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

// 定义最大顶点数
#define MAX_VERTICES 100

// 边节点结构（邻接表中的边）
typedef struct ArcNode {
    int adjvex;           // 邻接顶点的索引
    struct ArcNode* next; // 指向下一条边
} ArcNode;

// 顶点节点结构
typedef struct VNode {
    int data;             // 顶点数据（此处简单用索引）
    ArcNode* firstarc;    // 指向第一条边
} VNode;

// 图结构（邻接表）
typedef struct {
    VNode vertices[MAX_VERTICES]; // 顶点数组
    int vexnum;                   // 顶点数
    int arcnum;                   // 边数
} ALGraph;

// 全局访问标记数组
bool visited[MAX_VERTICES];

// 初始化访问标记数组
void InitVisited(int n) {
    for (int i = 0; i < n; i++) {
        visited[i] = false;
    }
```

```
}

// 创建边节点
ArcNode* CreateArc(int adjvex) {
    ArcNode* arc = (ArcNode*)malloc(sizeof(ArcNode));
    if (!arc) {
        printf("内存分配失败！\n");
        exit(1);
    }
    arc->adjvex = adjvex;
    arc->next = NULL;
    return arc;
}

// 构建图（无向图）
void CreateGraph(ALGraph* G) {
    // 初始化图
    G->vexnum = 4; // 4个顶点
    G->arcnum = 4; // 4条边
    for (int i = 0; i < G->vexnum; i++) {
        G->vertices[i].data = i;
        G->vertices[i].firstarc = NULL;
    }

    // 定义边：(0,1), (0,2), (1,3), (2,3)
    int edges[][2] = {{0, 1}, {0, 2}, {1, 3}, {2, 3}};

    for (int i = 0; i < G->arcnum; i++) {
        int v1 = edges[i][0]; // 起点
        int v2 = edges[i][1]; // 终点

        // 添加边 v1 -> v2
        ArcNode* arc1 = CreateArc(v2);
        arc1->next = G->vertices[v1].firstarc;
        G->vertices[v1].firstarc = arc1;
```

```
        // 添加边 v2 -> v1（无向图）
        ArcNode* arc2 = CreateArc(v1);
        arc2->next = G->vertices[v2].firstarc;
        G->vertices[v2].firstarc = arc2;
    }
}

// 访问顶点的函数
void Visit(int v) {
    printf("%d ", v);
}

// 深度优先搜索
void DFS(ALGraph* G, int i) {
    Visit(i);                       // 访问顶点 i
    visited[i] = true;              // 标记为已访问
    ArcNode* p = G->vertices[i].firstarc; // 获取第一条边
    while (p) {                     // 遍历所有邻接边
        int j = p->adjvex;          // 获取邻接顶点
        if (!visited[j]) {          // 若未访问
            DFS(G, j);              // 递归访问
        }
        p = p->next;                // 移动到下一条边
    }
}

// 遍历整个图（处理非连通图）
void DFSTraverse(ALGraph* G) {
    InitVisited(G->vexnum);         // 初始化访问标记
    printf("DFS 遍历顺序: ");
    for (int i = 0; i < G->vexnum; i++) {
        if (!visited[i]) {          // 若顶点未访问
            DFS(G, i);              // 从该顶点开始DFS
        }
```

```
    }
    printf("\n");
}

// 释放图的内存
void FreeGraph(ALGraph* G) {
    for (int i = 0; i < G->vexnum; i++) {
        ArcNode* p = G->vertices[i].firstarc;
        while (p) {
            ArcNode* temp = p;
            p = p->next;
            free(temp);
        }
    }
}

// 主函数
int main() {
    ALGraph G;
    CreateGraph(&G);        // 构建图
    DFSTraverse(&G);        // 执行DFS遍历
    FreeGraph(&G);          // 释放内存
    return 0;
}
```