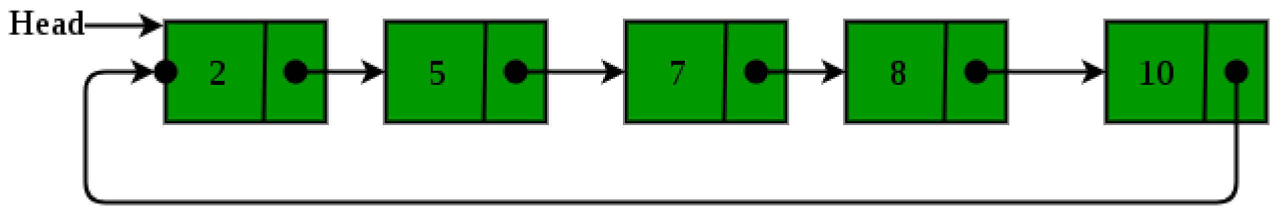
Article marked as read.

Dash
All
Articles
Videos
Problems
Quiz
Contest
« Prev
Next »

Circular Linked Lists

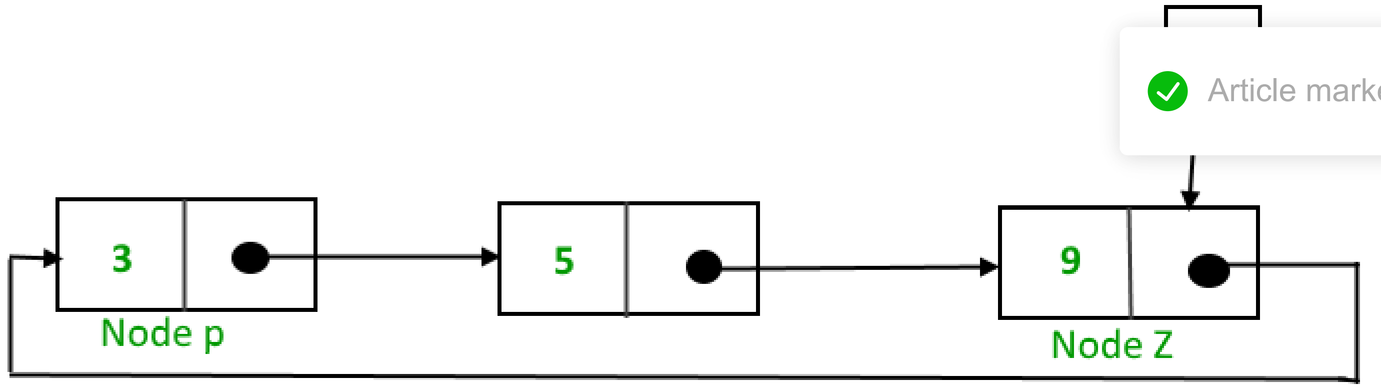
*A **circular linked** list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.*

Below is a pictorial representation of Circular Linked List:



Implementation:
To implement a circular singly linked list, we take an external pointer that points to the ***last*** node of the list. If we have a pointer ***last*** pointing to the ***last*** node, then ***last -> next*** will point to the first node.

Article marked as read.



The pointer ***last*** points to node **Z** and ***last -> next*** points to the node **P**.

Why have we taken a pointer that points to the last node instead of first node?

For insertion of node in the beginning we need to traverse the whole list. Also, for insertion at the end, the whole list has to be traversed. If instead of start pointer we take a pointer to the last node then in both the cases there won't be any need to traverse the whole list. So, insertion in the beginning or at the end takes constant time irrespective of the length of the list.

Below is a sample program to create and traverse in a Circular Linked List in both Java and C++:

C++ Java

Dash
All
Articles
Videos
Problems
Quiz
Contest

Article marked as read.

```
// A complete Java program to demonstrate the
// working of Circular Linked Lists

class CLL
{
    // A circular linked list node
    static class Node
    {
        int data;
        Node next;
    };

    // Function to insert a node in a Circular
    // linked list at the end
    static Node addEnd(Node last, int data)
    {
        if (last == null)
        {
            // Creating a node dynamically.
            Node temp = new Node();

            // Assigning the data.
            temp.data = data;
            last = temp;

            // Creating the link.
            last.next = last;
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

Article marked as read.

```
            return last;
        }

        Node temp = new Node();

        temp.data = data;
        temp.next = last.next;
        last.next = temp;
        last = temp;

        return last;
    }

    // Function to traverse a given Circular Linked
    // List using the Last pointer
    static void traverse(Node last)
    {
        Node p;

        // If list is empty, return.
        if (last == null)
        {
            System.out.println("List is empty.");
            return;
        }

        // Pointing to first Node of the list.
        p = last.next;
```

Article marked as read.

Dash
All
Articles
Videos

```
        // Traversing the list.
        do
        {
            System.out.print(p.data + " ");
            p = p.next;

        }
        while(p != last.next);

    }

    // Driver code
    public static void main(String[] args)
    {
```

Get 90% Refund!
Courses Tutorials Jobs Practice Contests

P

Quiz
Contest

```
        last = addEnd(last, 6);
        last = addEnd(last, 4);
        last = addEnd(last, 2);
        last = addEnd(last, 8);
        last = addEnd(last, 12);
        last = addEnd(last, 10);

        traverse(last);
    }
}
```

Output:

Dash
All
Articles
Videos
Problems
Quiz
Contest

```
6 4 2 8 12 10
```

Article marked as read.

Advantages of Circular Linked Lists:

1. Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again.
2. Useful for implementation of a queue. Unlike this implementation, we don't need to maintain two pointers for front and rear if we use a circular linked list. We can maintain a pointer to the last inserted node and the front can always be obtained as the next of last.
3. Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then to cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. It is convenient for the operating system to use a circular list so that when it reaches the end of the list it can cycle around to the front of the list.
4. Circular Doubly Linked Lists are used for implementation of advanced data structures like Fibonacci Heap.

Marked as Read

Report An Issue

If you are facing any issue on this page. Please let us know.