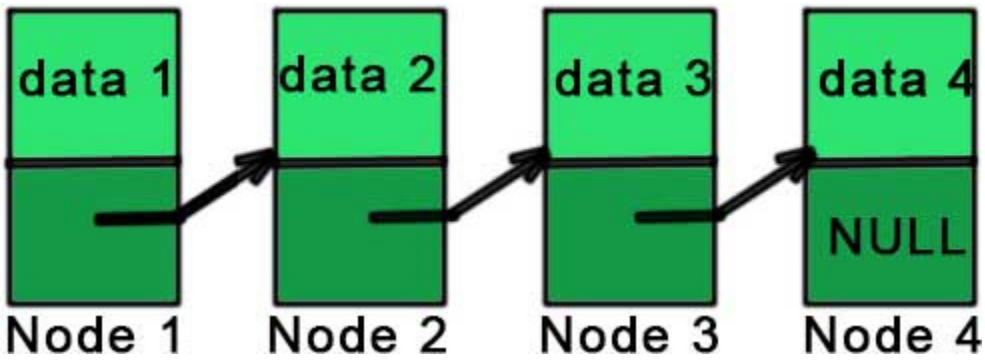
The linked list is one of the most important data structures. We often face situations, where the data is dynamic in nature and number of data can't be predicted or the number of data keeps changing during program execution. Linked lists are very useful in this type of situations.

The implementation of a linked list in C++ is done using pointers.

A linked list is made up of many nodes which are connected in nature. Every node is mainly divided into two parts, one part holds the data and the other part is connected to a different node. It is similar to the picture given below.



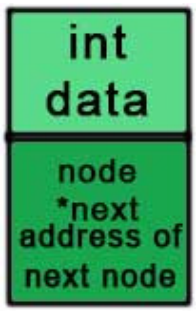
Here, each node contains a data member (the upper part of the picture) and link to another node (lower part of the picture).

Notice that the last node doesn't point to any other node and just stores NULL.

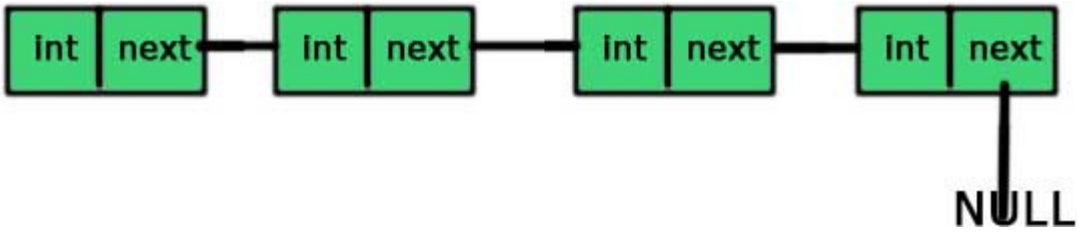
In C++, we achieve this functionality by using structures and pointers. Each structure represents a node having some data and also a pointer to another structure of the same kind. This pointer holds the address of the next node and creates the link between two nodes. So, the structure is something like:

```
struct node
{
   int data;
   struct node *next;
};
```

The first data member of the structure (named node) is an integer to hold an integer and the second data member is the pointer to a node (same structure). This means that the second data member holds the address of the next node and in this way, every node is connected as represented in the picture above. The picture representing the above structure is given below.



And the picture representing the linked list is:



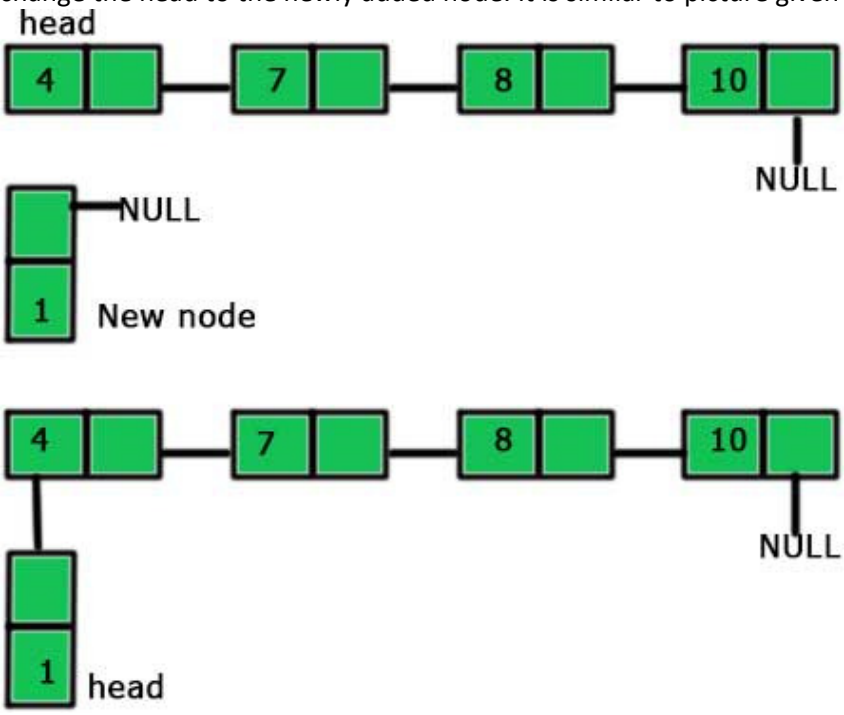
So, if we have access to the first node then we can access any node of the linked list. For example, if 'a' is a node then a->next is the node next to the 'a' (the pointer storing the address of the next node is named 'next').

One thing you should notice here is that we can easily access the next node but there is no way of accessing the previous node and this is the limitation of singly linked list.

There are three different possibilities for inserting a node into a linked list. These three possibilities are:

1. Insertion at the beginning of the list.
2. Insertion at the end of the list
3. Inserting a new node anywhere in between the list

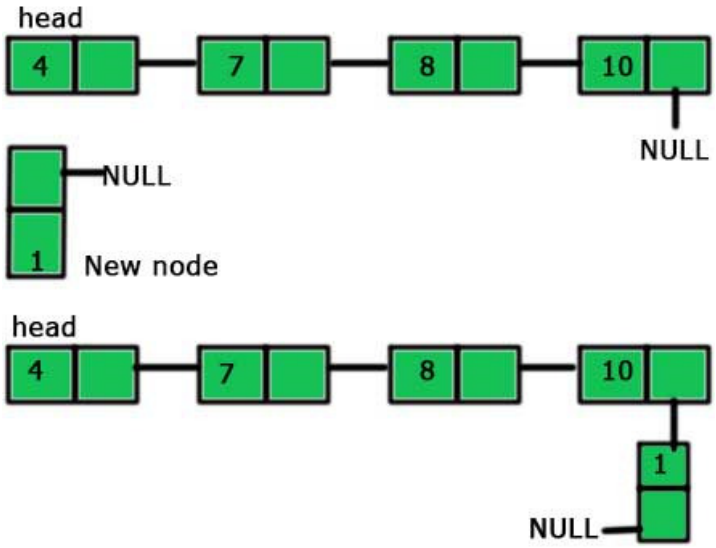
In the first case, we make a new node and points it's next to the head of the existing list and then change the head to the newly added node. It is similar to picture given below.



So, the steps to be followed are as follows:

1. Make a new node
2. Point the 'next' of the new node to the 'head' of the linked list.
3. Mark new node as 'head'.

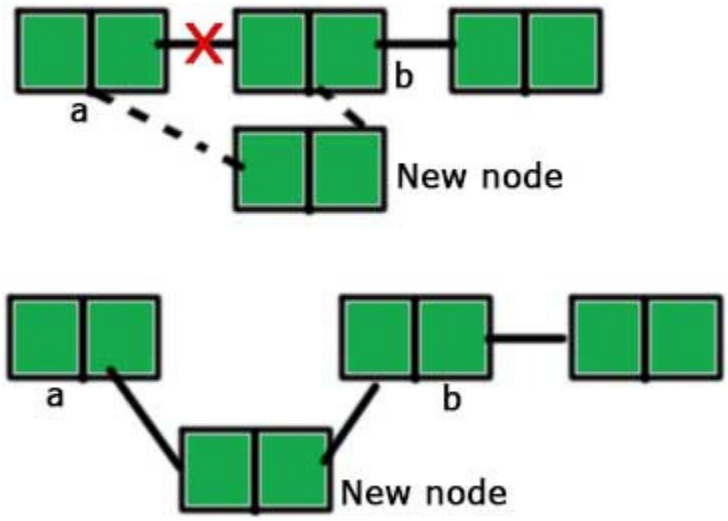
The second case is the simplest one. We just add a new node at the end of the existing list. It is shown in the picture given below:



So, the steps to add the end if a linked list are:

1. Make a new node
2. Point the last node of the linked list to the new node

The third and the last case is a little bit complicated. To insert a node in between a linked list, we need to first break the existing link and then create two new links. It will be clear from the picture given below.

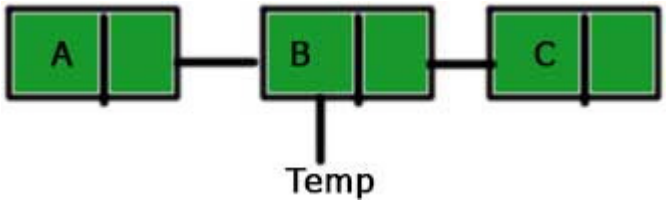


The steps for inserting a node after node ‘a’ (as shown in the picture) are:

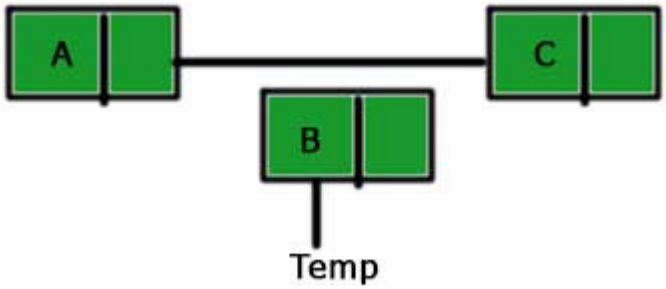
1. Make a new node
2. Point the ‘next’ of the new node to the node ‘b’ (the node after which we have to insert the new node). Till now, two nodes are pointing the same node ‘b’, the node ‘a’ and the new node.
3. Point the ‘next’ of ‘a’ to the new node.

We delete any node of a linked list by connecting the predecessor node of the node to be deleted by the successor node of the same node. For example, if we have a linked list a → b → c, then to delete the node 'b', we will connect 'a' to 'c' i.e., a → c. But this will make the node 'b' inaccessible and this type of inaccessible nodes are called garbage and we need to clean this garbage. We do this cleaning by the use of 'delete' operator. So, the steps to be followed for deletion of the node 'B' from the linked list A → B → C are as follows:

1. Create a temporary pointer to the node 'B'.



2. Connect node 'A' to 'C'.



3. Delete the node 'B'.