```
Definir clase Cliente:
   id: Entero
   nombre: Cadena
   scoring: Entero

Definir función EncontrarDosMaxScoring(clientes, inicio, fin):
   Si inicio == fin:
      // Caso base: solo un cliente en esta porción
      Retornar [clientes[inicio], NULO]

   mitad = (inicio + fin) / 2

   // Llamada recursiva para encontrar los dos máximos en cada mitad
   maxIzquierda = EncontrarDosMaxScoring(clientes, inicio, mitad)
   maxDerecha = EncontrarDosMaxScoring(clientes, mitad + 1, fin)

   // Inicializar arreglo para almacenar los dos máximos
   maxDos = [NULO, NULO]

   // Encontrar el cliente con el mayor scoring
   Si maxIzquierda[0].scoring > maxDerecha[0].scoring:
      maxDos[0] = maxIzquierda[0]
   Sino:
      maxDos[0] = maxDerecha[0]

   // Encontrar el segundo cliente con el mayor scoring
   Si maxIzquierda[0] == maxDos[0]:
      // El mayor está en la izquierda, comparar segundo mayor
      Si maxDerecha[0] no es NULO Y (maxIzquierda[1] es NULO O maxDerecha[0].scoring >
maxIzquierda[1].scoring):
         maxDos[1] = maxDerecha[0]
      Sino:
         maxDos[1] = maxIzquierda[1]
   Sino:
      // El mayor está en la derecha, comparar segundo mayor
      Si maxIzquierda[0] no es NULO Y (maxDerecha[1] es NULO O maxIzquierda[0].scoring >
maxDerecha[1].scoring):
         maxDos[1] = maxIzquierda[0]
      Sino:
         maxDos[1] = maxDerecha[1]

   Retornar maxDos

// Función principal para ejecutar el programa
```

```
Definir función Principal():
   clientes = [
      Cliente(1, "Juan", 85),
      Cliente(2, "Maria", 90),
      Cliente(3, "Pedro", 80),
      Cliente(4, "Ana", 95)
   ]

   maxClientes = EncontrarDosMaxScoring(clientes, 0, longitud(clientes) - 1)

   Imprimir "Los dos clientes con el máximo scoring son:"
   Imprimir maxClientes[0].nombre + " con puntuación de " + maxClientes[0].scoring
   Si maxClientes[1] no es NULO:
      Imprimir maxClientes[1].nombre + " con puntuación de " + maxClientes[1].scoring
```