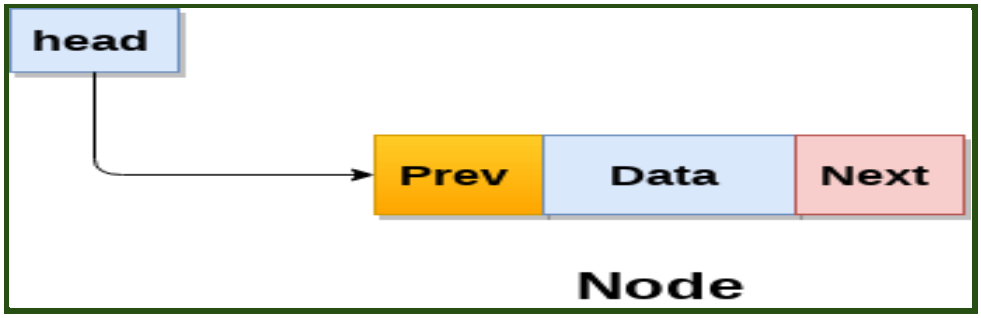
Doubly Linked List

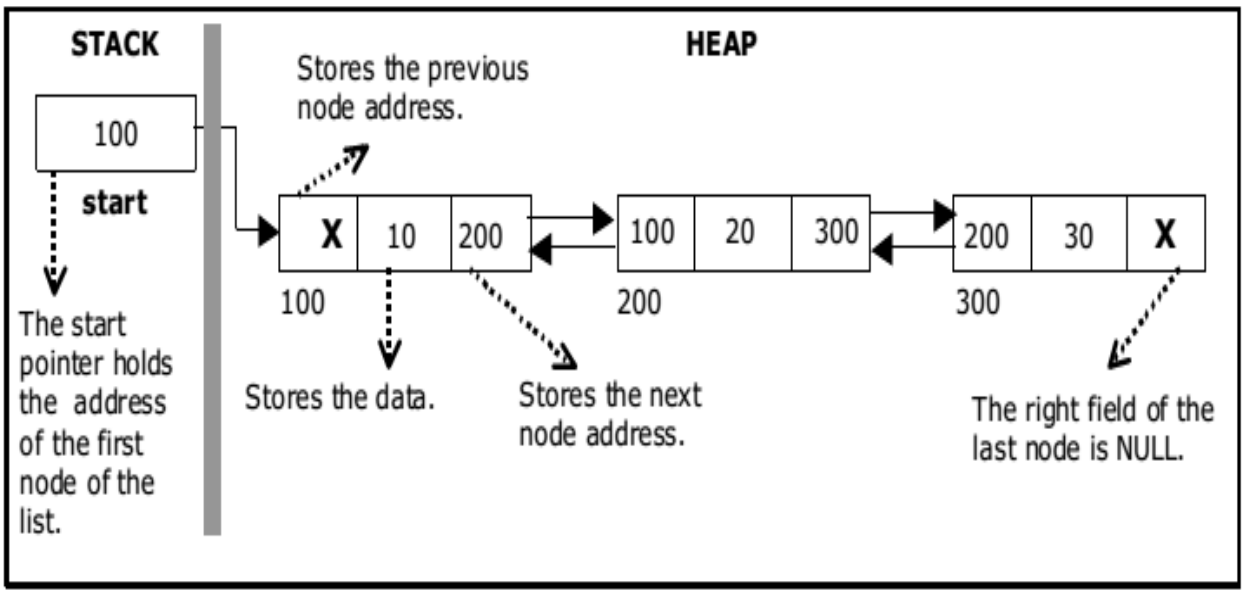
A **doubly linked list** is **a linked data structure that consists of a set of sequentially linked records called nodes**.

Each node contains **three fields**: two link fields (references to the previous and to the next node in the sequence of nodes) and one data field.

A sample node in a doubly linked list is shown in the figure.



A doubly linked list containing three nodes having numbers from 1 to 3 in their data part, is shown in the following image.



In C, structure of a node in doubly linked list can be given as :

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

The prev part of the first node and the next part of the last node will always contain null indicating end in each direction.

Operation on Doubly Link List

SN	Operation	Description
1	Insertion at beginning	Adding the node into the linked list at beginning.
2	Insertion at end	Adding the node into the linked list to the end.
3	Insertion after specified node	Adding the node into the linked list after the specified node.
4	Deletion at beginning	Removing the node from beginning of the list
5	Deletion at the end	Removing the node from end of the list.
6	Deletion of the node having given data	Removing the node which is present just after the node containing the given data.
7	Searching	Comparing each node data with the item to be searched and return the location of the item in the list if the item found else return null.
8	Traversing	Visiting each node of the list at least once in order to perform some specific operation like searching, sorting, display, etc.

Example

```
#include<stdio.h>
struct node
{
    int data;
    struct node *prev,*next;
}*head=NULL;
void create();
void display();
struct node *newnode,*temp;
void main()
{
    int ch;
    while(1)
    {
        printf("\n1.Create List \n2.Display \n3.Exit");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1:
            create();
            break;
        case 2:
            display();
            break;
        case 3:
            exit(0);
            break;
        default:
```

```
            printf("\nWrong choice....");
        }
    }
}
void create()
{
    newnode=(struct node*)malloc(sizeof(struct node));
    printf("\nEnter data:");
    scanf("%d",&newnode->data);
    newnode->prev=0;
    newnode->next=0;
    if(head==NULL)
    {
        head=temp=newnode;
    }
    else
    {
        temp->next=newnode;
        newnode->prev=temp;
        temp=newnode;
    }
}
void display()
{
    temp=head;
    printf("\n***Doubly Linked List***\n");
    while(temp!=0)
    {
        printf("%d\t",temp->data);
        temp=temp->next;
```

```
  }
}
```

```
1.Create List
2.Display
3.Exit
Enter your choice:1

Enter data:100

1.Create List
2.Display
3.Exit
Enter your choice:1

Enter data:200

1.Create List
2.Display
3.Exit
Enter your choice:1

Enter data:300

1.Create List
2.Display
3.Exit
Enter your choice:2

***Doubly Linked List***
100     200     300
1.Create List
2.Display
3.Exit
Enter your choice:3
```

Example : All operations on doubly linked lists….

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
```

```
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void insertion_specified();
void deletion_beginning();
void deletion_last();
void deletion_specified();
void display();
void search();
void main ()
{
    int choice;
    clrscr();
    while(1)
    {
        printf("\n*********Main Menu*********\n");
        printf("\nChoose one option from the following list ...\n");

printf("\n=====================================\n");
        printf("\n1.Insert in beginning\n2.Insert at last\n3.Insert at
```

```
any random location\n4.Delete from Beginning\n5.Delete from
last\n6.Delete the node after the given
data\n7.Search\n8.Show\n9.Exit\n");
        printf("\nEnter your choice?\n");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
            insertion_beginning();
            break;
            case 2:
            insertion_last();
            break;
            case 3:
            insertion_specified();
            break;
            case 4:
            deletion_beginning();
            break;
            case 5:
            deletion_last();
            break;
            case 6:
            deletion_specified();
            break;
```

```
            case 7:
            search();
            break;
            case 8:
            display();
            break;
            case 9:
            exit(0);
            break;
            default:
            printf("Please enter valid choice..");
        }
    }
}
void insertion_beginning()
{
   struct node *newnode;
   newnode = (struct node *)malloc(sizeof(struct node));
    printf("\nEnter Data:");
    scanf("%d",&newnode->data);

   if(head==NULL)
   {
      newnode->next = NULL;
      newnode->prev=NULL;
```

```
    head=newnode;
  }
  else
  {
    newnode->prev=NULL;
    newnode->next = head;
    head->prev=newnode;
    head=newnode;
  }
  printf("\nNode inserted.....\n");
}
void insertion_last()
{
  struct node *newnode,*temp;
  newnode=(struct node*)malloc(sizeof(struct node));
  printf("\nEnter data:");
  scanf("%d",&newnode->data);
  if(head == NULL)
  {
        newnode->next=NULL;
        newnode->prev=NULL;
        head=newnode;
  }
  else
  {
```

```
        temp=head;
        while(temp->next!=NULL)
        {
           temp=temp->next;
        }
        temp->next=newnode;
        newnode->prev=temp;
        newnode->next=NULL;
   }
   printf("\nNode inserted.....\n");
}
void insertion_specified()
{
   struct node *newnode,*temp;
   int loc,i;
   temp=head;
   newnode=(struct node*)malloc(sizeof(struct node));
   printf("Enter data:");
   scanf("%d",&newnode->data);

   printf("Enter the location:");
   scanf("%d",&loc);
   for(i=1;i<loc;i++)
   {
         temp=temp->next;
```

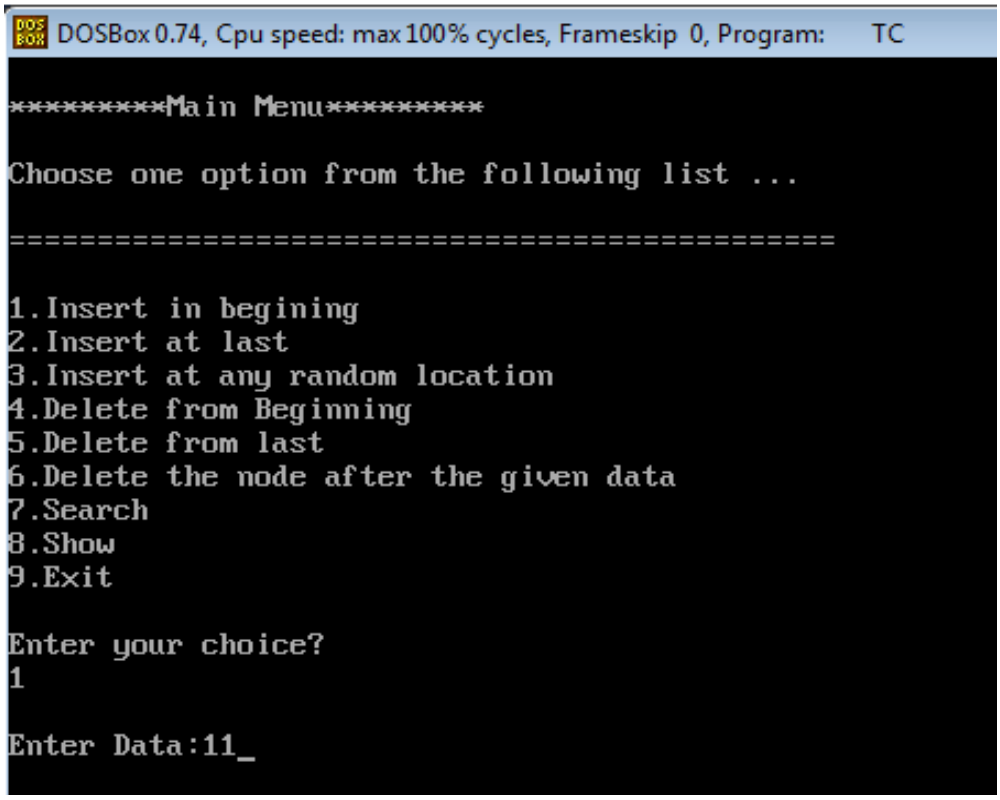
```
    }
    newnode->next=temp->next;
    newnode->prev=temp;
    temp->next=newnode;
    temp->next->prev=newnode;
    printf("\nNode inserted.....\n");
}
void deletion_beginning()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("\nUNDERFLOW.....");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nNode deleted......\n");
    }
    else
    {
        temp=head;
        head=head->next;
        head->prev=NULL;
```

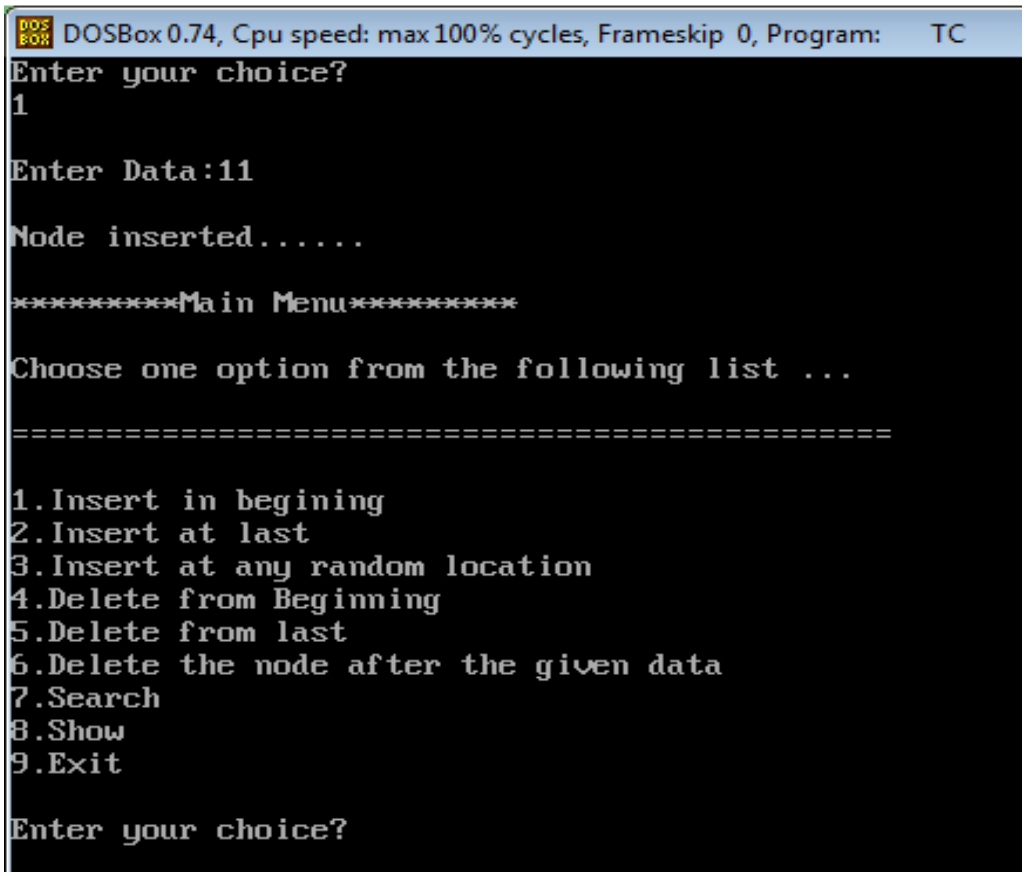
```
        free(temp);
        printf("\nNode deleted......\n");
    }
}
void deletion_last()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nNode deleted.....\n");
    }
    else
    {
        temp=head;
        while(temp->next!=NULL)
        {
           temp=temp->next;
        }
        temp->prev->next=NULL;
```

```
        free(temp);
        printf("\nNode deleted.....\n");
    }
}
void deletion_specified()
{
    struct node *ptr, *temp;
    int val;
    printf("\nEnter data after which the node is to be deleted :");
    scanf("%d", &val);
    ptr = head;
    while(ptr -> data != val)
    ptr = ptr -> next;
    if(ptr -> next == NULL)
    {
        printf("\nCan't delete…..\n");
    }
    else if(ptr -> next -> next == NULL)
    {
        ptr ->next = NULL;
    }
    else
    {
        temp = ptr -> next;
        ptr -> next = temp -> next;
```

```
        temp -> next -> prev = ptr;
        free(temp);
        printf("\nnode deleted\n");
    }
}
void display()
{
    struct node *temp;
    printf("\n printing values...\n");
    temp=head;
    while(temp != NULL)
    {
        printf("%d\n",temp->data);
        temp=temp->next;
    }
}
void search()
{
    struct node *temp;
    int item,i=0,flag;
    temp=head;
    if(temp == NULL)
    {
        printf("\nEmpty List\n");
    }
```

```
    else
    {
        printf("\nEnter item which you want to search?\n");
        scanf("%d",&item);
        while (temp!=NULL)
        {
            if(temp->data == item)
            {
                printf("\nitem found at location: %d ",i+1);
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
            i++;
            temp = temp -> next;
        }
        if(flag==1)
        {
            printf("\nItem not found……\n");
        }
    }
}
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
**********Main Menu**********
Choose one option from the following list ...
==========
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Show
9.Exit
Enter your choice?
1
Enter Data:11_


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter your choice?
1
Enter Data:11
Node inserted......
**********Main Menu**********
Choose one option from the following list ...
==========
1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Show
9.Exit
Enter your choice?