



Arduino Math and Logic Operations

This page explains the different types of math and logic possible on an Arduino.

Contents [\[hide\]](#)

- 1 [Arithmetic](#)
- 2 [Comparison operators](#)
- 3 [Boolean operators](#)
- 4 [Arduino variable types](#)
- 5 [Math functions](#)
- 6 [Trigonometry](#)

navigation

- [Main page](#)
- [RBE 1001](#)
- [RBE 2001](#)
- [RBE 2002](#)
- [RBE 3001](#)
- [RBE 3002](#)
- [Help](#)

search

tools

- [What links here](#)
- [Related changes](#)
- [Special pages](#)
- [Printable version](#)
- [Permanent link](#)
- [Page information](#)

Arithmetic

These are the basic math commands for arduinos and most other hardware and software.

- [assignment](#) [↗](#) operator is indicated by a `=`.
- [addition](#) [↗](#) operator is indicated by a `+`.
- [subtraction](#) [↗](#) operator is indicated by a `-`.
- [multiplication](#) [↗](#) operator is indicated by a `*`.
- [division](#) [↗](#) operator is indicated by a `/`.
- [modulo or remainder](#) [↗](#) operator, is indicated by a `%`.

Examples

```
int result = 7 + 5; // result assigned to value 12
int result = 7 * 5; // result assigned to value 35
int result = 7 % 5; // result assigned to value 2
```

Comparison operators

These are the basic logic or comparison operators for arduinos and most other hardware and software. If the relationship that they check for is true, they return a 1. Otherwise they will return a 0.

- `==` checks for an "equal to" [↗](#) relationship.
- `!=` checks for a "not equal to" [↗](#) relationship.
- `<` checks for a "less than" [↗](#) relationship.
- `>` checks for a "greater than" [↗](#) relationship.
- `<=` checks for a "less than or equal to" [↗](#) relationship.
- `>=` checks for a "greater than or equal to" [↗](#) relationship.

Examples


```
boolean result = 5<10; // result will be equal to true
boolean result = 10<5; // result will be equal to false
```

Boolean operators


These are the basic boolean operators, often also referred to as logic gates.

- **&&** stands for an **and**  gate. Using this operator, the resulting truth table is as follows:

Input A	Input B	Output
false	false	false
false	true	false
true	false	false
true	true	true










- **||** stands for an **"or"**  gate. Using this operator, the response table is as follows:



Input A	Input B	Output
false	false	false
false	true	true
true	false	true
true	true	true

- **!** stands for a **not**  gate. Using this operator, the response table is as follows:


Input	Output
0	1
1	0


Arduino variable types


Type	Explanation	Memory Size on Arduino	Range signed	Range unsigned
char 	Character. Smallest unit that can define a character	8 bit byte	-128 to 127	0 to 255
byte 	Stores an 8 bit value	8 bit byte		0-255
boolean 	Stores a true or false value	1 bit	0 or 1	
int 	Primary data type	2-8 bit bytes	-32,768 to 32,767.	0 to 65,535
short 	Same as int	2-8 bit bytes	-32,768 to 32,767	
long 	Extended variable for number storage	4- 8 bit bytes	-2,147,483,648 to 2,147,483,647	0 to 4,294,967,295
float 	Unit for handling decimals. Rounds to 6 decimals.	4- 8 bit bytes	3.4028235E+38 to -3.3028235E+38	
double 	That is, the double implementation is exactly the same as the float	4- 8 bit bytes	3.4028235E+38 to -3.3028235E+38	
char[]  (C-string)	An array of characters	Varies	Limited by memory size	


String  (object)	String class allows you to do more complex task at the cost of memory	Varies	Limited by memory size	
array 	A collection of variables that can be saved and edited by accessing its index number	Varies	Limited by memory size	

Math functions


[min\(x,y\)](#)  // returns value of smaller number


[max\(x,y\)](#)  // returns value of larger number


[abs\(x\)](#)  // absolute value of the value entered


[constrain\(x , low , hi\)](#)  // constrains first parameter by the following two parameters.

[map\(value, from low, from high, to low, to high\)](#)  // linearly maps a value from one range to another


[pow\(base,exponent\)](#)  // raises base to exponent


[sqrt\(\)](#)  // square root of value entered


[sq\(\)](#)  // squares the value entered

[random\(min,max\)](#)  // returns a random integer in the range [min,max)

Trigonometry

[sin\(x\)](#)  // returns the sin of an angle entered in radians

[cos\(x\)](#)  // returns the cos of an angle entered in radians

[tan\(x\)](#)  // returns the tan of an angle entered in radians