```cpp
#include"MGraph.h"

#define pathMatrix bool
#define DistancMattrix int

pathMatrix p[MAX_VERTEX_NUM][MAX_VERTEX_NUM][MAX_VERTEX_NUM];
DistancMattrix D[MAX_VERTEX_NUM][MAX_VERTEX_NUM];

void ShortestPath_FLOYD(MGraph G){
    //各节点之间初始已知路径及距离
    for(int v=0; v<G.vexnum; v++)
        for(int w=0; w<G.vexnum; w++){
            D[v][w] = G.arcs[v][w].adj;
            for(int u=0; u<G.vexnum; u++)
                p[v][w][u] = false;
            if(D[v][w]<INFINITY){    //从v到w有直接路径
                p[v][w][v] = true;
                p[v][w][w] = true;
            }//if
        }//for

    for(int u=0; u<G.vexnum; u++)
        for(int v=0; v<G.vexnum; v++)
            for(int w=0; w<G.vexnum; w++)
                //从v经u到w的一条路径更短
                if(D[v][u]+D[u][w] < D[v][w]){
                    D[v][w] = D[v][u] + D[u][w];
                    //修改最短路径
                    for(int i=0; i<G.vexnum; i++)
                        p[v][w][i] = p[v][u][i] || p[u][w][i];
                }//if

}//ShortestPath_FLOYD


int main(){
    MGraph G;
    CreateGraph(G);
    PrintAdjMatrix(G);
    ShortestPath_FLOYD(G);

    cout << G.arcs[2][2].adj << "\t" << G.arcs[3][2].adj<<endl;

    for(int i=0; i<G.vexnum; i++){
        for(int j=0; j<G.vexnum; j++){
            for(int k=0; k<G.vexnum; k++)
                cout << p[i][j][k] << "\t";
            cout << "\t" << D[i][j] << endl;
        }
        cout << endl << endl << endl;
    }

    return 0;
}

```