

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

Кафедра вычислительной техники

Отчет по лабораторной работе № 12  
по дисциплине «Программирование»  
Тема: «Двусвязный кольцевой  
список».

Студент гр. 9305

Леванович Р.О.

Преподаватель

Пересева Ю. В.

Санкт-Петербург

2020

## Содержание

<a href="#"><u>Введение</u></a> .....	2
<a href="#"><u>Задание</u></a> .....	2
<a href="#"><u>Описание структур</u></a> .....	3
<a href="#"><u>Схема вызова функций</u></a> .....	5
<a href="#"><u>Функций</u></a> .....	7
<a href="#"><u>Заключение</u></a> .....	33

## **Введение**

Получить практические навыки в разработке алгоритма и написании программы на языке Си. Для ознакомления работы с двусвязными кольцевыми списками, а также правилами их написания на языке Си.

## **Цель**

Получить практические навыки в разработке алгоритма и написании программы на языке Си для работы с двусвязным кольцевым списком.

## **Задание**

Разработать подалгоритм удаления в двусвязном кольцевом списке предпоследнего элемента. При недостаточном количестве элементов в списке вывести соответствующее сообщение.

## **Постановка задачи и описание решения**

Имеется файл, содержащий строки данных, которые должны стать значениями полей структур, на каждой строке — новая структура. Значения полей на одной строке отделены друг от друга специальным символом, который будет индикатором окончания считывания значения одного поля структуры и сигналом к началу считывания значения другой. Элемент списка имеет указатель на следующий элемент и на предыдущий, что позволяет нам сделать двусвязный список. Указатель на следующий элемент последнего списка головы направлен на первый, что позволяет нам сделать двусвязный кольцевой список. Элемент списка делается с помощью ф-ии `create_node` и записываем его в список начиная с головы. Чтобы удалить предпоследний элемент, и соответственно решить задачу лабораторной была написана функция `delete_penultimate`

**Описание структуры данных**  
*animals*

Имя поля	Тип	Назначение
name	char*	Название животного
kind	char*	Царство, к которому оно принадлежит
lifespan	int	Продолжительность жизни(средняя)
weight	float	Средний вес
height	float	Средний рост

*Head*

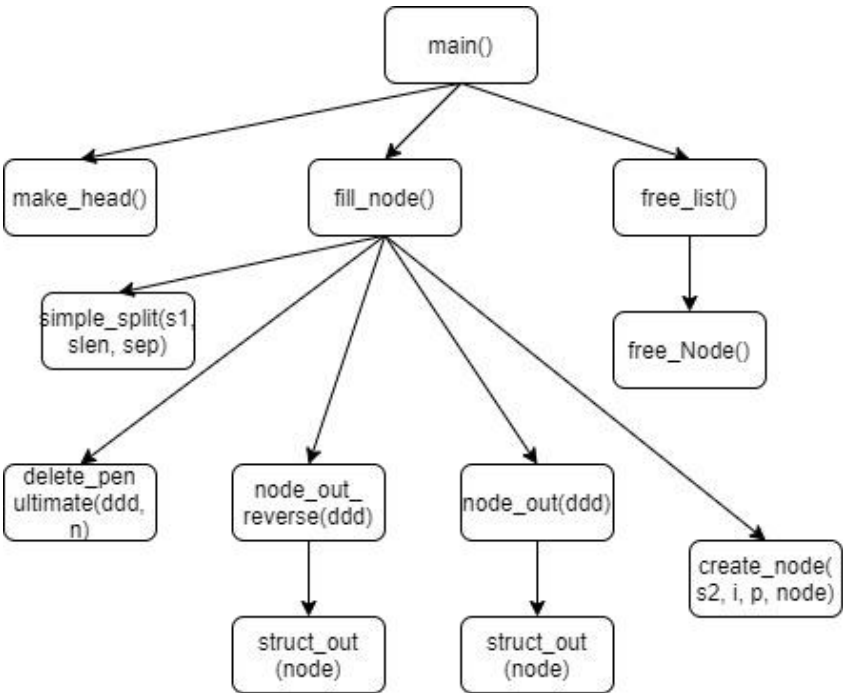
Имя поля	Тип	Назначение
cnt	int	Кол-во элементов
first	animals*	Первый элемент списка
last	animals*	Последний элемент списка

*Node*

Имя поля	Тип	Назначение
id	int	Номер элемента
structure	animals*	Структура внутри узла

next	Node*	Следующий элемент списка
prev	Node*	Предыдущий элемент списка

**Схемы вызова функций**



## Функции

### 1. fill\_node(Head \*ddd)

**Описание:** По сути тот же мейн, где совершаются все операции, начиная от получения текста из файла, заканчивая выводом списка.

#### Прототип:

```
void fill_node(Head *ddd)
```

#### Примеры вызова:

```
fill_node (ddd)
```

#### Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Локальная переменная	node	Node*	Элемент списка
Локальная переменная	p	Node*	Список
Локальная переменная	ddd	Head*	Голова списка
Локальная переменная	slen	int	Длина строки
Локальная переменная	n	int	Кол-во элементов
Локальная переменная	count	int	Номер структуры
Локальная переменная	choice	int	Переменная-выбор
Локальная переменная	s2	char**	Массив строк, каждая из которых служит элементом структуры
Локальная переменная	s1	char	Строка из элементов структуры

Локальная переменная	sep	char	Символ-разделитель
Локальная переменная	df	FILE*	Переменная для файла
Итератор	i	int	

**Возвращаемое значение:** отсутствует

**simple\_split**

**Описание:**

Функция получающая на вход строку, которую нужно разделить, возвращает массив строк из элементов структуры. Разделение по разделителю

**Прототип:**

char\*\* simple\_split(char \*str, int length, char sep)

**Примеры вызова**

simple\_split(s1, slen, sep)

**Описание переменных:**

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	str	char*	Строка для разделения по разделителю
Формальный аргумент	length	integer	Длина строки
Формальный аргумент	sep	char	Символ-разделитель
Локальная переменная	str_array	char**	Массив строк, получающийся из str
Локальная переменная	k	int	Помощь для цикла
Локальная переменная	m	int	Помощь для цикла
Локальная переменная	key	int	То же, что и булевая переменная
Локальная переменная	count	int	Итератор с сохраняемым значением

Итератор	i	int
Итератор	j	int

**Возвращаемое значение:** Массив строк получающийся из разделённой строки по символу-разделителю

## 2.print\_header

### Описание:

Вывод шапки для удобства чтения.

### Прототип:

```
void print_header ()
```

### Пример вызова:

```
print_header()
```

**Возвращаемое значение:** отсутствует

## 3.struct\_out

### Описание:

Выводит элементы структуры после шапки.

### Прототип:

```
void struct_out ()
```

### Пример вызова:

```
struct_out(node)
```

**Возвращаемое значение:** отсутствует.

## 4.create\_node

### Описание:

Создаёт элемент списка из структуры.

### Пример вызова:

```
node = create_node(s2, count)
```

### Описание переменных:



Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	s2	char**	Строка разделённая по разделителю
Формальный аргумент	i	int	Номер
Формальный аргумент	p	Node*	Элемент списка
Формальный аргумент	ddd	Head*	Голова списка
Локальная переменная	node	Node*	Готовый элемент

**Возвращаемое значение:** node

**5.make\_head**

**Описание:**

Создаёт голову списка

**Пример вызова:**

ddd = make\_head()

**Описание переменных:**

Вид переменной	Имя переменной	Тип	Назначение
Локальная переменная	ddd	Head*	Голова списка

**Возвращаемое значение:** ddd

**6.node\_out**

**Описание:**

Вид переменной	Имя переменной	Тип	Назначение
----------------	----------------	-----	------------

Формальный аргумент	head	Head*	Голова списка
Локальная переменная	node	Node*	Элемент списка

Выводит список

**Пример вызова:**

node\_out(ddd)

**7.node\_out\_reverse**

**Описание:**

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	head	Head*	Голова списка
Локальная переменная	node	Node*	Элемент списка

Выводит список наоборот

**Пример вызова:**

node\_out\_reverse(ddd)

## 7.delete\_penultimate

### Описание:

Удаление предпоследнего элемента списка

### Пример вызова:

delete\_penultimate(ddd, n,) **Описание**

### переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	ddd	Head*	Голова списка
Локальная переменная	p	Node*	Элемент списка
Формальный аргумент	n	int	Кол-во элементов списка
Локальная переменная	temp	Node*	Вспомогательный элемент списка

**Возвращаемое значение:** Отсутствует

## 8.free\_Node

### Описание:

Очистка определённого

узла

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	node	Node*	Элемент списка

### Пример вызова:

free\_node(ddd)

### Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	temp	Node*	Голова списка

**Возвращаемое значение:** Отсутствует

### Выводы:

При выполнении лабораторной работы были получены практические навыки в разработке алгоритма и написании программы на языке Си, а также получена информация о двусвязных списках.

Ссылка на github: <https://github.com/RitaStreet/labs/tree/master/lab11>