**Премахване на елемент (node)**

Премахванто на елемент(node) е по-лесно от вмъкването му, но изисква особен подход ако елемента(node) за премахване е първият(firstNode) или последният(lastnNode). :

function remove(List list, Node node)

if node.prev == null

list.firstNode := node.next

else

node.prev.next := node.next

if node.next == null

list.lastNode := node.prev

else

node.next.prev := node.prev

Коварното следвствие от горният пример при изтриване на последния елемент от списъка води до стойност null на firstNode и lastNode . Забележете че няма нужда от отделни методи "removeBefore" и "removeAfter" защото в двойно свъзаният списък можем просто да използваме "remove(node.prev)" или "remove(node.next)". Това също така гарантира, че елемента който се отстранява съществува. Ако елемента не съществува в списъка, ще възникне грешка която ще трябва да се обработи.

**Двойно свързани кръгли списъци**

**Присичане на списък :**

Ако приемем че, "someNode" е елемент в списък(не празен списък) следващия код пресича през списъка започвайки с "someNode"

**Напред:**

node := someNode

do

do something with node.value

node := node.next

while node ≠ someNode

**Назад :**

node := someNode

do

do something with node.value

node := node.prev

while node ≠ someNode

Забележете забавянето на проверката накрая на цикъла. Това е важно в случая когато списъка съдържа единичния елемент "somenode".

**Вмъкване на елемент (node)**

Тази функция вмъква node в двойно свързания кръгъл списък след зададен елемент:

function insertAfter(Node node, Node newNode)

newNode.next := node.next

newNode.prev := node

node.next.prev := newNode

node.next := newNode

**Вмъкване на елемент във вероятно празен списък изисква използването на по сложна функция.**

function insertEnd(List list, Node node)

if list.lastNode == null

node.prev := node

node.next := node

else

insertAfter(list.lastNode, node)

list.lastNode := node

**Премахването на елемент(node) става в случая когато списъка се опразва.**

function remove(List list, Node node)

if node.next == node

list.lastNode := null

else

node.next.prev := node.prev

node.prev.next := node.next

if node == list.lastNode

list.lastNode := node.prev;

destroy node