Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No.7
Implement Circular Linked List ADT.
Name: Aryan Parab
Roll No:36
Date of Performance:
Date of Submission:
Marks:
Sign:

Experiment No. 7: Circular Linked List Operations

Aim: Implementation of Circular Linked List ADT Objective:

In circular linked list last node is connected to first node. On other hand circular linked list can be used to implement traversal along web pages.

Theory:

In a circular linked list, the last node contains a pointer to the first node of the list. We can have a circular singly linked list as well as a circular doubly linked list. While traversing a circular linked list, we can begin at any node and traverse the list in any one direction, forward or backward, until we reach the same node where we started. Thus, a circular linked list has no beginning and no ending.

Inserting a New Node in a Circular Linked List

Case 1: The new node is inserted at the beginning.

Case 2: The new node is inserted at the end.

Deleting a Node from a Circular Linked List

Case 1: The first node is deleted.

Case 2: The last node is deleted.

Insertion and Deletion after or before a given node is same as singly linked list.

Algorithm

Algorithm to insert a new node at the beginning

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 9 [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL NEXT
Step 4: SET NEW_NODE-->DATA = VAL
Step 5: SET PTR=START
Repeat Step 6 while PTR NEXT != START
Step 6: SET PTR = PTR NEXT [END OF LOOP]

Step 7: SET NEW_NODE--> NEXT= START
Step 8: SET PTR-->NEXT = START
Step 9: SET START = NEW_NODE
Step 10: EXIT
```

Algorithm to insert a new node at the end

```
Step 1: IF AVAIL = NULL
            Write OVERFLOW
            Go to Step 11 [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL--> NEXT
Step 4: SET NEW_NODE -->DATA = VAL
Step 5: SET NEW_NODE-->NEXT = START
Step 6: SET PTR = START
Step 7: Repeat Step 8 while PTR--> NEXT != START
Step 8: SET PTR = PTR -->NEXT [END OF LOOP]
Step 9: SET PTR -->NEXT = NEW_NODE
Step 10: EXIT
```

Algorithm to delete the first node

```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 6 [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Step 4 while PTR--> NEXT != START
Step 4: SET PTR = PTR -->NEXT [END OF LOOP]
Step 4: SET PTR NEXT = START -->NEXT
Step 5: FREE START
Step 6: EXIT
```

Algorithm to delete the last node

```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 7 [END OF IF]
Step 2: SET PTR = START [END OF LOOP]
Step 3: Repeat Step 4 and Step 5 while PTR -->NEXT != START
Step 4: SET PREPTR = PTR
Step 5: SET PTR = PTR -->NEXT
Step 6: SET PREPTR-->NEXT = START
Step 7: FREE PTR
Step 8: EXIT
```

Code:

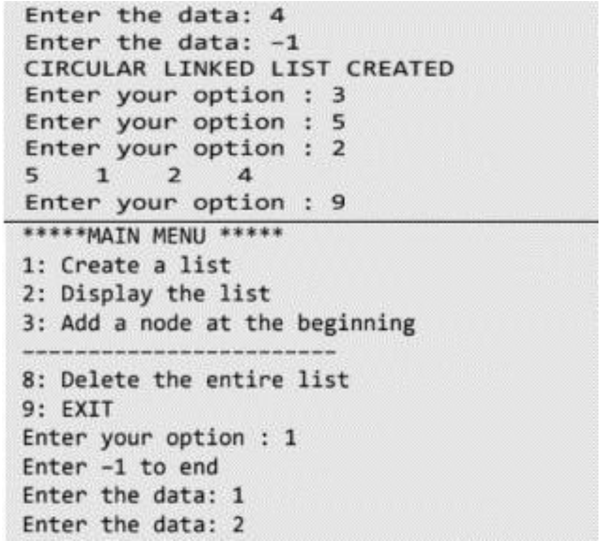
```
 #include
#include  #include  struct node {  int
data;  struct node *next;  }; struct
node *start = NULL;  struct node
*create_cll(struct node *);  struct
node *display(struct node *); struct
node *insert_beg(struct node *);
```

```
struct node *insert_end(struct node
*);  struct node *delete_beg(struct
node *);  struct node
*delete_end(struct node *);  struct
node *delete_after(struct node *);
struct node *delete_list(struct node
*);  int main() {  int option;  clrscr();
do {  printf("\n\n *****MAIN
MENU *****");
printf("\n 1: Create a list");  printf("\n 2:
Display the list");  printf("\n 3: Add a node at
the beginning");  printf("\n 4: Add a node at the
end");  printf("\n 5: Delete a node from the
beginning");  printf("\n 6: Delete a node from
the end");  printf("\n 7: Delete a node after a
given node");  printf("\n 8: Delete the entire
list");  printf("\n 9: EXIT");  printf("\n\n Enter
your option : ");  scanf("%d", &option);
switch(option) {  case 1:
 start = create_cll(start);
 printf("\n CIRCULAR LINKED LIST CREATED");
 break;  case 2:  start =
display(start);  break;
case 3:  start =
insert_beg(start);  break;
case 4:  start =
insert_end(start);  break;

case 5:  start =

delete_beg(start);  break;

case 6:  start =

delete_end(start);  break;

case 7:  start =

delete_after(start);

break;  case 8:

start = delete_list(start);

 printf("\n CIRCULAR LINKED LIST DELETED");

 break;

}

}while(option !=9);
 getch();  return 0; }  struct node *create_cll(struct

node *start)  { struct node *new_node, *ptr;  int num;

printf("\n Enter –1 to end");  printf("\n Enter the data :

");  scanf("%d", &num);  while(num!=–1) {

new_node = (struct node*)malloc(sizeof(struct

node));  new_node -> data = num;  if(start == NULL)

{  new_node -> next = new_node;  start = new_node;

 } else  {  ptr = start;

while(ptr -> next != start)  ptr

= ptr -> next;  ptr -> next =

new_node;  new_node -> next

= start;  } printf("\n Enter the
data : ");  scanf("%d",
&num);
} return start;
 } struct node *display(struct node *start)
{ struct node *ptr; ptr=start;  while(ptr -> next !=
start) {  printf("\t %d", ptr -> data);  ptr = ptr -> next;
}  printf("\t %d", ptr -> data);  return start; }  struct
node *insert_beg(struct node *start) {   struct node
*new_node, *ptr;  int num;  printf("\n Enter the data :
");  scanf("%d", &num);  new_node = (struct node
*)malloc(sizeof(struct node));  new_node -> data =
num;
 ptr = start;  while(ptr ->
next != start)  ptr = ptr ->
next;  ptr -> next =
new_node;  new_node ->
next = start;  start =
new_node;  return start;
} struct node *insert_end(struct node *start) {  struct
node *ptr, *new_node;  int num;  printf("\n Enter the
data : ");  scanf("%d", &num);  new_node = (struct
node *)malloc(sizeof(struct node));  new_node ->
data = num;

 ptr = start;  while(ptr -> next != start)  ptr =
ptr -> next;  ptr -> next = new_node;
```

```
new_node -> next = start;  return start;  }
struct node *delete_beg(struct node *start)
{  struct node *ptr;  ptr = start;  while(ptr ->
next != start)  ptr = ptr -> next;  ptr -> next
= start -> next;  free(start);  start = ptr ->
next;  return start;
} struct node *delete_end(struct node *start)
{  struct node *ptr,*preptr;  ptr = start;
while(ptr -> next != start) {  preptr = ptr;  ptr
= ptr -> next;  }  preptr -> next = ptr -> next;
free(ptr);
 return start;
} struct node *delete_after(struct node *start) {  struct node
*ptr, *preptr;  int val;  printf("\n Enter the value after which the
node has to deleted : ");  scanf("%d", &val);
 ptr = start;  preptr = ptr;
while(preptr -> data != val) {
preptr = ptr;  ptr = ptr -> next;  }
preptr -> next = ptr -> next;  if(ptr
== start) start = preptr -> next;
free(ptr);  return start;
 } struct node *delete_list(struct node *start)
{  struct node *ptr;  ptr = start;  while(ptr ->
next != start)  start = delete_end(start);
free(start);  return start;
```

}

Output:



```
Enter the data: 4
Enter the data: -1
CIRCULAR LINKED LIST CREATED
Enter your option : 3
Enter your option : 5
Enter your option : 2
5    1    2    4
Enter your option : 9
*****MAIN MENU *****
1: Create a list
2: Display the list
3: Add a node at the beginning
------------------------
8: Delete the entire list
9: EXIT
Enter your option : 1
Enter -1 to end
Enter the data: 1
Enter the data: 2
```

Conclusion:

Write an example of insertion and deletion in the circular linked list while traversing the web pages?

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



// Structure for a web page node in the circular linked list

struct WebPage {

   char title[50];

struct WebPage* next;

};
```

```
struct WebPage* current = NULL;


// Function to insert a new web page
void insertPage(char title[]) {     struct WebPage* newPage = (struct

WebPage*)malloc(sizeof(struct WebPage));     strcpy(newPage->title, title);



    if (current == NULL) {         current = newPage;         newPage->next

= newPage;  // Make it point to itself in a circular list.

    } else {         newPage->next =

current->next;         current->next =

newPage;         current = newPage;

    }

}


// Function to delete the current web

page void deletePage() {     if (current

== NULL) {         printf("No web page

to delete.\n");         return;

    }


    struct WebPage* nextPage = current->next;
    if (current == current->next) {
free(current);         current = NULL;
} else {         current->next =
nextPage->next;
free(nextPage);
    }
}

// Function to display the current web
page void displayCurrentPage() {     if
(current == NULL) {         printf("No
current web page.\n");
    } else {         printf("Current Page: %s\n",
current->title);
    }
}

int main() {
insertPage("Home Page");
insertPage("About Us");
insertPage("Contact Us");
```

```
displayCurrentPage();  //

Displays "Contact Us"



   deletePage();          // Deletes "Contact Us"

displayCurrentPage();  // Displays "About Us"


   deletePage();          // Deletes "About Us"

displayCurrentPage();  // Displays "Home Page"


   return 0;

}
```