Floating-Point Numbers in C Sharp

C# supports two floating-point types: `float` and `double`. These types follow the IEC 60559 standard, also known as the IEEE 754 standard for floating-point arithmetic.

IEC 60559 Standard

The IEC 60559 standard defines the representation for two floating-point types: single precision (`float` in C#) and double precision (`double` in C#). This standard provides a set of values including positive zero, negative zero, positive infinity, negative infinity, and Not-a-Number (NaN).

Mathematical Representation according to IEEE 754 and bits

The IEEE 754 standard represents floating-point numbers using a sign bit, an exponent, and a mantissa. The sign bit indicates the sign of the number (positive or negative), the exponent represents the scale of the number, and the mantissa contains the significant digits of the number.

For `float`, the IEEE 754 standard uses 32 bits to represent the number:

Formula: `±(1 + M) * 2^(E - 127)`

Where `M` is the mantissa (23 bits), `E` is the exponent (8 bits), and (1 bit) is the sign.

For `double`, the IEEE 754 standard uses 64 bits to represent the number:

Formula: `±(1 + M) * 2^(E - 1023)` Where `M` is the mantissa (52 bits), `E` is the exponent (11 bits), and (1 bit) is the sign.

Demo

Let's take the `float` value 0.15625 as an example:

1. Convert the number to binary. The integer part is 0 (which is 0 in binary), and the fractional part 0.15625 can be converted to binary by multiplying each fractional part by 2 until you get 0:
    - `0.15625 * 2 = 0.3125 (0)`
    - `0.3125 * 2 = 0.625 (0)`
    - `0.625 * 2 = 1.25 (1)`
    - `0.25 * 2 = 0.5 (0)`
    - `0.5 * 2 = 1.0 (1)`
    - `0.0 * 2 = 0 (0)`

    So, 0.15625 in binary is 0.00101.
2. Normalize the binary number. Move the binary point so that there's only one non-zero digit to its left. This gives us `1.01 * 2^-3`.
3. The sign bit is 0 (since the number is positive), the exponent is -3 (which needs to be represented as an unsigned 8-bit number by adding 127, giving us 124 or 01111100 in binary), and the mantissa is 01

(the part after the binary point in the normalized number, padded with zeros on the right to make 23 bits).

So, the binary representation of the `float` 0.15625 is:

- Sign: 0
- Exponent: 01111100
- Mantissa: 01000000000000000000000

Put together: 00111110001000000000000000000000

The process for `double` is similar, but the exponent is represented as an unsigned 11-bit number by adding 1023, and the mantissa is 52 bits.