```c
#define Max 100
#define Max_E 100
// duyet do thi theo dang dinh dinh
typedef struct
{
    int A[Max_E][Max_E];
    int n, m;
} Graph;
// tao so do dinh dinh
void init_graph(Graph *G, int n)
{
    int i, j;
    G->n = n;
    for (i = 1; i <= G->n; i++)
    {
        for (j = 1; j <= G->n; j++)
        {
            G->A[i][j] = 0;
        }
    }
}
// khoi tao do thi dinh cung
void init_graph1(Graph *G, int n, int m)
{
    int i, j;
    G->n = n;
    G->m = m;
    for (i = 1; i <= G->n; i++)
    {
        for (j = 1; j <= G->m; j++)
        {
            G->A[i][j] = 0;
        }
    }
}
// theo cung vao do thi dinh - dinh
void add_edge(Graph *G, int x, int y)
{
    G->A[x][y] = 1;
    // G->A[y][x] = 1;
}
// them cung vao do thi dinh cung
void add_edge1(Graph *G, int x, int y, int e)
{
    G->A[x][e] = 1;
    // G->A[y][e] = 1;
}
// kiem tra co phai lang gieng
int adjacent(Graph G, int x, int y)
{
    return G.A[x][y] == 1;
}
// tinhh Qo bac cua mot dinh
int degree(Graph *G, int x)
{
    int i;
    int deg = 0;
    for (i = 1; i <= G->n; i++)
    {
        if (G->A[x][i] == 1)
        {
            deg++;
        }
    }
    return deg;
}
//// in do thi dinh-dinh
void in(Graph G)
{
```

```
    int i, j;
    for (i = 1; i <= G.n; i++)
    {
        for (j = 1; j <= G.n; j++)
        {
            printf("%d ", G.A[i][j]);
        }
        printf("\n");
    }
}
// in do thi dinh-cung
void in1(Graph G)
{
    int i, j;
    for (i = 1; i <= G.n; i++)
    {
        for (j = 1; j <= G.m; j++)
        {
            printf("%d ", G.A[i][j]);
        }
        printf("\n");
    }
}
```