Insert at front:

1. #include<iostream>
2. using namespace std;
3. struct Node
4. {
5. int data;
6. Node\* next;
7. };
8. class LinkedList
9. {
10. private:
11. Node\* head;
12. public:
13. LinkedList()
14. {
15. head = NULL;
16. }
17. void insertFront(int data);
18. void printList();
19. };
20. void LinkedList::insertFront(int data)
21. {
22. Node\* temp = new Node();
23. temp->data = data;
24. temp->next = head;
25. head = temp;
26. cout << "Insert node: " << temp->data << endl;
27. }
28. void LinkedList::printList()
29. {
30. Node\* temp = new Node();
31. temp = head;
32. cout << "\nLinked List : ";
33. while (temp != NULL)
34. {
35. cout << temp->data << " ";
36. temp = temp->next;
37. }
38. cout << endl;
39. }
40. int main()
41. {
42. LinkedList\* new\_list = new LinkedList();
43. new\_list->insertFront(10);
44. new\_list->insertFront(15);
45. new\_list->insertFront(20);
46. new\_list->insertFront(25);
47. new\_list->insertFront(30);
48. new\_list->printList();
49. return 0;
50. }

Output:

Insert node: 10

Insert node: 15

Insert node: 20

Insert node: 25

Insert node: 30

Linked List : 30 25 20 15 10

Insert At End:

1. struct Node
2. {
3. int data;
4. Node\* next;
5. };
6. class LinkedList
7. {
8. private:
9. Node\* head;
10. public:
11. LinkedList()
12. {
13. head = NULL;
14. }
15. void insertLast(int data);
16. void printlist();
17. };
18. void LinkedList::insertLast(int data)
19. {
20. Node\* t1 = new Node();
21. t1->data = data;
22. t1->next = NULL;
23. if (head == NULL)
24. {
25. head = t1;
26. cout << t1->data << " inserted" << endl;
27. return;
28. }
29. Node\* temp = head;
30. while (temp->next != NULL)
31. temp = temp->next;
32. temp->next = t1; // assign last node's next to this t1
33. cout << t1->data << " inserted" << endl;
34. }
35. void LinkedList::printlist() // print the content of the list
36. {
37. Node\* temp = new Node();
38. temp = head;
39. while (temp != NULL) // linked list will end when temp is Null
40. {
41. cout << temp->data << " ";
42. temp = temp->next;
43. }
44. cout << endl;
45. }
46. int main() // test the code
47. {
48. LinkedList\* new\_list = new LinkedList();
49. new\_list->insertLast(1); //Add data at the end of the list
50. new\_list->insertLast(2);
51. new\_list->insertLast(3);
52. new\_list->insertLast(4);
53. new\_list->insertLast(5);
54. new\_list->insertLast(10);
55. new\_list->insertLast(50);
56. new\_list->printlist();
57. return 0;
58. }

Output:

1 inserted

2 inserted

3 inserted

4 inserted

5 inserted

10 inserted

50 inserted

1 2 3 4 5 10 50

Doubly Linked List

Insert at front:

1. #include <iostream>
2. using namespace std;
3. struct Node
4. {
5. int data;
6. struct Node\* prev;
7. struct Node\* next;
8. };
9. struct Node\* head = NULL;
10. void insert(int newdata)
11. {
12. struct Node\* newnode = (struct Node\*)malloc(sizeof(struct Node));
13. newnode->data = newdata;
14. newnode->prev = NULL;
15. newnode->next = head;
16. if (head != NULL)
17. head->prev = newnode;
18. head = newnode;
19. }
20. void display()
21. {
22. struct Node\* ptr;
23. ptr = head;
24. while (ptr != NULL)
25. {
26. cout << ptr->data << " ";
27. ptr = ptr->next;
28. }
29. }
30. int main()
31. {
32. insert(1);
33. insert(2);
34. insert(3);
35. insert(4);
36. insert(5);
37. cout << "The doubly linked list is: ";
38. display();
39. return 0;
40. }

Output:

The doubly linked list is: 5 4 3 2 1

Insertion at end:

1. #include <iostream>
2. #include <stdlib.h> // header defines four variable types, several macros, and various functions for performing general functions
3. using namespace std;
4. struct node //node structure
5. {
6. int num;
7. node\* next;
8. struct node\* preptr;
9. struct node\* nextptr;
10. }
11. \*stnode, \* ennode;
12. class LinkedList // declared class
13. {
14. private:
15. node\* head, \* previous; // declared pointers
16. public:
17. LinkedList() // constructor
18. {
19. head = previous = NULL;
20. }
21. void Listcreation(int n); // declared functions
22. void LinsertNodeAtEnd(int num);
23. void print(int a);
24. };
25. void Listcreation(int n) // list create
26. {
27. int i, num;
28. struct node\* fnNode;
29. if (n >= 1)
30. {
31. stnode = (struct node\*)malloc(sizeof(struct node));
32. if (stnode != NULL)
33. {
34. cout << " Input data for node 1: "; // assigning data in the first node
35. cin >> num;
36. stnode->num = num;
37. stnode->preptr = NULL;
38. stnode->nextptr = NULL;
39. ennode = stnode;
40. for (i = 2; i <= n; i++)
41. {
42. fnNode = (struct node\*)malloc(sizeof(struct node));
43. if (fnNode != NULL)
44. {
45. cout << " Input data for node " << i << ": ";
46. cin >> num;
47. fnNode->num = num;
48. fnNode->preptr = ennode; // new node is linking with the previous node
49. fnNode->nextptr = NULL;
50. ennode->nextptr = fnNode; // previous node is linking with the new node
51. ennode = fnNode; // assign new node as last node
52. }
53. else
54. {
55. cout << " Memory can not be allocated.";
56. break;
57. }
58. }
59. }
60. else
61. {
62. cout << " Memory can not be allocated.";
63. }
64. }
65. }
66. void LinsertNodeAtEnd(int num) // linkedlist insert node at the end
67. {
68. struct node\* newnode;
69. if (ennode == NULL)
70. {
71. cout << " No data found in the list!\n";
72. }
73. else
74. {
75. newnode = (struct node\*)malloc(sizeof(struct node));
76. newnode->num = num;
77. newnode->nextptr = NULL; // set next address field of new node is NULL
78. newnode->preptr = ennode; // previous address of new node is linking with ending node
79. ennode->nextptr = newnode; // next address of ending node is linking with new node
80. ennode = newnode; // set the new node as ending node
81. }
82. }
83. void print(int m)
84. {
85. struct node\* tmp;
86. int n = 1;
87. if (stnode == NULL)
88. {
89. cout << " No data found in the List yet.";
90. }
91. else
92. {
93. tmp = stnode;
94. if (m == 1)
95. {
96. cout << "\n Data entered in the list are :\n";
97. }
98. else
99. {
100. cout << "\n After insertion the new list are :\n";
101. }
102. while (tmp != NULL)
103. {
104. cout << " node" << n << ": " << tmp->num << endl;
105. n++;
106. tmp = tmp->nextptr; // current pointer moves to the next node
107. }
108. }
109. }
110. int main()
111. {
112. int n, num1, a;
113. stnode = NULL;
114. ennode = NULL;
115. cout << " Input the number of nodes : ";
116. cin >> n;
117. Listcreation(n);
118. a = 1;
119. print(a);
120. cout << " Input data for the last node : ";
121. cin >> num1;
122. LinsertNodeAtEnd(num1);
123. a = 2;
124. print(a);
125. return 0;
126. }

Output:

Input the number of nodes : 5

Input data for node 1: 10

Input data for node 2: 20

Input data for node 3: 30

Input data for node 4: 40

Input data for node 5: 50

Data entered in the list are :

node1: 10

node2: 20

node3: 30

node4: 40

node5: 50

Input data for the last node : 100

After insertion the new list are :

node1: 10

node2: 20

node3: 30

node4: 40

node5: 50

node6: 100