Program:

```
# Define infinity as a large enough number
INF = float('inf')


# Function to implement the Floyd-Warshall Algorithm
def floyd_warshall(graph, V):
    # dist[][] will be the output matrix that will finally have the shortest distances between every pair of vertices
    dist = [[INF] * V for _ in range(V)]


    # Initialize the solution matrix same as input graph matrix
    for i in range(V):
        for j in range(V):
            dist[i][j] = graph[i][j]


    # Dynamic Programming approach
    for k in range(V):
        for i in range(V):
            for j in range(V):
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])


    # Print the shortest distance matrix
    print_solution(dist, V)


# Utility function to print the solution
def print_solution(dist, V):
    print("The shortest distances between every pair of vertices are:")
    for i in range(V):
        for j in range(V):
            if dist[i][j] == INF:
                print("INF", end=" ")
```

```
            else:
                print(f"{dist[i][j]:7}", end=" ")
        print()

# Input function to take graph data from user
def input_graph():
    V = int(input("Enter the number of vertices: "))
    graph = []

    print("Enter the adjacency matrix (use INF for no edge): ")
    for i in range(V):
        row = list(map(str, input().split()))
        graph.append([int(x) if x != "INF" else INF for x in row])

    return graph, V

# Main execution
graph, V = input_graph()
floyd_warshall(graph, V)
```

Output: