

```

1  #define Max 100
2  #define Max_E 100
3  // duyệt đồ thị theo dạng đỉnh đỉnh
4  typedef struct
5  {
6      int A[Max_E][Max_E];
7      int n, m;
8  } Graph;
9  // tạo số đồ thị đỉnh đỉnh
10 void init_graph(Graph *G, int n)
11 {
12     int i, j;
13     G->n = n;
14     for (i = 1; i <= G->n; i++)
15     {
16         for (j = 1; j <= G->n; j++)
17         {
18             G->A[i][j] = 0;
19         }
20     }
21 }
22 // khởi tạo đồ thị đỉnh đỉnh
23 void init_graph1(Graph *G, int n, int m)
24 {
25     int i, j;
26     G->n = n;
27     G->m = m;
28     for (i = 1; i <= G->n; i++)
29     {
30         for (j = 1; j <= G->m; j++)
31         {
32             G->A[i][j] = 0;
33         }
34     }
35 }
36 // theo cùng vào đồ thị đỉnh - đỉnh
37 void add_edge(Graph *G, int x, int y)
38 {
39     G->A[x][y] = 1;
40     // G->A[y][x] = 1;
41 }
42 // thêm cùng vào đồ thị đỉnh cùng
43 void add_edge1(Graph *G, int x, int y, int e)
44 {
45     G->A[x][e] = 1;
46     // G->A[y][e] = 1;
47 }
48 // kiểm tra có phải láng giềng
49 int adjacent(Graph G, int x, int y)
50 {
51     return G.A[x][y] == 1;
52 }
53 // tính bậc của một đỉnh
54 int degree(Graph *G, int x)
55 {
56     int i;
57     int deg = 0;
58     for (i = 1; i <= G->n; i++)
59     {
60         if (G->A[x][i] == 1)
61         {
62             deg++;
63         }
64     }
65     return deg;
66 }
67 // in đồ thị đỉnh-đỉnh
68 void in(Graph G)
69 {

```

```
70     int i, j;
71     for (i = 1; i <= G.n; i++)
72     {
73         for (j = 1; j <= G.n; j++)
74         {
75             printf("%d ", G.A[i][j]);
76         }
77         printf("\n");
78     }
79 }
80 // in do thi dinh-cung
81 void in1(Graph G)
82 {
83     int i, j;
84     for (i = 1; i <= G.n; i++)
85     {
86         for (j = 1; j <= G.m; j++)
87         {
88             printf("%d ", G.A[i][j]);
89         }
90         printf("\n");
91     }
92 }
```