```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_VERTICES 10

// Adjacency Matrix Representation
int adjMatrix[MAX_VERTICES][MAX_VERTICES];

// Function to add an edge to the graph
void addEdgeMatrix(int u, int v) {
    adjMatrix[u][v] = 1;
    adjMatrix[v][u] = 1;  // For undirected graph
}

// Depth-First Search (DFS) for Adjacency Matrix
void DFSMatrix(int vertex, int visited[MAX_VERTICES]) {
    printf("%d ", vertex);
    visited[vertex] = 1;
    for (int i = 0; i < MAX_VERTICES; i++) {
        if (adjMatrix[vertex][i] == 1 && !visited[i]) {
            DFSMatrix(i, visited);
        }
    }
}

// Breadth-First Search (BFS) for Adjacency Matrix
void BFSMatrix(int start) {
    int visited[MAX_VERTICES] = {0};
    int queue[MAX_VERTICES], front = 0, rear = 0;
    visited[start] = 1;
    queue[rear++] = start;
```

```
    while (front < rear) {
        int current = queue[front++];
        printf("%d ", current);
        for (int i = 0; i < MAX_VERTICES; i++) {
            if (adjMatrix[current][i] == 1 && !visited[i]) {
                visited[i] = 1;
                queue[rear++] = i;
            }
        }
    }
}

// Main function to test the graph
int main() {
    // Initialize adjacency matrix
    for (int i = 0; i < MAX_VERTICES; i++) {
        for (int j = 0; j < MAX_VERTICES; j++) {
            adjMatrix[i][j] = 0;
        }
    }

    addEdgeMatrix(0, 1);
    addEdgeMatrix(0, 2);
    addEdgeMatrix(1, 3);
    addEdgeMatrix(2, 4);

    printf("DFS Traversal: ");
    int visited[MAX_VERTICES] = {0};
    DFSMatrix(0, visited);
    printf("\n");
```

```
    printf("BFS Traversal: ");
    BFSMatrix(0);
    printf("\n");

    return 0;
}
```

```
-Error-umxroegu.jnq' '--pid=Microsoft-MIEngine-Pid-y24wzlh0.xlg' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
DFS Traversal: 0 1 3 2 4
BFS Traversal: 0 1 2 3 4
PS C:\Users\bhand>
```