**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Структуры и работа со списками

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Фокин К.С. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Фокин К.С. | | |
| Группа 6303 | | |
| Тема работы: Линейные списки | | |
|  | | |
| Содержание пояснительной записки:   * Содержание * Цель работы * Ход работы * Примеры работы программы * Вывод * Список использованных источников * Приложение А. | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 02.12.2016 | | |
| Дата сдачи реферата: 16.12.2016 | | |
| Дата защиты реферата: 16.12.2016 | | |
| Студент |  | Фокин К.С. |
| Преподаватель |  | Берленко Т.А. |

**Аннотация**

В данной работе была разработана программа на языке программирования C, которая позволяет работать с набором функций, отвечающих засписок музыкальных композиций. Для функционирования списка были созданы и описаны необходимые функции, позволяющие добавлять, удалять, сортировать и выводить элементы списка и их количество в 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 клиента. Приведено полное описание исходного кода.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Цель работы | 5 |
| 1. | Ход работы | 5 |
| 1.1. | Созданиеэлемента | 5 |
| 1.2. | Создание двунаправленного списка связанных элементов | 5 |
| 1.3. | Добавление элемента в конец списка | 5 |
| 1.4. | Удаление элементов из списка | 5 |
| 1.5. | Подсчет количества элементов в списке | 5 |
| 1.6. | Вывод элементов списка в консоль | 5 |
| 1.7. | Сортировка списка | 5 |
| 2. | Примеры работы программы | 6 |
| 2.1. | Удаление первогоэлемента списка | 6 |
| 2.2. | Удалениеэлемента из середины списка | 6 |
| 2.3. | Удаление последнего (запушенного) элемента списка | 7 |
| 2.4. | Удаление всех элементов списка | 7 |
| 2.5. | Запушивание новой композиции после удаления всех элементов списка | 8 |
| 2.6. | Сортировка списка | 8 |

**Цель работы:**

Научиться использовать линейные списки, создать **api (application programming interface)** для работы со списком.

**Ход работы:**

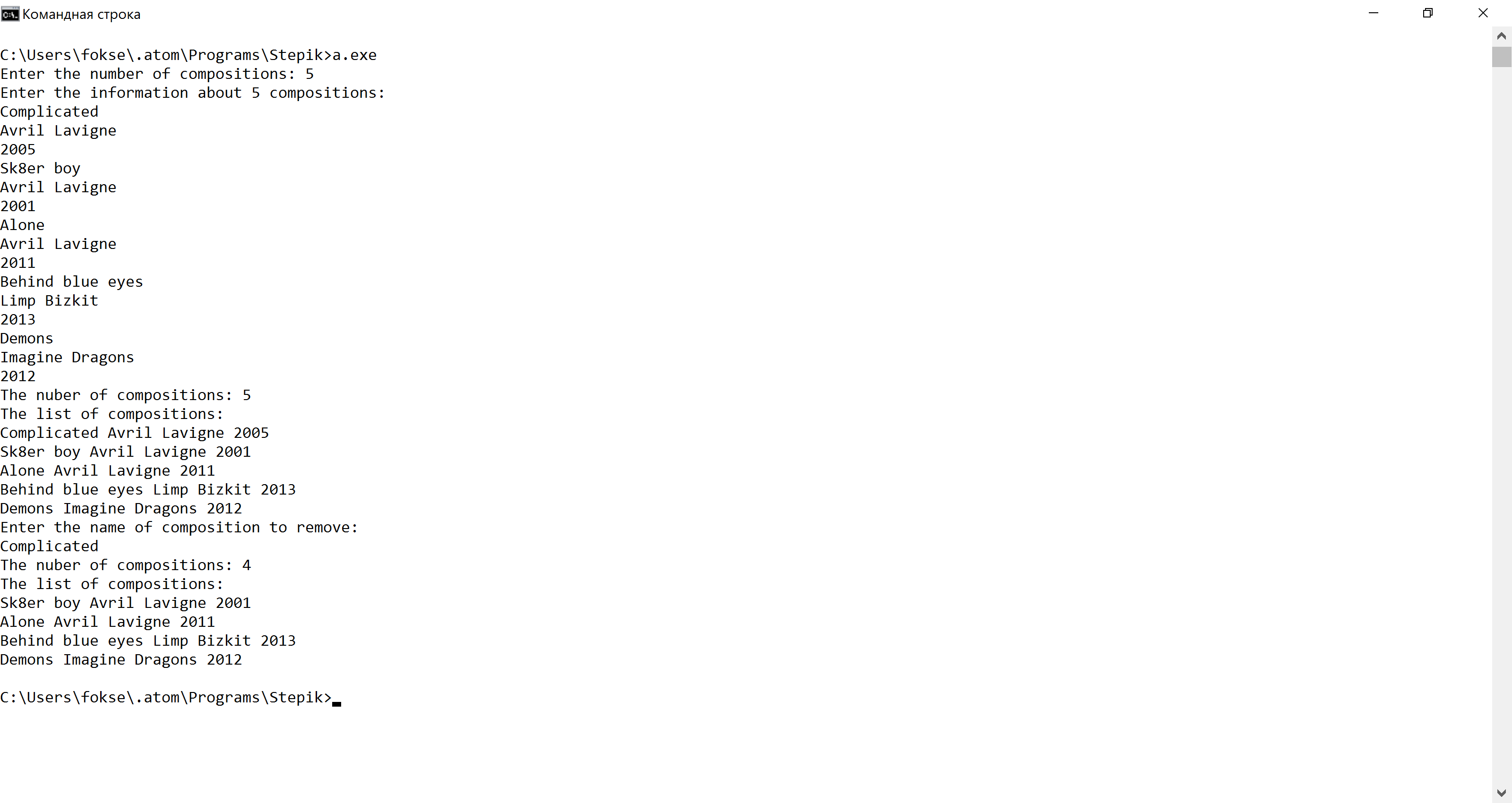
В ветке **fokin\_kurs** репозитория **pr-2016-6303** создал папку **Kursovaya** с проектом, состоящим из 2 файлов: **kursovaya.c**и**Conditions.txt.**

В функции **main** написана некоторая последовательность вызова команд из **api** для проверки работы списка.

1. Функция **MusicalComposition \*createMusicalComposition(char \*name, char \*author, int year)** создает и возвращает структуру типа **MusicalComposition.**
2. Функция **MusicalComposition \*createMusicalCompositionList()** создает двунаправленный список структур, содержащих информацию о музыкальных композициях.
3. Функция **void push(MusicalComposition \*head)** добавляет в конец списка структуру с информацией о новой композиции.
4. Функция **void removeEl(MusicalComposition \*head)** удаляет из списка все композиции, название которых соответствует имени, введенному пользователем.
5. Функция **int count(MusicalComposition \*head)** подсчитывает количество композиций в списке.
6. Функция **void print\_names(MusicalComposition \*head)** построчно выводит информацию (название, исполнитель, год) о каждой композиции, содержащейся в списке.
7. Функция **void sort(MusicalComposition \*head)** по названию композиции сортирует элементы списка в порядке возрастания.

**Примеры работы программы:**

1. Удаление первого элемента списка:



1. Удаление элемента из середины списка:



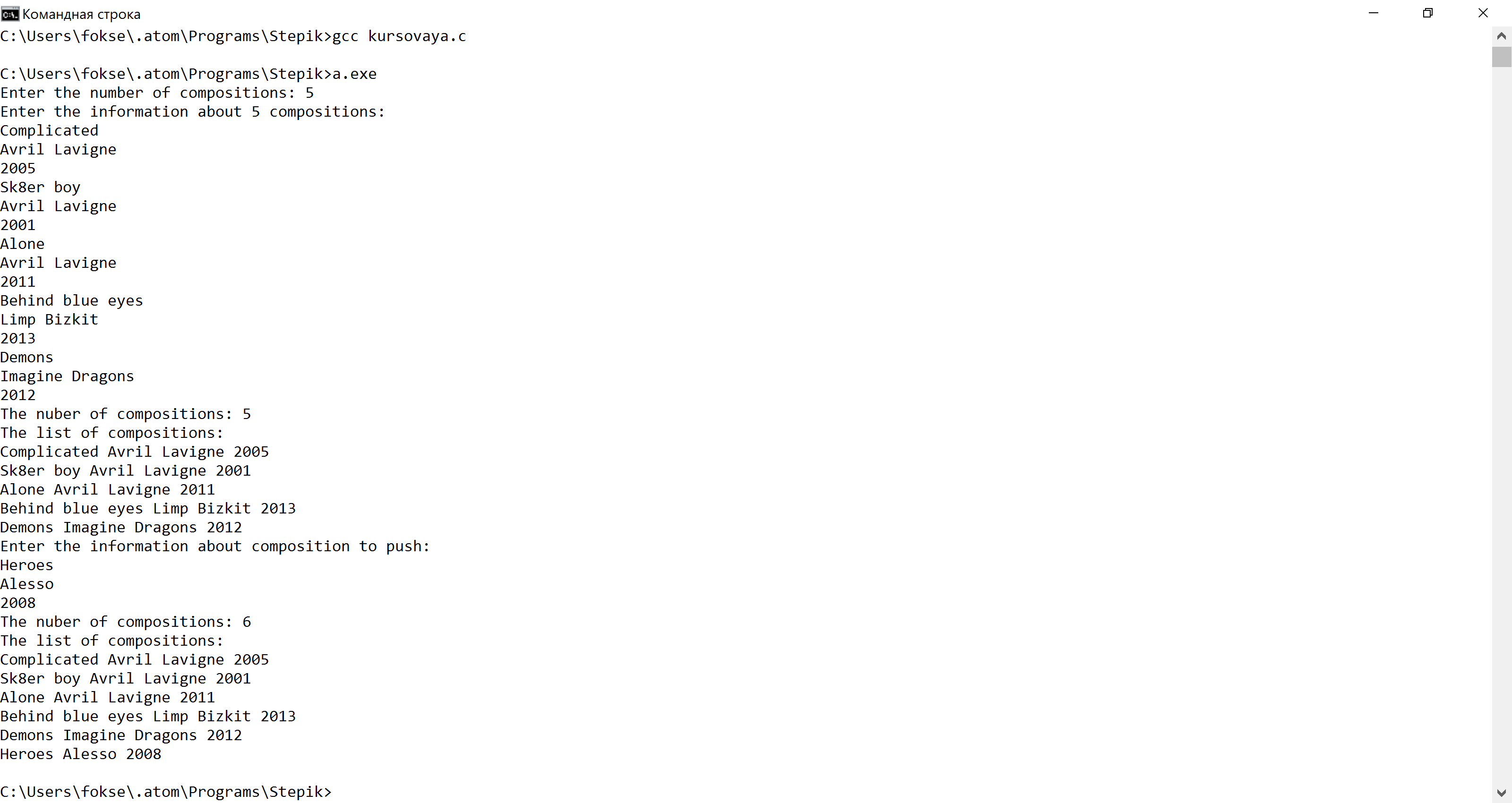
1. Удаление последнего (запушенного) элемента списка:



1. Удаление всех элементов списка:



1. Запушивание новой композиции после удаления всех элементов списка:



1. Сортировка списка:



**Вывод:**

В ходе выполнения курсовой работы были изучены и освоены принципы работы со структурами и объединением их в двунаправленные списки. Полученные навыки успешно применены на практике при разработке программы **Kursovaya.с**.

**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. 416 с.

**Приложение А**

**Код программы:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct MusicalComposition

{

char \*name;

char \*author;

int year;

int status;

struct MusicalComposition \*next, \*prev;

} MusicalComposition;

// Функции для работы со списком MusicalComposition

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

{

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

MusComp->name = name;

MusComp->author = author;

MusComp->year = year;

MusComp->status = 1;

MusComp->prev = NULL;

MusComp->next = NULL;

return MusComp;

}

MusicalComposition \*createMusicalCompositionList()

{

int length;

printf("Enter the number of compositions: ");

scanf("%d", &length);

getchar();

printf("Enter the information about %d compositions:\n", length);

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

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

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

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

{

name[i] = (char\*)malloc(sizeof(char\*)\*81);

author[i] = (char\*)malloc(sizeof(char\*)\*81);

gets(name[i]);

gets(author[i]);

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

getchar();

}

MusicalComposition \*head = createMusicalComposition(name[0], author[0], year[0]);

MusicalComposition \*prev = head;

MusicalComposition \*curr;

for (int i = 1; i < length; ++i)

{

curr = createMusicalComposition(name[i], author[i], year[i]);

curr->prev = prev;

prev->next = curr;

prev = curr;

}

return head;

}

void push(MusicalComposition \*head)

{

char \*name\_for\_push = (char\*)malloc(sizeof(char)\*81);

char \*author\_for\_push = (char\*)malloc(sizeof(char)\*81);

int year\_for\_push;

printf("Enter the information about composition to push:\n");

gets(name\_for\_push);

gets(author\_for\_push);

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

getchar();

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

if ((head->status) == 0)

memcpy(head, element\_for\_push, sizeof(MusicalComposition));

else

{

while (head->next)

head = head->next;

element\_for\_push->prev = head;

head->next = element\_for\_push;

}

}

int count(MusicalComposition \*head)

{

int count = 0;

if ((head->status) == 0)

return 0;

else

while (head)

{

++count;

head = head->next;

}

return count;

}

void removeEl(MusicalComposition \*head)

{

char name\_for\_remove[81];

int amount = count(head);

MusicalComposition \*temp = head;

MusicalComposition \*curr;

printf("Enter the name of composition to remove:\n");

gets(name\_for\_remove);

while (temp->next)

{

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

{

if ((temp->prev) == NULL)

{

temp = head->next;

memcpy(head, temp, sizeof(MusicalComposition));

head->prev = NULL;

free(temp);

temp = head;

}

else

{

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

curr = temp->next;

free(temp);

temp = curr;

}

--amount;

}

else

temp = temp->next;

}

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

{

if (amount != 1)

{

temp->prev->next = NULL;

free(temp);

}

else

head->status = 0;

}

}

void print\_names(MusicalComposition \*head)

{

printf("The list of compositions:\n");

if (head->status)

while (head)

{

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

head = head->next;

}

}

// Функция сортировки для курсовой

void sort(MusicalComposition \*head)

{

int amount = count(head);

MusicalComposition \*curr;

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

printf("The list of compositions after sort:\n");

for (int i = 0; i < (amount - 1); ++i)

{

curr = head;

while (curr->next)

{

if ((strcmp(curr->name, curr->next->name)) > 0)

{

memcpy(temp, curr->next, (sizeof(MusicalComposition) - 8));

memcpy(curr->next, curr, (sizeof(MusicalComposition) - 8));

memcpy(curr, temp, (sizeof(MusicalComposition) - 8));

}

curr = curr->next;

}

}

free(temp);

}

int main()

{

//Функция createMusicalCompositionList всегда вызывается первой и всего один раз

MusicalComposition \*head = createMusicalCompositionList();

push(head);

removeEl(head);

printf("The nuber of compositions: %d\n", count(head));

print\_names(head);

//Вызов фунции сортировки

sort(head);

printf("The nuber of compositions: %d\n", count(head));

print\_names(head);

return 0;

}