Here's the pseudocode in English for Dijkstra's algorithm as implemented in the code:

1. Initialize:
   * Set the distance to the starting node as 0
   * Set the distance to all other nodes as infinity
   * Create an empty priority queue
   * Create an array to track parent nodes (for path reconstruction)
   * Create an array to track visited nodes
2. Add the starting node to the priority queue with distance 0
3. While the priority queue is not empty:
   * Remove the node with the smallest distance from the priority queue (call it current node)
   * If the current node is the destination, break the loop
   * If the current node has already been visited, skip it
   * Mark the current node as visited
   * For each neighbor of the current node:
     + Calculate potential new distance = distance to current node + weight of edge to neighbor
     + If the neighbor is not visited AND the potential new distance is less than the current known distance to the neighbor:
       - Update the distance to the neighbor with the new shorter distance
       - Set the parent of the neighbor to be the current node
       - Add the neighbor to the priority queue with its new distance
4. After the loop:
   * If the distance to the destination is still infinity, no path exists
   * Otherwise, reconstruct the path using the parent array by working backwards from the destination
5. To print the path:
   * Use recursion to print in the correct order (from start to end)
   * Base case: when we reach a node with no parent (the start node)
   * Recursive step: first print the path to the parent, then print the current node

This algorithm guarantees finding the shortest path from a starting node to all other nodes in a weighted graph with non-negative edge weights.