```
 1  int Dim = 1; // Dimension del problema (1,2,3)
 2
 3  int NNE = 2; // Numero de nodos por elemento
 4               // 1D (2), 2D Rectangulos (4), 2D Triangulos (3), 3D Ortoedros (8)
 5  int NEN = 2; // Numero de elementos que comparten el nodo (Soporte del nodo)
 6               // 1D (2), 2D Rectangulos (4), 2D Trianguloss (6), 3D Ortiedros (8)
 7
 8  int NT       // Nodos Totales
 9  int ND       // Nodos Desconocidos
10
11
12
13  double Coord     [NT][Dim] // Cordenadas del problema
14  int    Elementos [NT][NNE] // Dado un nodo que elementos lo tienen (tiene -1 si no hay ese valor)
15  int    Nodos     [NT][NEN] // Dado un elemento que nodos tiene
16  int    TipoNodo  [NT]      // Tipo de nodo, Frontera (-1), Desconocido (>= 0)
17  int    Solución  [NT]      // Se llena con los valores de frontera y al finalizar el calculo
    con los resultados de x
18
19
20  double M[NNE][NNE] // Matriz de Carga
21  double F[NNE]      // Vector de Carga
22
23  double A[ND][ND]   // Matriz de trabajo
24  double x[ND]       // Vector solucion
25  double b[ND]       // Vector de trabajo
26
27  // Algoritmo para llenado de la Matriz A y Vector b, usando la Matriz M y Vector de Carga F
28  for (i = 0; i < NT; i++) { // Recorre sobre todos los nodos
29     if (TipoNodo == -1) continue; // Descarta los nodos de frontera
30     for (j = 0 ; j < NEN; j++) { // Recorre sobre el soporte (numero de elementos que comparten
    el nodo)
31        for(k = 0; k < NNE; k++) { // Recorre sobre los nodos del elemento (Numero de nodos por
    elemento)
32           nd = Nodos[Elementos[i][j]][k]; // Nodo sobre el elemento de soporte
33           if (TipoNodo[nd] == -1) continue; // Nodo es frontera se descarta
34           A[TipoNodo[i]][nd] + = M[j][k];
35           b[TipoNodo[i]] += F[k];
36        }
37     }
38  }
```

```
39
40
41  // Llenada la matriz A y el vector b resolver el sistema lineal
42  Ax=b
43
44
45  // Pasar el valor de x a la Solucion
46  for (i = 0; i < NT; i++) {
47     if (TipoNodo[i] >= 0) Solucion[i] = x[TipoNodo[i]]
48  }
49
50
51  // Visualizar ahora usando Coord y Solucion
52  for (i = 0; i < NT; i++) {
53     for (j = 0 ; j < NEN; j++) visualiza (Coord[i][j])
54     visualiza(Solucion[i]
55  }
56
57
58  Nota: Favor de checar para 1D, 2D y 3D
```