Circular Linked List

Source Code :

```
#include<iostream>
#include<malloc.h>

using namespace std;

//creating structure to use for the list
struct node {
   struct node *next;
   int data;
}*list=NULL,*p,*q,*r;
string line ="----------------------------------------";
int counter=0;

//Declaring function prototypes
void insertFirst();
void insertLast();
void display();
void insertAfter();
void insertBefore();
void deleteFirst();
void delLast();
void del();
void sortList();
void countElements();
void reverse();

class circular
{
   public:
   int ch,data1,data2;

   void menu()
   {
          do
          {
                 cout<<"\n\n"<<line<<"\n\t\tMenu:\n"<<line<<"\n1. Enter element at beginning\n2. Enter
element at end\n3. Enter element after an element\n4. Enter element before an element\n5. Delete an
element\n6. Delete First element\n7. Delete Last element\n8. Display\n9. Sort List\n10. Count the
Elements\n11. Reverse the List\n12. Exit\n\nEnter your choice: ";
                 cin>>ch;

                 switch(ch) {
                        case 1:
                               insertFirst();
```

```
            display();
            break;

case 2:
            insertLast();
            display();
            break;

case 3:
            insertAfter();
            display();
            break;

case 4:
            insertBefore();
            display();
            break;

case 5:
            del();
            display();
            break;

case 6:
            deleteStart();
            display();
            break;

case 7:
            deleteLast();
            display();
            break;

case 8:
            display();
            break;

case 9:
            sortList();
            break;

case 10:
            countElements();
            break;

case 11:
            reverse();
            display();
```

```
                                        break;


                                default:
                                        cout<<"Invalid Choice\n";
                                        break;
                        }
                } while(ch != 12);
}

void insertFirst() {
        p=(struct node*)malloc(sizeof(node));

        cout<<"Enter element: ";
        cin>>data1;
        p->data=data1;

        if(list==NULL) {
                p->next=p;
                list=p;
        } else {
                q=list;
                while(q->next!=list) {
                        q=q->next;
                }
                p->next=list;
                q->next=p;
                list=p;
        }
}

void insertAfter() {
        if(list==NULL) {
                cout<<"LIST IS EMPTY";
        } else {
                p=(struct node*) malloc(sizeof(node));
                cout<<"Enter element to insert: ";
                cin>>data1;
                p->data=data1;
                cout<<"Enter the element after which you want to insert: ";
                cin>>data2;

                q=list;
                while(q->data!=data2 && q->next!=list) {
                        q=q->next;
                        r=q->next;
                }
```

```
                if(q->next==list && q->data==data2) {
                        q->next=p;
                        p->next=list;
                } else if (q->data==data2) {
                        q->next=p;
                        p->next=r;
                } else {
                        cout<<"LIST ENDED, ELEMENT NOT FOUND";
                }
        }
}

void insertBefore() {
        if(list==NULL) {
                cout<<"LIST IS EMPTY";
        } else {
                p=(struct node*) malloc(sizeof(node));
                cout<<"Enter element to insert: ";
                cin>>data1;
                p->data=data1;
                cout<<"Enter the element before which you want to insert: ";
                cin>>data2;
                q=list;

                while(q->data!=data2 && q->next!=list) {
                        r=q;
                        q=q->next;
                }

                if(q->next==list && q->data!=data2) {
                        cout<<"LIST ENDED, ELEMENT NOT FOUND";
                } else if (q->data==data2 && list==q) {
                        p->next=q;
                        list=p;
                } else if (q->data==data2 && list!=q) {
                        r->next=p;
                        p->next=q;
                }
        }
}

void insertLast() {
        p=(struct node*) malloc(sizeof(node));
        cout<<"Enter element: ";
        cin>>data1;
        p->data=data1;

        if(list==NULL) {
```

```
                p->next=p;
                list=p;
        } else {
                q=list;
                while(q->next!=list) {
                        q=q->next;
                }
                q->next=p;
                p->next=list;
        }
}

void deleteStart() {
        if(list==NULL) {
                cout<<"CANNOT DELETE, LIST IS EMPTY";
        } else {
                q=list;
                if(list->next==list) {
                        delete(q);
                        cout<<"Element Deleted";
                        list=NULL;
                } else {
                        while(q->next!=list) {
                                q=q->next;
                        }
                        q->next=list->next;
                        delete(list);
                        cout<<"Element Deleted";
                        list=q->next;
                }
        }
}

void deleteLast() {
        if(list==NULL) {
                cout<<"CANNOT DELETE, LIST IS EMPTY";
        } else if(list->next==list) {
                deleteStart();
        } else {
                q=list;
                while(q->next!=list) {
                        r=q;
                        q=q->next;
                }
                r->next=list;
                delete(q);
                cout<<"Element Deleted";
        }
```

```
}

void del() {
	if(list==NULL) {
		cout<<"CANNOT DELETE, LIST IS EMPTY";
	} else {
		q=list;
		cout<<"Enter the element you want to delete: ";
		cin>>data1;

		while(q->data!=data1 && q->next!=list) {
			r=q;
		q=q->next;
	}

		if(q==list && q->data==data1) {
		deleteStart();
		} else if(q->next==list && q->data==data1) {
		deleteLast();
		} else if(q==NULL) {
			cout<<"Element Not Found ";
		} else {
			p=q->next;
			r->next=p;
			delete(q);
			cout<<"Element Deleted";
		}
	}
}

void display() {
	if(list==NULL && ch!=3 && ch!=6 && ch!=5 && ch!=7 && ch!=4) {
		cout<<"LIST IS EMPTY";
	} else {
		q=list;
		int flag=0;
		counter=0;
		cout<<endl;
		while(flag!=1)
		{
			if(q->next==list)
				flag=1;
			cout<<q->data<<"-->";
			counter++;
			q=q->next;
		}
	}
}
```

```
void sortList() {
	if(list==NULL) {
		display();
	} else {
		q=list;
		do {
			r=q->next;
			do {
				if(r->data<q->data) {
					int temp=r->data;
					r->data=q->data;
					q->data=temp;
				}
				r=r->next;
			} while(r!=list);
			q=q->next;
		} while(q->next!=list);
		display();
	}
}

void reverse() {
	if(list==NULL){
		cout<<"LIST IS EMPTY";
	} else {
		p=list;
		q=p->next;
		r=q->next;

		while(r->next!=list) {
			q->next=p;
			p=q;
			q=r;
			r=r->next;
		}

		if(r->next==list) {
			q->next=p;
			r->next=q;
			list->next=r;
			list=r;
		}
	}
}

void countElements() {
	display();
```

```
        cout<<"\nNumber of Elements : "<<counter;
    }

};

int main() {
    cout<<"\n"<<line<<"\n\tCIRCULAR LINKED LIST\n"<<line;
    circular obj;
    obj.menu();
    return 0;
}
```

Output :

```
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit

Enter your choice: 1
Enter element: 10

10-->

----------------------------------------
        Menu:
----------------------------------------
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit
```

```
Enter your choice: 2
Enter element: 20

10-->20-->

----------------------------------------
            Menu:
----------------------------------------
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit

Enter your choice: 3
Enter element to insert: 30
Enter the element after which you want to insert: 20

10-->20-->30-->

----------------------------------------
            Menu:
----------------------------------------
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit

Enter your choice: 4
Enter element to insert: 40
Enter the element before which you want to insert: 20

10-->40-->20-->30-->
```

```
----------------------------------------
            Menu:
----------------------------------------
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit

Enter your choice: 5
Enter the element you want to delete: 20
Element Deleted
10-->40-->30-->

----------------------------------------
            Menu:
----------------------------------------
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit

Enter your choice: 6
Element Deleted
40-->30-->

----------------------------------------
            Menu:
----------------------------------------
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
```

4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit

Enter your choice: 7
Element Deleted
40-->

Menu:
---
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit

Enter your choice: 8

40-->

Menu:
---
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List

12. Exit

Enter your choice: 9

40-->

Menu:
---
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit

Enter your choice: 10

40-->
Number of Elements : 1

Menu:
---
1. Enter element at beginning
2. Enter element at end
3. Enter element after an element
4. Enter element before an element
5. Delete an element
6. Delete First element
7. Delete Last element
8. Display
9. Sort List
10. Count the Elements
11. Reverse the List
12. Exit