# The Logical Operators

**In addition to the relational operators, C++ defines three logical operators** that take **Boolean operands** and **combine them** to form other Boolean values:

| Logical Operators | |
|---|---|
| `!` or ***`not`*** | Unary ***NOT*** (*true* if its operand is *false*) |
| `&&` or ***`and`*** | Binary ***AND*** (*true* if **both** operands are *true*) |
| `\|\|` or ***`or`*** | Binary ***OR*** (*true* if either or both operands are *true*) |

In C++ you can use either they operators `&&`, `||`, and `!` as you would in Java, **or** the English words ***`and`***, ***`or`***, and ***`not`***, as you would in Python.

Use the logical operators to **combine multiple conditions** like this:

```
if (percent >= 6.25 && percent < 78) { grade = "C"; }
```

Here, **both conditions** must be `true` for `grade` to be set to `"C"`. Here's another example:

```
if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u')
{
    result = "vowel";
}
```

Here, `result` is set to `"vowel"` if **any one** of the conditions is true.

*Remember, `&&` means all, and `||` means any!*

## Short-circuit Expressions

When C++ evaluates an expression with the logical operators:

- the sub-expressions are **always evaluated from left to right**.
- **evaluation ends** as soon as the result can be determined.

For example, if *expr1* is `false` in the expression `expr1 && expr2`, there is no need to evaluate `expr2` since the result will **always** be `false`.

Similarly, with `expr1 || expr2`, there is no need to evaluate `expr2` **when** `expr1` is `true`.

In both of these cases, evaluation which stops as soon as the result is known. This is called **short-circuit evaluation**.