C Operators

‹ Previous Next ›

Operators

Operators are used to perform operations on variables and values.

In the example below, we use the `+` **operator** to add together two values:

Example

```
int myNum = 100 + 50;
```

Try it Yourself »

Although the `+` operator is often used to add together two values, like in the example above, it can also be used to add together a variable and a value, or a variable and another variable:

Example

```
int sum1 = 100 + 50;         // 150 (100 + 50)
int sum2 = sum1 + 250;       // 400 (150 + 250)
int sum3 = sum2 + sum2;      // 800 (400 + 400)
```

Try it Yourself »

C divides the operators into the following groups:

- Comparison operators
- Logical operators
- Bitwise operators

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example	Try it
+	Addition	Adds together two values	x + y	Try it »
-	Subtraction	Subtracts one value from another	x - y	Try it »
*	Multiplication	Multiplies two values	x * y	Try it »
/	Division	Divides one value by another	x / y	Try it »
%	Modulus	Returns the division remainder	x % y	Try it »
++	Increment	Increases the value of a variable by 1	++x	Try it »
--	Decrement	Decreases the value of a variable by 1	--x	Try it »

Assignment Operators

Assignment operators are used to assign values to variables.

In the example below, we use the **assignment** operator (=) to assign the value **10** to a variable called **x**:

Example

```
int x = 10;
```

The **addition assignment** operator (`+=`) adds a value to a variable:

Example

```
int x = 10;
x += 5;
```

Try it Yourself »

A list of all assignment operators:

Operator	Example	Same As	Try it
=	x = 5	x = 5	Try it »
+=	x += 3	x = x + 3	Try it »
-=	x -= 3	x = x - 3	Try it »
*=	x *= 3	x = x * 3	Try it »
/=	x /= 3	x = x / 3	Try it »
%=	x %= 3	x = x % 3	Try it »
&=	x &= 3	x = x & 3	Try it »
|=	x |= 3	x = x | 3	Try it »
^=	x ^= 3	x = x ^ 3	Try it »
>>=	x >>= 3	x = x >> 3	Try it »
<<=	x <<= 3	x = x << 3	Try it »

Comparison Operators

The return value of a comparison is either `1` or `0`, which means **true** (`1`) or **false** (`0`). These values are known as **Boolean values**, and you will learn more about them in the Booleans and If..Else chapter.

In the following example, we use the **greater than** operator (`>`) to find out if 5 is greater than 3:

Example

```
int x = 5;
int y = 3;
printf("%d", x > y); // returns 1 (true) because 5 is greater than 3
```

Try it Yourself »

A list of all comparison operators:

Operator	Name	Example	Try it
==	Equal to	x == y	Try it »
!=	Not equal	x != y	Try it »
>	Greater than	x > y	Try it »
<	Less than	x < y	Try it »
>=	Greater than or equal to	x >= y	Try it »
<=	Less than or equal to	x <= y	Try it »

Logical Operators

You can also test for true or false values with logical operators.

Logical operators are used to determine the logic between variables or values:

&&	Logical and	Returns true if both statements are true	x < 5 && x < 10	Try it »
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4	Try it »
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)	Try it »

Sizeof Operator

The memory size (in bytes) of a data type or a variable can be found with the `sizeof` operator:

Example

```
int myInt;
float myFloat;
double myDouble;
char myChar;

printf("%lu\n", sizeof(myInt));
printf("%lu\n", sizeof(myFloat));
printf("%lu\n", sizeof(myDouble));
printf("%lu\n", sizeof(myChar));
```

Try it Yourself »

Note that we use the `%lu` format specifer to print the result, instead of `%d`. It is because the compiler expects the sizeof operator to return a `long unsigned int` (`%lu`), instead of `int` (`%d`). On some computers it might work with `%d`, but it is safer to use `%lu`.

C Exercises

Dark mode