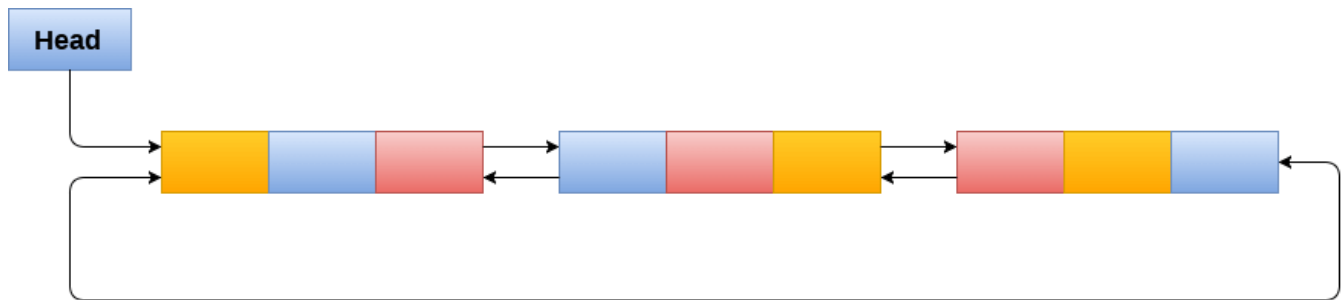


Home	Data Structure	C	C++	C#	Java	SQL	HTML	CSS	JavaScript	Ajax
------	----------------	---	-----	----	------	-----	------	-----	------------	------

# Circular Doubly Linked List

Circular doubly linked list is a more complexed type of data structure in which a node contain pointers to its previous node as well as the next node. Circular doubly linked list doesn't contain NULL in any of the node. The last node of the list contains the address of the first node of the list. The first node of the list also contain address of the last node in its previous pointer.

A circular doubly linked list is shown in the following figure.

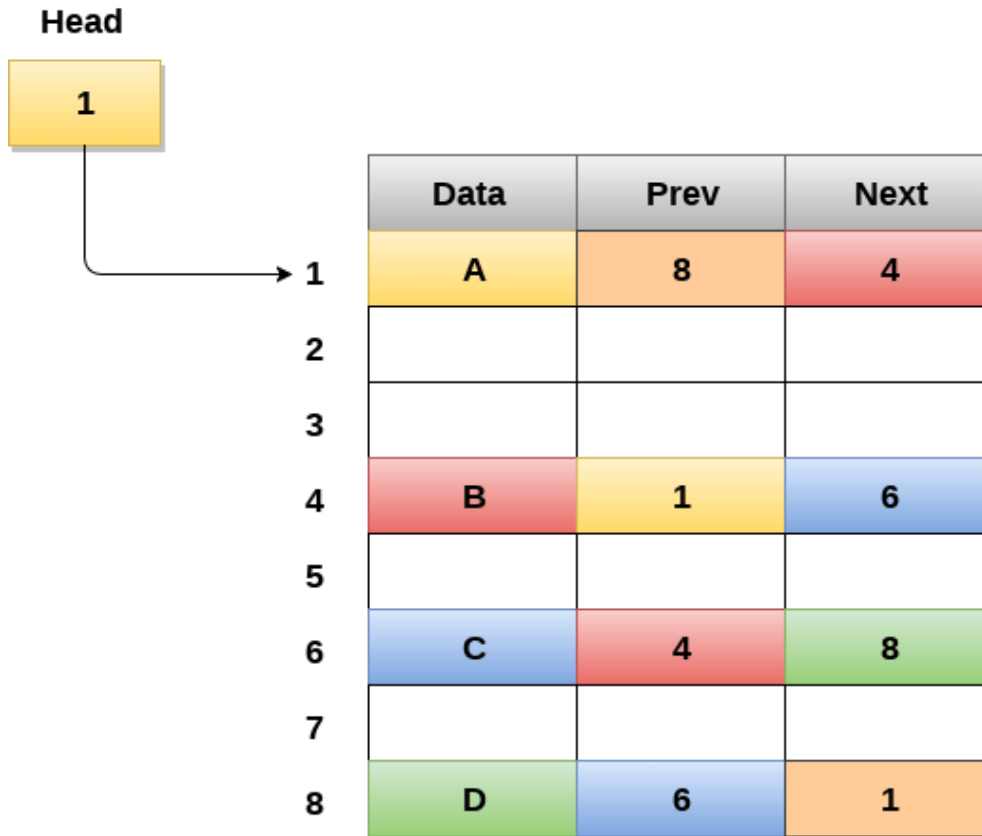


**Circular Doubly Linked List**

Due to the fact that a circular doubly linked list contains three parts in its structure therefore, it demands more space per node and more expensive basic operations. However, a circular doubly linked list provides easy manipulation of the pointers and the searching becomes twice as efficient.

## Memory Management of Circular Doubly linked list

The following figure shows the way in which the memory is allocated for a circular doubly linked list. The variable head contains the address of the first element of the list i.e. 1 hence the starting node of the list contains data A is stored at address 1. Since, each node of the list is supposed to have three parts therefore, the starting node of the list contains address of the last node i.e. 8 and the next node i.e. 4. The last node of the list that is stored at address 8 and containing data as 6, contains address of the first node of the list as shown in the image i.e. 1. In circular doubly linked list, the last node is identified by the address of the first node which is stored in the next part of the last node therefore the node which contains the address of the first node, is actually the last node of the list.



## Memory Representation of a Circular Doubly linked list

### Operations on circular doubly linked list :

There are various operations which can be performed on circular doubly linked list. The node structure of a circular doubly linked list is similar to doubly linked list. However, the operations on circular doubly linked list is described in the following table.

SN	Operation	Description
1	Insertion at beginning	Adding a node in circular doubly linked list at the beginning.
2	Insertion at end	Adding a node in circular doubly linked list at the end.
3	Deletion at beginning	Removing a node in circular doubly linked list from beginning.
4	Deletion at end	Removing a node in circular doubly linked list at the end.

Traversing and searching in circular doubly linked list is similar to that in the circular singly linked list.

## C program to implement all the operations on circular doubly linked list

```
#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 deletion_beginning();
void deletion_last();
void display();
void search();
void main ()
{
    int choice =0;
    while(choice != 9)
    {
        printf("\n*****Main Menu*****\n");
        printf("\nChoose one option from the following list ...\n");
        printf("\n=====");
        printf("\n1.Insert in Beginning\n2.Insert at last\n3.Delete from Beginning\n4.Delete from l");
        printf("\nEnter your choice?\n");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
                insertion_beginning();
```

```
        break;
        case 2:
            insertion_last();
        break;
        case 3:
            deletion_beginning();
        break;
        case 4:
            deletion_last();
        break;
        case 5:
            search();
        break;
        case 6:
            display();
        break;
        case 7:
            exit(0);
        break;
        default:
            printf("Please enter valid choice..");
    }
}

void insertion_beginning()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter Item value");
```

```
scanf("%d",&item);
ptr->data=item;
if(head==NULL)
{
    head = ptr;
    ptr -> next = head;
    ptr -> prev = head;
}
else
{
    temp = head;
    while(temp -> next != head)
    {
        temp = temp -> next;
    }
    temp -> next = ptr;
    ptr -> prev = temp;
    head -> prev = ptr;
    ptr -> next = head;
    head = ptr;
}
printf("\nNode inserted\n");
}

}

void insertion_last()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node *) malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
```

```
printf("\nEnter value");
scanf("%d",&item);
ptr->data=item;
if(head == NULL)
{
    head = ptr;
    ptr -> next = head;
    ptr -> prev = head;
}
else
{
    temp = head;
    while(temp->next !=head)
    {
        temp = temp->next;
    }
    temp->next = ptr;
    ptr ->prev=temp;
    head -> prev = ptr;
    ptr -> next = head;
}
printf("\nnode inserted\n");
}
```

```
void deletion_beginning()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
    }
}
```

```
    printf("\nnode deleted\n");
}
else
{
    temp = head;
    while(temp -> next != head)
    {
        temp = temp -> next;
    }
    temp -> next = head -> next;
    head -> next -> prev = temp;
    free(head);
    head = temp -> next;
}

}

void deletion_last()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != head)
        {
            ptr = ptr -> next;
        }
    }
}
```



```
ptr -> prev -> next = head;
head -> prev = ptr -> prev;
free(ptr);
printf("\nnode deleted\n");
}
}

void display()
{
    struct node *ptr;
    ptr=head;
    if(head == NULL)
    {
        printf("\nnothing to print");
    }
    else
    {
        printf("\n printing values ... \n");

        while(ptr -> next != head)
        {

            printf("%d\n", ptr -> data);
            ptr = ptr -> next;
        }
        printf("%d\n", ptr -> data);
    }
}

void search()
{
    struct node *ptr;
    int item,i=0,flag=1;
    ptr = head;
    if(ptr == NULL)
```

```
{
    printf("\nEmpty List\n");
}
else
{
    printf("\nEnter item which you want to search?\n");
    scanf("%d",&item);
    if(head ->data == item)
    {
        printf("item found at location %d",i+1);
        flag=0;
    }
    else
    {
        while (ptr->next != head)
        {
            if(ptr->data == item)
            {
                printf("item found at location %d ",i+1);
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
            i++;
            ptr = ptr -> next;
        }
    }
    if(flag != 0)
    {
        printf("Item not found\n");
    }
}
```

```
}
```

**Output:**

```
*****Main Menu*****
```

```
Choose one option from the following list ...
```

```
=====
```

- 1.Insert in Beginning
- 2.Insert at last
- 3.Delete from Beginning
- 4.Delete from last
- 5.Search
- 6.Show
- 7.Exit

```
Enter your choice?
```

```
1
```

```
Enter Item value123
```

```
Node inserted
```

```
*****Main Menu*****
```

```
Choose one option from the following list ...
```

```
=====
```

- 1.Insert in Beginning
- 2.Insert at last
- 3.Delete from Beginning
- 4.Delete from last
- 5.Search
- 6.Show
- 7.Exit

Enter your choice?

2

Enter value234

node inserted

\*\*\*\*\*Main Menu\*\*\*\*\*

Choose one option from the following list ...

=====

- 1.Insert in Beginning
- 2.Insert at last
- 3.Delete from Beginning
- 4.Delete from last
- 5.Search
- 6.Show
- 7.Exit

Enter your choice?

1

Enter Item value90

Node inserted

\*\*\*\*\*Main Menu\*\*\*\*\*

Choose one option from the following list ...

=====

- 1.Insert in Beginning
- 2.Insert at last
- 3.Delete from Beginning

```
4.Delete from last
5.Search
6.Show
7.Exit
```

Enter your choice?

2

Enter value80

node inserted

\*\*\*\*\*Main Menu\*\*\*\*\*

Choose one option from the following list ...

=====

```
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit
```

Enter your choice?

3

\*\*\*\*\*Main Menu\*\*\*\*\*

Choose one option from the following list ...

=====

```
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
```

```
4.Delete from last
5.Search
6.Show
7.Exit
```

Enter your choice?

4

node deleted

\*\*\*\*\*Main Menu\*\*\*\*\*

Choose one option from the following list ...

=====

```
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit
```

Enter your choice?

6

printing values ...

123

\*\*\*\*\*Main Menu\*\*\*\*\*

Choose one option from the following list ...

=====

```
1.Insert in Beginning
2.Insert at last
```

```
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit
```

Enter your choice?

5

Enter item which you want to search?

123

item found at location 1

\*\*\*\*\*Main Menu\*\*\*\*\*

Choose one option from the following list ...

=====

```
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit
```

Enter your choice?

7

← Prev

Next →

 Youtube For Videos Join Our Youtube Channel: [Join Now](#)
















Feedback

- Send your Feedback to [feedback@javatpoint.com](mailto:feedback@javatpoint.com)

## Help Others, Please Share



## Learn Latest Tutorials

 Splunk tutorial Splunk	 SPSS tutorial SPSS	 Swagger tutorial Swagger	 T-SQL tutorial Transact-SQL
 Tumblr tutorial Tumblr	 React tutorial ReactJS	 Regex tutorial Regex	 Reinforcement learning tutorial Reinforcement Learning
 R Programming tutorial R Programming	 RxJS tutorial RxJS	 React Native tutorial React Native	 Python Design Patterns Python Design Patterns
 Python Pillow tutorial Python Pillow	 Python Turtle tutorial Python Turtle	 Keras tutorial Keras	

## Preparation

 Aptitude Aptitude	 Logical Reasoning Reasoning	 Verbal Ability Verbal Ability	 Interview Questions Interview Questions
---	---	---	---





Company  
Interview  
Questions

Company Questions

## Trending Technologies



Artificial  
Intelligence  
Tutorial

Artificial  
Intelligence



AWS Tutorial  
AWS



Selenium  
tutorial  
Selenium



Cloud  
Computing  
tutorial

Cloud Computing



Hadoop tutorial  
Hadoop



ReactJS  
Tutorial  
ReactJS



Data Science  
Tutorial  
Data Science



Angular 7  
Tutorial  
Angular 7



Blockchain  
Tutorial  
Blockchain



Git Tutorial  
Git



Machine  
Learning Tutorial  
Machine Learning



DevOps  
Tutorial  
DevOps

## B.Tech / MCA



DBMS tutorial  
DBMS



Data Structures  
tutorial  
Data Structures



DAA tutorial  
DAA



Operating  
System tutorial  
Operating System



Computer  
Network tutorial  
Computer Network



Compiler  
Design tutorial  
Compiler Design




Computer  
Organization and  
Architecture  
Computer  
Organization



Discrete  
Mathematics  
Tutorial  
Discrete  
Mathematics




Ethical Hacking  
Tutorial  
Ethical Hacking



Computer  
Graphics Tutorial  
Computer Graphics



Software  
Engineering  
Tutorial  
Software  
Engineering




html tutorial  
Web Technology




Cyber Security  
tutorial  
Cyber Security




Automata  
Tutorial  
Automata



C Language  
tutorial  
C Programming




C++ tutorial  
C++




Java tutorial  
Java




.Net  
Framework  
tutorial  
.Net



Python tutorial  
Python



List of  
Programs  
Programs



Control  
Systems tutorial  
Control System



Data Mining  
Tutorial  
Data Mining



Data  
Warehouse  
Tutorial  
Data Warehouse