```
┌──────────────────────── MODULE Euclid ────────────────────────┐
EXTENDS Integers, GCD, TLC

CONSTANTS N

ASSUME  ∧ N ∈ Nat \ {0}

 *************************************************
--fair algorithm Euclid{
  variables x ∈ 1 . . N, y ∈ 1 . . N, x0 = x, y0 = y ;
  { abc: while ( x ≠ y )
    { d: if ( x < y ) { y := y − x ;  }
          else        { x   := x − y ;  }
     } ;

    at:    assert (x = y) ∧ (x = GCD(x0, y0)) ;
   }
 }
 *************************************************

 BEGIN TRANSLATION
VARIABLES x, y, x0, y0, pc

vars ≜ ⟨x, y, x0, y0, pc⟩

Init ≜  Global variables
          ∧ x ∈ 1 . . N
          ∧ y ∈ 1 . . N
          ∧ x0 = x
          ∧ y0 = y
          ∧ pc = "abc"

abc ≜  ∧ pc = "abc"
       ∧ IF x ≠ y
             THEN ∧ pc′ = "d"
             ELSE ∧ pc′ = "at"
       ∧ UNCHANGED ⟨x, y, x0, y0⟩

d ≜  ∧ pc = "d"
     ∧ IF x < y
           THEN ∧ y′ = y − x
                ∧ x′ = x
           ELSE ∧ x′ = x − y
                ∧ y′ = y
     ∧ pc′ = "abc"
     ∧ UNCHANGED ⟨x0, y0⟩

at ≜  ∧ pc = "at"
```

```
          ∧ Assert((x = y) ∧ (x = GCD(x0, y0)),
                   “Failure of assertion at line 16, column 9.”)
          ∧ pc′ = “Done”
          ∧ UNCHANGED ⟨x, y, x0, y0⟩

Next ≜ abc ∨ d ∨ at
           ∨ Disjunct to prevent deadlock on termination
             (pc = “Done” ∧ UNCHANGED vars)

Spec ≜ ∧ Init ∧ □[Next]_vars
       ∧ WF_vars(Next)

Termination ≜ ◇(pc = “Done”)

 END TRANSLATION
```