Práctica de Laboratorio N° 3:
Recorrido e Inserción en Listas Doblemente Enlazadas

I. OBJETIVOS DE LA PRÁCTICA:

- Comprender y aplicar las operaciones básicas de recorrido e inserción en una lista doblemente enlazada.
- Desarrollar habilidades en la manipulación de punteros para insertar elementos al inicio, al final y antes o después de un nodo específico en una lista doblemente enlazada.
- Consolidar los conocimientos adquiridos mediante la resolución de ejercicios que impliquen la implementación de operaciones de inserción en diferentes contextos de una lista doblemente enlazada

II. CONCEPTOS GENERALES:

- **Lista doblemente enlazada:** Es una estructura de datos lineal en la cual cada nodo contiene tres partes: un campo de datos, un puntero al nodo siguiente (LIGADER) y un puntero al nodo anterior (LIGAIZQ). Esto permite recorrer la lista en ambas direcciones.
- **Nodo:** Es la unidad básica de la lista doblemente enlazada. Un nodo contiene la información y las referencias (punteros) al nodo anterior y al siguiente.
- **Recorrido:** El proceso de visitar cada nodo de la lista para acceder o mostrar su información. Puede hacerse en sentido hacia adelante (desde la cabeza hacia la cola) o hacia atrás (desde la cola hacia la cabeza).
- **Inserción:**
  - **Al inicio:** En este caso el nuevo nodo se coloca al principio de la lista y se establecen las ligas correspondientes. El nuevo nodo insertado se convierte, entonces, en el primero de la lista doblemente ligada.
  - **Al final:** En este caso el nuevo nodo se coloca al final de la lista doblemente ligada, convirtiéndola en el último.
  - **Antes o después de un nodo con información X:** Implica localizar el nodo con la información X y modificar correctamente los punteros de los nodos adyacentes para insertar el nuevo nodo en la posición deseada.

III. DESARROLLO DE LA PRÁCTICA:

1. Implementación de lista doblemente enlazada en C++:

```cpp
#include <iostream>
using namespace std;

struct Nodo {
    int info;
    Nodo *ligader;
    Nodo *ligaizq;
};

// Recorrido de la lista desde el inicio
void recorrerDesdeInicio(Nodo *p) {
    Nodo *q = p;
    cout << "Recorrido de inicio a fin: ";
    while (q != NULL) {
        cout << q->info << " ";
        q = q->ligader;
    }
    cout << endl;
}

// Recorrido de la lista desde el final
void recorrerDesdeFinal(Nodo *f) {
    Nodo *q = f;
    cout << "Recorrido de fin a inicio: ";
    while (q != NULL) {
        cout << q->info << " ";
        q = q->ligaizq;
    }
    cout << endl;
}
```

```
32  // Inserción al inicio
33  void inserta_principio(Nodo *&p, Nodo *&f, int dato) {
34      Nodo* q = new Nodo();
35      q->info = dato;
36
37      if (p == NULL) {
38          p = f = q;
39      } else {
40          q->ligader = p;
41          p->ligaizq = q;
42          q->ligaizq = NULL;
43          p = q;
44      }
45
46  }
47
48  // Inserción al final
49  void inserta_final(Nodo *&p, Nodo *&f, int dato) {
50      Nodo* q = new Nodo();
51      q->info = dato;
52
53      if (f == NULL) {
54          p = f = q;
55      } else {
56          f->ligader = q;
57          q->ligaizq = f;
58          q->ligader = NULL;
59          f = q;
60      }
61  }
63  // Inserción antes de un nodo con dato X
64  void inserta_antes_X(Nodo *&p, int dato, int x) {
65      Nodo *q = p;
66
67      while (q->ligader != NULL && q->info != x) {
68          q = q->ligader;
69      }
70
71      if (q->info == x){
72          Nodo *t = new Nodo();
73          t->info = dato;
74          t->ligader = q;
75
76          Nodo *r = q->ligaizq;
77          q->ligaizq = t;
78
79          if (p == q){
80              p = t;
81              t->ligaizq = NULL;
82          }else{
83              r->ligader = t;
84              t->ligaizq = r;
85          }
86      }else{
87          cout << "No se encontró el dato " << x << " en la lista." << endl;
88          return;
89      }
90  }
```

```
// Menú interactivo
void menu() {
    Nodo *p = NULL;
    Nodo *f = NULL;
    int opcion, dato, x;

    do {
        cout << "\n--- MENÚ ---" << endl;
        cout << "1. Insertar al inicio" << endl;
        cout << "2. Insertar al final" << endl;
        cout << "3. Insertar antes de un nodo con dato X" << endl;
        cout << "4. Insertar después de un nodo con dato X" << endl;
        cout << "5. Mostrar lista de inicio a fin" << endl;
        cout << "6. Mostrar lista de fin a inicio" << endl;
        cout << "0. Salir" << endl;
        cout << "Opción: ";
        cin >> opcion;

        switch (opcion) {
            case 1:
                cout << "dato a insertar al inicio: ";
                cin >> dato;
                inserta_principio(p, f, dato);
                break;
            case 2:
                cout << "dato a insertar al final: ";
                cin >> dato;
                inserta_final(p, f, dato);
                break;
            case 3:
                cout << "dato a insertar: ";
                cin >> dato;
                cout << "Antes del nodo con dato: ";
                cin >> x;
                inserta_antes_X(p, dato, x);
                break;
            case 4:
                cout << "dato a insertar: ";
                cin >> dato;
                cout << "Después del nodo con dato: ";
                cin >> x;
                inserta_despues_X(p, f, dato, x);
                break;
            case 5:
                recorrerDesdeInicio(p);
                break;
            case 6:
                recorrerDesdeFinal(f);
                break;
            case 0:
                cout << "Programa finalizado." << endl;
                break;
            default:
                cout << "Opción inválida." << endl;
        }
    } while (opcion != 0);
}
```

```
217  // Función principal
218  int main() {
219      menu();
220      /*Nodo *p = NULL;
221      Nodo *f = NULL;
222      inserta_principio(p, f, 1);
223      inserta_principio(p, f, 3);
224      inserta_principio(p, f, 4);
225      inserta_principio(p, f, 5);
226      recorrerDesdeInicio(p);*/
227
228      /*int ocurrencias = 0;
229
230      Nodo *p = NULL;
231      Nodo *f = NULL;
232      inserta_final(p, f, 1);
233      inserta_final(p, f, 3);
234      inserta_final(p, f, 5);
235      inserta_final(p, f, 7);
236      recorrerDesdeInicio(p);
237      Invertir(p, f);
238      recorrerDesdeInicio(p);
239      inserta_principio(p, f, 1);
240      recorrerDesdeInicio(p);
241      int nro=1;
242      ocurrencias = contar_ocurrencias(p, nro);
243      cout << "El número "<< nro <<" se repite:"<< ocurrencias <<" veces.";*/
244
245      return 0;
246  }
```

2. Implementar la operación de Inserción después de un nodo con información X:

   void inserta_despues_X(Nodo *&p, Nodo *&f, int dato, int x)

3. Implementar una función que retorne el número de veces que se encuentra un dato dentro de la lista doblemente enlazada. En caso de no encontrarse, se debe mostrar un mensaje indicando que el dato no fue encontrado. La función debe recibir como parámetro el valor que se desea buscar.

   int contar_ocurrencias(Nodo *p, int dato)

4. Implementar una función que invierta los elementos de una lista doblemente enlazada en una sola pasada, sin usar una estructura de dato adicional y sin copiar el contenido de los nodos.

   void Invertir(Nodo *&p, Nodo *&f) {