Arithmetic Mostly Does What You Expect

Declare: `int A = 120;  int B = 42;`

Then…

`A + B`	**evaluates to**	**`162`**
`A - B`	evaluates to	**`78`**
`A * B`	evaluates to	**`5040`**
`A % B`	evaluates to	**`36`**
`A / B`	evaluates to…	**`2`**

What's going on with division?

A Few Pitfalls of C Arithmetic

No checks for overflow, so be careful.
- `unsigned int A = 0 - 1;`
- `A` is a large number!

Integer division
- Trying to **divide by 0** ends the program (floating-point produces **infinity** or **NaN**).
- Integer division **evaluates to an integer**, so **`(100 / 8) * 8` is not `100`**.

C Behavior Sometimes Depends on the Processor

Integer division is rounded to an integer.

Rounding **depends on the processor**.

Most modern processors **round towards 0**, so…

`11 / 3` evaluates to **`3`**

`-11 / 3` evaluates to **`-3`**

Modulus `A % B` is defined such that

`(A / B) * B + (A % B)` is equal to `A`

So `(-11 % 3)` evaluates to **`-2`**.

Modulus is not always positive.

Six Bitwise Operators on Integer Types

Bitwise operators in C include
- AND: `&`
- OR: `|`
- NOT: `~`
- XOR: `^`
- left shift: `<<`
- right shift: `>>`

In some languages, `^` means exponentation, but not in the **C** language.

