Teaching guide: Relational operations

This resource will help with understanding relational operations. It supports Section 3.2.4 of our GCSE Computer Science specification (8525). The guide is designed to address the following learning aims:

- Recognise the six relational operators.
- Evaluate Boolean expressions that use relational operators.

Ordering integers

Many data types (integer, character, string etc) have an 'order', which means that given two variables `a` and `b` of the same data type only one of three possibilities is true:

- the value of `a` is less than the value of `b` (`a` comes *before* `b` in that data type's ordering)
- the value of `a` is equal to the value of `b` (`a` and `b` are at the same position in that data type's ordering)
- the value of `a` is greater than the value of `b` (`a` comes *after* `b` in that data type's ordering)

In programming languages the following operators are used (they are known as logical operators because the result of evaluating an expression using one is a Boolean value, either `True` or `False`):

Logical operator	Meaning	Example
<	less than	`3 < 4` means `3` is less than `4`
>	greater than	`4 > 3` means `4` is greater than `3`
=	equal to	`3 = 3` means `3` is equal to `3`
≠	not equal to	`3 ≠ 4` means `3` is not equal to `4`
≤	less than or equal to	`3 ≤ 4` means `3` is less than or equal to `4`
≥	greater than or equal to	`4 ≥ 4` means `4` is greater than or equal to `4`

All the examples above use values (`3` and `4`) directly and so it's obvious what the results of the logical expression are (in fact they all evaluate to `True`). When programming using logical operators you will

- compare the value of a variable to a value, eg `a > 0`
- compare the value of one variable to the value of another, eg `a ≤ b`
- combine the results of two or more logical comparisons using the Boolean operators **`NOT`**, **`AND`** and **`OR`**, eg `(a < b)` **`AND`** `(b < c)`

Arithmetic operations are evaluated *before* relational operations and it is fairly simple to combine the two types of operations although it is more readable to use brackets around the arithmetic expressions. For example:

`(3 * 4) ≤ 15` evaluates to `12 ≤ 15` which evaluates to `True`

`(15 MOD 2) = 0` evaluates to `1 = 0` which evaluates to `False`

`(50 - 40) > (3 * (2 + 5))` evaluates to `10 > 21` which evaluates to `False`

Ordering other types

All real numbers can be compared using the six relational operators above. Strings can also be ordered using 'dictionary' or lexicographic order, eg `'aardvark'` is less than `'apple'` because it comes before it in the dictionary.

Using logical operations in programming

When programming you will use logical operations mainly in *selection* and *condition-controlled iteration* statements. There are Teaching Guides for these but here are a few examples:

Example code	Meaning

`IF a > b THEN`
`  temp ← a`
`  a ← b`
`  b ← temp`
`ENDIF`
	If the value of `a` is greater than the value of `b` then, using a temporary variable, exchange the two values.

`WHILE a ≤ b`
`  a ← a * 2`
`ENDWHILE`
	As long as the value of `a` is less than or equal to `b` double `a`.

`REPEAT`
`  a ← a * 2`
`UNTIL a > b`
	Double the value in `a` until `a > b`.
	To convert a **`WHILE`** statement to a **`REPEAT-UNTIL`** statement you invert the condition, ie ≤ becomes >, < becomes ≥, = becomes ≠ and so on.

The IF example

After this has executed what will be the value of `b ≤ a`?

The WHILE example

How many times will this loop be run if the initial values of `a` and `b` are `1` and `27`? What will the final value of `a` be?

How many times will this loop run if the initial values of `a` and `b` are `27` and `1`? What will the final value of `a` be?

The REPEAT UNTIL example

How many times will this loop be run if the initial values of `a` and `b` are `1` and `27`? What will the final value of `a` be?

How many times will this loop run if the initial values of `a` and `b` are `27` and `1`? What will the final value of `a` be?