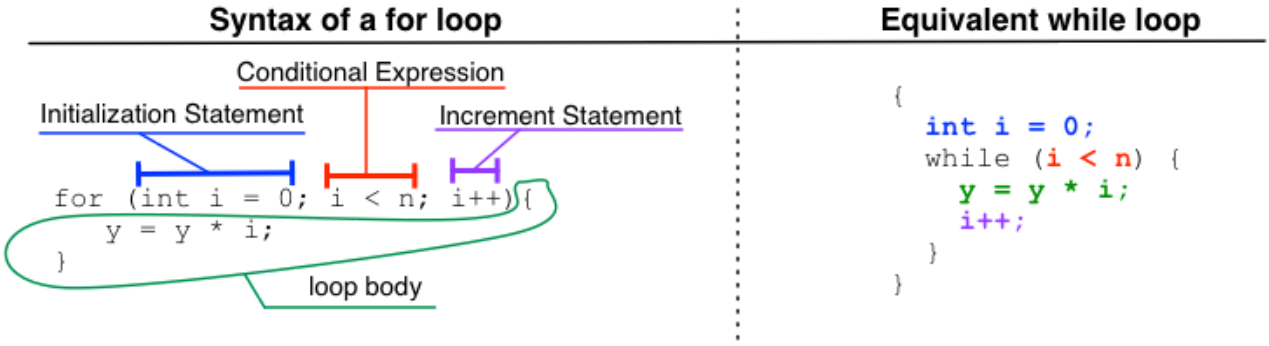
For Loops

coursera.org/learn/programming-fundamentals/supplement/kznvG/for-loops

The third type of loop in C is a for loop. The for loop is syntactic sugar—it does not introduce any new behavior, but instead provides a more convenient syntax for a common programming idiom. In the case of for loops, the common idiom is counting from one number to another. The following figure shows the syntax of a for loop, and how it is de-sugared into a while loop—that is, how we could write the for loop in terms of the already familiar while loop. Knowing how the for loop de-sugars to a while loop tells us how to execute it. We can imagine the equivalent while loop, and follow the execution rules we have already learned for it.



The for keyword is followed by three pieces, separated by semicolons, inside of parenthesis. The first of these is the “initialization statement”. It happens once before the first time the loop’s condition is checked. In the de-sugaring, this statement appears right before the while loop. The second piece is not a statement (even though it is followed by a semicolon), but rather the conditional expression for the loop. In the de-sugaring, this expression is the conditional expression of the while loop. The third statement is the “increment statement”. In the de-sugaring, it appears immediately before the close curly brace of the loop body. After all of these is the loop body, which (except for the addition of the “increment statement” at the end) is the loop body of the while loop in the de-sugared version.

If you examine the figure carefully, you will notice that there is a set of curly braces around the entire piece of while-based code. These curly braces are there for a subtle, but important reason. The scope of any variables declared in the “initialization statement” of the for loop have a scope which is limited to the for loop. Recall that a variable typically has a scope which is limited to the curly braces which enclose its declaration. For a variable declared in the start of the for loop, the scope appears to be an exception to this rule, however, it is not if we think of it in terms of the de-sugaring shown above with the curly braces surrounding the declaration.

Nesting

Just as if/else statements may be nested, loops may also be nested. Similarly, loops follow exactly the same rules no matter how they are nested. In fact, if/else statements and loops may be nested within each other in any combinations. The rules are always the same regardless of any combinations or depths of nesting.