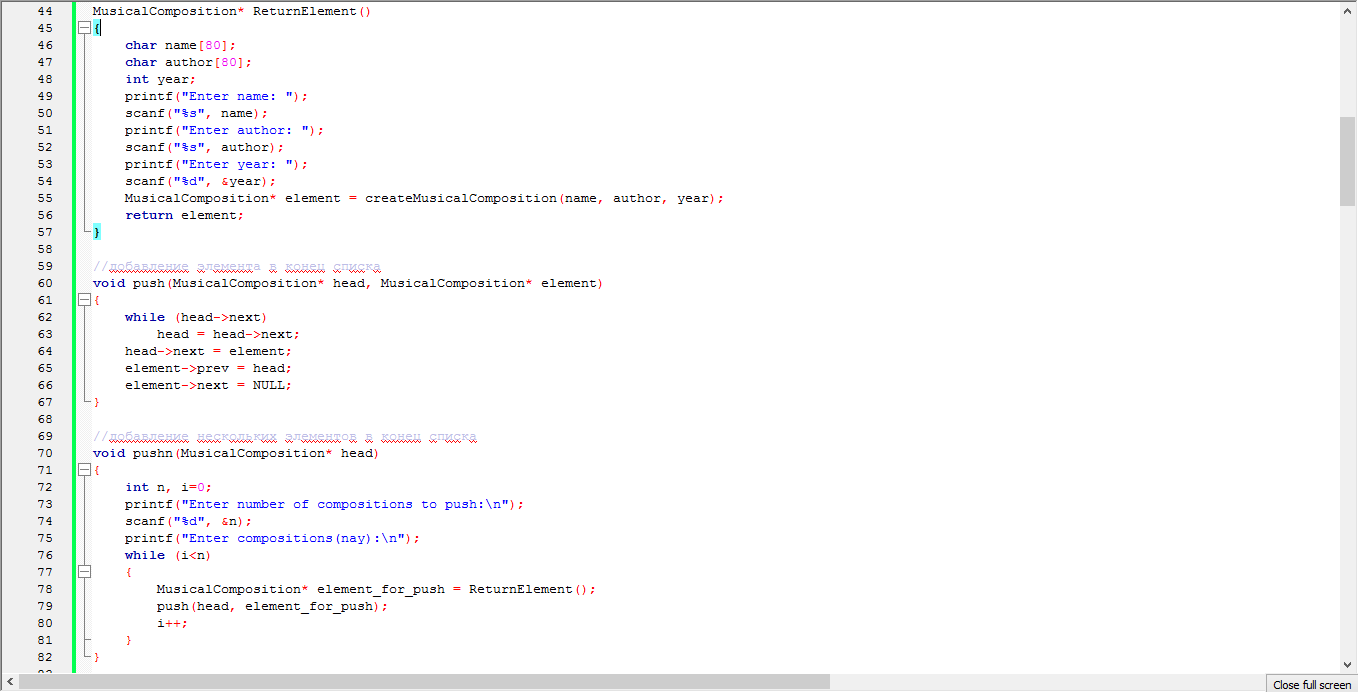


* 1. **Создание списка**



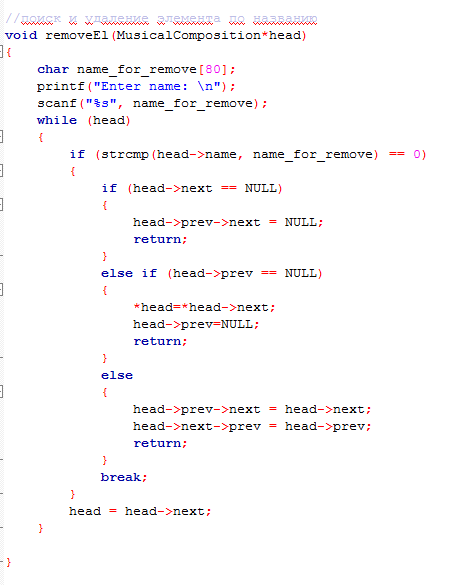
Функция createMusicalCompositionList получает на вход массивы, которые записывает в структуры. Сначала создается голова списка и в неё записываются первые значения массивов, а потом в head->next создается следующая структура, после чего происходит head = head->next. Так продолжается, пока не кончатся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

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

****

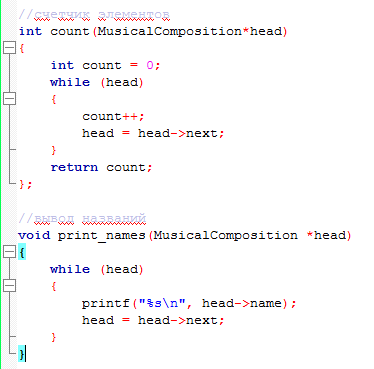
Происходит следующим образом – создается элемент типа структуры в функции ReturnElement, после чего добавляется в конец списка при помощи функции push. В случае с несколькими элементами предыдущие функции вызываются указанное в функции pushn количество раз.

* 1. **Удаление элемента**



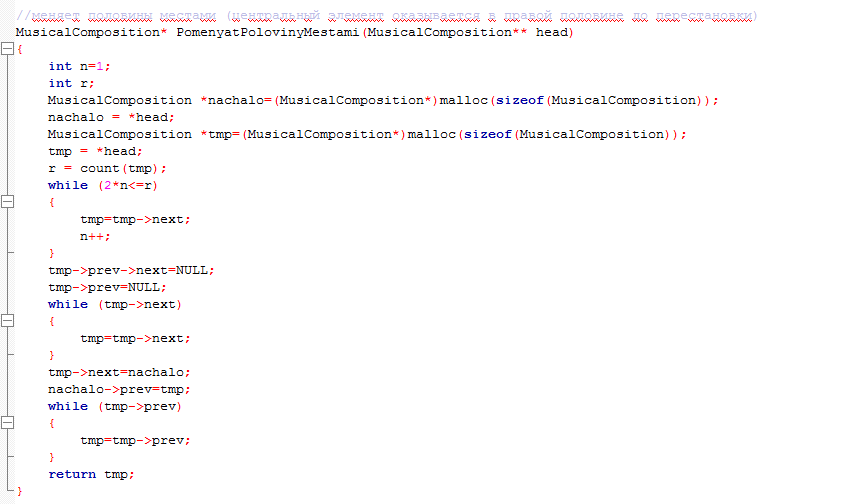
В случае, если элемент не в конце, указатели предыдущего и следующего элементов «перебрасываются» через текущий, а если элемент в конце – указателю next в предпоследнем элементе присваивается NULL. Если же элемент начале, то указатель на головной элемент смещается на следующий элемент, его указателю на предыдущий элемент присваивается NULL.

* 1. **Подсчёт элементов и вывод**



Подсчет элементов происходит прибавлением 1 к переменной и перемещением вперед по списку до достижения конца, т.е. NULL pointer. Вывод на печать названия каждого элемента происходит аналогично.

**3.6 Смена мест половин списка.**



В этой функции сначала указатель tmp устанавливается на элемент, с которого начинается вторая половина. Для этого вычисляется общее количество элементов при помощи функции count. После чего половины отделяются друг от друга и соединяются концом(второй) и началом(первой). В конце указатель tmp возвращается к новому первому элементу и функция его возвращает. Теперь head из функции main соответствует tmp из PomenyatPolovinyMestami.

**3.7 Смена элементов относительно одного из них.**



В этой функции сначала находится элемент с соответствующим номером. После чего список проходится одновременно поэлементно вперед и назад от этого элемента. В каждой итерации меняются местами значения названия, автора и года композиции.

**3.8 Код программы полностью.**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

//Описание структуры

typedef struct MusicalComposition

{

char name[80];

char author[80];

int year;

struct MusicalComposition\* next;

struct MusicalComposition\* prev;

} MusicalComposition;

//создание одного элемента типа структуры

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

{

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

strcpy(rr->name, name);

strcpy(rr->author, author);

rr->year = year;

rr->next = NULL;

rr->prev = NULL;

return rr;

}

//создание двусвязного списка из нескольких элементов типа структуры

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;

}

//ввод данных для создания нового элемента типа структуры

MusicalComposition\* ReturnElement()

{

char name[80];

char author[80];

int year;

printf("Enter name: ");

scanf("%s", name);

printf("Enter author: ");

scanf("%s", author);

printf("Enter year: ");

scanf("%d", &year);

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

return element;

}

//добавление элемента в конец списка

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

{

while (head->next)

head = head->next;

head->next = element;

element->prev = head;

element->next = NULL;

}

//добавление нескольких элементов в конец списка

void pushn(MusicalComposition\* head)

{

int n, i=0;

printf("Enter number of compositions to push:\n");

scanf("%d", &n);

printf("Enter compositions(nay):\n");

while (i<n)

{

MusicalComposition\* element\_for\_push = ReturnElement();

push(head, element\_for\_push);

i++;

}

}

//поиск и удаление элемента по названию

void removeEl(MusicalComposition\*head)

{

char name\_for\_remove[80];

printf("Enter name: \n");

scanf("%s", 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=\*head->next;

head->prev=NULL;

return;

}

else

{

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

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

return;

}

break;

}

head = head->next;

}

}

//счетчик элементов

int count(MusicalComposition\*head)

{

int count = 0;

while (head)

{

count++;

head = head->next;

}

return count;

};

//вывод названий

void print\_names(MusicalComposition \*head)

{

while (head)

{

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

head = head->next;

}

}

//меняет половины местами (центральный элемент оказывается в правой половине до перестановки)

MusicalComposition\* PomenyatPolovinyMestami(MusicalComposition\*\* head)

{

int n=1;

int r;

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

nachalo = \*head;

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

tmp = \*head;

r = count(tmp);

while (2\*n<=r)

{

tmp=tmp->next;

n++;

}

tmp->prev->next=NULL;

tmp->prev=NULL;

while (tmp->next)

{

tmp=tmp->next;

}

tmp->next=nachalo;

nachalo->prev=tmp;

while (tmp->prev)

{

tmp=tmp->prev;

}

return tmp;

}

//меняет местами элементы, симметрично расположенные относительно элемента с номером, соответствующим введенному

void PomenyatOtnosN(MusicalComposition\* head)

{

int n, i=1;

printf("Enter number: ");

scanf("%d", &n);

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

nachalo = head;

while (i<n)

{

nachalo=nachalo->next;

i++;

}

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

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

char\* temp[80];

int\* year;

frw = nachalo->next;

bck = nachalo->prev;

while (frw&&bck)

{

strcpy(temp, frw->author);

strcpy(frw->author, bck->author);

strcpy(bck->author, temp);

strcpy(temp, frw->name);

strcpy(frw->name, bck->name);

strcpy(bck->name, temp);

year=frw->year;

frw->year=bck->year;

bck->year=year;

frw=frw->next;

bck=bck->prev;

}

}

//меню и первоначальный ввод данных

int main()

{

printf("Enter length:\n");

int length;

scanf("%d", &length);

printf("Enter compositions:\n");

char \*\*names = (char\*\*)malloc(sizeof(char\*)\*length);

char \*\*authors = (char\*\*)malloc(sizeof(char\*)\*length);

int \*years = (int\*)malloc(sizeof(int)\*length);

int i;

for (i = 0; i < length; ++i)

{

char name[80];

char author[80];

printf("Enter name %d: ", i+1);

scanf("%s", name);

printf("Enter author %d: ", i+1);

scanf("%s", author);

printf("Enter year %d: ", i+1);

scanf("%d", &years[i]);

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 (1)

{

int j;

printf("Choose action:\n1.Add element\n2.Add some elements\n3.Delete element by name\n4.Swap halves\n5.Swap elements from both sides of N\n6.Print list\n");

scanf("%d", &j);

switch (j)

{

case 1:

{

MusicalComposition\* elfpush = ReturnElement();

push (head, elfpush);

break;

}

case 2:

{

pushn(head);

break;

}

case 3:

{

removeEl(head);

break;

}

case 4:

{

head=PomenyatPolovinyMestami(&head);

break;

}

case 5:

{

PomenyatOtnosN(head);

break;

}

case 6:

{

print\_names(head);

break;

}

}

getchar();

getchar();

if (system("cls"))

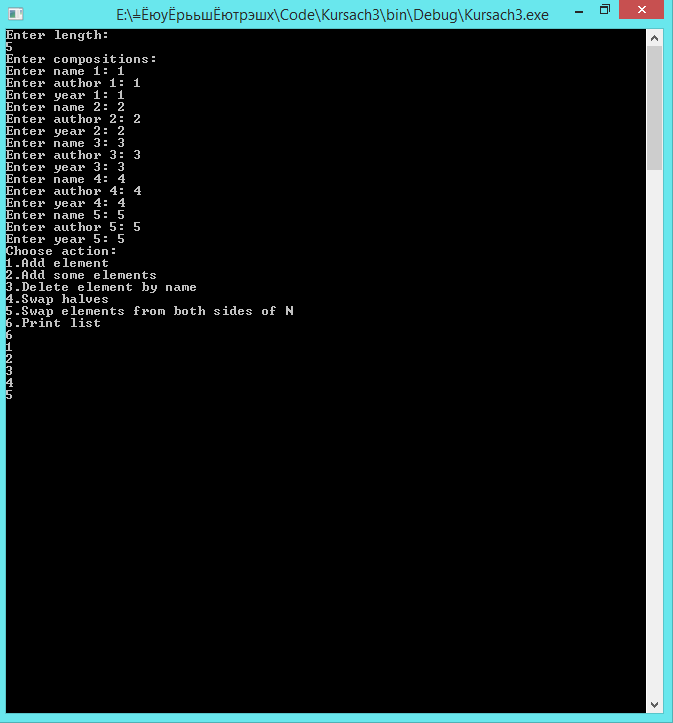
system("clear");

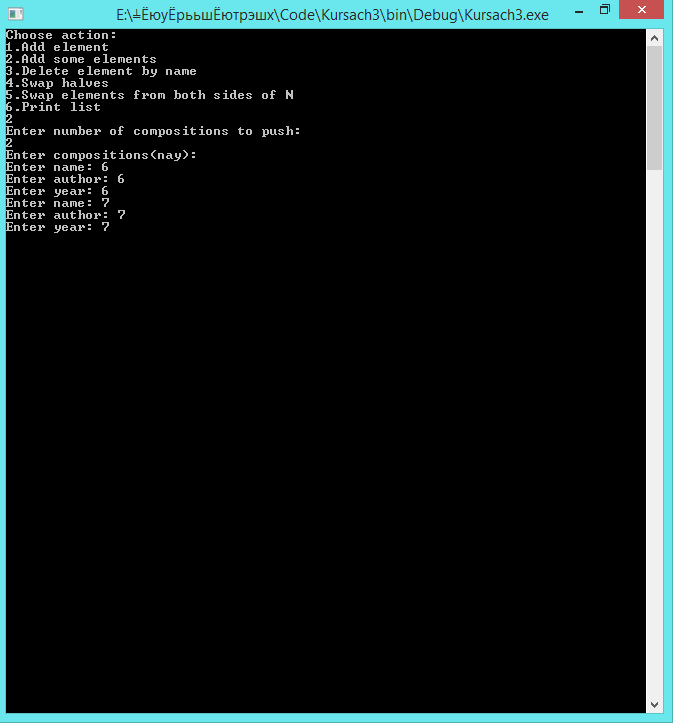
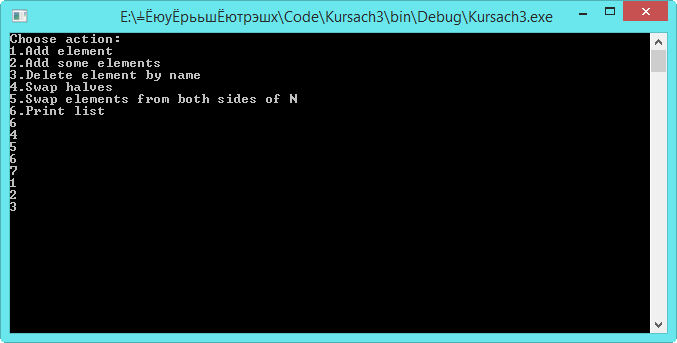
}

return 0;

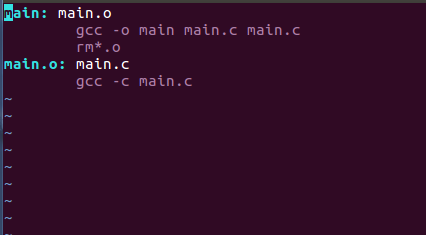
}

**3.10 Примеры работы программы.**





1. **MAKEFILE**



**заключение**

Цель работы достигнута – создан двунаправленный список структур, реализована возможность добавления и удаления элементов, сделан подсчёт элементов и смена их относительно друг друга.

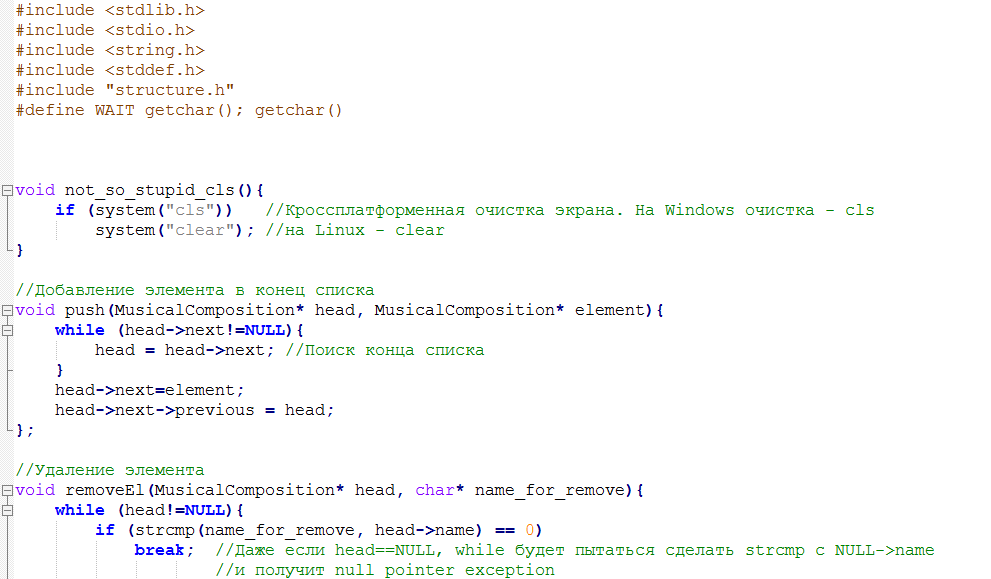
**список использованных источников**

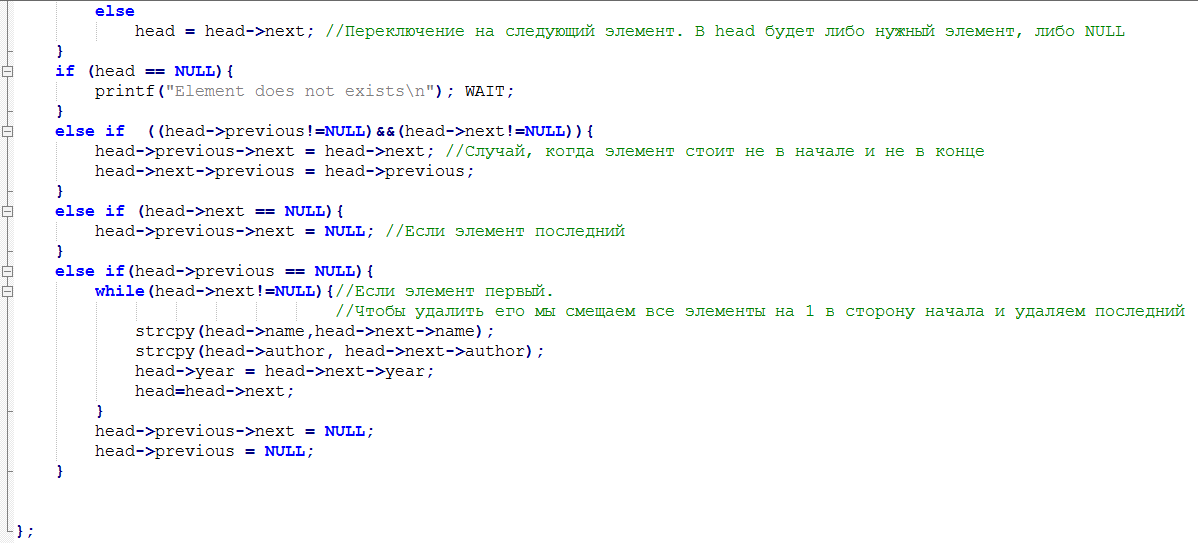
* Команда man в терминале Linux
* Керниган, Ритчи. Язык программирования Си.

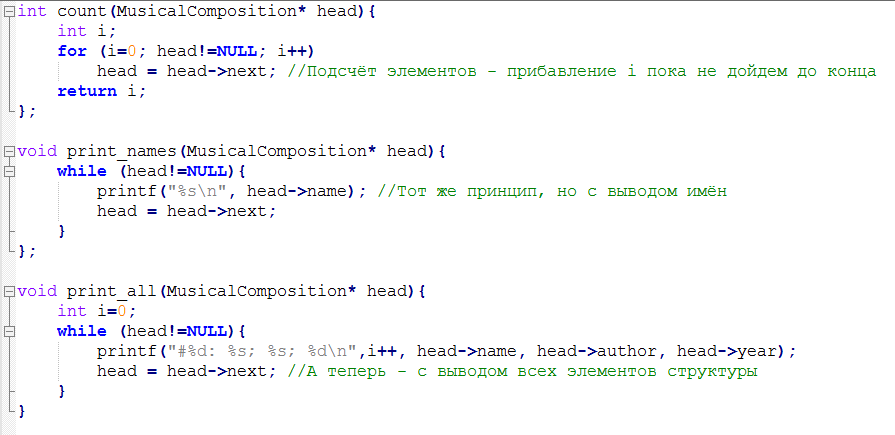
# **приложение А. КОд программы**

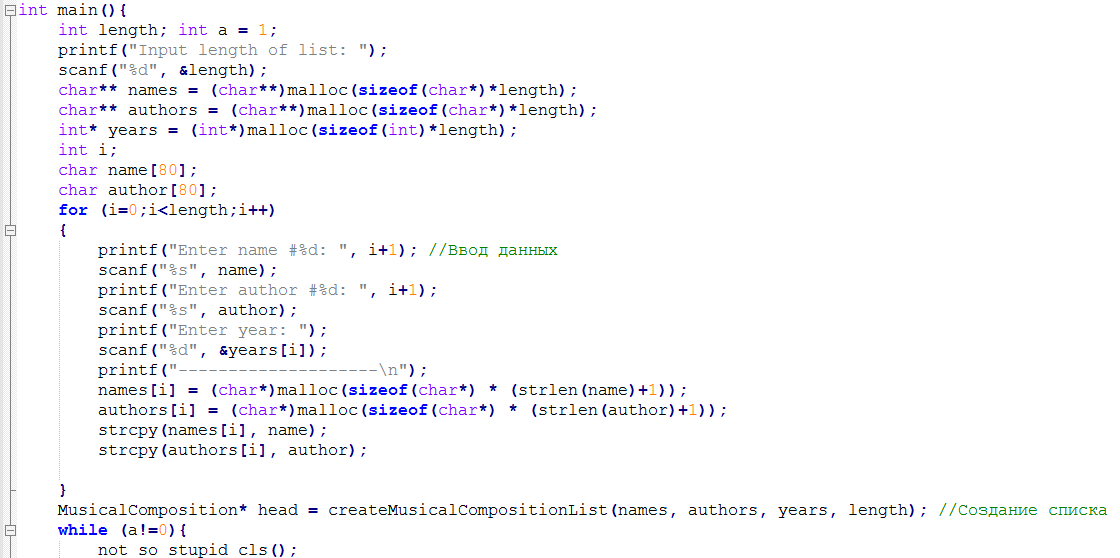
Для сохранения красоты форматирования кода было решено использовать скриншоты вместо копирования текста.

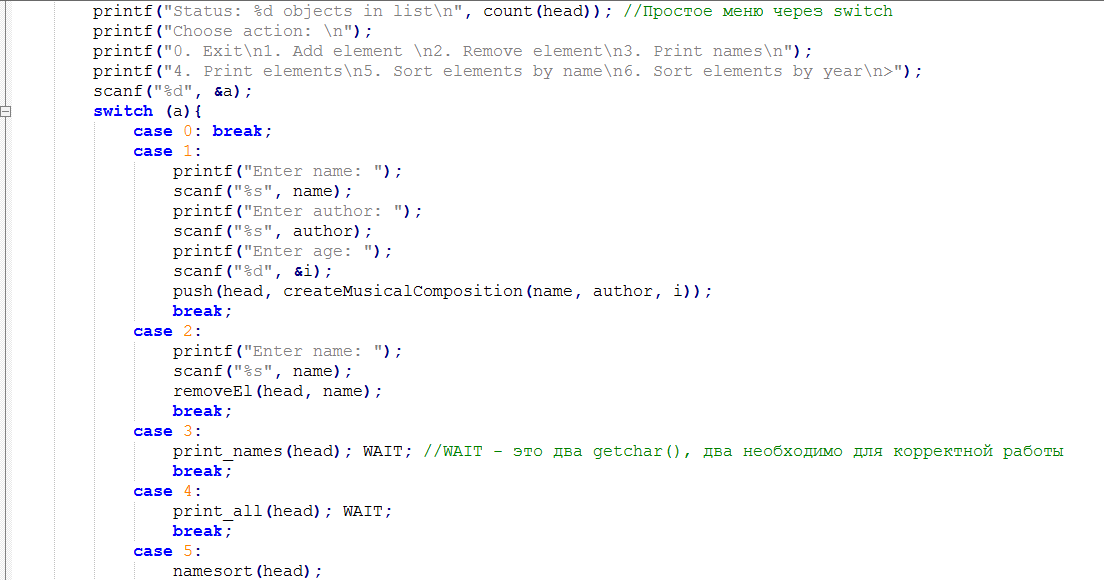
1. **Файл main.c**

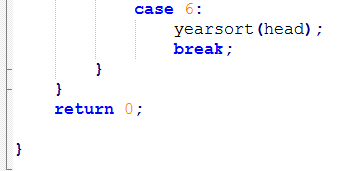




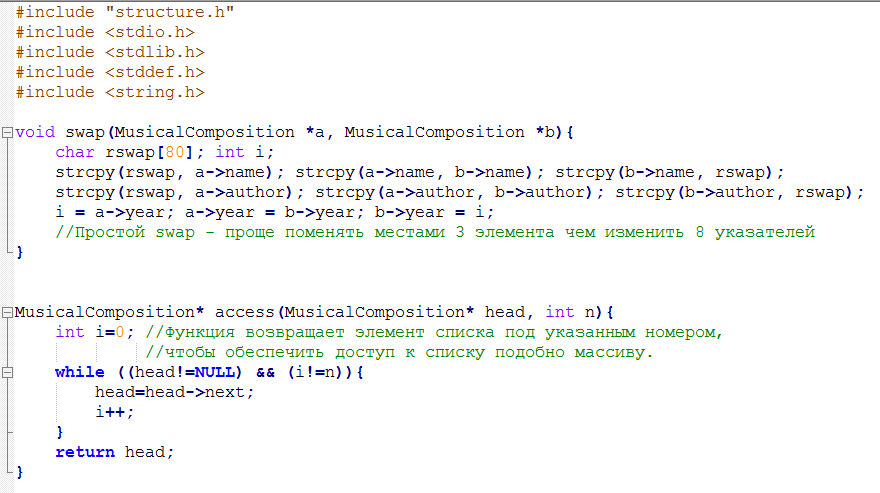


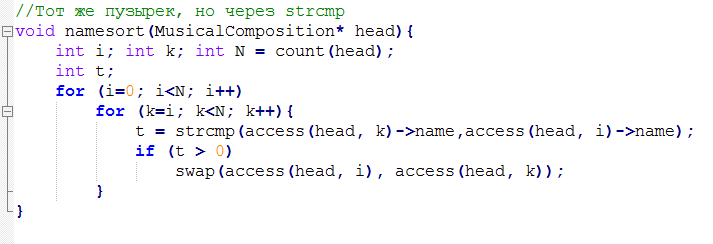




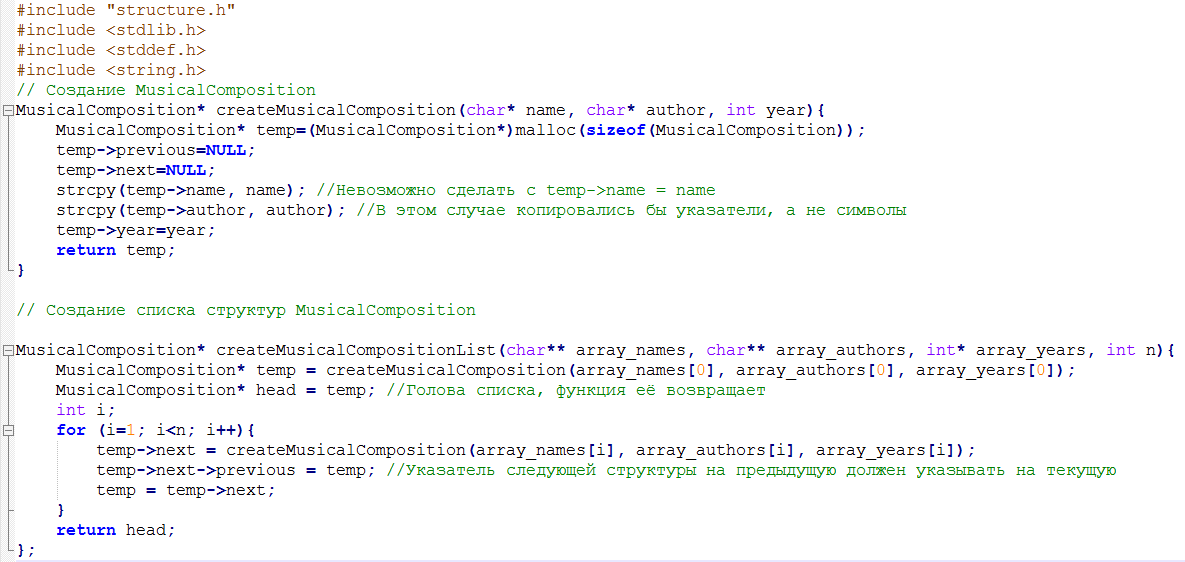


1. **Файл sort.c**

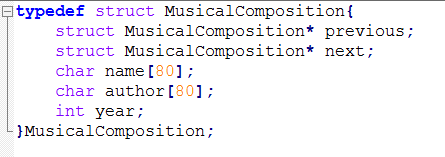




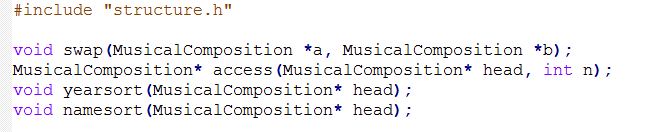
1. **Файл structure.c**



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



* 1. **sort.h.**



* 1. **create.h**

