Lab Task #2: Write a C++ program to implement a singly linked list using object-oriented programming concepts. Your program should support the following operations:

•Insert a node at the beginning of the list.

•Insert a node at the end of the list.

•Delete a node from the beginning of the list.

•Delete a node from the end of the list.

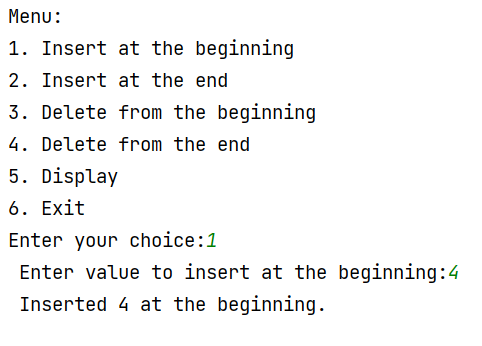
•Display the elements of the linked list.

Additionally, provides a menu-driven interface to allow the user to perform these operations.

Code:

#include <iostream>  
using namespace std;  
class Node {  
public:  
 int data;  
 Node\* next;  
  
 Node(int value) : data(value), next(nullptr) {}  
};  
  
class LinkedList {  
private:  
 Node\* head;  
  
public:  
 LinkedList() : head(nullptr) {}  
  
 *// Insert a node at the beginning of the list* void insertAtBeginning(int value) {  
 Node\* newNode = new Node(value);  
 newNode->next = head;  
 head = newNode;  
 cout << "Inserted " << value << " at the beginning." << endl;  
 }  
  
 *// Insert a node at the end of the list* void insertAtEnd(int value) {  
 Node\* newNode = new Node(value);  
 if (!head) {  
 head = newNode;  
 } else {  
 Node\* current = head;  
 while (current->next) {  
 current = current->next;  
 }  
 current->next = newNode;  
 }  
 cout << "Inserted " << value << " at the end." << endl;  
 }  
  
 *// Delete a node from the beginning of the list* void deleteFromBeginning() {  
 if (!head) {  
 cout << "List is empty. Cannot delete from the beginning." << std::endl;  
 return;  
 }  
 Node\* temp = head;  
 head = head->next;  
 delete temp;  
 cout << "Deleted from the beginning." << endl;  
 }  
  
 *// Delete a node from the end of the list* void deleteFromEnd() {  
 if (!head) {  
 cout << "List is empty. Cannot delete from the end." << endl;  
 return;  
 }  
 if (!head->next) {  
 delete head;  
 head = nullptr;  
 cout << "Deleted from the end." << endl;  
 return;  
 }  
 Node\* current = head;  
 while (current->next->next) {  
 current = current->next;  
 }  
 delete current->next;  
 current->next = nullptr;  
 cout << "Deleted from the end." << endl;  
 }  
  
 *// Display the elements of the linked list* void display() {  
 Node\* current = head;  
 if (!current) {  
 cout << "List is empty." << endl;  
 } else {  
 std::cout << "Linked List: ";  
 while (current) {  
 cout << current->data << " ";  
 current = current->next;  
 }  
 cout << endl;  
 }  
 }  
  
 *// Destructor to free memory* ~LinkedList() {  
 while (head) {  
 Node\* temp = head;  
 head = head->next;  
 delete temp;  
 }  
 }  
};  
  
int main() {  
 LinkedList list;  
 int choice, value;  
  
 do {  
 cout << "\nMenu:\n";  
 cout << "1. Insert at the beginning\n";  
 cout << "2. Insert at the end\n";  
 cout << "3. Delete from the beginning\n";  
 cout << "4. Delete from the end\n";  
 cout << "5. Display\n";  
 cout << "6. Exit\n";  
 cout << "Enter your choice: ";  
 cin >> choice;  
  
 switch (choice) {  
 case 1:  
 cout << "Enter value to insert at the beginning: ";  
 cin >> value;  
 list.insertAtBeginning(value);  
 break;  
 case 2:  
 cout << "Enter value to insert at the end: ";  
 cin >> value;  
 list.insertAtEnd(value);  
 break;  
 case 3:  
 list.deleteFromBeginning();  
 break;  
 case 4:  
 list.deleteFromEnd();  
 break;  
 case 5:  
 list.display();  
 break;  
 case 6:  
 cout << "Exiting program." << endl;  
 break;  
 default:  
 cout << "Invalid choice. Try again." << endl;  
 }  
 } while (choice != 6);  
  
 return 0;  
}

**Output:**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated