

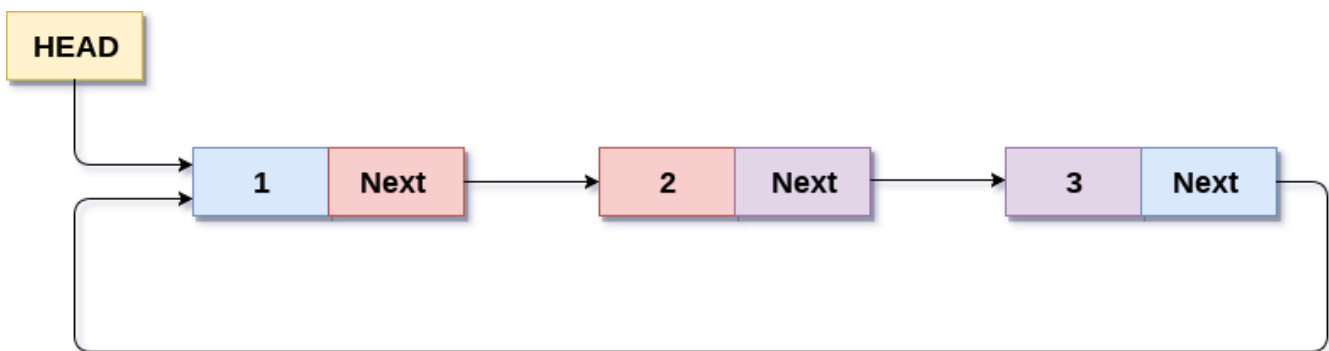
Home	Data Structure	C	C++	C#	Java	SQL	HTML	CSS	JavaScript	Ajax
------	----------------	---	-----	----	------	-----	------	-----	------------	------

Circular Singly Linked List

In a circular Singly linked list, the last node of the list contains a pointer to the first node of the list. We can have circular singly linked list as well as circular doubly linked list.

We traverse a circular singly linked list until we reach the same node where we started. The circular singly linked list has no beginning and no ending. There is no null value present in the next part of any of the nodes.

The following image shows a circular singly linked list.



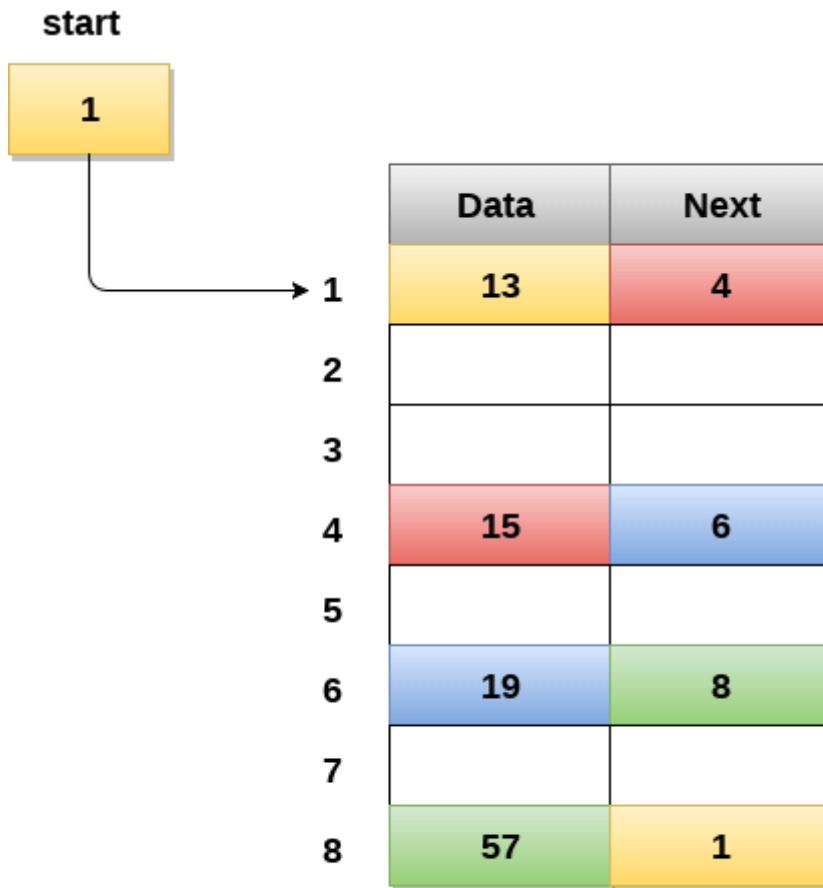
Circular Singly Linked List

Circular linked list are mostly used in task maintenance in operating systems. There are many examples where circular linked list are being used in computer science including browser surfing where a record of pages visited in the past by the user, is maintained in the form of circular linked lists and can be accessed again on clicking the previous button.

Memory Representation of circular linked list:

In the following image, memory representation of a circular linked list containing marks of a student in 4 subjects. However, the image shows a glimpse of how the circular list is being stored in the memory. The start or head of the list is pointing to the element with the index 1 and containing 13 marks in the data part and 4 in the next part. Which means that it is linked with the node that is being stored at 4th index of the list.

However, due to the fact that we are considering circular linked list in the memory therefore the last node of the list contains the address of the first node of the list.



Memory Representation of a circular linked list

We can also have more than one number of linked list in the memory with the different start pointers pointing to the different start nodes in the list. The last node is identified by its next part which contains the address of the start node of the list. We must be able to identify the last node of any linked list so that we can find out the number of iterations which need to be performed while traversing the list.

Operations on Circular Singly linked list:

Insertion

SN	Operation	Description
1	Insertion at beginning	Adding a node into circular singly linked list at the beginning.
2	Insertion at the end	Adding a node into circular singly linked list at the end.

Deletion & Traversing

SN	Operation	Description
1	Deletion at beginning	Removing the node from circular singly linked list at the beginning.
2	Deletion at the end	Removing the node from circular singly linked list at the end.
3	Searching	Compare each element of the node with the given item and return the location at which the item is present in the list otherwise return null.
4	Traversing	Visiting each element of the list at least once in order to perform some specific operation.

Menu-driven program in C implementing all operations on circular singly linked list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *head;

void begininsert ();
void lastinsert ();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();
void main ()
{
```

```
int choice =0;
while(choice != 7)
{
    printf("\n*****Main Menu*****\n");
    printf("\nChoose one option from the following list ...\n");
    printf("\n=====");
    printf("\n1.Insert in begining\n2.Insert at last\n3.Delete from Beginning\n4.Delete from la
    printf("\nEnter your choice?\n");
    scanf("\n%d",&choice);
    switch(choice)
    {
        case 1:
            begininsert();
            break;
        case 2:
            lastinsert();
            break;
        case 3:
            begin_delete();
            break;
        case 4:
            last_delete();
            break;
        case 5:
            search();
            break;
        case 6:
            display();
            break;
        case 7:
            exit(0);
            break;
        default:
            printf("Please enter valid choice..");
    }
}
```

```
}  
  
void begininsert()  
{  
    struct node *ptr,*temp;  
    int item;  
    ptr = (struct node *)malloc(sizeof(struct node));  
    if(ptr == NULL)  
    {  
        printf("\nOVERFLOW");  
    }  
    else  
    {  
        printf("\nEnter the node data?");  
        scanf("%d",&item);  
        ptr -> data = item;  
        if(head == NULL)  
        {  
            head = ptr;  
            ptr -> next = head;  
        }  
        else  
        {  
            temp = head;  
            while(temp->next != head)  
                temp = temp->next;  
            ptr->next = head;  
            temp -> next = ptr;  
            head = ptr;  
        }  
        printf("\nnode inserted\n");  
    }  
}  
  
void lastinsert()  
{  
    struct node *ptr,*temp;
```

```
int item;
ptr = (struct node *)malloc(sizeof(struct node));
if(ptr == NULL)
{
    printf("\nOVERFLOW\n");
}
else
{
    printf("\nEnter Data?");
    scanf("%d",&item);
    ptr->data = item;
    if(head == NULL)
    {
        head = ptr;
        ptr -> next = head;
    }
    else
    {
        temp = head;
        while(temp -> next != head)
        {
            temp = temp -> next;
        }
        temp -> next = ptr;
        ptr -> next = head;
    }

    printf("\nnode inserted\n");
}
}
```

```
void begin_delete()
{
    struct node *ptr;
    if(head == NULL)
```

```
{
    printf("\nUNDERFLOW");
}
else if(head->next == head)
{
    head = NULL;
    free(head);
    printf("\nnode deleted\n");
}

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

void last_delete()
{
    struct node *ptr, *preptr;
    if(head==NULL)
    {
        printf("\nUNDERFLOW");
    }
    else if (head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
```



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

```
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");
}
}
```

```
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);  
}  
  
}
```

Output:

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

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

```
=====
```

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

```
Enter your choice?
```

```
1
```

```
Enter the node data?10
```

```
node inserted
```

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

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

```
=====
```

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

Enter your choice?

2

Enter Data?20

node inserted

*****Main Menu*****

Choose one option from the following list ...

=====

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

Enter your choice?

2

Enter Data?30

node inserted

*****Main Menu*****

Choose one option from the following list ...

=====

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

Enter your choice?

3

node deleted

*****Main Menu*****

Choose one option from the following list ...

=====

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

Enter your choice?

4

node deleted

*****Main Menu*****

Choose one option from the following list ...

=====

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

Enter your choice?

5

Enter item which you want to search?

20

item found at location 1

*****Main Menu*****

Choose one option from the following list ...

=====

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

Enter your choice?

6

printing values ...

20

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

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

```
=====
```

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

```
Enter your choice?
```

```
7
```

[← Prev](#)[Next →](#)

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












Feedback

- Send your Feedback to 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++

