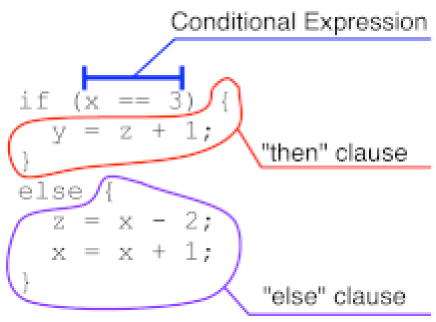
If/Else

coursera.org/learn/programming-fundamentals/supplement/zZFOA/if-else

if/else

Now that we understand comparison operators, and can compare expressions, we can discuss the evaluation of if/else statements. The syntax for an if/else statement is shown in figure below.

The keyword if is followed by an expression in parenthesis. This expression is evaluated to a value, to determine whether the "then" block or the "else" block is executed. The "then" block of code comes immediately after the expression. C does not have a then keyword (although some languages do), however, this block of code serves the same purpose regardless of the syntactic particulars of the language—it is executed if the conditional expression evaluates to true. After the "then" block, we have the keyword else, followed by the "else" block. This block of code is executed if the conditional expression evaluates to false.



```
if (x == 3) {
  y = z + 1;
}
else {
  z = x - 2;
  x = x + 1;
}
```

When your execution arrow reaches an if statement, evaluate the conditional expression. Evaluating this expression proceeds just like evaluating any other expression. If the result is true, move the execution arrow immediately inside the "then" block and continue executing statements as usual. When your execution reaches the close curly brace that ends the "then" block, skip over the else block, placing your execution arrow immediately after the close curly brace of the "else" block, and continue executing statements from there.

If the result of the conditional expression is false, you should instead skip the "then" block and execute the "else" block. Move your execution arrow into the start of the "else" block, and continue executing statements from there. When your execution arrow reaches the close curly brace that ends the "else" block, simply move it past that curly brace (which has no effect—it just denotes the end of the block) and continue executing statements normally.

C permits if with no else, which is equivalent to an empty "else" block (as if the programmer had written else {}). If you execute an if with no else, then simply imagine the empty "else" block. If the conditional expression evaluates to true, you should execute the "then" block as previously described, however, there is no "else" block to skip. Instead, continue executing statements immediately after the end of the "then" block (skipping over the non-existent "else" block). If the conditional expression evaluates to false, then skip the "then" block, and execute whatever statements follow it (doing nothing for the "else" block).

if/else statements may be nested—one (or more) may occur in the “then” or “else” block of another if/else statement. When you encounter nested statements, the same rules apply. The inner statement is just one of the (possibly) many statements in the block, and is executed according to its rules—the condition is evaluated, whichever of the “then” or “else” blocks is appropriate is executed, and then execution continues after the end of the “else” block. When the execution arrow reaches the end of the outer “then” or “else” block, it behaves no differently than if there were no inner if statement. The next video demonstrates the execution of some if/else statements.