Floating Point Numbers

What are floating points?

Since computer memory is limited, you cannot store numbers with infinite precision, no matter whether you use binary fractions or decimal ones: at some point you have to cut off. But how much accuracy is needed? And where is it needed? How many integer digits and how many fraction digits?

- To an engineer building a highway, it does not matter whether it's 10 meters or 10.0001 meters wide - his measurements are probably not that accurate in the first place.
- To someone designing a microchip, 0.0001 meters (a tenth of a millimeter) is a huge difference - But he'll never have to deal with a distance larger than 0.1 meters.
- A physicist needs to use the speed of light (about 300000000) and Newton's gravitational constant (about 0.0000000000667) together in the same calculation.

To satisfy the engineer and the chip designer, a number format has to provide accuracy for numbers at very different magnitudes. However, only relative accuracy is needed. To satisfy the physicist, it must be possible to do calculations that involve numbers with different magnitudes.

Basically, having a fixed number of integer and fractional digits is not useful - and the solution is a format with a **floating point**.

How floating-point numbers work

The idea is to compose a number of two main parts:

- A **significand** that contains the number's digits. Negative significands represent negative numbers.
- An **exponent** that says where the decimal (or binary) point is placed relative to the beginning of the significand. Negative exponents represent numbers that are very small (i.e. close to zero).

Such a format satisfies all the requirements:

- It can represent numbers at wildly different magnitudes (limited by the length of the exponent)
- It provides the same relative accuracy at all magnitudes (limited by the length of the significand)
- It allows calculations across magnitudes: multiplying a very large and a very small number preserves the accuracy of both in the result.

Decimal floating-point numbers usually take the form of scientific notation with an explicit point always between the 1st and 2nd digits. The exponent is either written explicitly including the base, or an e is used to separate it from the significand. The table below shows a few examples.

Significand	Exponent	Scientific notation	Fixed-point value
1.5	4	$1.5 * 10^4$	15000
-2.001	2	$-2.001 * 10^2$	-200.1
5	-3	$5 * 10^{-3}$	0.005
6.667	-11	6.667e-11	0.00000000006667

The standard

Nearly all hardware and programming languages use floating-point numbers in the same binary formats. The usual formats are 32 or 64 bits in total length:

Format	Total bits	Significand bits	Exponent bits	Smallest number	Largest number
Single precision	32	23 + 1 sign	8	$1.2 * 10^{-38}$	$3.4 * 10^{38}$
Double precision	64	52 + 1 sign	11	$5.0 * 10^{-324}$	$1.8 * 10^{308}$

Note: It is not required to understand how floating points are implemented. Focus on how a floating point can result in round-off and other errors due to the relative accuracy that it results.

Source: http://floating-point-gui.de/formats/fp/