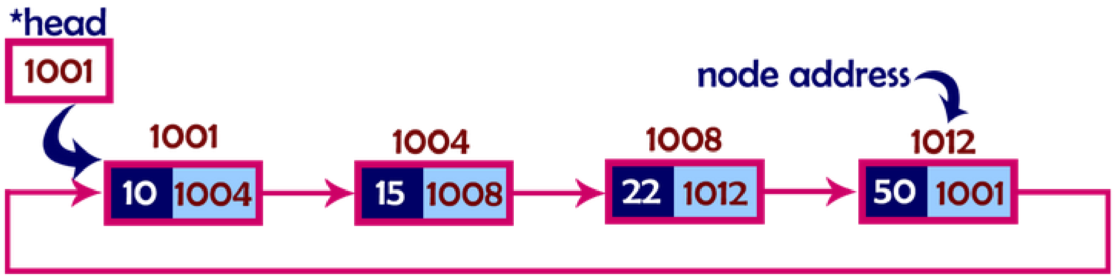
Circular Linked List

In single linked list, every node points to its next node in the sequence and the last node points NULL. But in circular linked list, every node points to its next node in the sequence but the last node points to the first node in the list.

A circular linked list is a sequence of elements in which every element has a link to its next element in the sequence and the last element has a link to the first element.

That means circular linked list is similar to the single linked list except that the last node points to the first node in the list.



Operations

In a circular linked list, we perform the following operations...

1. Insertion
2. Deletion
3. Display

Insertion

In a circular linked list, the insertion operation can be performed in three ways. They are as follows...

1. Inserting At Beginning of the list
2. Inserting At End of the list
3. Inserting At Specific location in the list

Inserting at Beginning of the list

We can use the following steps to insert a new node at beginning of the circular linked list...

- Step 1 - Create a **newNode** with given value.
- Step 2 - Check whether list is **Empty** (**head** == **NULL**)

- Step 3 - If it is **Empty** then, set **head** = **newNode** and **newNode→next** = **head** .
- Step 4 - If it is **Not Empty** then, define a Node pointer '**temp**' and initialize with '**head**'.
- Step 5 - Keep moving the '**temp**' to its next node until it reaches to the last node (until '**temp → next** == **head**').
- Step 6 - Set '**newNode → next** =**head**', '**head** = **newNode**' and '**temp → next** = **head**'.

Inserting at End of the list

We can use the following steps to insert a new node at end of the circular linked list...

- Step 1 - Create a **newNode** with given value.
- Step 2 - Check whether list is **Empty** (**head** == **NULL**).
- Step 3 - If it is **Empty** then, set **head** = **newNode** and **newNode → next** = **head**.
- Step 4 - If it is **Not Empty** then, define a node pointer **temp** and initialize with **head**.
- Step 5 - Keep moving the **temp** to its next node until it reaches to the last node in the list (until **temp → next** == **head**).
- Step 6 - Set **temp → next** = **newNode** and **newNode → next** = **head**.

Inserting at Specific location in the list (After a Node)

We can use the following steps to insert a new node after a node in the circular linked list...

- Step 1 - Create a **newNode** with given value.
- Step 2 - Check whether list is **Empty** (**head** == **NULL**)
- Step 3 - If it is **Empty** then, set **head** = **newNode** and **newNode → next** = **head**.
- Step 4 - If it is **Not Empty** then, define a node pointer **temp** and initialize with **head**.
- Step 5 - Keep moving the **temp** to its next node until it reaches to the node after which we want to insert the newNode (until **temp1 → data** is equal to **location**, here location is the node value after which we want to insert the newNode).
- Step 6 - Every time check whether **temp** is reached to the last node or not. If it is reached to last node then display **'Given node is not found in the list!!! Insertion not possible!!!'** and terminate the function. Otherwise move the **temp** to next node.
- Step 7 - If **temp** is reached to the exact node after which we want to insert the newNode then check whether it is last node (temp → next == head).
- Step 8 - If **temp** is last node then set **temp → next** = **newNode** and **newNode → next** = **head**.
- Step 8 - If **temp** is not last node then set **newNode → next** = **temp → next** and **temp → next** = **newNode**.

Deletion

In a circular linked list, the deletion operation can be performed in three ways those are as follows...

1. Deleting from Beginning of the list
2. Deleting from End of the list

3. Deleting a Specific Node

Deleting from Beginning of the list

We can use the following steps to delete a node from beginning of the circular linked list...

- Step 1 - Check whether list is **Empty** (**head** == **NULL**)
- Step 2 - If it is **Empty** then, display **'List is Empty!!! Deletion is not possible'** and terminate the function.
- Step 3 - If it is **Not Empty** then, define two Node pointers **'temp1'** and '**temp2**' and initialize both '**temp1**' and '**temp2**' with **head**.
- Step 4 - Check whether list is having only one node (**temp1 → next** == **head**)
- Step 5 - If it is **TRUE** then set **head** = **NULL** and delete **temp1** (Setting **Empty** list conditions)
- Step 6 - If it is **FALSE** move the **temp1** until it reaches to the last node. (until **temp1 → next** == **head**)
- Step 7 - Then set **head** = **temp2 → next**, **temp1 → next** = **head** and delete **temp2**.

Deleting from End of the list

We can use the following steps to delete a node from end of the circular linked list...

- Step 1 - Check whether list is **Empty** (**head** == **NULL**)
- Step 2 - If it is **Empty** then, display '**List is Empty!!! Deletion is not possible**' and terminate the function.
- Step 3 - If it is **Not Empty** then, define two Node pointers **'temp1'** and '**temp2'** and initialize '**temp1**' with **head**.
- Step 4 - Check whether list has only one Node (**temp1 → next** == **head**)
- Step 5 - If it is **TRUE**. Then, set **head** = **NULL** and delete **temp1**. And terminate from the function. (Setting **Empty** list condition)
- Step 6 - If it is **FALSE**. Then, set '**temp2 = temp1** ' and move **temp1** to its next node. Repeat the same until **temp1** reaches to the last node in the list. (until **temp1 → next** == **head**)
- Step 7 - Set **temp2 → next** = **head** and delete **temp1**.

Deleting a Specific Node from the list

We can use the following steps to delete a specific node from the circular linked list...

- Step 1 - Check whether list is **Empty** (**head** == **NULL**)
- Step 2 - If it is **Empty** then, display **'List is Empty!!! Deletion is not possible'** and terminate the function.
- Step 3 - If it is **Not Empty** then, define two Node pointers **'temp1'** and '**temp2**' and initialize '**temp1**' with **head**.

- Step 4 - Keep moving the **temp1** until it reaches to the exact node to be deleted or to the last node. And every time set '**temp2 = temp1**' before moving the '**temp1**' to its next node.
- Step 5 - If it is reached to the last node then display **'Given node not found in the list! Deletion not possible!!!'**. And terminate the function.
- Step 6 - If it is reached to the exact node which we want to delete, then check whether list is having only one node (**temp1 → next** == **head**)
- Step 7 - If list has only one node and that is the node to be deleted then set **head** = **NULL** and delete **temp1** (**free(temp1)**).
- Step 8 - If list contains multiple nodes then check whether **temp1** is the first node in the list (**temp1 == head**).
- Step 9 - If **temp1** is the first node then set **temp2** = **head** and keep moving **temp2** to its next node until **temp2** reaches to the last node. Then set **head = head → next**, **temp2 → nex**t = **head** and delete **temp1**.
- Step 10 - If **temp1** is not first node then check whether it is last node in the list (**temp1 → next == head**).
- Step 1 1- If **temp1** is last node then set **temp2 → next** = **head** and delete **temp1** (**free(temp1)**).
- Step 12 - If **temp1** is not first node and not last node then set **temp2 → next** = **temp1 → next** and delete **temp1** (**free(temp1)**).

Displaying a circular Linked List

We can use the following steps to display the elements of a circular linked list...

- Step 1 - Check whether list is **Empty** (**head** == **NULL**)
- Step 2 - If it is **Empty**, then display **'List is Empty!!!'** and terminate the function.
- Step 3 - If it is **Not Empty** then, define a Node pointer **'temp'** and initialize with **head**.
- Step 4 - Keep displaying **temp → data** with an arrow (**--->**) until **temp** reaches to the last node
- Step 5 - Finally display **temp → data** with arrow pointing to **head → data**.