|  |  |
| --- | --- |
| **Linked List (Add at index) in C++** | |
| #include <iostream>  using namespace std;  // Node class definition  class Node {  public:  int data;  Node\* next;  // Constructor  Node(int d) {  data = d;  next = nullptr;  }  };  // LinkedList class definition  class LinkedList {  private:  Node\* head;  Node\* tail;  int size;  public:  // Constructor  LinkedList() {  head = nullptr;  tail = nullptr;  size = 0;  }  // Method to add a node at the end of the list  void addLast(int val) {  Node\* temp = new Node(val);  if (size == 0) {  head = tail = temp;  } else {  tail->next = temp;  tail = temp;  }  size++;  }  // Method to get the size of the list  int getSize() {  return size;  }  // Method to display the elements of the list  void display() {  Node\* temp = head;  while (temp != nullptr) {  cout << temp->data << " ";  temp = temp->next;  }  cout << endl;  }  // Method to remove the first node  void removeFirst() {  if (size == 0) {  cout << "List is empty" << endl;  } else if (size == 1) {  head = tail = nullptr;  size = 0;  } else {  head = head->next;  size--;  }  }  int getFirst() {  if (size == 0) {  cout << "List is empty" << endl;  return -1;  } else {  return head->data;  }  }  int getLast() {  if (size == 0) {  cout << "List is empty" << endl;  return -1;  } else {  return tail->data;  }  }  int getAt(int idx) {  if (size == 0) {  cout << "List is empty" << endl;  return -1;  } else if (idx < 0 || idx >= size) {  cout << "Invalid arguments" << endl;  return -1;  } else {  Node\* temp = head;  for (int i = 0; i < idx; i++) {  temp = temp->next;  }  return temp->data;  }  }  // Method to add a node at the beginning of the list  void addFirst(int val) {  Node\* temp = new Node(val);  temp->next = head;  head = temp;  if (size == 0) {  tail = temp;  }  size++;  }  // Method to add a node at a specified index  void addAt(int idx, int val) {  if (idx < 0 || idx > size) {  cout << "Invalid arguments" << endl;  } else if (idx == 0) {  addFirst(val);  } else if (idx == size) {  addLast(val);  } else {  Node\* node = new Node(val);  Node\* temp = head;  for (int i = 0; i < idx - 1; i++) {  temp = temp->next;  }  node->next = temp->next;  temp->next = node;  size++;  }  }  };  // Main function to demonstrate LinkedList operations  int main() {  LinkedList list;  // Hardcoded sequence of operations  list.addFirst(10);  cout << list.getFirst() << endl; // Should display: 10  list.addAt(0, 20);  cout << list.getFirst() << endl; // Should display: 20  cout << list.getLast() << endl; // Should display: 10  list.display(); // Should display: 20 10  cout << list.getSize() << endl; // Should display: 2  list.addAt(2, 40);  cout << list.getLast() << endl; // Should display: 40  list.addAt(1, 50);  list.addFirst(30);  list.removeFirst();  cout << list.getFirst() << endl; // Should display: 20  list.removeFirst();  list.removeFirst();  list.addAt(2, 60);  list.display(); // Should display: 50 10 60  cout << list.getSize() << endl; // Should display: 3  list.removeFirst();  list.removeFirst();  cout << list.getFirst() << endl; // Should display: 60  return 0;  } | ****Dry Run Table****  | **Step** | **Operation** | **List State** | **Output** | **Notes** | | --- | --- | --- | --- | --- | | 1 | addFirst(10) | 10 |  | Adds 10 at front | | 2 | getFirst() | 10 | 10 |  | | 3 | addAt(0, 20) | 20 → 10 |  | Insert 20 at index 0 | | 4 | getFirst() | 20 → 10 | 20 |  | | 5 | getLast() | 20 → 10 | 10 |  | | 6 | display() | 20 → 10 | 20 10 |  | | 7 | getSize() | 20 → 10 | 2 |  | | 8 | addAt(2, 40) | 20 → 10 → 40 |  | Insert 40 at end | | 9 | getLast() | 20 → 10 → 40 | 40 |  | | 10 | addAt(1, 50) | 20 → 50 → 10 → 40 |  | Insert 50 at index 1 | | 11 | addFirst(30) | 30 → 20 → 50 → 10 → 40 |  | Adds 30 at front | | 12 | removeFirst() | 20 → 50 → 10 → 40 |  | Removes 30 | | 13 | getFirst() | 20 → 50 → 10 → 40 | 20 |  | | 14 | removeFirst() | 50 → 10 → 40 |  | Removes 20 | | 15 | removeFirst() | 10 → 40 |  | Removes 50 | | 16 | addAt(2, 60) | 10 → 40 → 60 |  | Adds 60 at index 2 | | 17 | display() | 10 → 40 → 60 | 10 40 60 |  | | 18 | getSize() | 10 → 40 → 60 | 3 |  | | 19 | removeFirst() | 40 → 60 |  | Removes 10 | | 20 | removeFirst() | 60 |  | Removes 40 | | 21 | getFirst() | 60 | 60 |  | |
| 10  20  10  20 10  2  40  20  10 40 60  3  60 | |