```
1     // Return true if odd integer n is prime
2
3     public static boolean isPrime( long n )
4     {
5         for( long i = 3; i * i <= n; i += 2 )
6             if( n % i == 0 )
7                 return false;    // Not prime
8
9         return true;             // Prime
10    }
```

Figure 9.8 Primality testing by trial division

The simplest algorithm for testing if an odd number N is prime is *trial division*. In this algorithm, an odd number greater than 3 is prime if it is not divisible by any other odd number smaller than or equal to $\sqrt{N}$. A direct implementation of this strategy is shown in Figure 9.8.

Trial division is a simple algorithm for primality testing. It is fast for small (32-bit) numbers but cannot be used for larger numbers.

Trial division is reasonably fast for small (32-bit) numbers, but it is unusable for even 64-bit `long`s because it could require the testing of roughly $\sqrt{N}/2$ divisors, thus using $O(\sqrt{N})$ time. What we want is a test whose running time is of the same order of magnitude as the `power` routine in Section 7.4.2. A well-known theorem, called *Fermat's Little Theorem*, looks very promising. (A proof of this theorem is provided in Theorem 9.1 for completeness, but it is not needed to understand the primality-testing algorithm.)

Theorem 9.1 *(Fermat's Little Theorem): If P is prime and $0 < A < P$, then $A^{P-1} \equiv 1 \pmod{P}$.*

Proof *Consider any $1 \le k < P$. $Ak \equiv 0 \pmod{P}$ is impossible, since P is prime and greater than A and k. Now consider any $1 \le i < j < P$. $Ai \equiv Aj \pmod{P}$ would imply $A(j-i) \equiv 0 \pmod{P}$. This is impossible by the previous argument because $1 \le j - i < P$. Thus the sequence $A, 2A, \ldots, (P-1)A$, when considered $\pmod{P}$, is a permutation of $1, 2, \ldots, P-1$. The product of both sequences $\pmod{P}$ must be equivalent, thus yielding the equivalence $A^{P-1}(P-1)! \equiv (P-1)! \pmod{P}$ from which the theorem follows.*

Fermat's Little Theorem is necessary but not sufficient to establish primality.

If the converse of Fermat's Little Theorem was true, then we would have a primality-testing algorithm that would be computationally equivalent to modular exponentiation (that is, $O(\log N)$). Unfortunately, the converse is not true. It is easily verified that $2^{340} \equiv 1 \pmod{341}$, but 341 is composite ($11 \cdot 31$).