CONDITIONAL STATEMENTS IN LOOPS

Evaluating expressions as part of Loops:

When writing a CONDITIONAL STATEMENT for a loop, you must also use a Boolean expression. In a loop, the code inside the loop will continue to execute repeatedly as long as the Boolean expression evaluates as `true`.
Lets look at several examples of Boolean expressions using the variables on the right:

Assume these variables

Variable	Value
i	0
j	1

Loop using relational operators:

```
while ( i < 5 )
    i = i + 2;
```

We need to observe the loop through multiple iterations. Since the same code will be executed many times, we will evaluate the Boolean expression each time, then, if the loop continues, record the change in any variables:

Code	1st Iteration	2nd Iteration	3rd Iteration	4th Iteration
`while (i < 5)`	(0 < 5) -> (true)	(2 < 5) -> (true)	(4 < 5) -> (true)	(6 < 5) -> (false)
`i = i + 2;`	i = 0 + 2 -> i = 2	i = 2 + 2 -> i = 4	i = 4 + 2 -> i = 6	Skipped

Conditional expression using relational operators and mathematical expressions:

```
while ( i * 2 < i + j +1) {
    i = i + 2;
    j = j + 1;
}
```

This time, the loop contains two statements to execute, so we need a code block (curly braces) to group our statements together:

Code	1st Iteration	2nd Iteration	3rd Iteration
`while (i * 2 < i + j) {`	(0 * 2 < 0 + 1 + 1) -> (0 < 2) -> (true)	(2 * 2 < 2 + 2 + 1) -> (4 < 5) -> (true)	(4 * 2 < 4 + 3 + 1) -> (8 < 8) -> (false)
`i = i + 2;`	i = 0 + 2 -> i = 2	i = 2 + 2 -> i = 4	Skip
`j = j + 1;`	j = 1 + 1 -> j = 2	j = 2 + 1 -> j = 3	Skip
`}`			

Loops can also be based on logical operators of course, but they most often use relational operators so that is where we will focus our attention!