**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Двунаправленные линейные списки

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Прозорова А. Д. |
| Преподаватель |  | Берленко Т. А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| **Студентка** Прозорова Анастасия Дмитриевна | | |
| **Группа** 6304 | | |
| **Тема работы:** Двунаправленные линейные списки | | |
| **Исходные данные:**  Требуется создать двунаправленный линейный список и API( ***a****pplication* ***p****rogramming* ***i****nterface)* для работы с ним. | | |
| **Содержание пояснительной записки:**   * Аннотация * Введение * Описание необходимых для работы со списком функций * Создание makefile * Описание работы с git и github * Примеры работы программы * Вывод * Приложение с исходным кодом программы | | |
| **Предполагаемый объем пояснительной записки:**  25 страниц. | | |
| **Дата выдачи задания:** 07.12.2016 | | |
| **Дата сдачи реферата:** 29.12.2016 | | |
| **Дата защиты реферата:** 29.12.2016 | | |
| Студентка |  | Прозорова А. Д. |
| Преподаватель |  | Берленко Т. А. |

# **АННОТАЦИЯ**

Данная работа направлена на создание двунаправленного линейного списка музыкальных композиций, а также API для работы с помощью языка программирования С, в частности, с использованием структур, указателей и работы с динамической памятью. В работе реализуются стандартные операции, такие как добавление нового элемента, удаление, подсчет количества элементов и т.д. В работе использовались файлы исходного кода(\*.c), заголовочные файлы(\*.h), makefile. Также продемонстрированы основные этапы работы с Git.

1. Оглавление

[АННОТАЦИЯ 3](#__RefHeading__5436_189884073)

[ВВЕДЕНИЕ 5](#__RefHeading___Toc470274170)

[Цель работы 5](#__RefHeading___Toc470274171)

[Формулировка задачи 5](#__RefHeading___Toc470274172)

[ТЕОРИЯ 1](#__RefHeading___Toc470274174)

[РЕШЕНИЕ ЗАДАЧИ 2](#__RefHeading___Toc470274175)

[1) Создание структуры данных 2](#__RefHeading___Toc470274176)

[2) Функция создания нового элемента структуры MusicalComposition 2](#__RefHeading___Toc470274177)

[3) Функция создания списка 2](#__RefHeading___Toc470274179)

[4) Функция добавления элемента в конец списка 3](#__RefHeading___Toc470274180)

[5) Функция удаления элемента 3](#__RefHeading___Toc470274181)

[6) Функция подсчета элементов списка 4](#__RefHeading__5438_189884073)

[7) Функция вывода имен 4](#__RefHeading___Toc470274183)

[8) Функция вывода всего списка(реализуется аналогично предыдущей) 4](#__RefHeading___Toc470274184)

[9) Вспомогательная функция 4](#__RefHeading___Toc470274185)

[Для реализации «индивидуального задания», принимает на вход два элемента и меняет их местами. 4](#__RefHeading__5440_189884073)

[10) Вторая вспомогательная функция 5](#__RefHeading__5442_189884073)

[11) Итоговая функция 6](#__RefHeading__5444_189884073)

[12) Функция, копирующая первую часть списка и вставляющую её в центр. 6](#__RefHeading__5446_189884073)

[13) Главная функция, в которой реализовано меню. 7](#__RefHeading__5448_189884073)

[Makefile 9](#__RefHeading___Toc470274187)

[Работа с GitHub: 9](#__RefHeading___Toc470274188)

[ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ 10](#__RefHeading___Toc470274189)

[ЗАКЛЮЧЕНИЕ 12](#__RefHeading___Toc470274190)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 13](#__RefHeading__7345_189884073)

[ПРИЛОЖЕНИЕ 14](#__RefHeading__5450_189884073)

[Файл main.c 14](#__RefHeading___Toc470274192)

[Файл funk.c 16](#__RefHeading___Toc470274193)

[Файл funk2.c 17](#__RefHeading___Toc470274194)

[Файл struct.h 19](#__RefHeading__5452_189884073)

[Файл funk.h 19](#__RefHeading___Toc470274195)

[Файл funk2.h 19](#__RefHeading__5454_189884073)

# **ВВЕДЕНИЕ**

## **Цель работы**

Обучение применению на практике сложных типов (struct) в языке С. Использование их для реализации сложных структур данных, в частности, одно- и двунаправленных линейных списков, создание API для работы с ними.

## 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. Разделить список на две части относительно элемента с индексом n, разделить вторую часть на группы по m элементов и поменять в каждой группе начальный и конечный элемент местами. Соединить все элементы в список.

2. Разделить список на две части (индекс центрального элемента округлить в большую сторону), сделать копию первой части и вставить между разделенными частями, соединить все части.

# **ТЕОРИЯ**

*Список* – линейная последовательность элементов, каждый из которых содержит указатели (ссылается) на своих соседей.

·        элемент списка доступен в программе через указатель. «Смысл» этого указателя отражает функциональное назначение элемента списка в программе: первый, последний, текущий, предыдущий, новый и т.п.. Между указателем и элементом списка имеется такая же взаимосвязь, как между индексом в массиве и элементом массива;

·        в программе список задается посредством заголовка – указателя на первый элемент списка;

·        порядок следования элементов определяется последовательностью связей между элементами. Изменение порядка следования элементов (вставка, удаление) осуществляются изменением переустановкой указателей на соседние элементы.

·        логический (порядковый) номер элемента списка также задается его естественной нумерацией в цепочке элементов;

·        список является структурой данных  с последовательным доступом. Для получения n-го по счету элемента необходимо последовательно пройти по цепочке от элемента, на который имеется указатель (например, от заголовка);

·        список удобен для использования именно как *динамическая структура данных*: элементы списка обычно создаются как динамические переменные, а связи между ними устанавливаются программно (динамически);

·        список обладает свойством локальности изменений: при вставке/удалении элемента изменения касаются только текущего и его соседей.

# **РЕШЕНИЕ ЗАДАЧИ**

## *1) Создание структуры данных*

Создадим описание структуры MusicalComposition:

struct MusicalComposition { //объявление структуры

char name[80];

char author[80];

int year;

struct MusicalComposition \* next;//*следующий элемент списка*

struct MusicalComposition \* prior; //*предыдущий элемент списка*

};

C помощью typedef пределим новый тип MusicalComposition, чтобы впоследствии не использовать struct.

typedef struct MusicalComposition MusicalComposition;

## *2) Функция создания нового элемента структуры MusicalComposition*

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

MusicalComposition\* head = (MusicalComposition\*)malloc(sizeof(MusicalComposition)); *//выделение памяти на элемент*

strcpy(head->name, name); *//копирование данных в элемент*

strcpy(head->author, author);

head->year = year;

head->prior=NULL; *//нет ссылок на предыдущий и следующий элемент*

head->next=NULL;

return head;

}

## *3) Функция создания списка*

Создадим один элемент и проинициализируем его поля, после с помощью цикла создадим остальные элементы, свяжем их и вернем указатель на первый элемент в списке.

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

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

MusicalComposition\* head = tmp;

int i;

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

tmp->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

tmp->next->prior = tmp; *//связывание со следующим элементом*

tmp = tmp->next; *//переход на следующий элемент*

}

return head;

}

## *4) Функция добавления элемента в конец списка*

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

while(head->next){ *//перемещаемся по списку до последнего элемента*

head=head->next;

}

head->next=element; *//связываем последний элемент в списке с новым элементом*

element->prior=head;

}

## *5) Функция удаления элемента*

Найдем нужный элемент и удалим его, изменив связь с предыдущим и следующим. Рассмотрим три случая, когда элемент находится в начале списка, когда он в середине и когда в конце.

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

MusicalComposition \*tmp = head;

while (tmp->next){ *//передвижение по списку*

if (strcmp(tmp->name, name\_for\_remove) == 0) { *//если имя элемента совпало с нужным*

if ((tmp->prior) == NULL){ *//если это первый элемент*

MusicalComposition \*ntmp = tmp->next; /*/убираем связь со следующим элементом, связываем с новым*

strcpy(tmp->name, ntmp->name);

strcpy(tmp->author, ntmp->author);

tmp->year = ntmp->year;

ntmp->next->prior = tmp;

tmp->next = ntmp->next;

free(ntmp);

}

else{ *//если элемент не первый*

tmp->prior->next = tmp->next; *//меняем значение next у предыдущего элемента и значение prior у следующего*

tmp->next->prior = tmp->prior;

free(tmp); *//очищаем tmp*

}

}

tmp = tmp->next; *//переходим на следующий элемент*

}

if ((strcmp(tmp->name, name\_for\_remove) == 0) && (tmp->next == NULL)){ *//если это последний элемент*

tmp->prior->next = NULL; *//аналогично убираем элемент*

free(tmp);

}

}

## *6) Функция подсчета элементов списка*

int count(MusicalComposition\* head){

int count=0;

while (head){ *//движемся по списку до конца, увеличивая count на 1*

count++;

head=head->next;

}

return count;

}

## *7) Функция вывода имен*

void print\_names(MusicalComposition\* head){

MusicalComposition \*tmp = head;

while (tmp->next){ *//передвигаемся по списку, пока есть следующий элемент*

printf("%s\n", tmp->name); *//выводим название композиции*

tmp = tmp->next;

}

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

}

## *8) Функция вывода всего списка(реализуется аналогично предыдущей)*

void print\_list(MusicalComposition\* head){

while (head->next){

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

printf("%s ", head->author);

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

head = head->next;

}

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

printf("%s ", head->author);

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

}

## *9) Вспомогательная функция*

## *Для реализации «индивидуального задания», принимает на вход два элемента и меняет их местами.*

void exchange(MusicalComposition\* a, MusicalComposition\* b){

char rubbish[80];

int rubb;

strcpy(rubbish, a->name);

strcpy(a->name, b->name);

strcpy (b->name, rubbish);

strcpy(rubbish, a->author);

strcpy(a->author, b->author);

strcpy (b->author, rubbish);

rubb=a->year;

a->year=b->year;

b->year=rubb;

}

## *10) Вторая вспомогательная функция*

*Рразбивает список на группы по m элементов и в каждой группе меняет первый и последний элемент местами с помощью предыдущей функции. Если список на группы по m элементов не делится ровно, то данные группы отделяются и в них происходит обмен, а оставшиеся элементы просто присоединяются.*

MusicalComposition\* ChangeList1(MusicalComposition\* head, int m){

int i=0;

int j=0;

int n=count(head);

MusicalComposition\*\* arr = (MusicalComposition\*\*)malloc((n+1)\*sizeof(MusicalComposition\*));

for (i = 0; i<n; i++) arr[i] = (MusicalComposition\*)malloc(sizeof(MusicalComposition\*));

for (i = 0; i<n; i++){

arr[i] = head;

head = head->next;

}

if (n%m==0){ //если кол-во элементов делится на m

for(i = 0; i<n-1; i = i + m){ *//разбиваем список на группы по m элементов*

arr[i+m-1]->next = NULL;

arr[i]->prior = NULL;

exchange(arr[i], arr[i+m-1]); *//в каждой группе меняем первый и последний элемент местами*

}

for (i = 1; i<n-1; i = i + m){ *//объединяем группы обратно в список*

arr[i+m-2]->next = arr[i+m-1];

arr[i+1]->prior = arr[i];

}

}

else {

for(i = 0; i<n-2; i = i + m){ *//отделяем в списке группы*

arr[i+m-1]->next = NULL;

arr[i]->prior = NULL;

exchange(arr[i], arr[i+m-1]);  *//меняем начальный и конечный элемент местами*

}

for (i = 1; i<n-2; i = i + m){ *//соединяем группы в список*

arr[i+m-2]->next = arr[i+m-1];

arr[i+1]->prior = arr[i];

}

i=n-2;

arr[i+1]->next=arr[i+2]*//присоединяем оставшиеся элементы*

arr[i+1]->prior=arr[i];

}

head = arr[0];

return head; *//возвращаем указатель на первый элемент в списке*}

## *11) Итоговая функция*

*Разделяет список на две части относительно индекса n, вторую часть разбивает на группы по m элементов и в каждой группе меняет первый и последний символ местами, затем соединяет все элементы обратно в список.*

MusicalComposition\* ChangeList2(MusicalComposition\* head, int n, int m){

MusicalComposition \*tmp = head;

int j;

for(j=0;j<(n-1);j++) tmp=tmp->next; *//доходим до элемента с индексом n*

MusicalComposition \*curr=tmp->next;

tmp->next=NULL; *//отделяем первую часть*

ChangeList1(curr, m); *//вызываем предыдущую функцию для второй части*

tmp->next=curr; *//соединяем части обратно в общий список*

return head;

}

## *12) Функция, копирующая первую часть списка и вставляющую её в центр.*

*Если количество элементов нечетное, то индекс центрального элемента округляется в большую сторону.*

MusicalComposition\* ChangeList(MusicalComposition\* head){

MusicalComposition\* current = head;

int l = count(head);

int i;

float n;

if (l%2==0) n=l/2;

else n=l/2+0.5;

for (i=0; i<n; i++){

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

strcpy(tmp->name, head->name);

strcpy(tmp->author, head->author);

tmp->year = head->year;

current=pushfirst(current, tmp); *//вставляет элемент в начало списка*

head=head->next;

}

print\_list(current);

return current;

}

## *13)Функция, вставляющая элемент в начало списка*

MusicalComposition\* pushfirst(MusicalComposition\* head, MusicalComposition\* element) {

head->prior = element;

element->next = head;

return element;

}

## *14) Главная функция, в которой реализовано меню.*

int main(){

int a = 0;

int l;

int year;

char name[80];

char author[80];

printf("Создание списка\nВведите длину списка\n");

scanf("%d", &l);

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

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

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

int i;

for(i=0;i<l;i++){ *//вводим соответствующие массивы «имя» «автор» и «дата»*

printf("Введите название, автора композиции и год создания\n");

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

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);

years[i]=year;

}

MusicalComposition\* head = createMusicalCompositionList(names, authors, years, l); *//создаем из введенных массивов линейный список*

for(i=0; i<l; i++){

free(names[i]);

free(authors[i]);

}

free(years); *//очищаем массивы*

while(1==1){ *//зацикливаем меню*

printf("1)Добавление нового элемента в конец списка;\n2)Удаление определенного элемента;\n3)Вывод длины списка;\n4)Вывод названий композиций;\n5)Разделить список на две части относительно индекса n, вторую часть разделить на группы по m элементов, соединить элементы обратно в список;\n6)Разделись список на две равные части(или же индекс центрального элемента округлить в большую сторону), скопировать первую часть и вставить в середину\n7)Выход\n");

scanf("%d", &a);

if(head==0){ *//проверка на отсутствие списка и его создание*

while((a != 1) || (a != 0)){

printf("Если хотите продолжить работу, введите 1, если хотите выйти, введите 0\n");

scanf("%d", &a);

}

if(a == 0) exit(0);

else{

printf("Введите длину списка\n");

scanf("%d", &l);

printf("Введите название, автора композиции и год создания\n");

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

head = createMusicalComposition(name, author, year);

for (i=0;i<l-1;i++) {

printf("Введите название, автора и год создания\n");

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

push(head, createMusicalComposition(name, author, year));

}

}

}

else

switch(a){ *//реализуем пункты меню с помощью переключателя*

case 1:{  *//ввод нового элемента*

printf("Введите название, автора композиции и год создания\n");

scanf("%s%s%d", name, author, &year); *//добавление элемента в список*

push(head, createMusicalComposition(name, author, year));

break;

}

case 2:{ *//удаление элемента*

printf("Введите название нужной композиции\n");

scanf("%s", name);

removeEl(head, name);

break;}

case 3:{ *//вывод кол-ва элементов*

int c = count(head);

printf("%d - количество композиций в списке\n", c);

break;

}

case 4:{ *//вывод имен*

print\_names(head);

break;

}

case 5:{ *//реализация первой части индивидуального задания*

int m;

int n;

printf("Введите m и n\n");

scanf("%d %d",&n, &m);

ChangeList2(head,n,m);

print\_list(head); *//вывод функции*

break;

}

case 6:{ *//реализация второй части индивидуального задания*

head = ChangeList(head);

break;

}

case 7: { *//выход из программы*

exit(0);

}

}

printf("Если вы хотите продолжить работу, введите 1, если хотите выйти, введите 0\n");

scanf("%d", &a);

if(a == 0) break;  *//если a=0 выход из программы, иначе продолжается цикл*

}

return 0; }

## *Makefile*

Создаем объектные файлы функций, после компилируем их вместе, затем очищаем объектные файлы после сборки проекта. В отдельные файлы вынесем общие функции для работы со списком (funk.c), функции по индивидуальному заданию (funk2.c) и главную функцию с меню (main.c). Для каждого из этих файлов создадим заголовочные файлы \*.h c прототипами соответствующих функций и заголовочный файл с описанием структуры MusicalComposition.

main: struct.h main.o funk.o funk2.o

gcc main.o -o main funk.o funk2.o

rm \*.o

main.o: main.c struct.h

gcc -c main.c

funk.o: funk.c funk.h struct.h

gcc -c funk.c

funk2.o: funk2.c funk2.h struct.h

gcc -c funk2.c

## *Работа с GitHub:*

* Создадим новую ветку git checkout –b “Prozorova\_kurswork”
* Создадим новую папку mkdir kurswork
* Добавим файлы проекта в данную папку
* Подтвердим изменения git add Prozorova\_kurswork
* Создадим коммит git commit –m “#Prozorova\_kurswork kurswork added”
* Отправим изменения на удаленный репозиторий git push origin

# **ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ**

|  |  |  |
| --- | --- | --- |
| **Функция** | **Ввод** | **Результат** |
| Создание списка | Sprinter Kalafina 2015  moon AbneyPark 2010  oblivious Kalafina 2011  canon Okawari 2016  cold Kasabian 2013  train Osbourn 2013  dark Komori 2011  vilify Device 2015  immortalized Disturbed 2011  ego Oomph 2014 | - |
| Удаление элемента | cold | Sprinter Kalafina 2015  moon AbneyPark 2010  oblivious Kalafina 2011  canon Okawari 2016  train Osbourn 2013  dark Komori 2011  vilify Device 2015  immortalized Disturbed 2011  ego Oomph 2014 |
| Добавление элемента | cold Kasabian 2013 | Sprinter Kalafina 2015  moon AbneyPark 2010  oblivious Kalafina 2011  canon Okawari 2016  train Osbourn 2013  dark Komori 2011  vilify Device 2015  immortalized Disturbed 2011  ego Oomph 2014  cold Kasabian 2013 |
| Разделение элемента на две части относительно n, разделение второй части на группы по m и замена в них первого и последнего элемента местами | 4  3 | Sprinter Kalafina 2015  moon AbneyPark 2010  oblivious Kalafina 2011  canon Okawari 2016  vilify Device 2015  dark Komori 2011  train Osbourn 2013  cold Kasabian 2013 ego Oomph 2014 immortalized Disturbed 2011 |
| Копирование первой половины списка и вставка ее в середину |  | Sprinter Kalafina 2015  moon AbneyPark 2010  oblivious Kalafina 2011  canon Okawari 2016  vilify Device 2015  Sprinter Kalafina 2015  moon AbneyPark 2010  oblivious Kalafina 2011  canon Okawari 2016  vilify Device 2015  dark Komori 2011  train Osbourn 2013  cold Kasabian 2013 ego Oomph 2014 immortalized Disturbed 2011 |

# **ЗАКЛЮЧЕНИЕ**

При выполнении данной работы мы закрепили на практике написание функций создания одно- и двунаправленных линейных списков, а также API для работы с ними, а именно функций выврда списка, удаления и добавления элемента, подсчета кол-ва элементов и т.д.. В работе использовались файлы исходного кода(\*.c), заголовочные файлы(\*.h), makefile. Также продемонстрированы основные этапы работы с Git.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство "Невский Диалект", 2001. 352 с.

# **ПРИЛОЖЕНИЕ**

## ***Файл main.c***

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include"struct.h"

#include"funk.h"

#include"funk2.h"

int main(){

int a = 0;

int l;

int year;

char name[80];

char author[80];

printf("Создание списка\nВведите длину списка\n");

scanf("%d", &l);

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

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

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

int i;

for(i=0;i<l;i++){

printf("Введите название, автора композиции и год создания\n");

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

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);

years[i]=year;

}

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

for(i=0; i<l; i++){

free(names[i]);

free(authors[i]);

}

free(years);

while(1==1){

printf("1)Добавление нового элемента в конец списка;\n2)Удаление определенного элемента;\n3)Вывод длины списка;\n4)Вывод названий композиций;\n5)Разделить список на две части относительно индекса n, вторую часть разделить на группы по m элементов, соединить элементы обратно в список;\n6)Разделись список на две равные части(или же индекс центрального элемента округлить в большую сторону), скопировать первую часть и вставить в середину\n7)Выход\n");

scanf("%d", &a);

if(head==0){

while((a != 1) || (a != 0)){

printf("Если хотите продолжить работу, введите 1, если хотите выйти, введите 0\n");

scanf("%d", &a);

}

if(a == 0) exit(0);

else{

printf("Введите длину списка\n");

scanf("%d", &l);

printf("Введите название, автора композиции и год создания\n");

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

head = createMusicalComposition(name, author, year);

for (i=0;i<l-1;i++) {

printf("Введите название, автора и год создания\n");

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

push(head, createMusicalComposition(name, author, year));

}

}

}

else

switch(a){

case 1:{

printf("Введите название, автора композиции и год создания\n");

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

push(head, createMusicalComposition(name, author, year));

break;

}

case 2:{

printf("Введите название нужной композиции\n");

scanf("%s", name);

removeEl(head, name);

break;}

case 3:{

int c = count(head);

printf("%d - количество композиций в списке\n", c);

break;

}

case 4:{

print\_names(head);

break;

}

case 5:{

int m;

int n;

printf("Введите m и n");

scanf("%d %d",&n, &m);

ChangeList2(head,n,m);

print\_list(head);

break;

}

case 6:{

head = ChangeList(head);

break;

}

case 7: {

exit(0);

}

}

printf("Если вы хотите продолжить работу, введите 1, если хотите выйти, введите 0\n");

scanf("%d", &a);

if(a == 0) break;

}

return 0;

}

## ***Файл funk.c***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "struct.h"

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

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

strcpy(head->name, name);

strcpy(head->author, author);

head->year = year;

head->prior=NULL;

head->next=NULL;

return head;

}

//создание списка

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

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

MusicalComposition\* head = tmp;

int i;

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

tmp->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

tmp->next->prior = tmp;

tmp = tmp->next;

}

return head;

}

//вставка элемента в конец списка

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

while(head->next){

head=head->next;

}

head->next=element;

element->prior=head;

}

////удаление определенного элемента

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

MusicalComposition \*tmp = head;

while (tmp->next){

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

if ((tmp->prior) == NULL){

MusicalComposition \*ntmp = tmp->next;

strcpy(tmp->name, ntmp->name);

strcpy(tmp->author, ntmp->author);

tmp->year = ntmp->year;

ntmp->next->prior = tmp;

tmp->next = ntmp->next;

free(ntmp);

}

else{

tmp->prior->next = tmp->next;

tmp->next->prior = tmp->prior;

free(tmp);

}

}

tmp = tmp->next;

}

if ((strcmp(tmp->name, name\_for\_remove) == 0) && (tmp->next == NULL)){

tmp->prior->next = NULL;

free(tmp);

}

}

//кол-во элементов

int count(MusicalComposition\* head){

int count=0;

while (head){

count++;

head=head->next;

}

return count;

}

//вывод имен

void print\_names(MusicalComposition\* head){

MusicalComposition \*tmp = head;

while (tmp->next){

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

tmp = tmp->next;

}

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

}

//вывод списка

void print\_list(MusicalComposition\* head){

while (head->next){

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

printf("%s ", head->author);

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

head = head->next;

}

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

printf("%s ", head->author);

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

}

## ***Файл funk2.c***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "struct.h"

MusicalComposition\* ChangeList(MusicalComposition\* head){

MusicalComposition\* current = head;

int l = count(head);

int i;

float n;

if (l%2==0) n=l/2;

else n=l/2+0.5;

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

MusicalComposition\* curr = tmp;

for (i=0; i<n; i++){

strcpy(tmp->name, head->name);

strcpy(tmp->author, head->author);

tmp->year = head->year;

head=head->next;

tmp=tmp->next;

}

MusicalComposition\* rubb = head;

head = curr;

tmp = rubb;

return current;

}

//поменять местами элементы

void exchange(MusicalComposition\* a, MusicalComposition\* b){

char rubbish[80];

int rubb;

strcpy(rubbish, a->name);

strcpy(a->name, b->name);

strcpy (b->name, rubbish);

strcpy(rubbish, a->author);

strcpy(a->author, b->author);

strcpy (b->author, rubbish);

rubb=a->year;

a->year=b->year;

b->year=rubb;

}

//разбить список на группы и поменять в нем начальный и конечный элементы местами

MusicalComposition\* ChangeList1(MusicalComposition\* head, int m){

int i=0;

int j=0;

int n=count(head);

MusicalComposition\*\* arr = (MusicalComposition\*\*)malloc((n+1)\*sizeof(MusicalComposition\*));

for (i = 0; i<n; i++) arr[i] = (MusicalComposition\*)malloc(sizeof(MusicalComposition\*));

for (i = 0; i<n; i++){

arr[i] = head;

head = head->next;

}

if (n%m==0){

for(i = 0; i<n-1; i = i + m){

arr[i+m-1]->next = NULL;

arr[i]->prior = NULL;

exchange(arr[i], arr[i+m-1]);

}

for (i = 1; i<n-1; i = i + m){

arr[i+m-2]->next = arr[i+m-1];

arr[i+1]->prior = arr[i];

}

}

else {

for(i = 0; i<n-2; i = i + m){

arr[i+m-1]->next = NULL;

arr[i]->prior = NULL;

exchange(arr[i], arr[i+m-1]);

}

for (i = 1; i<n-2; i = i + m){

arr[i+m-2]->next = arr[i+m-1];

arr[i+1]->prior = arr[i];

}

i=n-2;

arr[i+1]->next=NULL;

arr[i+1]->prior=arr[i];

}

head = arr[0];

return head;

}

//разбить список на две части относительно индекса n, вторую часть разбить группы по m элементов, поменять в каждой начальный и конечный элемент местами, соединить обратно

MusicalComposition\* ChangeList2(MusicalComposition\* head, int n, int m){

MusicalComposition \*tmp = head;

int j;

for(j=0;j<(n-1);j++) tmp=tmp->next;

MusicalComposition \*curr=tmp->next;

tmp->next=NULL;

ChangeList1(curr, m);

tmp->next=curr;

return head;

}

## *Файл struct.h*

struct MusicalComposition {

char name[80];

char author[80];

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prior;

};

typedef struct MusicalComposition MusicalComposition;

## *Файл funk.h*

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);

void print\_list(MusicalComposition\* head);

## *Файл funk2.h*

MusicalComposition\* ChangeList(MusicalComposition\* head);

void exchange(MusicalComposition\* a, MusicalComposition\* b);

MusicalComposition\* ChangeList1(MusicalComposition\* head, int m);

MusicalComposition\* ChangeList2(MusicalComposition\* head, int n, int m);