Arithmetic in PBASIC

- The usual operators

  `+, -, *, /`

- But
  - order of evaluation is left-to-right
    - `5 + 3 * 2 = 11`
  - results are limited to 16 bits
    - from `-32768` to `32767`
  - don't divide negative numbers
    - compute `-200/15` as `-(200/15)`

Signed vs. unsigned

- C++ integers can be signed and unsigned
- PBASIC variables are n-bits
  - If you use a word as unsigned
    - the range is `0` to `65535`
  - If you use a word as signed
    - the range is `-32768` to `32767`
  - **But**, think before you use

Avoiding overflow

- Don’t store a large value in a byte
  - Bytes can hold `-128` to `127`
    - or `0` to `255` if you are careful
- Don’t add two large numbers
  - if signed
    - two positives may sum to a negative
  - if unsigned
    - two large numbers may sum to a small one
  - **think and plan**

Variations on multiplication and division

`//` remainder

```
1017//100 = 17
```

`**` high 16 bits of multiplication

```
10000 ** 10000 = 1525 = $5F5
```

`*/` middle 16 bits of division

```
10000 */ 10000 = 62945 = $F5E1
```

By the way

```
10000 * 10000 = $05F5E100
```

Rounding with integers

- Division truncates

```
11760/100 = 117
```

- To round, add in half the divisor

```
(11760+50)/100 = 118
```

- Or in PBASIC

```
X = (N+(D/2))/D
```

Multiplying by “reals”

- Suppose you want to compute

```
Y = 17.37 * X
```

- You could do

```
Y = (17*X) + ((37*X+50)/100)
```

- Or you could do

```
Y = X */ $115F
```

- How to get `$115F`
  - In Windows calculator, compute

```
17.37 * 256 + 0.5
```