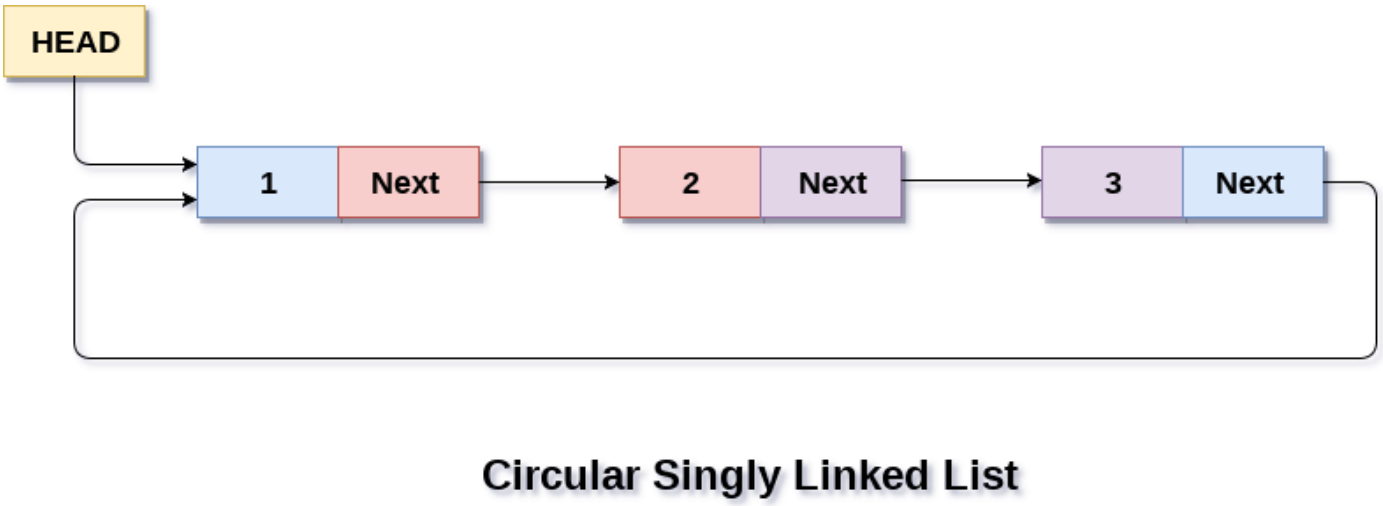
Circular Singly Linked List

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 liked 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.



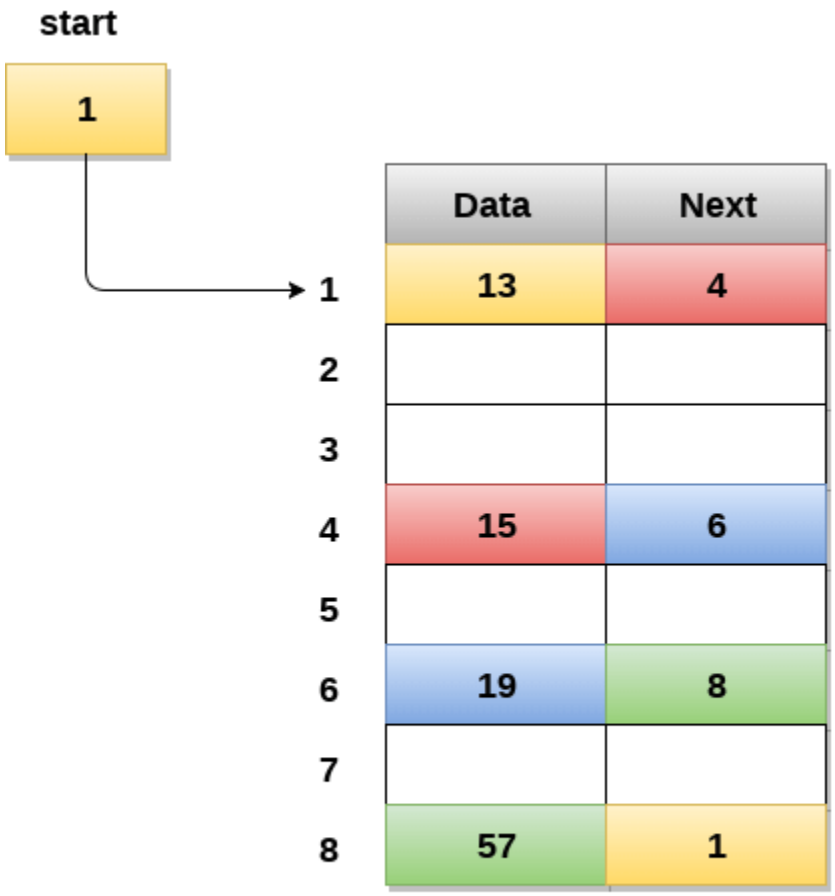
Circular Singly Linked List

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;
```

```
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

```
449.
4510. void beginsert ();
4611. void lastinsert ();
4712. void randominsert();
4813. void begin_delete();
4914. void last_delete();
5015. void random_delete();
5116. void display();
5217. void search();
5318. void main ()
5419. {
5520.     int choice =0;
5621.     while(choice != 7)
5722.     {
5823.         printf("\n*********Main Menu*********\n");
5924.         printf("\nChoose one option from the following list ...\n");
6025.         printf("\n==========================================\
61      n");
6226.         printf("\n1.Insert in begining\n2.Insert at last\n3.Delete from Beginning\n4.Delete from last
63      \n5.Search for an element\n6.Show\n7.Exit\n");
6427.         printf("\nEnter your choice?\n");
6528.         scanf("\n%d",&choice);
6629.         switch(choice)
6730.         {
6831.             case 1:
6932.             beginsert();
7033.             break;
7134.             case 2:
7235.             lastinsert();
7336.             break;
7437.             case 3:
7538.             begin_delete();
7639.             break;
7740.             case 4:
```

```
7841.            last_delete();
7942.           break;
8043.            case 5:
8144.           search();
8245.            break;
8346.           case 6:
8447.            display();
8548.           break;
8649.            case 7:
8750.           exit(0);
8851.            break;
8952.           default:
9053.            printf("Please enter valid choice..");
9154.       }
9255.    }
9356. }
9457. void beginsert()
9558. {
9659.    struct node *ptr,*temp;
9760.    int item;
9861.    ptr = (struct node *)malloc(sizeof(struct node));
9962.    if(ptr == NULL)
10063.   {
10164.      printf("\nOVERFLOW");
10265.   }
10366.   else
10467.   {
10568.      printf("\nEnter the node data?");
10669.       scanf("%d",&item);
10770.      ptr -> data = item;
10871.       if(head == NULL)
10972.      {
11073.          head = ptr;
11174.          ptr -> next = head;
```

```
11275.        }
11376.       else
11477.        {
11578.           temp = head;
11679.            while(temp->next != head)
11780.               temp = temp->next;
11881.            ptr->next = head;
11982.           temp -> next = ptr;
12083.            head = ptr;
12184.        }
12285.        printf("\nnode inserted\n");
12386.    }
12487.
12588. }
12689. void lastinsert()
12790. {
12891.    struct node *ptr,*temp;
12992.    int item;
13093.    ptr = (struct node *)malloc(sizeof(struct node));
13194.    if(ptr == NULL)
13295.    {
13396.       printf("\nOVERFLOW\n");
13497.    }
13598.    else
13699.    {
137100.                printf("\nEnter Data?");
138101.                 scanf("%d",&item);
139102.                ptr->data = item;
140103.                 if(head == NULL)
141104.                {
142105.                    head = ptr;
143106.                   ptr -> next = head;
144107.                 }
145108.                else
```

```
146109.             {
147110.                temp = head;
148111.                 while(temp -> next != head)
149112.                {
150113.                    temp = temp -> next;
151114.                }
152115.                 temp -> next = ptr;
153116.                ptr -> next = head;
154117.             }
155118.
156119.             printf("\nnode inserted\n");
157120.          }
158121.
159122.       }
160123.
161124.       void begin_delete()
162125.       {
163126.          struct node *ptr;
164127.          if(head == NULL)
165128.          {
166129.             printf("\nUNDERFLOW");
167130.          }
168131.          else if(head->next == head)
169132.          {
170133.             head = NULL;
171134.             free(head);
172135.             printf("\nnode deleted\n");
173136.          }
174137.
175138.          else
176139.          {   ptr = head;
177140.             while(ptr -> next != head)
178141.                ptr = ptr -> next;
179142.             ptr->next = head->next;
```

```
180143.             free(head);
181144.            head = ptr->next;
182145.             printf("\nnode deleted\n");
183146.
184147.         }
185148.     }
186149.     void last_delete()
187150.     {
188151.         struct node *ptr, *preptr;
189152.         if(head==NULL)
190153.         {
191154.             printf("\nUNDERFLOW");
192155.         }
193156.         else if (head ->next == head)
194157.         {
195158.             head = NULL;
196159.             free(head);
197160.             printf("\nnode deleted\n");
198161.
199162.         }
200163.         else
201164.         {
202165.             ptr = head;
203166.             while(ptr ->next != head)
204167.             {
205168.                 preptr=ptr;
206169.                 ptr = ptr->next;
207170.             }
208171.             preptr->next = ptr -> next;
209172.             free(ptr);
210173.             printf("\nnode deleted\n");
211174.
212175.         }
213176.     }
```

```
214177.
215178.     void search()
216179.     {
217180.         struct node *ptr;
218181.         int item,i=0,flag=1;
219182.         ptr = head;
220183.         if(ptr == NULL)
221184.         {
222185.             printf("\nEmpty List\n");
223186.         }
224187.         else
225188.         {
226189.             printf("\nEnter item which you want to search?\n");
227190.             scanf("%d",&item);
228191.             if(head ->data == item)
229192.             {
230193.             printf("item found at location %d",i+1);
231194.             flag=0;
232195.             }
233196.             else
234197.             {
235198.             while (ptr->next != head)
236199.             {
237200.                 if(ptr->data == item)
238201.                 {
239202.                     printf("item found at location %d ",i+1);
240203.                      flag=0;
241204.                     break;
242205.                 }
243206.                 else
244207.                 {
245208.                     flag=1;
246209.                 }
247210.                 i++;
```

```
248211.                 ptr = ptr -> next;
249212.             }
250213.             }
251214.             if(flag != 0)
252215.             {
253216.                printf("Item not found\n");
254217.             }
255218.         }
256219.
257220.     }
258221.
259222.     void display()
260223.     {
261224.         struct node *ptr;
262225.         ptr=head;
263226.         if(head == NULL)
264227.         {
265228.             printf("\nnothing to print");
266229.         }
267230.         else
268231.         {
269232.             printf("\n printing values ... \n");
270233.
271234.             while(ptr -> next != head)
272235.             {
273236.
274237.                 printf("%d\n", ptr -> data);
275238.                ptr = ptr -> next;
276239.             }
277240.             printf("%d\n", ptr -> data);
278241.         }
279242.
280243.     }
```

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
```