**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 |  | Путьков Д.А. |
| Преподаватель |  | Берленко Т.А, |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студент Путьков Дмитрий 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: 24.12.2016 | | |
| Дата защиты реферата: 24.12.2016 | | |
| Студент |  | Путьков Д.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 добавления нескольких элементов в конец списка 9](#_Toc470137655)

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

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

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

[10. Функция сортировки 11](#_Toc470137659)

[11. Функция удаления нечетных элементов списка 12](#_Toc470137660)

[12. Функция вывода ошибки при отсутствии списка 13](#_Toc470137661)

[13. Функция очистки экрана 13](#_Toc470137662)

[14. Функция задержки экрана 13](#_Toc470137663)

[15. 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. Поменять все взаимно противоположные элементы относительно элемента с индексом n(пример для индекса 3 : было 12345 стало 54321, пример для индекса 4 : было 12345 стало 12543)
2. Разделить список на две части(индекс центрального элемента округлить в большую сторону), поменять их местами и соединить.

**Решение задачи**

## Создание структуры данных

С помощью типа *struct* объявим 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

Выделим 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(rr->name, name);

strcpy(rr->author, author);

rr->year = year;

rr->next = NULL;

rr->prev = NULL;

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

return rr;

}

## Функция создания списка

Создадим сначала один элемент и проинициализируем его поля, а затем с помощью еще двух указателей в цикле создадим остальные элементы и свяжем их и вернем 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). Считаем название композиции, автора и год написания, которые мы хотим добавить в список, и заполним 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 перестановки взаимно противоположных элементов

В основе функции перестановки лежит метод функции swap.То есть меняться местами будут не сами элементы списка, а их содержимое. Мы воспроизведем эту функцию с той лишь разницей, что замещаться будут массивы.

void PermutationRegarding\_n(MusicalComposition\* ptr, int n){

//Создаем указатель, счетчик i, чтобы дойти до нужного индекса и промежуточные переменные, нужные для swap функции

MusicalComposition \*headprev;

int i=1;

char fname[80];

char fauthor[80];

int fyear;

// доходим до нужного элемента в списке

while (i!=n){ptr=ptr->next; i++;}

//присваиваем его адрес созданному нами ранее указателю

headprev=ptr;

//пока существует предыдущие и последующие элементы в списке

while ((headprev->prev) && (ptr->next)){

// функция swap, главной особенностью здесь является то, что мы перставляем значения массивов

strcpy(fname,headprev->prev->name);

strcpy(fauthor,headprev->prev->author);

fyear=headprev->prev->year;

strcpy(ptr->next->name,fname);

strcpy(ptr->next->author,fauthor);

ptr->next->year=fyear;

// присваиваем указателю адрес предыдущего элемента

headprev=headprev->prev;

// присваиваем указателю адрес следующего элемента

ptr=ptr->next;

}

}

## Функция перестановки частей списка местами

Если номер элемента в списке нечетен (нумерация начинается с нуля), тогда функции удаления передается имя элемента и происходит его удаление.

/\*Функция перестановки частей списка\*/

MusicalComposition\* PermutationRegardingMiddle(MusicalComposition \*head, int k){

//Создание двух указателей

MusicalComposition \*ptr=head;

MusicalComposition \*saveHead=head;

//Проверяем на нечетность

if ((k % 2)==0) k=k/2; else k=(k-1)/2;

//доходим до последнего элемета

while (ptr->next) ptr=ptr->next;

//Связываем последний и первый элементы

ptr->next=head;

head->prev=ptr;

//Доходим до среднего элемента

for (int i=1;i<k;i++) head=head->next;

//Делаем средний элемент первым

head->next->prev=NULL;

//передаем его адрес

saveHead=head->next;

//делаем последний элемент первой части последним в списке

head->next=NULL;

//Возвращаем адрес первого элемента получившегося списка

return saveHead;

}

## Makefile

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

#Главная цель

kurswork.out : main.cpp functionsFromLab.cpp kursFunctions.cpp

g++ main.cpp kursFunctions.cpp functionsFromLab.cpp -o kurswork.out

#Объектный файл для функции main

main.o: main.cpp

g++ -c main.cpp

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

|  |
| --- |
|  |
| functionsFromLab.o:/functionsFromLab.cpp/functionsFromLab.h |
|  |

g++ -c functionsFromLab.cpp

#Объектный файл для функций для курсовой

kursFunctions.0: kursFunctions.cpp kursFunctions.h

g++ -c kursFunctions.cpp

#очистка

clean:

rm -rf \*.o

# **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 “Putkov\_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 Putkov\_kursework*

1. Коммитим изменения:

*Git commit –m “The kursework by PutkovDmitriy”*

1. Отправляем изменения в репозиторий командой

*Git push origin*

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

**примеры работы программы:**

Для демонстрации всех возможностей программы используется следующий формат ввода:

* 1. Выбор пункта меню
  2. Количество элементов списке
  3. Ввод названия, имени автора и год создания элемента списка
  4. Выбор в меню следующего действия
  5. Новый элемент списка
  6. Выбор в меню следующего действия
  7. Название композиции, которую нужно удалить
  8. Индекс, относительно которого будет просходить перестановка элементов

Для вывода:

1. Название, автор и год написания элементов списка
2. Количество элементов в результате каждого нашего действия
3. Список названий композиций после добавления
4. Количество элементов после удаления
5. Список элементов после удаления

|  |  |  |
| --- | --- | --- |
| № | ВВОд | вЫВОД |
| 1 | 6 // количество элементов списка  Highway to Hell // элементы для формирования списка  AC/DC  1980  Turn Down for What  DJ Lil John  2015  Know your enemy  Green Day  2004  Numb  Linkin Park  2003  Shape of my heart  Sting  2000  Романс  Сплин  2002  1//выбор в меню добавления нового элемента  Грязь // элемент для добавления в список  Schokk  2013  2//выбор в меню удаления элемента  Shape of my heart //имя элемента для удаления из списка  5// запрос на вывод списка на экран | Highway to Hell//список названий композиций  Turn Down for What //после добавления и удаления элемента  Know your enemy  Numb  Романс  Грязь  6 //количество элементов после добавления и удаления |
| 2 | 4 // количество элементов для формирования списка  Name1 // элементы для формирования списка  author1  1  Name2  author2  2  Name3  author3  3  Name4  author4  4  4//Переставить части списка местами  5// запрос на вывод списка на экран | Name1 // имена элементов списка создания  Name2  Name3  Name4  4  // список элемента после перестановки частей  Name3  Name4  Name2  Name1  4// количество элементов списка |
| 3 | 3// количество элементов  Name1  Athor1  1  …  Name5  Author5  5  // созданныйсписок  3// перестановка относительно индекса  2// индекс 2  5// запрос на вывод списка на экран | //список после перестановки  Name3  Name2  Name1  Name4  Name5  5// количество элементов 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**

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

strcpy(rr->name, name);

strcpy(rr->author, author);

rr->year = year;

rr->next = NULL;

rr->prev = NULL;

return rr;

}

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;

}

//MusicalComposition\* createList(){

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

{

while (head->next)

head = head->next;

head->next = element;

element->prev = head;

element->next = NULL;

}

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

{

MusicalComposition \*ptr;

ptr=head;

while (head)

{

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

{

if (head->next == NULL)

{

head->prev->next = NULL;

head=ptr;

return head;

}

else if (head->prev == NULL)

{

head->next->prev = NULL;

head=ptr;

return head->next;

}

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

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

head=ptr;

break;

}

head = head->next;

}

return head;

}

void PermutationRegarding\_n(MusicalComposition\* ptr, int n){

MusicalComposition \*headprev; // создаем указатель, который будет идти в сторону первого эл-та списка

int i=1;

char fname[80];

char fauthor[80];

int fyear;

while (i!=n){ptr=ptr->next; i++;}

headprev=ptr;// присваиваем headprev значение ptr

while ((headprev->prev) && (ptr->next)){

// Берем промежуточные переменные для записи имени из элемента структуры перед n

strcpy(fname,headprev->prev->name);

strcpy(fauthor,headprev->prev->author);

fyear=headprev->prev->year;

//Записываем в элемент списка перед n значения элемента списка после n

strcpy(headprev->prev->name,ptr->next->name);

strcpy(headprev->prev->author,ptr->next->author);

headprev->prev->year=ptr->next->year;

//Записываем в элемент списка после n значения элемента списка перед n

strcpy(ptr->next->name,fname);

strcpy(ptr->next->author,fauthor);

ptr->next->year=fyear;

// идем по списку в противоположные стороны пока "условие цикла"

headprev=headprev->prev;

ptr=ptr->next;

}

}

MusicalComposition\* PermutationRegardingMiddle(MusicalComposition \*head, int k){ // Функция делит список на две части, меняет их местами и склеивает

// создаем локальные указатели на первый элемент

MusicalComposition \*ptr=head;

MusicalComposition \*saveHead=head;

// проверяем на четность. При отрицательном результате индекс центрального элемента относим во вторую часть согласно условию

if ((k % 2)==0) k=k/2; else k=(k-1)/2;

while (ptr->next) ptr=ptr->next; // Указатель ptr указывает на последний элемент

// переставляем элементы списка

ptr->next=head;

head->prev=ptr;

for (int i=1;i<k;i++) head=head->next; // доходим до центрального элемента

// первый элемент второй части списка становится первым элементом списка

head->next->prev=NULL;

saveHead=head->next;//указателю saveHead присваиваем адрес первого элемента списка

//а последний элемент первой части списка-последним элементом списка.

head->next=NULL;

return saveHead;// функция возвращает указатель на первый элемент списка

}

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;

}

}

int main()

{

int length;

printf("Enter the length of the Musical list\n");

scanf("%d", &length);

int tmp;

while((tmp = getchar()) != '\n');

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

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

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

printf("Enter the composition's name, author and year of origin:\n");

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

{

char name[80];

char author[80];

fgets(name, 80, stdin);

fgets(author, 80, stdin);

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

while((tmp = getchar()) != '\n');

printf("Enter the composition's name, author and year of origin:\n");

(\*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));

strcpy(names[i], name);

strcpy(authors[i], author);

}

MusicalComposition \*head = createMusicalCompositionList(names, authors, years, length);

MusicalComposition \*ptr=head;

int action=0;

int elements=0;

int choise=2;

int k;

while (choise!=1)

{

printf("What actions are you want to do with the list:\n 1.Add an element\n 2.Remove an elements\n 3.Change the list regarding the number you will write\n 4.Change the list by the parties\n 5.Print the list\n");

scanf("%d", &action);

int tmp;

while((tmp = getchar()) != '\n');// очистка буфера

switch (action) // выбор действия

{

case 1: // если выбрали "добавить элемент"

{

char name\_for\_push[80]; // выделение памяти по имя

char author\_for\_push[80]; // выделение памяти под имя автора

int year\_for\_push; // выделение памяти под год создания

printf("Enter the composition's name, author and year of origin:\n");

fgets(name\_for\_push, 80, stdin); // ввод строки

fgets(author\_for\_push, 80, stdin);

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

while((tmp = getchar()) != '\n'); // очистка стдин

(\*strstr(name\_for\_push, "\n")) = 0;

(\*strstr(author\_for\_push, "\n")) = 0;

MusicalComposition \*element\_for\_push = createMusicalComposition(name\_for\_push, author\_for\_push, year\_for\_push); // функция создания элемента

push(head, element\_for\_push); // функция добавления элемента в плейлист

k=count(head); // счет количества композиций в плейлисте

break;

}

case 2: // если "выбрали удалить элемент"

{

char name\_for\_remove[80]; // выделение памяти под имя

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

fgets(name\_for\_remove, 80, stdin); // ввод строки

(\*strstr(name\_for\_remove, "\n")) = 0;

head=removeEl(head, name\_for\_remove); // вызов функции удаления элемента

k=count(head);// счет количества композиций в плейлисте

break;

}

case 3: // если выбрали " переставить элементы относительно введенного индекса

{

int n;// выделения памяти под индекс

printf("Choose the number which the list will be changed regarding\n");

fscanf(stdin, "%d", &n);// ввод индекса

PermutationRegarding\_n(ptr, n); // вызов функции перестановки

break;

}

case 4: // переставить части плейлиста местами

{

k=count(head); //счет количества элементов в плейлисте

head=PermutationRegardingMiddle(head, k); // передача указателю на первый элемент значения, которое возвращает функция

break;

}

case 5: // вывод плейлиста на экран

{

k=count(head); // счет количества элементов

print\_names(head); // вызов функции вывода названий

printf("The amount of musical composition is %d\n", k); // вывод количества песен

break;

}

}

printf("Are you want to exit?\n 1.Yes\n 2.No\n");

fscanf(stdin, "%d", &choise); // предложение продолжить работу

}

return 0;

}