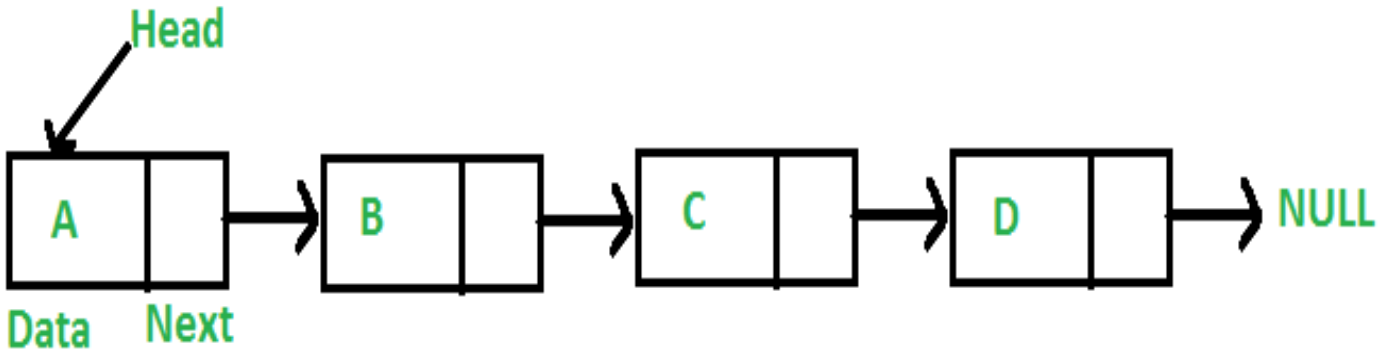
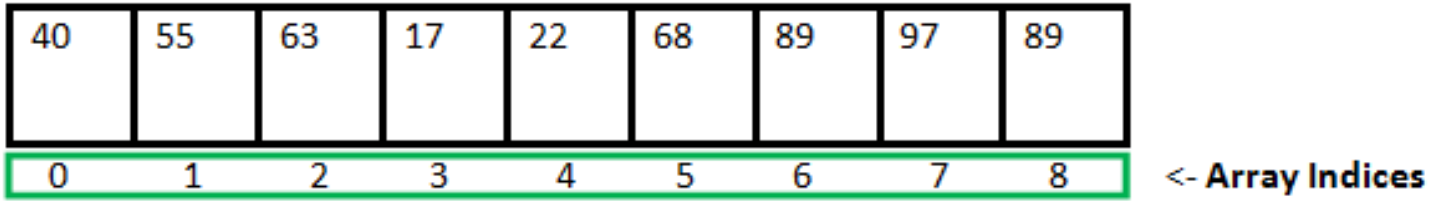
Linked list

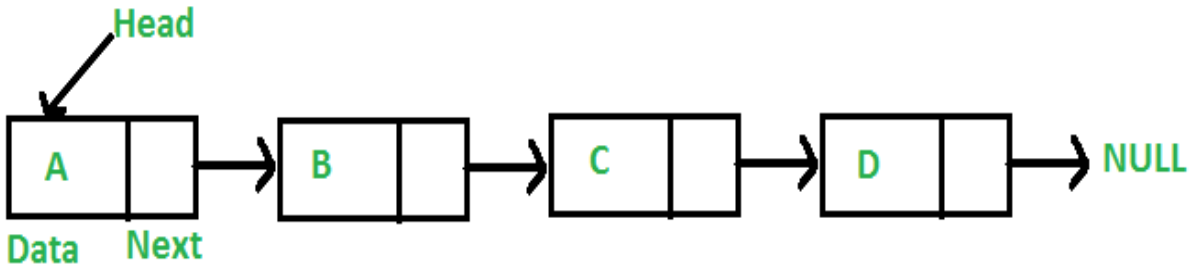


- Linear data structure
- Linked using pointers

A linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.



Array Length = 9
First Index = 0
Last Index = 8



Advantage

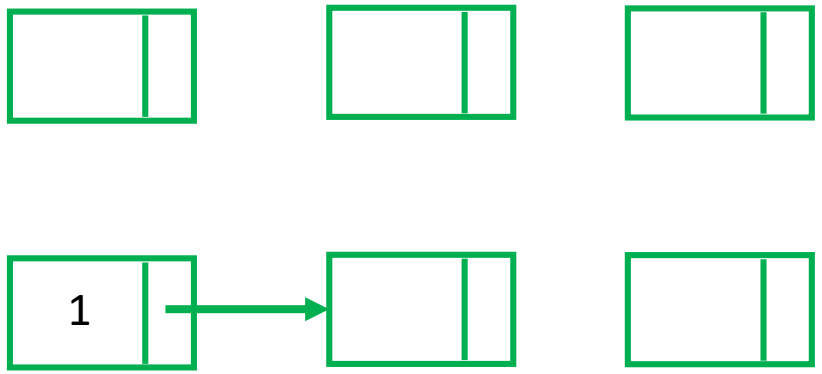
- ✓ Dynamic size
- ✓ Ease of insertion/deletion

Drawbacks

- Random cases is not allowed
- Extra memory space for pointer
- Not cache friendly

LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contains a reference of Node class type.

```
class Node {
public:
    int data;
    Node* next;
};

int main()
{
    Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;

    // allocate 3 nodes in the heap
    head = new Node();
    second = new Node();
    third = new Node();

    head->data = 1; // assign data in first node
    head->next = second; // Link first node with
    // the second node
    ...
    third->data = 3; // assign data to third node
    third->next = NULL;

     return 0;
 }
```



Add a node at the front

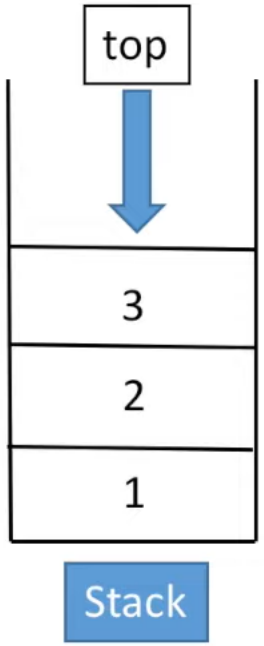
```
/* Given a reference (pointer to pointer)
to the head of a list and an int,
inserts a new node on the front of the list. */
void push(Node** head_ref, int new_data)
{
    /* 1. allocate node */
    Node* new_node = new Node();

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. Make next of new node as head */
    new_node->next = (*head_ref);

    /* 4. move the head to point to the new node */
    (*head_ref) = new_node;
}
```

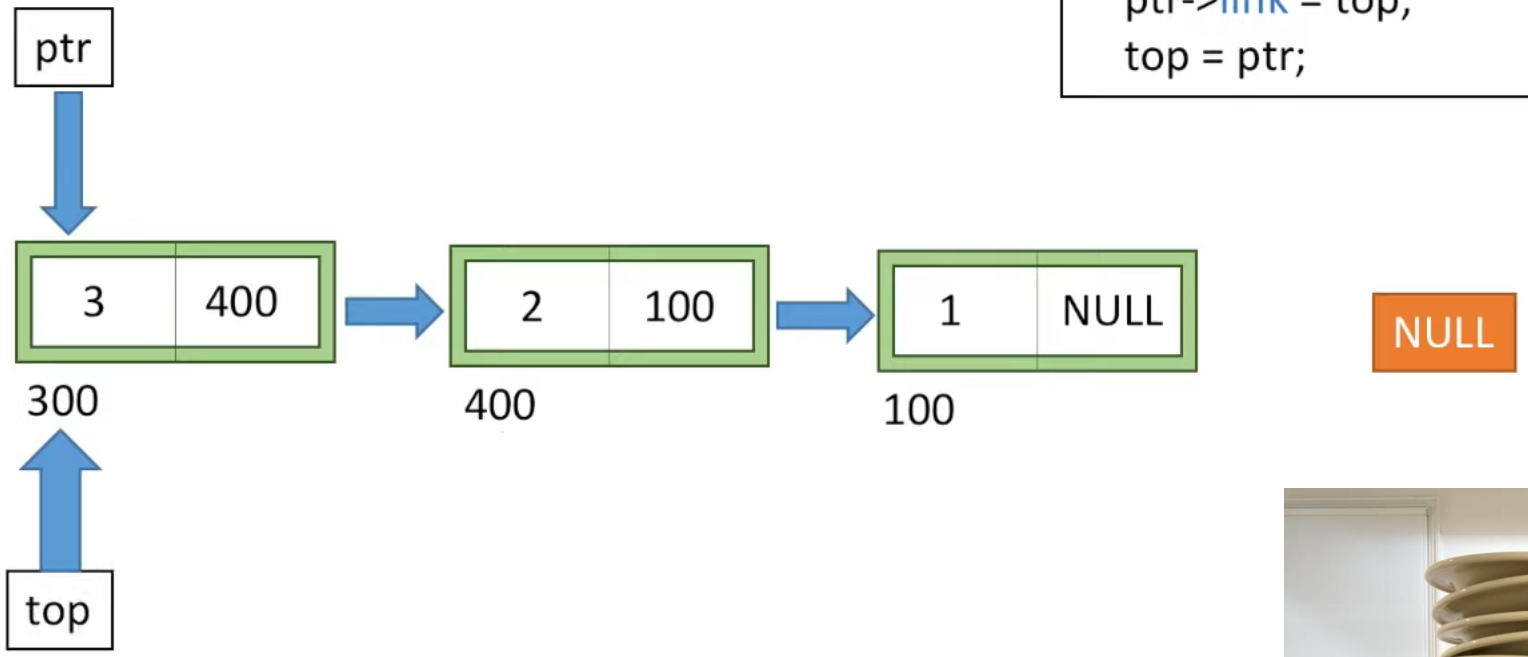
Add a node after a given node

```
/* Given a node prev_node, insert a new node after the given
   prev_node */
void insertAfter(struct Node* prev_node, int new_data)
{
    /*1. check if the given prev_node is NULL */
    if (prev_node == NULL)
    {
       printf("the given previous node cannot be NULL");
       return;
    }

    /* 2. allocate new node */
    struct Node* new_node =(struct Node*) malloc(sizeof(struct Node));

    /* 3. put in the data  */
    new_node->data  = new_data;

    /* 4. Make next of new node as next of prev_node */
    new_node->next = prev_node->next;

    /* 5. move the next of prev_node as new_node */
    prev_node->next = new_node;
}
```

Stack using Linked List

```
Insert at start
  Node *ptr = new Node();
  ptr->data = value;
  ptr->link = top;
  top = ptr;
```





1. head can be called top
2. Push (insert element in stack) = insert node at starting of linked list

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

struct Node
{
int data;

Node *link;
};

Node *top = NULL;

bool isempty()
{
  if(top == NULL)
  return true;
  else
  return false;
}
```

```
void  push (int value)
{
     Node *ptr = new Node();
     ptr->data = value;
     ptr->link = top;
     top = ptr;
}

void  pop ( )
{
 if ( isempty() )
     cout<<“Stack is Empty”;
  else
      {
         Node  *ptr = top;
         top = top -> link;
         delete(ptr);
      }
}
```

```
void showTop()
{
  if ( isempty() )
    cout<<“Stack is Empty”;
  else
    cout<<“Element at top is : ”<< top->data;

}

int main()
{
 push(1);
 push(2);
 pop();

return 0;
}
```

- References

- https://www.geeksforgeeks.org/data-structures/linked-list/