Learning Objectives: Boolean Operators

- **Recognize the difference between = and ==**
- **Understand the && and || operators' functions**
- **Evaluate boolean expressions**

Equal To & Not Equal To

Boolean operators are operators that return a boolean value (true or false).

Equal To

C++ uses the `==` operator to determine *equality*. Beginners often confuse the `=` and the `==` operators. Remember, `=` is the *assignment* operator.

```
int a = 5;
int b = 5;
cout << boolalpha << (a == b) << endl;
```

challenge

What happens if you:

- Assign `b` to `1`?
- Change `a` to `bool a = true;` and `b` to `bool b = false;`?

Not Equal To

The `!=` operator checks to see if two values are *not equal*.

```
int a = 5;
int b = 5;
cout << boolalpha << (a != b) << endl;
```

challenge

What happens if you:

- Assign `b` to `1`?
- Change `a` to `bool a = true;` and assign `b` to `1`?
- Change `b` to `bool b = false;`?

Less Than & Less Than or Equal To

Less Than

The `<` operator is used to check if one value is *less than* another value.

```
int a = 5;
int b = 7;
cout << boolalpha << (a < b) << endl;
```

challenge

What happens if you:

- Assign `b` to `1`?
- Assign `b` to `5`?
- Assign `b` to `false`?

▼ **Hint(s)**

It is possible to declare and assign `int b = false;` because `false` is just a value of `0`. Since `5` is not less than `0`, `false` is returned.

Less Than or Equal To

The `<=` operator is used to check if one value is *less than or equal to* another value.

```
int a = 5;
int b = 7;
cout << boolalpha << (a <= b) << endl;
```

challenge

What happens if you:

- Assign `b` to `1`?
- Assign `b` to `5`?
- Assign `a` to `false` and assign `b` to `true`?

▼ **Hint(s)**

`false` is less than `true` because `0` is less than `1`.

Greater Than & Greater Than or Equal To

Greater Than

The `>` operator is used to check if one value is *greater* than another value.

```
int a = 9;
int b = 17;
cout << boolalpha << (a > b) << endl;
```

challenge

What happens if you:

- Assign `b` to `1`?
- Assign `b` to `9`?
- Assign `b` to `false`?
- Assign `b` to `true`?

▼ **Hint(s)**

`9` is both greater than the value of `false`, which is `0`, and the value of `true`, which is `1`.

Greater Than or Equal To

The `>=` operator is used to check if one value is *greater than or equal* to another value.

```
int a = 9;
int b = 17;
cout << boolalpha << (a >= b) << endl;
```

challenge

What happens if you:

- Assign `b` to `1`?
- Assign `b` to `9`?
- Assign `a` to `true` and assign `b` to `false`?

▼ **Hint(s)**

`true` is greater than `false`.

And

The && Operator

The && (and) operator allows for compound (more than one) boolean expressions. **All** boolean expressions **must** be true in order for the whole thing to be true. If at least **one** boolean expressions is false, then the whole thing is false.

```
bool a = true;
bool b = true;
bool c = false;
cout << boolalpha << (a && b) << endl;
```

▼ **How do I type &&?**
It is located towards the top of the keyboard, on the same key as the number 7. Hold `shift` and press the 7 key to type &.

challenge

What happens if you:

- Replace `(a && b)` in the code above with `(a && c)`?
- Replace `(a && b)` in the code above with `(b && c)`?

Multiple && Statements

You can chain several && expressions together. They are evaluated in a left-to-right manner.

```
bool a = true;
bool b = true;
bool c = false;
cout << boolalpha << (a && b && c) << endl;
```

challenge

What happens if you:

- Replace `(a && b && c)` in the code above with `(a && b && a && b && a)`?
- Replace `(a && b && c)` in the code above with `(a && b && a && b && c)`?

▼ **Hint(s)**

`c` is the only variable is that is `false`. Thus, if `c` is involved in an `&&` expression, the entire thing will evaluate to `false`. Any combinations of `a`s and/or `b`s will result in `true`.

Or

The || Operator

The || (or) operator allows for compound (more than one) boolean expressions. If at least **one** boolean expression is true, then the whole thing is true. To be false, **all** boolean expressions **must** be false.

```
bool a = true;
bool b = true;
bool c = false;
bool d = false;
cout << boolalpha << (a || b) << endl;
```

▼ **How do I type ||?**
It is towards the right-hand side, below the backspace or delete key and above the enter or return key. The | symbol is located on the same key as the </code> symbol. Hold shift and press the </code> key to type |.

challenge

What happens if you:

- Replace (a || b) in the code above with (a || c)?
- Replace (a || b) in the code above with (c || d)?

Multiple || Statements

You can chain several || expressions together. They are evaluated in a left-to-right manner.

```
bool a = true;
bool b = true;
bool c = false;
cout << boolalpha << (a || b || c) << endl;
```

challenge

What happens if you:

- Replace `(a || b || c)` in the code above with `(a || c || c || c || c)`?
- Replace `(a || b || c)` in the code above with `(c && c && c && c && c)`?

Not

The ! Operator

The `!` (not) operator produces the *opposite* result of the boolean expression that it modifies.

```
cout << boolalpha << (! true) << endl;
```

challenge

What happens if you:

- Replace `(! true)` in the code above with `(! true && false)`?
- Replace `(! true)` in the code above with `(! (true && false))`?
- Replace `(! true)` in the code above with `(! ! true)`?

▼ **Hint(s)**

The `!` operator works similarly to how a `-` (negative) sign works in mathematics. The `-` of a positive number is a negative number and the `-` of a negative number is a positive number.

Order of Boolean Operators

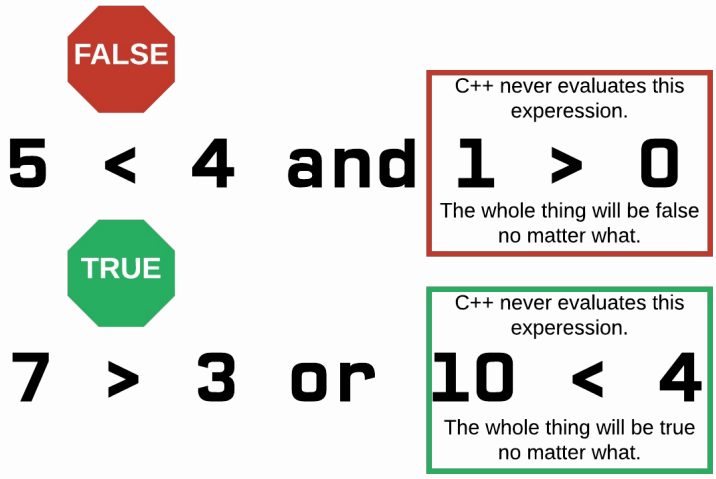
Much like how arithmetic operators are evaluated in a certain order, boolean operators also work according to their priority level. Boolean operations are evaluated in the following order from highest to lowest priority:

1. Parentheses `()`
2. Not `!`
3. And `&&`
4. Or `||`

Short Circuiting

Short Circuiting

If C++ can determine the result of a boolean expression before evaluating the entire thing, it will stop and return the value.



.guides/img/ShortCircuiting

```
cout << boolalpha << (false
                              &&
/*C++ never reaches this line*/ true) << endl;

cout << boolalpha << (true
                             ||
/*C++ never reaches this line*/ false) << endl;
```