# The Continue Statement

The continue statement passes control to the next iteration of the do, for, or while statement in which it appears, bypassing any remaining statements in the do, for, or while statement body. A typical use of the continue statement is to return to the start of a loop from within a deeply nested loop.

**Syntax**

```
jump-statement :
  continue;
```

**The next iteration of a do, for, or while statement is determined as follows:**

- Within a do or a while statement, the next iteration starts by reevaluating the expression of the do or while statement.

- A continue statement in a for statement causes the first expression of the for statement to be evaluated. Then the compiler reevaluates the conditional expression and, depending on the result, either terminates or iterates the statement body.

**This is an example of the continue statement:**

```
while ( i-- > 0 )
{
   x = f( i );
   if ( x == 1 )
      continue;
   y += x * x;
}
```

In this example, the statement body is executed while **i** is greater than 0. First f(i) is assigned to x; then, if x is equal to 1, the continue statement is executed. The rest of the statements in the body are ignored, and execution resumes at the top of the loop with the evaluation of the loop's test.