

Folder SEM 3\Exp2

1 printable files

(file list disabled)

SEM 3\Exp2\DLL_implementation.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  // Node structure for the doubly linked list
5  struct Node
6  {
7      int data;
8      struct Node *prev;
9      struct Node *next;
10 };
11
12 // Insert at the end of the doubly linked list
13 void insert(struct Node **head_ref, int new_data)
14 {
15     struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
16     struct Node *last = *head_ref;
17     new_node->data = new_data;
18     new_node->next = NULL;
19
20     if (*head_ref == NULL)
21     {
22         new_node->prev = NULL;
23         *head_ref = new_node;
24         return;
25     }
26
27     while (last->next != NULL)
28         last = last->next;
29
30     last->next = new_node;
31     new_node->prev = last;
32 }
33
34 // Display the doubly linked list
35 void display(struct Node *node)
36 {
37     struct Node *last;
38     printf("Traversal in forward direction:\n");
39     while (node != NULL)
40     {
41         printf("%d ", node->data);
42         last = node;
43         node = node->next;
44     }
45     printf("\n");
```

```
46 }
47
48 // Delete a node from the doubly linked list
49 void deleteNode(struct Node **head_ref, int key)
50 {
51     struct Node *temp = *head_ref;
52
53     if (*head_ref == NULL)
54         return;
55
56     while (temp != NULL && temp->data != key)
57         temp = temp->next;
58
59     if (temp == NULL)
60         return;
61
62     if (*head_ref == temp)
63         *head_ref = temp->next;
64
65     if (temp->next != NULL)
66         temp->next->prev = temp->prev;
67
68     if (temp->prev != NULL)
69         temp->prev->next = temp->next;
70
71     free(temp);
72 }
73
74 // Search for a key in the doubly linked list
75 void search(struct Node *head, int key)
76 {
77     struct Node *temp = head;
78     int pos = 0;
79     while (temp != NULL)
80     {
81         if (temp->data == key)
82         {
83             printf("Element %d found at position %d\n", key, pos);
84             return;
85         }
86         temp = temp->next;
87         pos++;
88     }
89     printf("Element %d not found in the list\n", key);
90 }
91
92 // Count the number of nodes in the doubly linked list
93 int count(struct Node *head)
94 {
95     int count = 0;
96     struct Node *temp = head;
97     while (temp != NULL)
98     {
99         count++;
```

```
100     temp = temp→next;
101 }
102 return count;
103 }
104
105 int main()
106 {
107     struct Node *head = NULL;
108     int choice, value, key;
109
110     while (1)
111     {
112         printf("\nDoubly Linked List Operations:\n");
113         printf("1. Insert\n");
114         printf("2. Display\n");
115         printf("3. Delete\n");
116         printf("4. Search\n");
117         printf("5. Count\n");
118         printf("6. Exit\n");
119         printf("Enter your choice: ");
120         scanf("%d", &choice);
121
122         switch (choice)
123         {
124             case 1:
125                 printf("Enter the value to insert: ");
126                 scanf("%d", &value);
127                 insert(&head, value);
128                 break;
129             case 2:
130                 display(head);
131                 break;
132             case 3:
133                 printf("Enter the value to delete: ");
134                 scanf("%d", &key);
135                 deleteNode(&head, key);
136                 break;
137             case 4:
138                 printf("Enter the value to search: ");
139                 scanf("%d", &key);
140                 search(head, key);
141                 break;
142             case 5:
143                 printf("The number of nodes in the list: %d\n", count(head));
144                 break;
145             case 6:
146                 exit(0);
147             default:
148                 printf("Invalid choice!\n");
149         }
150     }
151
152     return 0;
153 }
```

