**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Двунаправленные линейные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Ковынев М.В. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

# **Аннотация**

Создание двунаправленного линейного списка, а также API для работы с ним, средствами языка С. В частности, с использованием сложных типов, указателей на них и работы с динамической памятью. Реализация стандартных операций таких как: вставка, удаление, создание элемента списка, поиск по значениям полей и т.д. Работа с проектом в целом: разбиение его на заголовки (header files) и файлы кода (source files), а также написание makefile для сборки программы.

**Содержание**

[Аннотация 2](#_Toc469166413)

[Введение 4](#_Toc469166414)

[Цель работы 4](#_Toc469166415)

[Формулировка задачи 4](#_Toc469166416)

[Индивидуальное задание: 5](#_Toc469166417)

[Решение задачи 5](#_Toc469166418)

[1. Создание структуры данных: 5](#_Toc469166419)

[2. Функция создания нового экземпляра данного типа: 5](#_Toc469166420)

[3. Функция создания списка из элементов: 6](#_Toc469166421)

[4. Функция добавления элемента в конец списка: 6](#_Toc469166422)

[5. Функция удаления элемента: 7](#_Toc469166423)

[6. Функция подсчета элементов списка: 7](#_Toc469166424)

[7. Функция печати имен: 8](#_Toc469166425)

[8. Функция сортировки: 8](#_Toc469166426)

[9. Функция удаления нечетных элементов списка: 9](#_Toc469166427)

[10. Makefile: 9](#_Toc469166428)

[Заключение 11](#_Toc469166429)

# **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. Отсортировать список по году, по возрастанию (не вывести, а именно поменять местами, если придется, элементы списка).
2. Удалить все нечетные элементы списка.

**Решение задачи**

1. *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

};

typedef struct MusicalComposition MusicalComposition; //Объявление нового типа данных

1. *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;

}

1. *Функция создания списка из элементов*

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;

/\* Заполенение элментов\*/

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;

}

return head;

}

1. *Функция добавления элемента в конец списка*

С помощью указателя на новый элемент, который был передан в функцию, свяжем его с последним в списке.

/\*Функция добавления элемента в конец списка\*/

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

{

/\* Двигаем указатель до последнего элемента \*/

while (head->next)

head = head->next;

head->next = element;

element->prev = head;

element->next = NULL;

}

1. *Функция удаления элемента*

Найдем нужный элемент и изменим связь с предыдущим и следующим.

/\*Функция удаления элемента\*/

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

{

/\* Двигаем указатель до последнего элемента \*/

while (head)

{

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

{

/\* Если элемент, который надо удалить последний то разрываем связь с предыдущим \*/

if (head->next==NULL)

{

head->prev->next=NULL;

return;

}

/\* Если элемент, который надо удалить первый то разрываем связь со следующим\*/

else if (head->prev==NULL)

{

head->next->prev=NULL;

return;

}

/\* Меняем значения поля next у предыдущего элемента \*/

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

/\* Меняем значение поля prev у следующего элемента \*/

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

break;

}

head=head->next;

}

}

1. *Функция подсчета элементов списка*

/\*Функция подсчета элементов списка:\*/

int count(MusicalComposition\*head)

{

/\*Переменная count - счетчик\*/

int count = 0;

/\* Двигаем указатель до последнего элемента \*/

while (head)

{

count++;

head = head->next;

}

return count;

};

1. *Функция печати имен*

/\*Функция печати имен\*/

void print\_names(MusicalComposition \*head)

{

/\* Двигаем указатель до последнего элемента \*/

while (head)

{

/\*Печать имени\*/

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

head = head->next;

}

}

## *Функция сортировки*

Сортируем данные списка методом пузырька, обменивая данные 2 элементов между собой, используя указатели.

/\*Функция сортировки\*/

void sort(MusicalComposition \*head)

{

MusicalComposition \*curr\_i = head;

MusicalComposition \*curr\_j;

/\* Двигаем указатель до последнего элемента \*/

while (curr\_i->next)

{

curr\_j = head;

while (curr\_j->next)

{

if (curr\_j->year>curr\_j->next->year)

{

/\*Выделяем память под переменную temp, необходимую для обмена значений\*/

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

/\*Обмениваеcмя данными с 2 элементами\*/

strcpy(temp->name, curr\_j->next->name); strcpy(temp->author, curr\_j->next->author); temp->year = curr\_j->next->year;

strcpy(curr\_j->next->name, curr\_j->name); strcpy(curr\_j->next->author, curr\_j->author); curr\_j->next->year = curr\_j->year;

strcpy(curr\_j->name, temp->name); strcpy(curr\_j->author, temp->author); curr\_j->year = temp->year;

/\*Особождаем память\*/

free(temp);

}

curr\_j = curr\_j->next;

}

curr\_i = curr\_i->next;

}

}

## *Функция удаления нечетных элементов списка*

Если номер элемента нечетен, то функции удаления передадим имя, элемент которого необходимо удалить.

/\*Функция удаления нечетных элементов списка\*/

void removeoddel(MusicalComposition \*head)

{

int i = 0;

/\* Двигаем указатель до последнего элемента \*/

while (head)

{

/\*Если номер элемента нечетен, то передаем функции removeEl имя элемента\*/

if (((++i) % 2) == 0) removeEl(head, head->name);

head = head->next;

}

}

1. *Makefile*

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

# Главная цель

Kursach: themainesthnik.h sort.o removeoddel.o main.o print\_names.o count.o push.o createList.o

gcc main.o -o Kursach\_Main.out sort.o removeoddel.o print\_names.o count.o push.o createList.o

rm \*.o

# Объектный файл для функции sort

sort.o: sort.c sort.h struct.h

gcc -c sort.c

# Объектный файл для функции removeodderl

removeoddel.o: removeoddel.c removeoddel.h struct.h

gcc -c removeoddel.c

# Объектный файл для функции print\_names

print\_names.o: print\_names.c print\_names.h struct.h

gcc -c print\_names.c

# Объектный файл для функции count

count.o: count.c count.h struct.h

gcc -c count.c

# Объектный файл для функции main

main.o: main.c themainesthnik.h

gcc -c main.c

# Объектный файл для функции push

push.o: push.c push.h struct.h

gcc -c push.c

# Объектный файл для функции createList

createList.o: createList.c createList.h struct.h

gcc -c createList.c

# **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.