```
/* ALGORITMO "TEORICO PARA LA PRUEBA DE ACICLICIDAD DE UN GRAFO (NO DIRIGIDO) QUE
 * TIENE N VERTICES Y DEL CUAL SE CONOCE SU LISTA DE ADYACENCIA
 *
 * NOTA: LA LISTA DE ADYACENCIA SE IMPLEMENTA MEDIANTE
 * UN ARREGLO QUE CONTIENE APUNTADORES A LISTAS ENLAZADAS
 * DE LOS VERTICES ADYACENTES:
 *        Lista[N]
 *        0 [0|·]-> NULL          NO HAY VERTICE CERO
 *        1 [1|·]-> [x|·]-> ··· -> [y|·]-> NULL
 *        1 [2|·]-> [f|·]-> ··· -> [g|·]-> [h|·]-> NULL
 *        .
 *        .
 *        .
 *    n-1 [n-1|·]-> [p|·]-> ··· -> [q|·]-> NULL
 */
int vertices[n];          // N VETICES

// ESTRUCTURA PARA LA LISTA DE ADYACENCIA
struct ady {
        int valor;                // VERTICE ADYACENTE
        struct ady * siguiente; // PUNTERO A "struct ady" siguiente
}

//SINONIMO
typedef struct ady sady;      // sady es una "struct ady"
typedef sady * psady;                 // psady es un "puntero a struct ady"

//LISTA DE ADYACENCIA                 UN ARREGLO DE "n punteros a struct ady"
psady lista[n];

/* FUNCION: prueba
 * RETORNA:   1 en caso de haber ciclo en el grafo
 *            0 en caso de no haberlos
 * ARGUMENTOS:
 *            psady lista[n]
 */
int prueba( psady lista[n] )
{
        psady tempi;
        psady tempj;
        int i, j;
        for( i = 1 ; i < n ; i++ )
        {
                tempi = lista[i];
                for( j = n-1 ; j > i ; j--)
                {
                        tempj = lista[j];
                        while( tempi->siguiente != NULL && tempj->siguiente != NULL )
                        {
                                if( tempj->valor == i && j == tempi->valor )
                                        return 1; //HAY CICLO
                                else
                                {
                                        tempj = tempj->siguiente;
                                        tempi = tempi->siguiente;
                                }
                        }
                }
        }
        //AQUI LA LISTA HA SIDO RECORRIDA TODA SIN ENCONTRAR CICLOS
        return 0; //NO HAY CICLOS
}
```