



Discover. Learn. Empower.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Experiment-4

Student Name: Gurveer Singh Mangat

UID: 23BCS11074

Branch: B.E-C.S.E

Section/Group: 23KRG-1A

Semester: 5th

Date of Performance: 25/08/2025

Subject Name: DAA

Subject Code: 23CSH-301

1. Aim: Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List.

2. Objective: To understand doubly and circular linked list

3. Input/Apparatus Used: Doubly and circular Linked List is used.

4. Procedure/Algorithm: Pseudocode:

Procedure for beginning of circular linked list:

Step1. Create the new node

Step2. Set the new node's next to itself (circular) Step3. If the list is empty, return new node.

Step4. Set our new node's next to the front. Step5. Set tail's next to our new node.

Step6. Return the end of the list.

Procedure for end of circular linked list:

Step1. Create the new node

Step2. Set the new node's next to itself (circular) Step3. If the list is empty, return new node.

Step4. Set our new node's next to the front. Step5. Set tail's next to our new node.

Step6. Return the end of the list.

5. Code:

Exp 4 > C++ Exp_4.cpp > ...

```
1  #include <iostream>
2  using namespace std;
3  class DoublyNode {
4  public:
5      int data;
6      DoublyNode* prev;
7      DoublyNode* next;
8      DoublyNode(int val) : data(val), prev(NULL), next(NULL) {}
9  };
10 class DoublyLinkedList {
11     DoublyNode* head;
12     DoublyNode* tail;
13 public:
14     DoublyLinkedList() : head(NULL), tail(NULL) {}
15     void insertAtBeginning(int val) {
16         DoublyNode* node = new DoublyNode(val);
17         if (!head) {
18             head = tail = node;
19         } else {
20             node->next = head;
21             head->prev = node;
22             head = node;
23         }
24     }
25     void insertAtEnd(int val) {
26         DoublyNode* node = new DoublyNode(val);
27         if (!tail) {
28             head = tail = node;
29         } else {
30             tail->next = node;
31             node->prev = tail;
```

```
32         tail = node;
33     }
34 }
35 void deleteAtBeginning() {
36     if (!head) return;
37     DoublyNode* temp = head;
38     if (head == tail) {
39         head = tail = NULL;
40     } else {
41         head = head->next;
42         head->prev = NULL;
43     }
44     delete temp;
45 }
46 void deleteAtEnd() {
47     if (!tail) return;
48     DoublyNode* temp = tail;
49     if (head == tail) {
50         head = tail = NULL;
51     } else {
52         tail = tail->prev;
53         tail->next = NULL;
54     }
55     delete temp;
56 }
57 void display() {
58     DoublyNode* curr = head;
59     while (curr) {
```

```
60         cout << curr->data << " ";
61         curr = curr->next;
62     }
63     cout << endl;
64 }
65 };
66 class CircularNode {
67 public:
68     int data;
69     CircularNode* next;
70     CircularNode(int val) : data(val), next(NULL) {}
71 };
72 class CircularLinkedList {
73     CircularNode* tail;
74 public:
75     CircularLinkedList() : tail(NULL) {}
76     void insertAtBeginning(int val) {
77         CircularNode* node = new CircularNode(val);
78         if (!tail) {
79             tail = node;
80             tail->next = tail;
81         } else {
82             node->next = tail->next;
83             tail->next = node;
84         }
85     }
86     void insertAtEnd(int val) {
87         CircularNode* node = new CircularNode(val);
```

```
88         if (!tail) {
89             tail = node;
90             tail->next = tail;
91         } else {
92             node->next = tail->next;
93             tail->next = node;
94             tail = node;
95         }
96     }
97     void deleteAtBeginning() {
98         if (!tail) return;
99         CircularNode* head = tail->next;
100        if (tail == head) {
101            delete head;
102            tail = NULL;
103        } else {
104            tail->next = head->next;
105            delete head;
106        }
107    }
108    void deleteAtEnd() {
109        if (!tail) return;
110        CircularNode* curr = tail->next;
111        if (curr == tail) {
112            delete tail;
113            tail = NULL;
114        } else {
115            while (curr->next != tail) {
116                curr = curr->next;
```

```
117         }
118         curr->next = tail->next;
119         delete tail;
120         tail = curr;
121     }
122 }
123 void display() {
124     if (!tail) {
125         cout << "List empty\n";
126         return;
127     }
128     CircularNode* head = tail->next;
129     CircularNode* curr = head;
130     do {
131         cout << curr->data << " ";
132         curr = curr->next;
133     } while (curr != head);
134     cout << endl;
135 }
136 };
137 int main() {
138     cout << "Doubly Linked List:\n";
139     DoublyLinkedList dll;
140     dll.insertAtBeginning(10);
141     dll.insertAtEnd(20);
142     dll.insertAtBeginning(5);
143     dll.display();
```

```
144     dll.deleteAtBeginning();
145     dll.deleteAtEnd();
146     dll.display();
147
148     cout << "\nCircular Linked List:\n";
149     CircularLinkedList cll;
150     cll.insertAtBeginning(10);
151     cll.insertAtEnd(20);
152     cll.insertAtBeginning(5);
153     cll.display();
154     cll.deleteAtBeginning();
155     cll.deleteAtEnd();
156     cll.display();
157
158     return 0;
159 }
160
```

6. Output:

```
PS D:\DAA> cd "d:\DAA\Exp 4\" ; if ($?) { g++ Exp_4.cpp -o Exp_4 } ; if ($?) { .\Exp_4 }  
● Doubly Linked List:  
5 10 20  
10  
  
Circular Linked List:  
5 10 20  
10
```