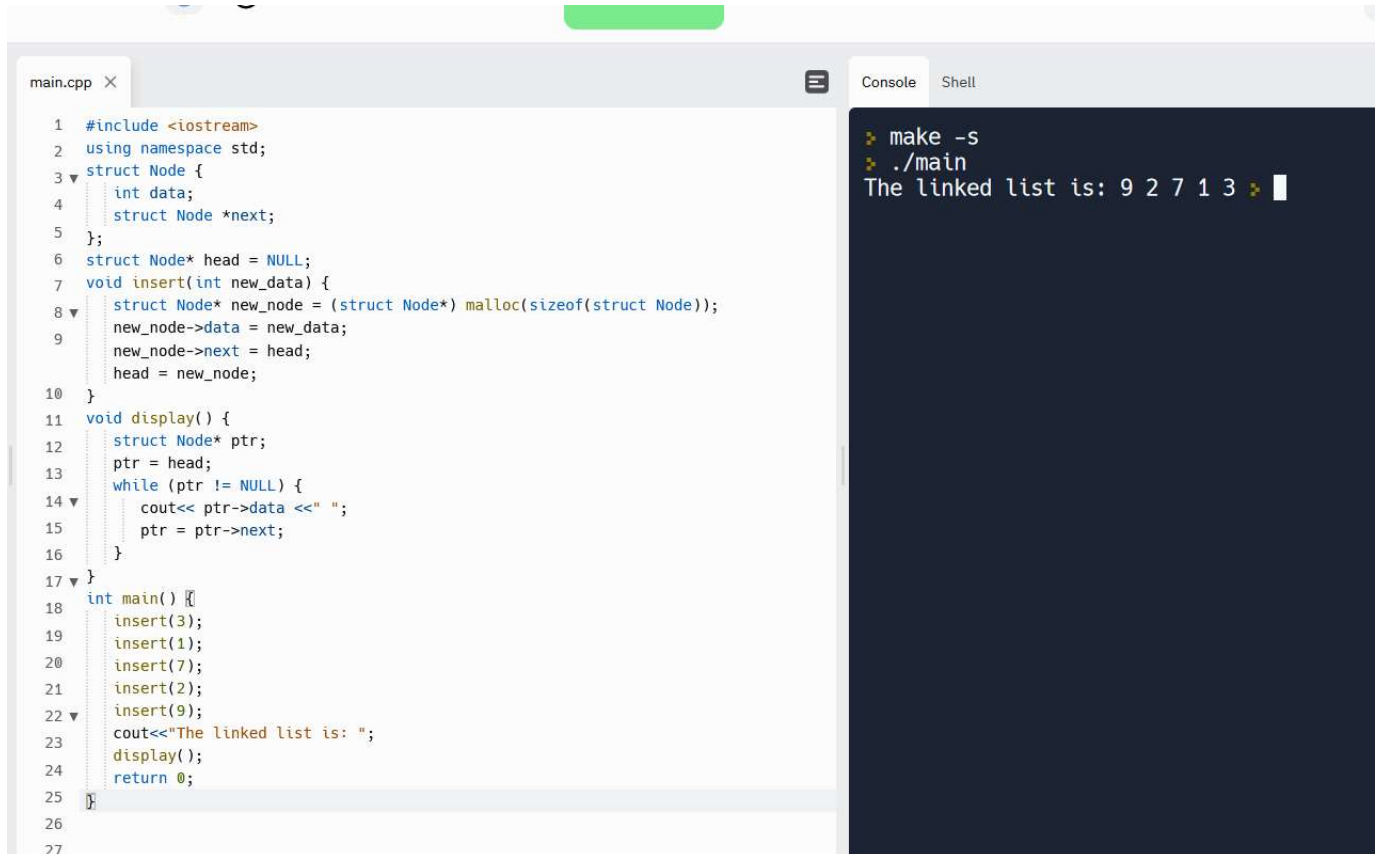


AMAN PRAJAPATI

Linked List Implementation

C++ Program to Implement Singly Linked List



```
main.cpp x
1 #include <iostream>
2 using namespace std;
3 struct Node {
4     int data;
5     struct Node *next;
6 };
7 struct Node* head = NULL;
8 void insert(int new_data) {
9     struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
10    new_node->data = new_data;
11    new_node->next = head;
12    head = new_node;
13 }
14 void display() {
15     struct Node* ptr;
16     ptr = head;
17     while (ptr != NULL) {
18         cout<< ptr->data <<" ";
19         ptr = ptr->next;
20     }
21 }
22 int main() {
23     insert(3);
24     insert(1);
25     insert(7);
26     insert(2);
27     insert(9);
28     cout<<"The linked list is: ";
29     display();
30     return 0;
31 }
```

```
❏ make -s
❏ ./main
The linked list is: 9 2 7 1 3 ❏
```

Implement Doubly Linked List

```
main.cpp X Console Shell

1  #include <iostream>
2  using namespace std;
3  struct Node {
4      int data;
5      struct Node *prev;
6      struct Node *next;
7  };
8  struct Node* head = NULL;
9  void insert(int newdata) {
10     struct Node* newnode = (struct Node*) malloc(sizeof(struct Node));
11     newnode->data = newdata;
12     newnode->prev = NULL;
13     newnode->next = head;
14     if(head != NULL)
15         head->prev = newnode ;
16     head = newnode;
17 }
18 void display() {
19     struct Node* ptr;
20     ptr = head;
21     while(ptr != NULL) {
22         cout<< ptr->data <<" ";
23         ptr = ptr->next;
24     }
25 }
26 int main() {
27     insert(3);
28     insert(1);
29     insert(7);
30     insert(2);
31     insert(9);
32     cout<<"The doubly linked list is: ";
33     display();
34     return 0;
}
```

```
➤ make -s
➤ ./main
The doubly linked list is: 9 2 7 1 3 ➤
```

Circular Linked List

```

1  #include <iostream>
2  using namespace std;
3  struct Node {
4      int data;
5      struct Node* next;
6  };
7  struct Node* addToEmpty(struct Node* last, int data) {
8      if (last != NULL) return last;
9      struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
10     newNode->data = data;
11     last = newNode;
12     last->next = last;
13     return last;
14 }
15 struct Node* addFront(struct Node* last, int data) {
16     if (last == NULL) return addToEmpty(last, data);
17     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
18     newNode->data = data;
19     newNode->next = last->next;
20     last->next = newNode;
21     return last;
22 }
23 struct Node* addEnd(struct Node* last, int data) {
24     if (last == NULL) return addToEmpty(last, data);
25     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
26     newNode->data = data;
27     newNode->next = last->next;
28     last->next = newNode;
29     last = newNode;
30     return last;
31 }
32 struct Node* addAfter(struct Node* last, int data, int item) {
33     if (last == NULL) return NULL;
34     struct Node *newNode, *p;
35     struct Node* addToAfter(struct Node* last, int data, int item) {
36         if (last == NULL) return NULL;
37         struct Node *newNode, *p;
38         p = last->next;
39         do {
40             if (p->data == item) {
41                 newNode = (struct Node*)malloc(sizeof(struct Node));
42                 newNode->data = data;
43                 newNode->next = p->next;
44                 p->next = newNode;
45                 if (p == last) last = newNode;
46                 return last;
47             }
48             p = p->next;
49         } while (p != last->next);
50         cout << "\nThe given node is not present in the list" << endl;
51         return last;
52     }
53 void deleteNode(Node** last, int key) {
54     if (*last == NULL) return;
55     if ((*last)->data == key && (*last)->next == *last) {
56         free(*last);
57         *last = NULL;
58         return;
59     }
60     Node *temp = *last, *d;
61     if ((*last)->data == key) {
62         while (temp->next != *last) temp = temp->next;
63         temp->next = (*last)->next;
64         free(*last);
65         *last = temp->next;
66     }
67     while (temp->next != *last && temp->next->data != key) {
68         temp = temp->next;
69     }

```

```

❯ make -s
❯ ./main
2 10 6 8
10 6 2

```

```

❯ make -s
❯ ./main
2 10 6 8
10 6 2

```

```
main.cpp X
64 while (temp->next != *last && temp->next->data != key) {
65     temp = temp->next;
66 }
67 if (temp->next->data == key) {
68     d = temp->next;
69     temp->next = d->next;
70     free(d);
71 }
72 }
73 void traverse(struct Node* last) {
74     struct Node* p;
75     if (last == NULL) {
76         cout << "The list is empty" << endl;
77         return;
78     }
79     p = last->next;
80     do {
81         cout << p->data << " ";
82         p = p->next;
83     } while (p != last->next);
84 }
85
86 int main() {
87     struct Node* last = NULL;
88     last = addToEmpty(last, 6);
89     last = addEnd(last, 8);
90     last = addFront(last, 2);
91     last = addAfter(last, 10, 2);
92     traverse(last);
93     deleteNode(&last, 8);
94     cout << endl;
95     traverse(last);
96     return 0;
97 }
```

```
❖ make -s
❖ ./main
2 10 6 8
10 6 2
```

Implement Stack using linked list

main.cpp x



Console

Shell

```

1  #include <iostream>
2  using namespace std;
3  struct Node {
4      int data;
5      struct Node *next;
6  };
7  struct Node* top = NULL;
8  void push(int val) {
9      struct Node* newNode = (struct Node*) malloc(sizeof(struct
10         Node));
11     newNode->data = val;
12     newNode->next = top;
13     top = newNode;
14 }
15 void pop() {
16     if(top==NULL)
17         cout<<"Stack Underflow"<<endl;
18     else {
19         cout<<"The popped element is "<< top->data <<endl;
20         top = top->next;
21     }
22 }
23 void display() {
24     struct Node* ptr;
25     if(top==NULL)
26         cout<<"stack is empty";
27     else {
28         ptr = top;
29         while(ptr != NULL) {
30             cout<<ptr->data<<" ";
31             ptr = ptr->next;
32         }
33         cout<<endl;
34     }
35 }
36 int main() {
37     int ch, val;
38     cout<<"1) Push in stack"<<endl;
39     cout<<"2) Pop from stack"<<endl;
40     cout<<"3) Display stack"<<endl;
41     cout<<"4) Exit"<<endl;
42     do {
43         cout<<"Enter choice: "<<endl;
44         cin>>ch;
45         switch(ch) {
46             case 1: {
47                 cout<<"Enter value to be pushed:"<<endl;
48                 cin>>val;
49                 push(val);
50                 break;
51             }
52             case 2: {
53                 pop();
54                 break;
55             }
56             case 3: {
57                 display();
58                 break;
59             }
60             case 4: {
61                 exit(0);
62             }
63         }
64     } while(ch != 4);
65 }

```

```

❖ make -s
❖ ./main
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
4
Enter choice:
1
Enter value to be pushed:
3
Enter choice:
1
Enter value to be pushed:
6
Enter choice:
2
The popped element is 6
Enter choice:
3
Stack elements are: 3 4
Enter choice:

```

main.cpp x



Console

Shell

```

23 struct Node* ptr;
24 if(top==NULL)
25     cout<<"stack is empty";
26 else {
27     ptr = top;
28     cout<<"Stack elements are: ";
29     while (ptr != NULL) {
30         cout<<ptr->data<<" ";
31         ptr = ptr->next;
32     }
33 }
34 cout<<endl;
35 }
36 int main() {
37     int ch, val;
38     cout<<"1) Push in stack"<<endl;
39     cout<<"2) Pop from stack"<<endl;
40     cout<<"3) Display stack"<<endl;
41     cout<<"4) Exit"<<endl;
42     do {
43         cout<<"Enter choice: "<<endl;
44         cin>>ch;
45         switch(ch) {
46             case 1: {
47                 cout<<"Enter value to be pushed:"<<endl;
48                 cin>>val;
49                 push(val);
50                 break;
51             }
52             case 2: {
53                 pop();
54                 break;
55             }
56             case 3: {
57                 display();
58                 break;
59             }
60             case 4: {
61                 exit(0);
62             }
63         }
64     } while(ch != 4);
65 }

```

```

❖ make -s
❖ ./main
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
4
Enter choice:
1
Enter value to be pushed:
3
Enter choice:
1
Enter value to be pushed:
6
Enter choice:
2
The popped element is 6
Enter choice:
3
Stack elements are: 3 4
Enter choice:

```

```
main.cpp x
47      cout<<"Enter value to be pushed:"<<endl;
48      cin>>val;
49      push(val);
50      break;
51  }
52  case 2: {
53      pop();
54      break;
55  }
56  case 3: {
57      display();
58      break;
59  }
60  case 4: {
61      cout<<"Exit"<<endl;
62      break;
63  }
64  default: {
65      cout<<"Invalid Choice"<<endl;
66  }
67  }
68  }while(ch!=4);
69  return 0;
70  }
```

Console Shell

```
> make -s
> ./main
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
4
Enter choice:
1
Enter value to be pushed:
3
Enter choice:
1
Enter value to be pushed:
6
Enter choice:
2
The popped element is 6
Enter choice:
3
Stack elements are: 3 4
Enter choice:

```

Implement Queue using Linked List

main.cpp x

```

1  #include <iostream>
2  using namespace std;
3  struct node {
4      int data;
5      struct node *next;
6  };
7  struct node* front = NULL;
8  struct node* rear = NULL;
9  struct node* temp;
10 void Insert() {
11     int val;
12     cout<<"Insert the element in queue : "<<endl;
13     cin>>val;
14     if (rear == NULL) {
15         rear = (struct node *)malloc(sizeof(struct node));
16         rear->next = NULL;
17         rear->data = val;
18         front = rear;
19     } else {
20         temp=(struct node *)malloc(sizeof(struct node));
21         rear->next = temp;
22         temp->data = val;
23         temp->next = NULL;
24         rear = temp;
25     }
26 }
27 void Delete() {
28     temp = front;

```

Console

Shell

```

> make -s
> ./main
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice :
1
Insert the element in queue :
4
Enter your choice :
1
Insert the element in queue :
5
Enter your choice :
1
Insert the element in queue :
6
Enter your choice :
2
Element deleted from queue is : 4
Enter your choice :
3
Queue elements are: 5 6
Enter your choice :

```

main.cpp x

```

29     if (front == NULL) {
30         cout<<"Underflow"<<endl;
31         return;
32     }
33     else
34     if (temp->next != NULL) {
35         temp = temp->next;
36         cout<<"Element deleted from queue is : "<<front->data<<endl;
37         free(front);
38         front = temp;
39     } else {
40         cout<<"Element deleted from queue is : "<<front->data<<endl;
41         free(front);
42         front = NULL;
43         rear = NULL;
44     }
45 }
46 void Display() {
47     temp = front;
48     if ((front == NULL) && (rear == NULL)) {
49         cout<<"Queue is empty"<<endl;
50         return;
51     }
52     cout<<"Queue elements are: ";
53     while (temp != NULL) {
54         cout<<temp->data<<" ";
55         temp = temp->next;
56     }
57     cout<<endl;

```

Console

Shell

```

> make -s
> ./main
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice :
1
Insert the element in queue :
4
Enter your choice :
1
Insert the element in queue :
5
Enter your choice :
1
Insert the element in queue :
6
Enter your choice :
2
Element deleted from queue is : 4
Enter your choice :
3
Queue elements are: 5 6
Enter your choice :

```

main.cpp x



Console

Shell

```
58 }
59 int main() {
60     int ch;
61     cout<<"1) Insert element to queue"<<endl;
62     cout<<"2) Delete element from queue"<<endl;
63     cout<<"3) Display all the elements of queue"<<endl;
64     cout<<"4) Exit"<<endl;
65     do {
66         cout<<"Enter your choice : "<<endl;
67         cin>>ch;
68         switch (ch) {
69             case 1: Insert();
70                 break;
71             case 2: Delete();
72                 break;
73             case 3: Display();
74                 break;
75             case 4: cout<<"Exit"<<endl;
76                 break;
77             default: cout<<"Invalid choice"<<endl;
78         }
79     } while(ch!=4);
80     return 0;
81 }
```

```
* make -s
* ./main
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice :
1
Insert the element in queue :
4
Enter your choice :
1
Insert the element in queue :
5
Enter your choice :
1
Insert the element in queue :
6
Enter your choice :
2
Element deleted from queue is : 4
Enter your choice :
3
Queue elements are: 5 6
Enter your choice :

```