**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «ПРОГРАММИРОВАНИЕ»**

Тема: ЛИНЕЙНЫЕ СПИСКИ

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6303 |  | Матюшина М.Е. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студентка Матюшина М.Е. | | |
| Группа 6303 | | |
| Тема работы: Линейные списки | | |
|  | | |
| Содержание пояснительной записки:   * Содержание * Введение * Описание функций для работы с двунаправленным линейным списком * Описание функции сортировки двунаправленного линейного списка * Примеры работы программы * Заключение * Список использованных источников * Приложение А. Исходный код программы | | |
| Предполагаемый объем пояснительной записки:  Не менее 10страниц. | | |
| Дата выдачи задания: 02.12.2016 | | |
| Дата сдачи реферата: 29.12.2016 | | |
| Дата защиты реферата: 29.12.2016 | | |
| Студентка |  | Матюшина М.Ю. |
| Преподаватель |  | Берленко Т.А. |

**Аннотация**

В ходе работы была реализована программа на языке программирования С, с помощью которой создается двунаправленный линейный список, содержащий сведения о музыкальных композициях, а также функции для работы с ним, позволяющие добавлять, удалять, сортировать и выводить элементы списка и их количество. Также описана структура элемента списка. Приведено полное описание исходного кода.

**содержание**

|  |
| --- |
| [**введение** 5](#_Toc470082407)  [**1.** **функции для работы с двунаправленным линейным списком** 6](#_Toc470082408)  [1.1. Создание элемента 6](#_Toc470082409)  [Создание двунаправленного списка связанных элементов 6](#_Toc470082410)  [1.2. Добавление элемента в конец списка 6](#_Toc470082411)  [1.3. Удаление элементов из списка 7](#_Toc470082413)  [1.4. Подсчет количества элементов в списке 7](#_Toc470082414)  [1.5. Вывод названий музыкальных композиций в консоль 8](#_Toc470082415)  [**2.** **функция ДОБАВЛЕНИЯ ЭЛЕМЕНТОВ СПИСКА** 8](#_Toc470082417)  [2.1. Добавление 3-х элементов в середину списка ……..](#_Toc470082418)  [**3.** **примеры работы программы** ……………………...**Ошибка! Закладка не определена.**](#_Toc470082420)  [**заключение** 11](#_Toc470082421)  [**список использованных источников** 12](#_Toc470082422)  [**приложение А** 13](#_Toc470082423)  [**Исходный код программы** 13](#_Toc470082424) |
|  |
|  |
|  |

# **введение**

В ходе работы необходимо создать двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication****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.

ссо

Функции, необходимые для работы программы

* Создание элемента типа MusicalComposition
* Создание двунаправленного списка связанных элементов типа MusicalComposition
* Добавление элемента в конец списка
* Удаление определенного элемента (элементов) из списка
* Подсчет количества элементов в списке
* Вывод элементов списка в консоль
* Добавление 3-х элементов в середину списка

Описание структуры MusicalComposition.

typedef struct MusicalComposition {

char\* name;

char\* author;

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

} MusicalComposition;

MusicalComposition \* head; //создаем глобальную переменную head – первый элемент списка

# 

# **1.функции для работы с двунаправленным линейным списком**

## Создание элемента

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

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

music->name = name;

music->author = author;

music->year = year;

music->next = NULL;

music->prev = NULL;

return music;

}

Функция принимает в качестве аргументов указатели на название композиции (**char**\*name), ее автора (**char**\* author), а также год написания (**int** year). Происходит выделение памяти для структуры типа MusicalComposition и заполняются ее переменные.

## Создание двунаправленного списка связанных элементов

|  |
| --- |
|  |

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

head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);

MusicalComposition \*tmp = head;

int i;

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

MusicalComposition \* pp = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

tmp->next = pp;

pp->prev = tmp;

tmp = tmp->next;

}

return;

}

Функция принимает в качестве аргументов указатели на указатель на массив названий композиций (**char**\*\* array\_names) и их авторов (**char**\*\* array\_authors), указатель на массив лет написания (**int\***array\_years), размер массива (**int** n). Сначала создается “голова” списка (первый элемент) и происходит заполнение ее переменных данными, затем аналогичные операции происходят для всех последующих n-1 элементов списка

## Добавление элемента в конец списка

MusicalComposition \* push( MusicalComposition\* element) {

MusicalComposition \* tmp = head;

if (tmp->next!=NULL){

while ( tmp->next != NULL) {

tmp = tmp->next;

}

}

tmp->next = element;

element->prev= tmp;

element->next=NULL;

return head;

}

Функция принимает в качестве аргументов указатель на элемент списка (MusicalComposition\* element), который нужно добавить в конец списка. В первую очередь происходит поиск последнего элемента списка, далее новый элемент вставляется после текущего.

## Удаление элементов из списка

Функция принимает в качестве аргумента указатель на название композиции (**char**\*name\_for\_remove) и производит удаление из списка всех элементов с таким названием.

void removeEl( char\* name\_for\_remove){

MusicalComposition \*pp;

pp=head;

MusicalComposition \*tmp;

tmp=head;

MusicalComposition\* next;

MusicalComposition\* prev;

while ( pp!=NULL){

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

tmp=head;

while (\*tmp->name!=\*name\_for\_remove) {

tmp=tmp->next;

}

if (tmp==head) {

head=head->next;

}

else if(tmp->next==NULL) {

prev=tmp->prev;

prev->next=NULL;

}

else{

next=tmp->next;

prev=tmp->prev;

next->prev=prev;

prev->next=next;

}

}

pp=pp->next;

}

}

Функция принимает в качестве аргументов указатель на имя элемента, который нужно удалить (char\* name\_for\_remove). Далее имена элементов списка сравниваются с name\_for\_remove с помощью функции strcmp. В случае совпадения, указатель на предыдущий элемент присваивается предыдущему, а на следующий - следующему (если элемент не крайний). Если удалиться должен первый элемент, то head - ом становится следующий элемент, а текущий удаляется. Если удалить нужно последний элемент списка, то указатель переходит на предыдущий элемент, а указатель на следующий становится NULL.

## Подсчет количества элементов в списке

Функция производит подсчет элементов в списке, пока следующий элемент существует. Возвращает количество элементов списка.

## int count() {

## MusicalComposition \* tmp = head;

## int count = 0;

## while (tmp != NULL) {

## ++count;

## tmp = tmp->next;

## }

## return count;

}

## Вывод названий музыкальных композиций в консоль

Функция производит вывод на консоль названий музыкальных композиций, пока следующий элемент списка существует.

void print\_names() {

MusicalComposition \* tmp = head;

while (tmp != NULL) {

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

tmp = tmp->next;

}

if(tmp==NULL) { printf("" );}

}

# **функция добавления 3-х элементов в середину списка**

MusicalComposition \* PUSH(MusicalComposition\* element1, MusicalComposition\* element2, MusicalComposition\* element3) {

MusicalComposition \* tmp = head;

int k=count();

int schet=0;  
while(schet!=(k/2-1)) {  
 tmp=tmp->next;  
 ++schet;  
}   
MusicalComposition\* second = tmp->next ;  
tmp->next = element1;  
element1->prev=tmp;  
element1-> next = element2;  
element2->prev= element1;  
  
element2->next = element3;  
element3->prev= element2;  
element3->next=second;  
second->prev=element3;

return head;

}

Функция принимает в качестве аргументов указатели элементы списка ( MusicalComposition \* element1, MusicalComposition\* element2, MusicalComposition\* element3), которы12:32 нужно добавить в середину списка. В первую очередь происходит поиск середины списка, для этого заводится переменная schet, значение которой становится равным длине списка, когда указатель tmp( изначально tmp=head) доходит до его конца. Далее указатель темп возвращается к середине списка, делая (schet/2+1) шаг, затем новые элементы вставляется после центрального. Далее последний из добавленных элементов связывается с второй половиной первоначального списка.

1. **Тестирование функций**

г

Для демонстрации функций push, removeEl, count, print\_names используется следующий формат ввода:

* Количество элементов в списке
* Элементы списка
* Элемент для добавления в список
* Название композиции, которую нужно удалить
* Три элемента, добавляемые в середину списка

Формат вывода представлен ниже:

* Название, автор и год написания первого элемента списка
* Количество элементов до добавления 3-х элементов в середину
* Элементы списка после добавления 3-х в середину
* Количество элементов после добавления 3-х элементов в середину
* Количество элементов после добавления элемента в конец списка
* Названия всех композиций после удаления
* Количество элементов после удаления

|  |  |  |
| --- | --- | --- |
| № | ВВОд | вЫВОД |
| 1 | 2 pesnya1 author1 2000 pesnya2 author2 2000 pesnya\_v\_end  author\_end 2001 pesnya\_remove pesnya\_middle1  author\_middle1 1991 pesnya\_middle2 author\_middle2 1992 pesnya\_middle3 author\_middle3 1993 | pesnya1 author1 2000 2 pesnya1 author1 2000 pesnya\_middle1 author\_middle1 1991 pesnya\_middle2 author\_middle2 1992 pesnya\_middle3 author\_middle3 1993 pesnya2 author2 2000 5 6 pesnya1 pesnya\_middle1 pesnya\_middle2 pesnya\_middle3 pesnya2 pesnya\_v\_end  6  Без удаления элементов |
| 2 | 2 pesnya author 2000 pesnya author 2000 pesnya author\_end 2000 pesnya pesnya author\_middle1 1991 pesnya author\_middle2 1992 pesnya author\_middle3 1993 | pesnya author 2000 2 pesnya author 2000 pesnya author\_middle1 1991 pesnya author\_middle2 1992 pesnya author\_middle3 1993 pesnya author 2000 5 6 0  Удаление всех элементов |
| 3 | 4 AAA pops 1 BBB pops 2 CCC pops3 3 DDD pops 4 AAA pops 1 AAA EEE ops 1 EEE ops  2 EEE ops 3 | AAA pops 1 4 AAA pops 1 BBB pops 2 EEE ops 1 EEE ops 2 EEE ops 3 CCC pops3 3 DDD pops 4 7 8 BBB EEE EEE EEE CCC DDD 6  Удаление первого элемента и элемента, добавленного в конец. |
| 4 | 4 AAA pop 1 BBB pop 2 CCC pop 3 DDD pop 4 END  end 10 DELETE LLL ops 1 DELETE ops 2 LLL ops 3 | AAA pop 1 4 AAA pop 1 BBB pop 2 LLL ops 1 DELETE ops 2 LLL ops 3 CCC pop 3 DDD pop 4 7 8 AAA BBB LLL LLL CCC DDD END  7 |

# 

# **заключение**

В ходе работы была реализована программа на языке Си для работы с двунаправленным линейным списком музыкальных композиций. В частности, был создан и описан сам двунаправленный список музыкальных композиций MusicalComposition и разработан api для работы со списком. Также были созданы и описаны все необходимые функции.

# **список использованных источников**

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство"Невский Диалект", 2001. 352 с.
2. Моли Б. UNIX/Linux: теория и практика программирования.М.: КУДИЦ-ОБРАЗ, 2004. 576 с.

# **приложение А**

# **Исходный код программы**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition {

char\* name;

char\* author;

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

} MusicalComposition;

MusicalComposition \* head;

// Создание структуры MusicalComposition

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

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

music->name = name;

music->author = author;

music->year = year;

music->next = NULL;

music->prev = NULL;

return music;

}

// Функции для работы со списком MusicalComposition

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

head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);

MusicalComposition \*tmp = head;

int i;

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

MusicalComposition \* pp = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

tmp->next = pp;

pp->prev = tmp;

tmp = tmp->next;

}

return;

}

MusicalComposition \* push( MusicalComposition\* element) {

MusicalComposition \* tmp = head;

if (tmp->next!=NULL){

while ( tmp->next != NULL) {

tmp = tmp->next;

}

}

tmp->next = element;

element->prev= tmp;

element->next=NULL;

return head;

}

void removeEl( char\* name\_for\_remove){

MusicalComposition \*pp;

pp=head;

MusicalComposition \*tmp;

tmp=head;

MusicalComposition\* next;

MusicalComposition\* prev;

while ( pp!=NULL){

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

tmp=head;

while (\*tmp->name!=\*name\_for\_remove) {

tmp=tmp->next;

}

if (tmp==head) {

head=head->next;

}

else if(tmp->next==NULL) {

prev=tmp->prev;

prev->next=NULL;

}

else{

next=tmp->next;

prev=tmp->prev;

next->prev=prev;

prev->next=next;

}

}

pp=pp->next;

}

}

int count() {

MusicalComposition \* tmp = head;

int count = 0;

while (tmp != NULL) {

++count;

tmp = tmp->next;

}

return count;

}

void print\_names() {

MusicalComposition \* tmp = head;

while (tmp != NULL) {

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

tmp = tmp->next;

}

if(tmp==NULL) { printf("" );}

}

MusicalComposition \* PUSH(MusicalComposition\* element1, MusicalComposition\* element2, MusicalComposition\* element3) {

MusicalComposition \* tmp = head;

int k=count();

int schet=0;

while(schet!=(k/2-1)) {

tmp=tmp->next;

++schet;

}

MusicalComposition\* second = tmp->next ;

tmp->next = element1;

element1->prev=tmp;

element1-> next = element2;

element2->prev= element1;

element2->next = element3;

element3->prev= element2;

element3->next=second;

second->prev=element3;

return head;

}

int main(){

int length;

scanf("%d\n", &length);

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

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

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

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

char name[80];

char author[80];

fgets(name, 80, stdin);

fgets(author, 80, stdin);

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

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

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

names[i] = (char\*)malloc(sizeof(char\*) \* (strlen(name)+4));

authors[i] = (char\*)malloc(sizeof(char\*) \* (strlen(author)+4));

strcpy(names[i], name);

strcpy(authors[i], author);

}

createMusicalCompositionList(names, authors, years, length);

char name\_for\_push1[80];

char author\_for\_push1[80];

int year\_for\_push1;

char name\_for\_push2[80];

char author\_for\_push2[80];

int year\_for\_push2;char name\_for\_push3[80];

char author\_for\_push3[80];

int year\_for\_push3;

char name\_for\_push[80];

char author\_for\_push[80];

int year\_for\_push;

char name\_for\_remove[80];

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

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

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

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

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

fgets(name\_for\_remove, 80, stdin);

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

fgets(name\_for\_push1, 80, stdin);

fgets(author\_for\_push1, 80, stdin);

fscanf(stdin, "%d\n", &year\_for\_push1);

(\*strstr(name\_for\_push1,"\n"))=0;

(\*strstr(author\_for\_push1,"\n"))=0;

fgets(name\_for\_push2, 80, stdin);

fgets(author\_for\_push2, 80, stdin);

fscanf(stdin, "%d\n", &year\_for\_push2);

(\*strstr(name\_for\_push2,"\n"))=0;

(\*strstr(author\_for\_push2,"\n"))=0;

fgets(name\_for\_push3, 80, stdin);

fgets(author\_for\_push3, 80, stdin);

fscanf(stdin, "%d", &year\_for\_push3);

(\*strstr(name\_for\_push3,"\n"))=0;

(\*strstr(author\_for\_push3,"\n"))=0;

MusicalComposition\* element\_for\_push1 = createMusicalComposition(name\_for\_push1, author\_for\_push1, year\_for\_push1);

MusicalComposition\* element\_for\_push2 = createMusicalComposition(name\_for\_push2, author\_for\_push2, year\_for\_push2);

MusicalComposition\* element\_for\_push3 = createMusicalComposition(name\_for\_push3, author\_for\_push3, year\_for\_push3);

MusicalComposition\* element\_for\_push = createMusicalComposition(name\_for\_push, author\_for\_push, year\_for\_push);printf("%s %s %d\n", head->name, head->author, head->year);

int k = count();

printf("%d\n", k);

PUSH(element\_for\_push1, element\_for\_push2, element\_for\_push3);

int el;

MusicalComposition\* t = head;

for(el=0; el<(length+3); ++el) {

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

t=t->next;

}

k = count();

printf("%d\n", k);

push(element\_for\_push);

k = count();

printf("%d\n", k);

removeEl( name\_for\_remove);

print\_names();

k = count();

printf("%d\n", k);

return 0;

}