

```
doublylinkedlist.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran wsSmith Tools Tools+ Plugins DocyBlocks Settings Help
Start here X doublylinkedlist.c X
1 #include <stdio.h>
2 #include <stdlib.h>
3
4
5 struct Node {
6     int data;
7     struct Node *prev;
8     struct Node *next;
9 };
10
11
12
13
14 struct Node* createList(struct Node* head, int data) {
15     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
16     newNode->data = data;
17     newNode->prev = NULL;
18     newNode->next = NULL;
19
20     if (head == NULL) {
21         return newNode;
22     }
23
24     struct Node* temp = head;
25     while (temp->next != NULL)
26         temp = temp->next;
27
28     temp->next = newNode;
29     newNode->prev = temp;
30
31     return head;
32 }
33
34
35 struct Node* insertAtBeginning(struct Node* head, int data) {
36     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
37     newNode->data = data;
38     newNode->prev = NULL;
39     newNode->next = head;
40
41     if (head != NULL)
42         head->prev = newNode;
43
44     head = newNode;
45
46     return head;
47 }
48
49
50
51 struct Node* deleteNode(struct Node* head, int value) {
52     struct Node* temp = head;
53
54     while (temp != NULL && temp->data != value)
```

```
doublylinkedlist.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran wsSmith Tools Tools+ Plugins DocyBlocks Settings Help
Start here X doublylinkedlist.c X
52     struct Node* temp = head;
53
54     while (temp != NULL && temp->data != value)
55         temp = temp->next;
56
57     if (temp == NULL) {
58         printf("Node with value %d not found!\n", value);
59         return head;
60     }
61
62     if (temp->prev != NULL)
63         temp->prev->next = temp->next;
64     else
65         head = temp->next;
66
67     if (temp->next != NULL)
68         temp->next->prev = temp->prev;
69
70     free(temp);
71     printf("Node %d deleted.\n", value);
72
73     return head;
74 }
75
76
77 void display(struct Node* head) {
78     struct Node* temp = head;
79
80     if (head == NULL) {
81         printf("List is empty.\n");
82         return;
83     }
84
85     printf("Doubly Linked List: ");
86     while (temp != NULL) {
87         printf("%d <> ", temp->data);
88         temp = temp->next;
89     }
90     printf("NULL\n");
91 }
92
93
94 int main() {
95     struct Node* head = NULL;
96     int choice, value, key;
97
98     while (1) {
99         printf("\n--- MENU ---\n");
100         printf("1. Create Node\n");
101         printf("2. Insert at Beginning\n");
102         printf("3. Delete Node by Value\n");
103         printf("4. Display List\n");
104         printf("5. Exit\n");
105         printf("Enter choice: ");
```

```
doublylinkedlist.c - CodeBlocks 20.03
File Edit View Search Project Build Debug Fortran wsSmith Tools Plugins DocyBlocks Settings Help
Start here X doublylinkedlist.c X
90 printf("\n\n");
91 }
92
93
94 int main() {
95     struct Node* head = NULL;
96     int choice, value, key;
97
98     while (1) {
99         printf("\n--- MENU ---\n");
100         printf("1. Create Node\n");
101         printf("2. Insert at beginning\n");
102         printf("3. Delete Node by Value\n");
103         printf("4. Display List\n");
104         printf("5. Exit\n");
105         printf("Enter choice: ");
106         scanf("%d", &choice);
107
108         switch (choice) {
109             case 1:
110                 printf("Enter value to insert: ");
111                 scanf("%d", &value);
112                 head = createNode(head, value);
113                 break;
114
115             case 2:
116                 printf("Enter value to insert at beginning: ");
117                 scanf("%d", &value);
118                 head = insertAtBeginning(head, value);
119                 break;
120
121             case 3:
122                 printf("Enter value to delete: ");
123                 scanf("%d", &value);
124                 head = deleteNode(head, value);
125                 break;
126
127             case 4:
128                 display(head);
129                 break;
130
131             case 5:
132                 exit(0);
133
134             default:
135                 printf("Invalid choice!\n");
136         }
137     }
138
139     return 0;
140 }
141
142
```

```
D:\chethan\DP\doublylinkedlist.c
--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 1
Enter value to insert: 233
--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 1
Enter value to insert: 566
--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 4
Doubly Linked List: 233 <-> 566 <-> NULL
--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 2
Enter value to insert at beginning: 1222
--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 3
Enter value to delete: 233
Node 233 Deleted.
--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice: 4
Doubly Linked List: 1222 <-> 566 <-> NULL
--- MENU ---
1. Create Node
2. Insert at beginning
3. Delete Node by Value
4. Display List
5. Exit
Enter choice:
```

LeetCode - Problem: Linked List Cycle

Accepted 29 / 29 testcases passed

chethanm23 submitted at Dec 08, 2025 09:32

Runtime: 8 ms Beats 80.25% Memory: 11.89 MB Beats 51.63%

Analyze Complexity

Code: C++

```
1 class Solution {
2 public:
3     bool hasCycle(ListNode *head) {
4         if (!head || !head->next)
5             return false;
6
7         ListNode* slow = head;
8         ListNode* fast = head;
9
10        while (fast && fast->next) {
11            slow = slow->next;
12            fast = fast->next->next;
13
14            if (slow == fast)
15                return true;
16        }
17        return false;
18    }
19 };
20
```

More challenges

- 142. Linked List Cycle II
- 202. Happy Number

Testcase Test Result

3) Implementation of double linked list

pseudocode

Function createNode (int n):

create a node = (newNode)

newNode->data = data

newNode->prev = newNode->next = NULL

if (tail == NULL) head = tail = newNode;

else

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

end

Function insertAtBeginning (int data)

if head is null

head = tail = newNode;

else

head->prev = newNode;

~~head~~ newNode->next = head;

head = newNode

end if

Function insertAtEnd (int data):

if (tail == NULL)

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

end if

Function ~~Delete~~ (int val):

Start temp = head

while (temp != null && temp->data != val) {

temp = temp->next;

}

temp->prev->next = temp->next

temp->next->prev = temp->prev



Shot on OnePlus

chE Reddy

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

```
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};
```

```
struct Node* createlist(struct Node* head, int data)
```

```
{
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    if (head == NULL) {
        return newNode;
    }
```

```
    struct Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
    return head;
}
```

```
struct Node* insertAtBeginning(struct Node* head, int data) {  
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
```

```
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = head;
    if (head != NULL)
        head->prev = newNode;
    return head;
```



Shot on OnePlus

chE_Reddy

DATE: _____

```

struct node* deleteNode(struct node* head, int value) {
    struct node* temp = head;
    while (temp != NULL && temp->data != value)
        temp = temp->next;
    if (temp == NULL) {
        printf("Node with value %d not found.\n", value);
        return head;
    }
    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    else
        head = temp->next;
    if (temp->next != NULL)
        temp->next->prev = temp->prev;
    free(temp);
    printf("Node %d deleted.\n", value);
    return head;
}

```

```

void display(struct node* head) {
    struct node* temp = head;
    if (head == NULL)
        printf("List is empty.\n");
    return;
}

printf("Doubly linked list:");
while (temp != NULL) {
    printf("%d <-> ", temp->data);
    temp = temp->next;
}

printf("NULL\n");

```



DATE: PAGE:

```

int main() {
    struct Node * head = NULL;
    int choice, value, key;
    while (1) {
        printf("\n---Menu---\n");
        printf("1. Create Node\n");
        printf("2. Insert at beginning\n");
        printf("3. Delete Node by value\n");
        printf("4. Display list\n");
        printf("5. Exit\n");
        printf("Enter choice:");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter value to insert:");
                scanf("%d", &value);
                head = createList(head, value);
                break;
            case 2:
                printf("Enter value to insert at beginning:");
                scanf("%d", &value);
                head = insertatbegining(head, value);
                break;
            case 3:
                printf("Enter value to delete:");
                scanf("%d", &value);
                head = deleteNode(head, value);
                break;
            case 4:
                display(head);
                break;
            case 5:
                exit(0);
        }
    }
}

```



Output:

-- Menu --

1. Create Node
2. Insert at beginning
3. Delete Node by value
4. Display List
5. Exit

Enter choice: 1

Enter value to insert: 23

Enter choice: 1

Enter value to insert: 45

Enter choice: 4

Doubly Linked list: 23C-45C-NULL

Enter choice: 3

Enter value to delete: 45

Node 45 deleted

Enter your choice: 4

Doubly Linked list: 23C-NULL

Enter choice: 3

Enter value to delete: 444

Node with value 444 not found!

Enter your choice: 1

Enter value to insert: 555

Enter your choice: 4

Doubly Linked list: 23C-555C-2 NULL



Shot on OnePlus

chE_Reddy

8) To construct a binary search tree and traverse the tree using inorder, preorder, postorder and display elements in the tree.

Leetcode : linked list cycle

class solution {

public:

bool hasCycle(ListNode * head) {

if (!head || !head->next)

return false;

ListNode * slow = head

ListNode * fast = head;

while (fast && fast->next)

slow = slow->next;

fast = fast->next->next;

if (slow == fast)

return true;

}

return false;

}

}

~~Q~~
8/11/15

