Programmers use the **ternary operator** for decision making in place of longer **if** and **else** conditional statements.

The ternary operator take three arguments:

1. The first is a comparison argument
2. The second is the result upon a true comparison
3. The third is the result upon a false comparison

It helps to think of the ternary operator as a shorthand way or writing an if-else statement. Here’s a simple decision-making example using **if** and **else**:

int a = 10, b = 20, c;

if (a < b) {

c = a;

}

else {

c = b;

}

printf("%d", c);

This example takes more than 10 lines, but that isn’t necessary. You can write the above program in just 3 lines of code using a ternary operator.

**Syntax**

condition ? value\_if\_true : value\_if\_false

The statement evaluates to value\_if\_true if condition is met, and value\_if\_false otherwise.

Here’s the above example rewritten to use the ternary operator:

int a = 10, b = 20, c;

c = (a < b) ? a : b;

printf("%d", c);

Output of the example above should be:10

c is set equal to a, because the condition a < b was true.

Remember that the arguments value\_if\_true and value\_if\_false must be of the same type, and they must be simple expressions rather than full statements.

Ternary operators can be nested just like if-else statements. Consider the following code:

int a = 1, b = 2, ans;

if (a == 1) {

if (b == 2) {

ans = 3;

} else {

ans = 5;

}

} else {

ans = 0;

}

printf ("%d\n", ans);

Here's the code above rewritten using a nested ternary operator:

int a = 1, b = 2, ans;

ans = (a == 1 ? (b == 2 ? 3 : 5) : 0);

printf ("%d\n", ans);

The output of both sets of code above should be:

3