DOULBY LINKED LIST: This code defines a structure for a doubly linked list node with pointers to both the previous and next nodes. It also provides functions to insert a new node at the beginning of the list (`insertAtBeginning()`), and print the list both forward (`printListForward()`) and backward (`printListBackward()`). Finally, the `main()` function demonstrates how to use these functions to create and manipulate a doubly linked list.

```
#include <iostream>

// Define the structure for a node
struct Node {
    int data;
    Node* prev;
    Node* next;
};

// Function to insert a new node at the beginning of the doubly linked list
void insertAtBeginning(Node** head_ref, int new_data) {
    // Allocate memory for new node
    Node* new_node = new Node;

    // Assign data to the new node
    new_node->data = new_data;

    // Set previous of new node as NULL since it will be the first node
    new_node->prev = nullptr;

    // Set next of new node as current head
    new_node->next = *head_ref;

    // If the list is not empty, change the previous of the current head node to new
node
    if (*head_ref != nullptr)
        (*head_ref)->prev = new_node;

    // Move the head to point to the new node
    *head_ref = new_node;
}

// Function to print the doubly linked list forward
void printListForward(Node* node) {
    while (node != nullptr) {
        std::cout << node->data << " ";
        node = node->next;
    }
    std::cout << std::endl;
}

// Function to print the doubly linked list backward
void printListBackward(Node* node) {
    // Move to the last node
    while (node->next != nullptr) {
        node = node->next;
    }

    // Traverse backward and print the data
    while (node != nullptr) {
        std::cout << node->data << " ";
        node = node->prev;
    }
    std::cout << std::endl;
}

// Test the code
int main() {
    // Initialize an empty doubly linked list
    Node* head = nullptr;

    // Insert some elements at the beginning
    insertAtBeginning(&head, 5);
    insertAtBeginning(&head, 10);
    insertAtBeginning(&head, 15);

    // Print the doubly linked list forward
    std::cout << "Doubly linked list (forward): ";
    printListForward(head);

    // Print the doubly linked list backward
    std::cout << "Doubly linked list (backward): ";
    printListBackward(head);

    return 0;
}
```