```c
// Dijkstra's Algorithm in C

#include <stdio.h>
#define INFINITY 9999
#define MAX 10

void Dijkstra(int Graph[MAX][MAX], int n, int start);

void Dijkstra(int Graph[MAX][MAX], int n, int start) {
  int cost[MAX][MAX], distance[MAX], pred[MAX];
  int visited[MAX], count, mindistance, nextnode, i, j;

  // Creating cost matrix
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (Graph[i][j] == 0)
        cost[i][j] = INFINITY;
      else
        cost[i][j] = Graph[i][j];

  for (i = 0; i < n; i++) {
    distance[i] = cost[start][i];
    pred[i] = start;
    visited[i] = 0;
  }

  distance[start] = 0;
  visited[start] = 1;
  count = 1;

  while (count < n - 1) {
    mindistance = INFINITY;

    for (i = 0; i < n; i++)
      if (distance[i] < mindistance && !visited[i]) {
        mindistance = distance[i];
        nextnode = i;
      }

```

```
40:     visited[nextnode] = 1;
41:     for (i = 0; i < n; i++)
42:       if (!visited[i])
43:         if (mindistance + cost[nextnode][i] < distance[i]) {
44:           distance[i] = mindistance + cost[nextnode][i];
45:           pred[i] = nextnode;
46:         }
47:     count++;
48:   }
49:
50:   // Printing the distance
51:   for (i = 0; i < n; i++)
52:     if (i != start) {
53:       printf("\nDistance from source to %d: %d", i, distance[i]);
54:     }
55: }
56: int main() {
57:   int Graph[MAX][MAX], i, j, n, u;
58:   n = 7;
59:
60:   Graph[0][0] = 0;
61:   Graph[0][1] = 0;
62:   Graph[0][2] = 1;
63:   Graph[0][3] = 2;
64:   Graph[0][4] = 0;
65:   Graph[0][5] = 0;
66:   Graph[0][6] = 0;
67:
68:   Graph[1][0] = 0;
69:   Graph[1][1] = 0;
70:   Graph[1][2] = 2;
71:   Graph[1][3] = 0;
72:   Graph[1][4] = 0;
73:   Graph[1][5] = 3;
74:   Graph[1][6] = 0;
75:
76:   Graph[2][0] = 1;
77:   Graph[2][1] = 2;
78:   Graph[2][2] = 0;
```

```
 79:   Graph[2][3] = 1;
 80:   Graph[2][4] = 3;
 81:   Graph[2][5] = 0;
 82:   Graph[2][6] = 0;
 83:
 84:   Graph[3][0] = 2;
 85:   Graph[3][1] = 0;
 86:   Graph[3][2] = 1;
 87:   Graph[3][3] = 0;
 88:   Graph[3][4] = 0;
 89:   Graph[3][5] = 0;
 90:   Graph[3][6] = 1;
 91:
 92:   Graph[4][0] = 0;
 93:   Graph[4][1] = 0;
 94:   Graph[4][2] = 3;
 95:   Graph[4][3] = 0;
 96:   Graph[4][4] = 0;
 97:   Graph[4][5] = 2;
 98:   Graph[4][6] = 0;
 99:
100:   Graph[5][0] = 0;
101:   Graph[5][1] = 3;
102:   Graph[5][2] = 0;
103:   Graph[5][3] = 0;
104:   Graph[5][4] = 2;
105:   Graph[5][5] = 0;
106:   Graph[5][6] = 1;
107:
108:   Graph[6][0] = 0;
109:   Graph[6][1] = 0;
110:   Graph[6][2] = 0;
111:   Graph[6][3] = 1;
112:   Graph[6][4] = 0;
113:   Graph[6][5] = 1;
114:   Graph[6][6] = 0;
115:
116:   u = 0;
117:   Dijkstra(Graph, n, u);
```

```
118:
119:    return 0;
120: }
```