Dijikstra's Algorithm:

```
#include <stdio.h>
#include <limits.h>
#define V 5 // Number of vertices in the graph
// Function to find the vertex with the minimum distance value
int minDistance(int dist[], int sptSet[]) {
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (sptSet[v] == 0 && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}

// Function to print the constructed distance array
void printSolution(int dist[]) {
    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}

// Function implementing Dijkstra's algorithm for a graph represented as an adjacency matrix
void dijkstra(int graph[V][V], int src) {
    int dist[V]; // dist[i] will hold the shortest distance from src to i
    int sptSet[V]; // sptSet[i] will be true if vertex i is included in the shortest path tree

    // Initialize all distances as INFINITE and sptSet[] as false
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = 0;

    // Distance of source vertex from itself is always 0
```

```
    dist[src] = 0;

    // Find the shortest path for all vertices
    for (int count = 0; count < V - 1; count++) {
        // Pick the minimum distance vertex from the set of vertices not yet processed
        int u = minDistance(dist, sptSet);

        // Mark the picked vertex as processed
        sptSet[u] = 1;

        // Update dist[] of the adjacent vertices of the picked vertex
        for (int v = 0; v < V; v++)
            // Update dist[v] if it's not in the shortest path tree, there's an edge from u to v,
            // and the total weight of path from src to v through u is smaller than the current value of
dist[v]
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
    }

    // Print the constructed distance array
    printSolution(dist);
}

int main() {
    // Graph represented as an adjacency matrix
    int graph[V][V] = {
        {0, 10, 0, 0, 5},
        {10, 0, 1, 0, 2},
        {0, 1, 0, 4, 0},
        {0, 0, 4, 0, 3},
        {5, 2, 0, 3, 0}
    };
```

```
    dijkstra(graph, 0); // Call Dijkstra's algorithm with source vertex 0

    return 0;
}
```