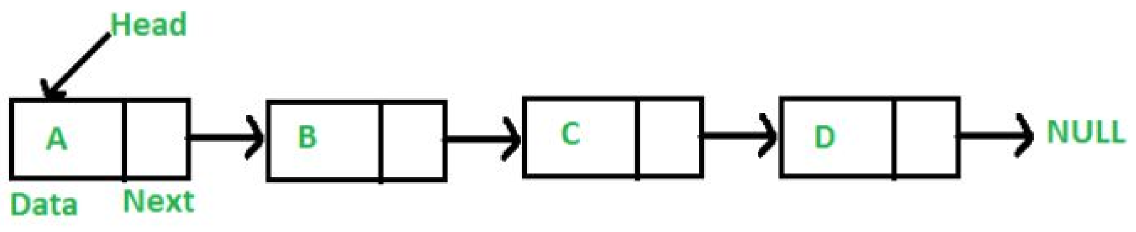
Singly linked list

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



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. 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.

Code 1st part

Now, we will create a class 'linked_list' which will contain all the functions and data members required for a linked list. This class will use the structure 'node' for the creation of the linked list.

The second and the most important part of a linked list is to always keep the track of the first node because access to the first node means access to the entire list. So, let's call our first node as ' head'.

```
#include <iostream>
using namespace std;

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

Code 2nd part

We have made two nodes – head and tail. We will store the first node in 'head' and the last node in 'tail'. The constructor of the linked list is making both 'head ' and ' tail' NULL because we have not yet added any element to our linked list and thus both are NULL.

```
class linked_list
{
private:
   node *head,*tail;
public:
   linked_list()
   {
      head = NULL;
      tail = NULL;
   }
};
```

Code part 3

Now, let's create a function of adding a node to our linked list.

```
void add_node(int n)
   {
      /*We are allocating the space required for a node by the new operator. Now,
'tmp' points to a node (or space allocated for the node) */

      node *tmp = new node;

//We are giving a value to the 'data' of 'tmp' as passed to the function

      tmp->data = n;

/*We have given the value to 'data' in the previous line and a value of the pointer
'next' (NULL) in this line and thus making our node 'tmp' complete*/

      tmp->next = NULL;

      if(head == NULL)
      {
         head = tmp;
         tail = tmp;
      }
      else
```

```
        {
           tail->next = tmp;
           tail = tail->next;
        }
    }
```

Code part 3

The next part after the creation of a node is to join the nodes and create the linked list. We will first check if the 'head' is NULL or not. If the 'head' is NULL, it means that there is no linked list yet and our current node(tmp) will be the 'head'.

If 'head' is NULL, our current node (tmp) is the first node of the linked list and this it will be 'head' and 'tail' both (as it is also the last element right now).

```
if(head == NULL)

 {

   head = tmp;

   tail = tmp;

 }
```

If 'head' is not NULL, it means that we have a linked list and we just have to add the node at the end of the linked list.The new node (tmp) will go after the 'tail' and then we are changing the tail because the new node is the new 'tail'.

```
else

 {

    tail->next = tmp;

   tail = tail->next;

 }
```