UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA, SOLOLA

Ingeniería En Sistemas

Tercer Semestre

Curso: Programación 1

Catedrático: Neftalí García

Estudiante: Pascual Evelio Yaxon Tziquin **No. Carné:** 2290-24-26037

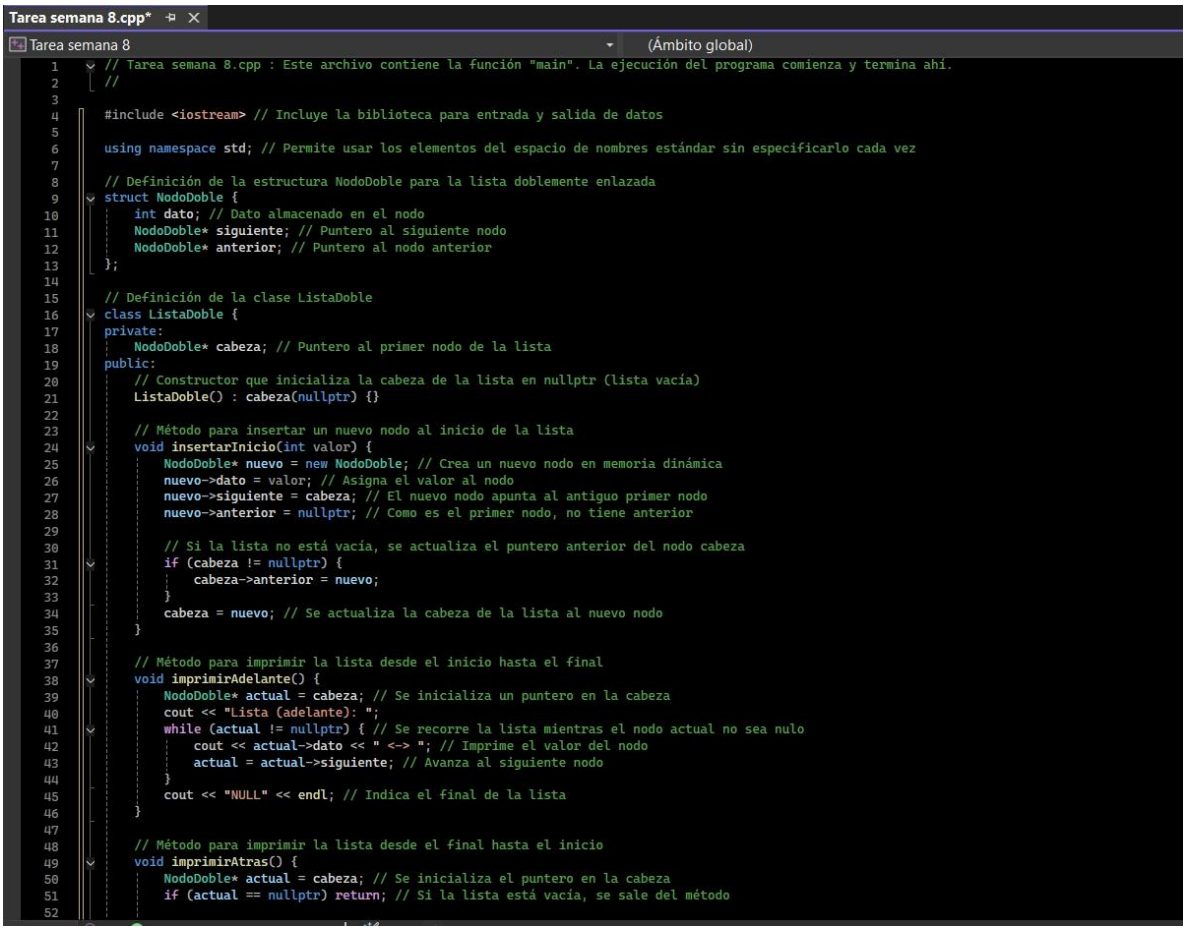
Selvin Geovany Tzunún Balán 2290-24-44

Tema

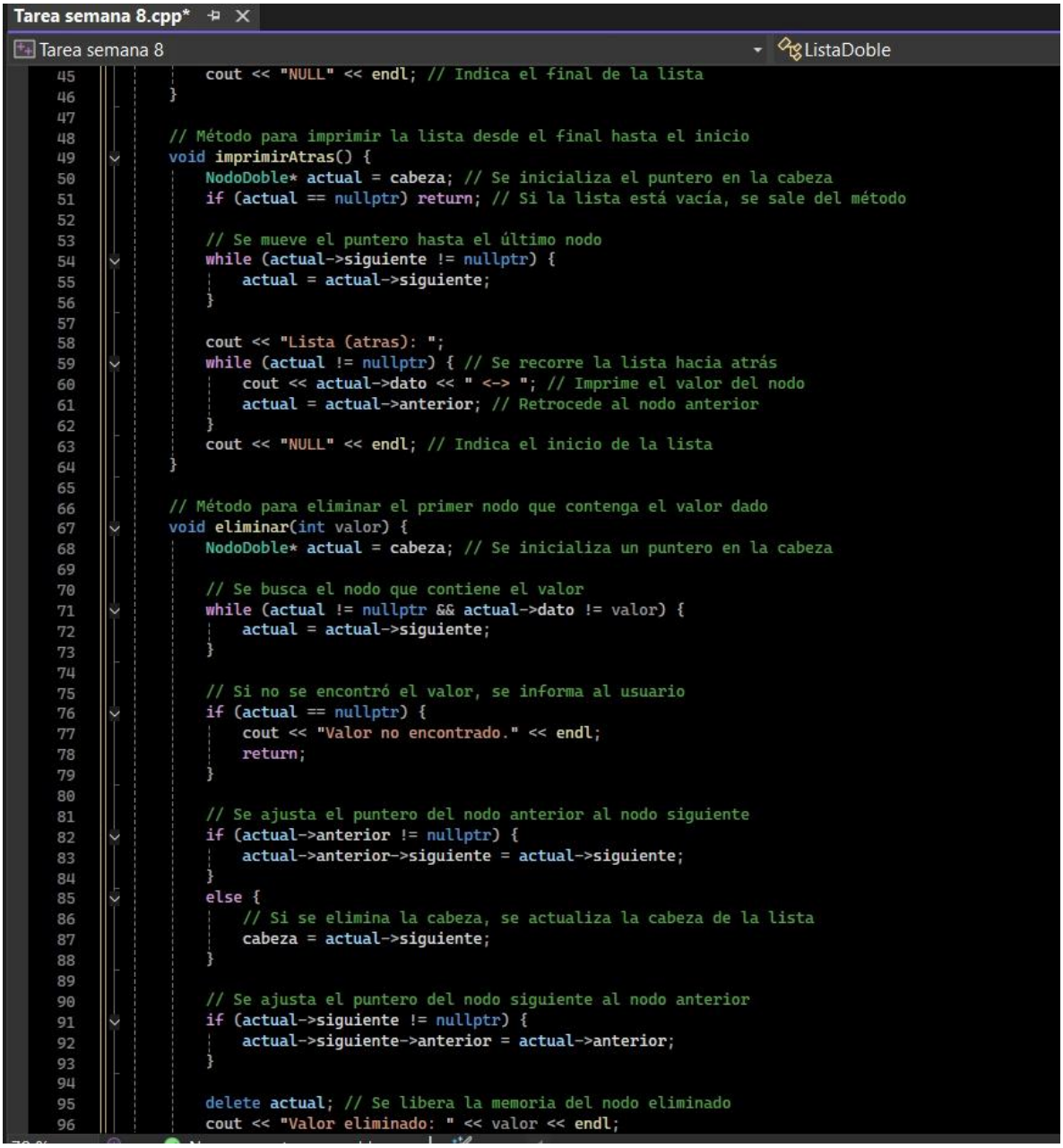
Tarea Semana 8

LISTAS DOBLEMENTE ENLAZADAS

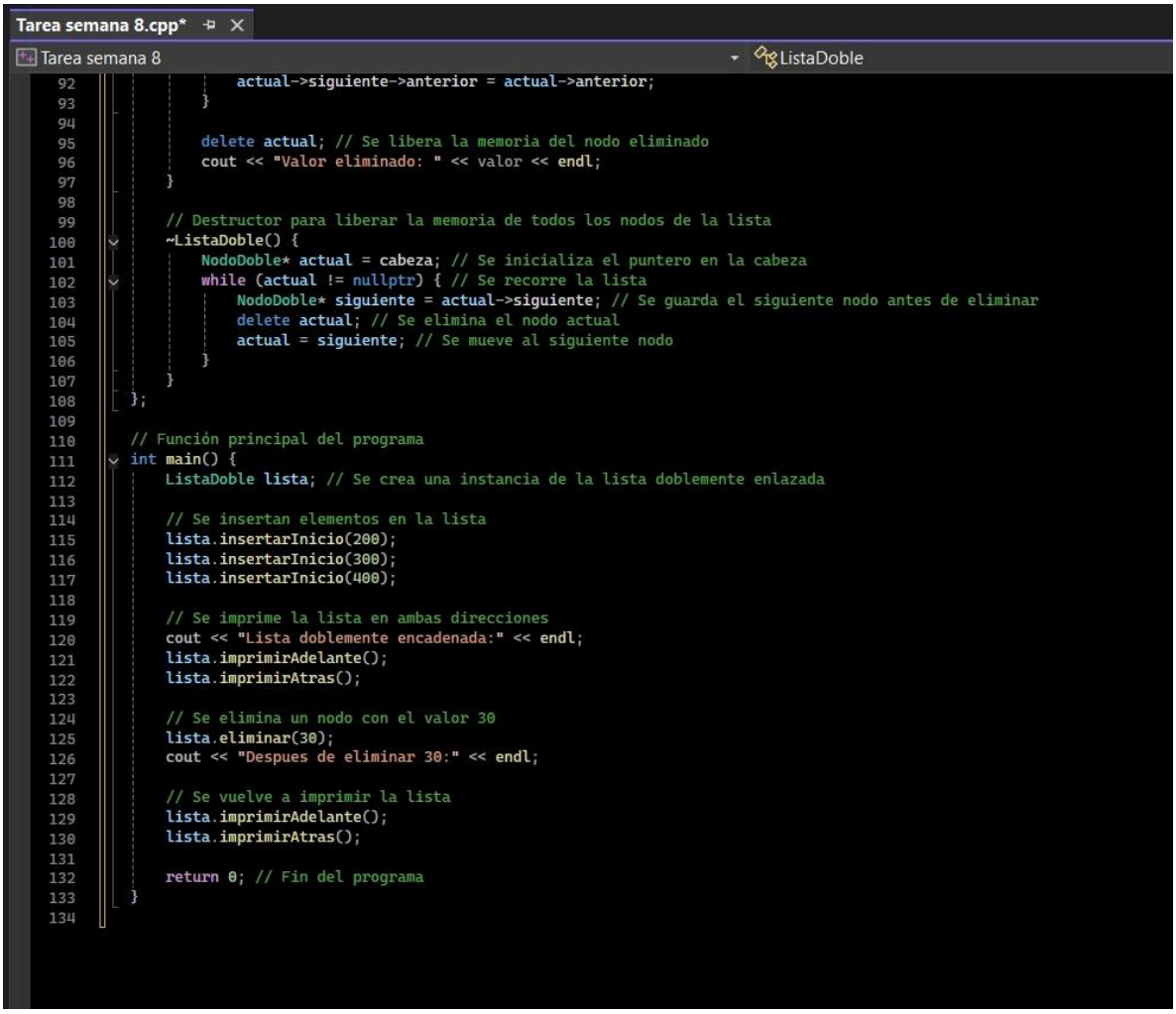
• Código comentado:




```cpp
// Tarea semana 8.cpp : Este archivo contiene la función "main". La ejecución del programa comienza y termina ahí.
//

#include <iostream> // Incluye la biblioteca para entrada y salida de datos

using namespace std; // Permite usar los elementos del espacio de nombres estándar sin especificarlo cada vez

// Definición de la estructura NodoDoble para la lista doblemente enlazada
struct NodoDoble {
    int dato; // Dato almacenado en el nodo
    NodoDoble* siguiente; // Puntero al siguiente nodo
    NodoDoble* anterior; // Puntero al nodo anterior
};

// Definición de la clase ListaDoble
class ListaDoble {
private:
    NodoDoble* cabeza; // Puntero al primer nodo de la lista
public:
    // Constructor que inicializa la cabeza de la lista en nullptr (lista vacía)
    ListaDoble() : cabeza(nullptr) {}

    // Método para insertar un nuevo nodo al inicio de la lista
    void insertarInicio(int valor) {
        NodoDoble* nuevo = new NodoDoble; // Crea un nuevo nodo en memoria dinámica
        nuevo->dato = valor; // Asigna el valor al nodo
        nuevo->siguiente = cabeza; // El nuevo nodo apunta al antiguo primer nodo
        nuevo->anterior = nullptr; // Como es el primer nodo, no tiene anterior

        // Si la lista no está vacía, se actualiza el puntero anterior del nodo cabeza
        if (cabeza != nullptr) {
            cabeza->anterior = nuevo;
        }
        cabeza = nuevo; // Se actualiza la cabeza de la lista al nuevo nodo
    }

    // Método para imprimir la lista desde el inicio hasta el final
    void imprimirAdelante() {
        NodoDoble* actual = cabeza; // Se inicializa un puntero en la cabeza
        cout << "Lista (adelante): ";
        while (actual != nullptr) { // Se recorre la lista mientras el nodo actual no sea nulo
            cout << actual->dato << " <-> "; // Imprime el valor del nodo
            actual = actual->siguiente; // Avanza al siguiente nodo
        }
        cout << "NULL" << endl; // Indica el final de la lista
    }

    // Método para imprimir la lista desde el final hasta el inicio
    void imprimirAtras() {
        NodoDoble* actual = cabeza; // Se inicializa el puntero en la cabeza
        if (actual == nullptr) return; // Si la lista está vacía, se sale del método
```

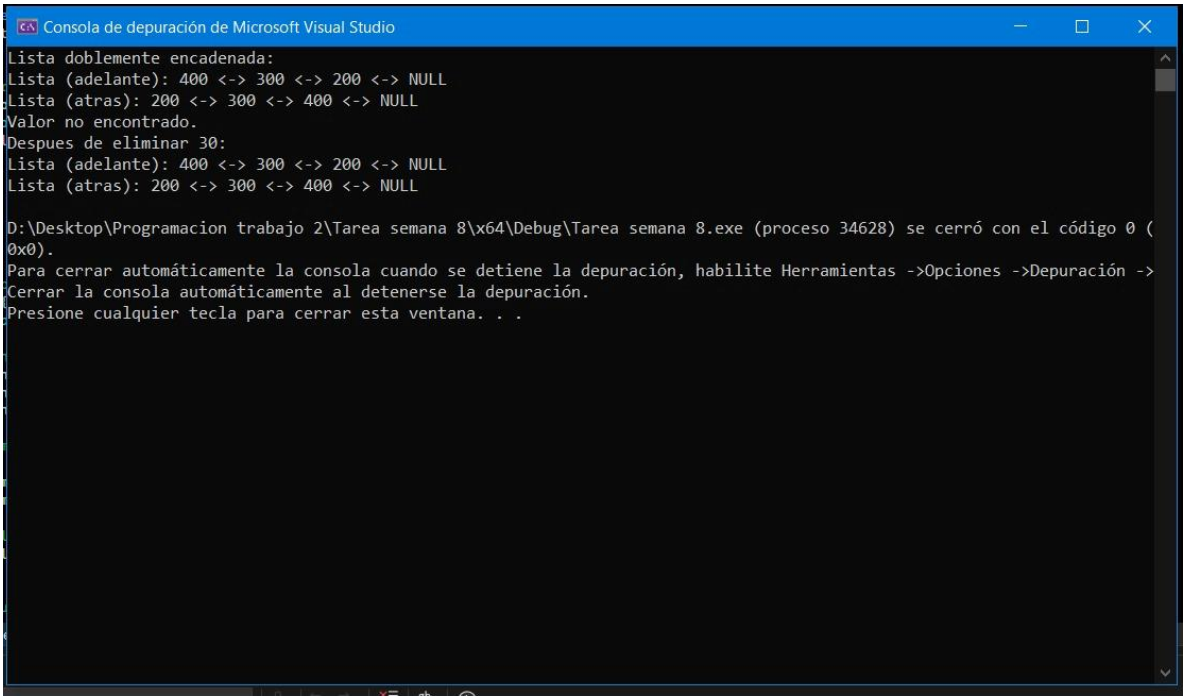



```
45          cout << "NULL" << endl; // Indica el final de la lista
46      }
47
48      // Método para imprimir la lista desde el final hasta el inicio
49      void imprimirAtras() {
50          NodoDoble* actual = cabeza; // Se inicializa el puntero en la cabeza
51          if (actual == nullptr) return; // Si la lista está vacía, se sale del método
52
53          // Se mueve el puntero hasta el último nodo
54          while (actual->siguiente != nullptr) {
55              actual = actual->siguiente;
56          }
57
58          cout << "Lista (atras): ";
59          while (actual != nullptr) { // Se recorre la lista hacia atrás
60              cout << actual->dato << " <-> "; // Imprime el valor del nodo
61              actual = actual->anterior; // Retrocede al nodo anterior
62          }
63          cout << "NULL" << endl; // Indica el inicio de la lista
64      }
65
66      // Método para eliminar el primer nodo que contenga el valor dado
67      void eliminar(int valor) {
68          NodoDoble* actual = cabeza; // Se inicializa un puntero en la cabeza
69
70          // Se busca el nodo que contiene el valor
71          while (actual != nullptr && actual->dato != valor) {
72              actual = actual->siguiente;
73          }
74
75          // Si no se encontró el valor, se informa al usuario
76          if (actual == nullptr) {
77              cout << "Valor no encontrado." << endl;
78              return;
79          }
80
81          // Se ajusta el puntero del nodo anterior al nodo siguiente
82          if (actual->anterior != nullptr) {
83              actual->anterior->siguiente = actual->siguiente;
84          }
85          else {
86              // Si se elimina la cabeza, se actualiza la cabeza de la lista
87              cabeza = actual->siguiente;
88          }
89
90          // Se ajusta el puntero del nodo siguiente al nodo anterior
91          if (actual->siguiente != nullptr) {
92              actual->siguiente->anterior = actual->anterior;
93          }
94
95          delete actual; // Se libera la memoria del nodo eliminado
96          cout << "Valor eliminado: " << valor << endl;
```



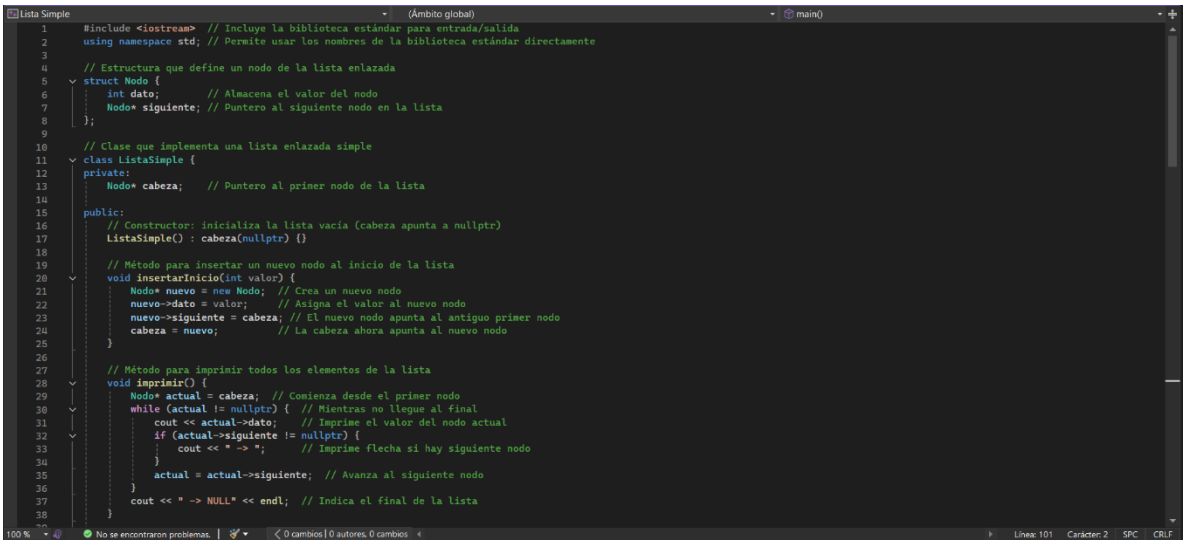

```
92              actual->siguiente->anterior = actual->anterior;
93          }
94
95          delete actual; // Se libera la memoria del nodo eliminado
96          cout << "Valor eliminado: " << valor << endl;
97      }
98
99      // Destructor para liberar la memoria de todos los nodos de la lista
100     ~ListaDoble() {
101         NodoDoble* actual = cabeza; // Se inicializa el puntero en la cabeza
102         while (actual != nullptr) { // Se recorre la lista
103             NodoDoble* siguiente = actual->siguiente; // Se guarda el siguiente nodo antes de eliminar
104             delete actual; // Se elimina el nodo actual
105             actual = siguiente; // Se mueve al siguiente nodo
106         }
107     }
108 };
109
110 // Función principal del programa
111 int main() {
112     ListaDoble lista; // Se crea una instancia de la lista doblemente enlazada
113
114     // Se insertan elementos en la lista
115     lista.insertarInicio(200);
116     lista.insertarInicio(300);
117     lista.insertarInicio(400);
118
119     // Se imprime la lista en ambas direcciones
120     cout << "Lista doblemente encadenada:" << endl;
121     lista.imprimirAdelante();
122     lista.imprimirAtras();
123
124     // Se elimina un nodo con el valor 30
125     lista.eliminar(30);
126     cout << "Despues de eliminar 30:" << endl;
127
128     // Se vuelve a imprimir la lista
129     lista.imprimirAdelante();
130     lista.imprimirAtras();
131
132     return 0; // Fin del programa
133 }
134
```

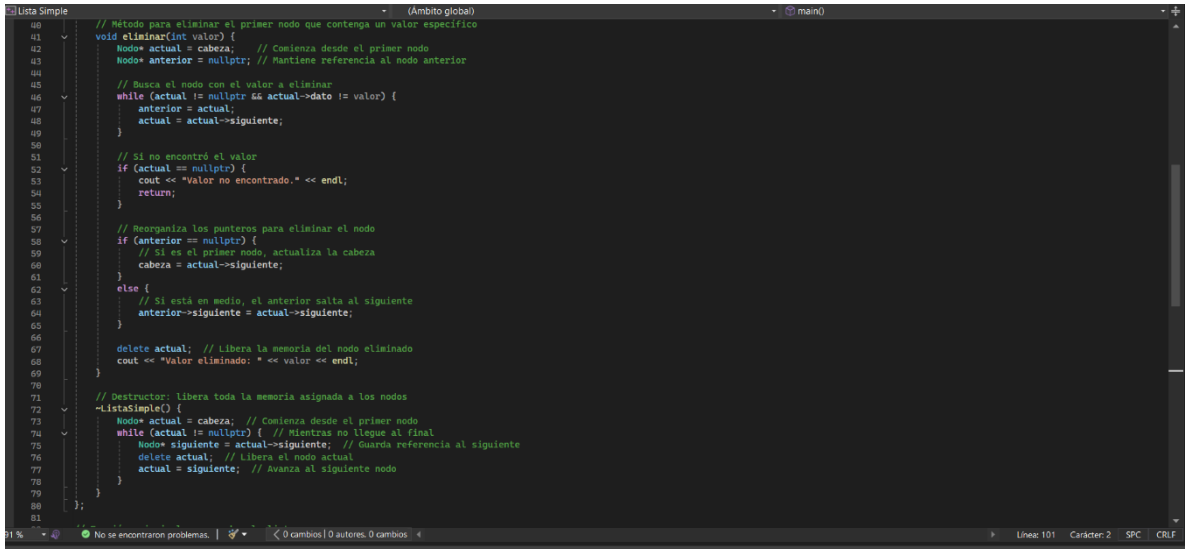
• Compilación:

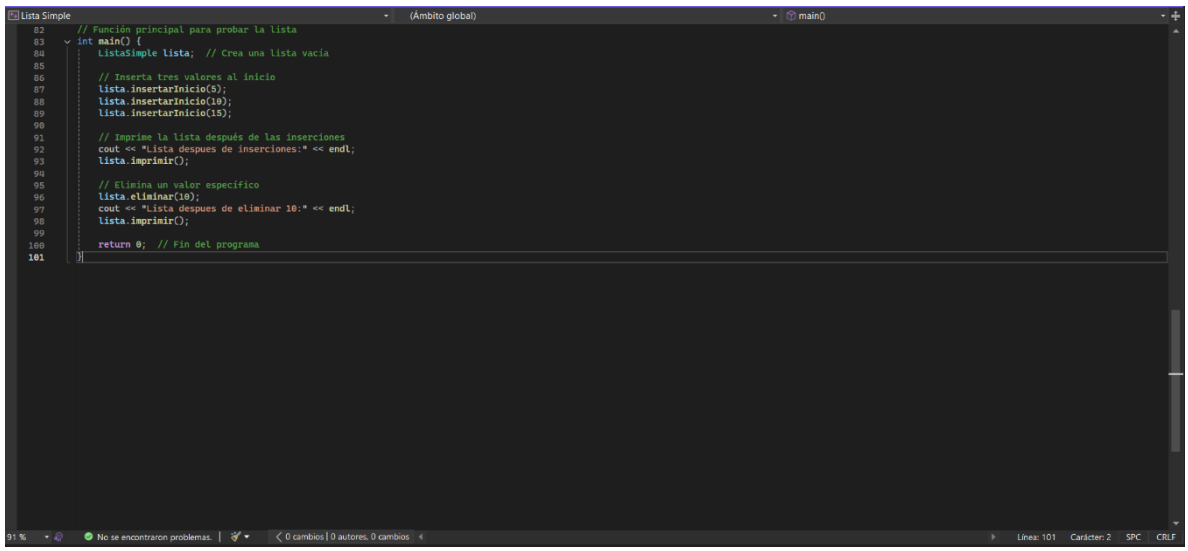


LISTA SIMPLE

• Código comentado:







• Compilación:

LINK DEL REPOSITORIO PARA ESTA TAREA:

https://github.com/E-lev10/Tarea-Semana-8-Progra.git