

Writing circuits

circom allows programmers to define the [constraints](#) that define the arithmetic circuit. All constraints must be of the form $A * B + C = 0$, where A, B and C are linear combinations of signals. More details about these equations can be found [here](#).

The arithmetic circuits built using circom operate on signals. Let us define our first circuit that simply multiplies two input signals and produces an output signal.

```
pragma circom 2.0.0;
```

```
/This circuit template checks that c is the multiplication of a and b.
```

```
template Multiplier2 () {
```

```
  // Declaration of signals.
```

```
  signal input a;
```

```
  signal input b;
```

```
  signal output c;
```

```
  // Constraints.
```

```
  c <== a * b;
```

```
} First, the pragma instruction is used to specify the compiler version. This is to ensure that the circuit is compatible with the compiler version indicated after the pragma instruction. Otherwise, the compiler will throw a warning.
```

Then, we use the reserved keyword `template` to define the shape of a new circuit, called `Multiplier2`. Now, we have to define its [signals](#). Signals can be named with an identifier, e.g., a, b, c. In this circuit, we have two input signals a, b and an output signal c. Finally, we use `<==` to set that the value of c is the result of multiplying the values of a and b. Equivalently, we could have also used the operator `==>`, e.g., `a * b ==> c`.

Let us notice that in each template, we first declare its signals, and after that, the associated constraints.