Types of Linked List

Before knowing about the types of a linked list, we should know what is ***linked list***. So, to know about the linked list, click on the link given below:

Types of Linked list

**The following are the types of linked list:**

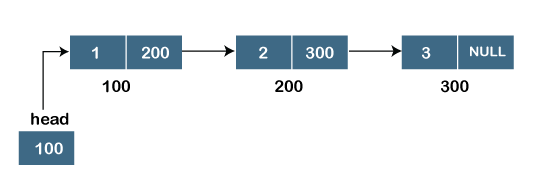
* [Singly Linked list](https://www.javatpoint.com/ds-types-of-linked-list#Singly)
* [Doubly Linked list](https://www.javatpoint.com/ds-types-of-linked-list#Doubly)
* [Circular Linked list](https://www.javatpoint.com/ds-types-of-linked-list#Circular)
* [Doubly Circular Linked list](https://www.javatpoint.com/ds-types-of-linked-list#Doubly-Circular)

Singly Linked list

It is the commonly used linked list in programs. If we are talking about the linked list, it means it is a singly linked list. The singly linked list is a data structure that contains two parts, i.e., one is the data part, and the other one is the address part, which contains the address of the next or the successor node. The address part in a node is also known as a **pointer**.

Suppose we have three nodes, and the addresses of these three nodes are 100, 200 and 300 respectively. The representation of three nodes as a linked list is shown in the below figure:

Play Video



We can observe in the above figure that there are three different nodes having address 100, 200 and 300 respectively. The first node contains the address of the next node, i.e., 200, the second node contains the address of the last node, i.e., 300, and the third node contains the NULL value in its address part as it does not point to any node. The pointer that holds the address of the initial node is known as a ***head pointer***.

The linked list, which is shown in the above diagram, is known as a singly linked list as it contains only a single link. In this list, only forward traversal is possible; we cannot traverse in the backward direction as it has only one link in the list.

**Representation of the node in a singly linked list**

1. struct node
2. {
3. **int** data;
4. struct node \*next;
5. }

In the above representation, we have defined a user-defined structure named a **node** containing two members, the first one is data of integer type, and the other one is the pointer (next) of the node type.

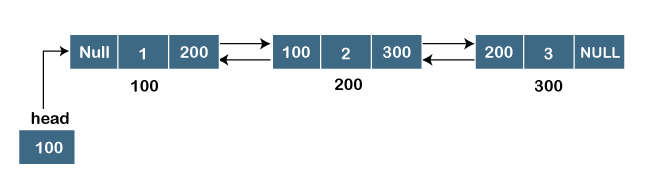
To know more about a singly linked list, click on the link given below:

<https://www.javatpoint.com/singly-linked-list>

Doubly linked list

As the name suggests, the doubly linked list contains two pointers. We can define the doubly linked list as a linear data structure with three parts: the data part and the other two address part. In other words, a doubly linked list is a list that has three parts in a single node, includes one data part, a pointer to its previous node, and a pointer to the next node.

Suppose we have three nodes, and the address of these nodes are 100, 200 and 300, respectively. The representation of these nodes in a doubly-linked list is shown below:



As we can observe in the above figure, the node in a doubly-linked list has two address parts; one part stores the ***address of the next*** while the other part of the node stores the ***previous node's address***. The initial node in the doubly linked list has the **NULL** value in the address part, which provides the address of the previous node.

**Representation of the node in a doubly linked list**

1. struct node
2. {
3. **int** data;
4. struct node \*next;
5. struct node \*prev;
6. }

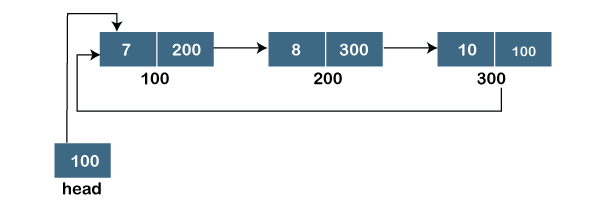
In the above representation, we have defined a user-defined structure named ***a node*** with three members, one is **data** of integer type, and the other two are the pointers, i.e., **next and prev** of the node type. The **next pointer** variable holds the address of the next node, and the **prev pointer** holds the address of the previous node. The type of both the pointers, i.e., **next and prev** is **struct node** as both the pointers are storing the address of the node of the ***struct node*** type.

Circular linked list

A circular linked list is a variation of a singly linked list. The only difference between the ***singly linked list*** and a ***circular linked*** list is that the last node does not point to any node in a singly linked list, so its link part contains a NULL value. On the other hand, the circular linked list is a list in which the last node connects to the first node, so the link part of the last node holds the first node's address. The circular linked list has no starting and ending node. We can traverse in any direction, i.e., either backward or forward. The diagrammatic representation of the circular linked list is shown below:

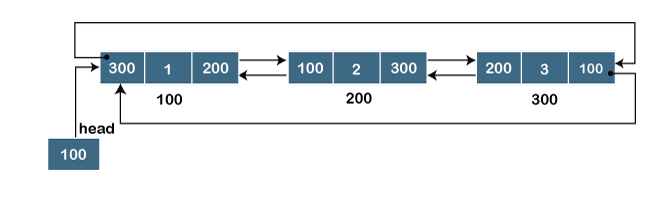
1. struct node
2. {
3. **int** data;
4. struct node \*next;
5. }

A circular linked list is a sequence of elements in which each node has a link to the next node, and the last node is having a link to the first node. The representation of the circular linked list will be similar to the singly linked list, as shown below:



Doubly Circular linked list

The doubly circular linked list has the features of both the ***circular linked list*** and ***doubly linked list***.



The above figure shows the representation of the doubly circular linked list in which the last node is attached to the first node and thus creates a circle. It is a doubly linked list also because each node holds the address of the previous node also. The main difference between the doubly linked list and doubly circular linked list is that the doubly circular linked list does not contain the NULL value in the previous field of the node. As the doubly circular linked contains three parts, i.e., two address parts and one data part so its representation is similar to the doubly linked list.

1. struct node
2. {
3. **int** data;
4. struct node \*next;
5. struct node \*prev;
6. }