October 01, 2021

IMPLEMENTATION OF DOUBLY CIRCULAR LINK LIST

—

```
/* creation & Operations On Doubly Circular Linked List
        ->insertion
                ->at begning
                ->at end
                ->at specific position

        -> Deletion
                ->at begning
                ->at end
                ->at specific position

        ->find the length of list
        ->reverce the list
// Created By SMIT
*/


#include<stdio.h>
#include<stdlib.h>

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

struct node *head=0,*tail,*temp;

void CreateDcll();      //creates Doubly Circular link List
```

```
30. void DisplayDcll();     //Displays Doubly Circular Link List
31. void InsAtBeg();        //Insert At Begning
32. void InsAtPos();        //Insert At given Position
33. void InsAtEnd();        //insert at end
34. void DelAtBeg();        //Delete At Begning
35. void DelAtEnd();        //Delete At End
36. void DelAtPos();        //Delete At Position
37. void GetLen();          //Get The Length of List
38. void ReverseDcll();     //Reverses The Link List
39.
40. //Main Program
41. int main(){
42.         int choice=1;
43.         while(choice!=11){
44.                 printf("\n\n***MAIN MENU***\n\n");
45.                 printf("1. Create Doubly Circular Link list\n");
46.                 printf("2. Display\n");
47.                 printf("3. Insert At Begning\n");
48.                 printf("4. Insert At Position\n");
49.                 printf("5. Insert At end\n");
50.                 printf("6. Delete At Begning\n");
51.                 printf("7. Delete At Position\n");
52.                 printf("8. Delete At End\n");
53.                 printf("9. Get Length Of list\n");
54.                 printf("10. Reverse Link List\n");
55.                 printf("11. exit\n");
56.                 printf("Enter Choice ");
57.                 scanf("%d",&choice);
58.                 switch(choice){
59.                         case 1:
60.                                 CreateDcll();
61.                                 break;
62.                         case 2:
63.                                 DisplayDcll();
64.                                 break;
65.                         case 3:
66.                                 InsAtBeg();
67.                                 break;
68.                         case 4:
69.                                 InsAtPos();
70.                                 break;
71.                         case 5:
72.                                 InsAtEnd();
73.                                 break;
74.                         case 6:
```

```
75.                                     DelAtBeg();
76.                                     break;
77.                             case 7:
78.                                     DelAtPos();
79.                             case 8:
80.                                     DelAtEnd();
81.                                     break;
82.                             case 9:
83.                                     GetLen();
84.                                     break;
85.                             case 10:
86.                                     ReverseDcll();
87.                                     break;
88.                             case 11:
89.                                     exit(0);
90.                                     break;
91.                             default:
92.                                     printf("error");
93.                     }
94.             }
95.             return 0;
96. }
97.
98. // To create Dcll
99. void CreateDcll(){
100.            struct node *newnode;
101.            int choice=1;
102.            while(choice){
103.                    newnode=(struct node*)malloc(sizeof(struct node));
104.                    printf("Enter Data ");
105.                    scanf("%d",&newnode->data);
106.                    if(head==0){
107.                            head=tail=newnode;
108.                            newnode->next=head;
109.                            newnode->prev=tail;
110.                    }
111.                    else{
112.                            newnode->next=head;
113.                            newnode->prev=tail;
114.                            head->prev=newnode;
115.                            tail->next=newnode;
116.                            tail=newnode;
117.                    }
118.                    printf("0 to exit 1 to add ");
119.                    scanf("%d",&choice);
```

```
120.         }
121. }
122.
123. //To insert At Begning
124. void InsAtBeg(){
125.         struct node *newnode;
126.         newnode=(struct node*)malloc(sizeof(struct node));
127.         printf("Enter data ");
128.         scanf("%d",&newnode->data);
129.         newnode->prev=tail;
130.         newnode->next=head;
131.         if(head==0){
132.                 head=tail=newnode;
133.         }
134.         else{
135.                 head->prev=newnode;
136.                 tail->next=newnode;
137.                 head=newnode;
138.         }
139.         printf("\nCreated Successfully");
140. }
141.
142.
143. //To insert At end
144. void InsAtEnd(){
145.         struct node *newnode;
146.         temp=head;
147.         newnode=(struct node*)malloc(sizeof(struct node));
148.                 printf("Enter Data ");
149.                 scanf("%d",&newnode->data);
150.         if(head==0){
151.                 head=tail=newnode;
152.                 newnode->next=head;
153.                 newnode->prev=head;
154.         }
155.         else{
156.                 newnode->next=head;
157.                 newnode->prev=tail;
158.                 tail->next=newnode;
159.                 head->prev=newnode;
160.                 tail=newnode;
161.         }
162.         printf("\nCreated Successfully");
163. }
164.
```

```
165. //To insert at given position
166. void InsAtPos(){
167.                 struct node *newnode;
168.                 int i,pos;
169.                 temp=head;
170.                 printf("enter position ");
171.                 scanf("%d",&pos);
172.                 for(i=1;i<pos-1;i++){
173.                         temp=temp->next;
174.                 }
175.                 if(pos==1){
176.                         InsAtBeg();
177.                 }
178.                 else{
179.                         newnode=(struct node*)malloc(sizeof(struct node));
180.                         printf("Enter Data ");
181.                         scanf("%d",&newnode->data);
182.                         newnode->next=temp->next;
183.                         newnode->prev=temp;
184.                         temp->next->prev=newnode;
185.                         temp->next=newnode;
186.                         printf("\nCreated Successfully");
187.                 }
188. }
189.
190. //To Delete At begning
191. void DelAtBeg(){
192.         if(head==0){
193.                 printf("list is empty");
194.         }
195.         else{
196.                 temp=head;
197.                 head=head->next;
198.                 head->prev=tail;
199.                 tail->next=head;
200.                 free(temp);
201.                 printf("\nDeleted Successfully");
202.         }
203. }
204.
205.
206. //To delete At End
207. void DelAtEnd(){
208.         if(head==0){
209.                 printf("list is empty");
```

```
210.         }
211.         else{
212.                 temp=tail;
213.                 tail=tail->prev;
214.                 tail->next=head;
215.                 head->prev=tail;
216.                 free(temp);
217.                 printf("\nDeleted Successfully");
218.         }
219. }
220.
221.
222. //To delete At given Position
223. void DelAtPos(){
224.         int pos,i;
225.         struct node *prev,*next;
226.         printf("Entro position ");
227.         scanf("%d",&pos);
228.         if(pos==1){
229.                 DelAtBeg();
230.         }
231.         else{
232.                 temp=head;
233.                 for(i=1;i<pos;i++){
234.                         temp=temp->next;
235.                 }
236.                 prev=temp->prev;
237.                 next=temp->next;
238.                 prev->next=temp->next;
239.                 next->prev=temp->prev;
240.                 free(temp);
241.                 printf("\nDeleted Successfully");
242.         }
243. }
244.
245.
246. //To get the length of Dcll
247. void GetLen(){
248.         temp=head;
249.         int count=1;
250.         if(head==0){
251.                 printf("list is empty");
252.         }
253.         else{
254.                 do{
```

```
                        temp=temp->next;
                        count++;
                }while(temp->next!=tail->next);
                printf("%d",count);
        }
}

//To Display Dcll
void DisplayDcll(){
        temp=head;
        do{
                printf("%d ",temp->data);
                temp=temp->next;
        }while(temp!=tail->next);
}

//To reverse Link list
void ReverseDcll(){
        struct node *nextnode;
        if(head==0){
                printf("list is empty");
        }
        else{
                temp=head;
                temp=temp->next;
                while(temp!=head){
                        nextnode=temp->next;
                        temp->next=temp->prev;
                        temp->prev=nextnode;
                        temp=nextnode;
                }
                nextnode=temp->next;
                temp->next=temp->prev;
                temp->prev=nextnode;
                head=tail;
                tail=temp;
                printf("link revresed successfully");
        }
}
//Created By SMIT
```

OUTPUT

run the code in your machine.

notify if you found error during run time