Quick reference

Daniel Berio

October 10, 2023

Contents

1 Variables

2 C/C++ operators

Operators are symbols that represents a specific mathematical or logical operation.

2.1 Assignment

In C++ the `=` symbol is thet operator that assigns one value to another. For example

```
a = 10;
```

assigns the value `10` to the variable `a` (assuming it has been previously declared).

2.2 Arithmetic operators

Operator	Result	Notes
`+`	Addition	
`-`	Subtraction	
`*`	Multiplication	
`/`	Division	
`%`	Modulo (remainder of division)	Applies only to integers

Arithmetic operators have different *precedence*, with multiplication, division and modulo (`*`, `/` and `%`) being applied before addition and subtraction (`+` and `-`). We can use parentheses to force a desired precedence. For example `a+b*c` will not be equal to `(a+b)*c`, where in the first `a` is added to `b*c`, while in the second `(a + b)` is multiplied by `c`. That is similar to the way we would express this with mathematical notation (which makes it somewhat more obvious), where the first would be $a + bc$ and the second $(a + b)c$.

2.3 Relational operators

Relational (or comparison) operators are used to test the relation between two variables. They always result in a boolean (`bool` type) value being either `true` or `false`

Operator	Relation	Notes
`<`	Less than	E.g. `5 < 10` is `true`
`>`	Greater than	E.g. `5 > 10` ia `false`
`<=`	Less or equal to	E.g. `10 <= 10` is `true`
`>=`	Greater or equal to	E.g. `12 >= -10` is `true`
`==`	Equal	E.g. `a==b` is `true` if `a` and `b` have the same value

2.4 Logical operators

Logical operators are used to compose expressions made of boolean (`true` or `false`) values. C++ gives three logical operators `&&` (AND), `||` (OR) and `!` (NOT). Similarly to English, the first two are always applied to two values, one on the left and one on the right. E.g `a && b` will be `true` only if both `a` and `b` are true, while `a || b` will be `true` if either of `a` or `b` is `true`. Instead, the NOT (`!`) operators applies to the value on its right. E.g. `!true` is `false`, which with the relational operators can be expressed as `!true==false`. We can use relational operators together with logical operator as a powerful way

to test different conditions, e.g. to store whether a variable `v` is betwen two numbers `a` and `b` we could do

```
bool isBetween = (v >= a) && (v <= b);
```

This can also be written in a perhaps more concise (but cryptic?) way as:

```
bool isBetween = a <= v <= b;
```

Together with an `if` statement we can use this kind of expression to perform some actions if `v` this condition is **not** true:

```
if (!((v >= a) && (v <= b))) {
  // Do some action
}
```

Note that we wrapped the whole expression in parentheses in order to apply the NOT operator to the result. This results in many parentheses and the result would be more readable if we use the `isBetween` variable and write

```
if (!isBetween) {
  // Do stuff
}
```