Data Types in C

So, somewhat unlike Python, you must tell the compiler the type of every single variable that you use, without exception! This makes C fast, but also somewhat annoying. It also brings up several topics that we need to discuss and understand.

1. Basic / Primitive Data Types.

   a) char - single character
   b) int - integer
   c) float - single precision floating point
   d) double - double precision floating point

There are also modifiers:

- short and long

Single-precision examples [edit]

These examples are given in bit *representation*, in hexadecimal and binary, of the floating-point value. This includes the sign, (biased) exponent, and significand.

$0\ 00000000\ 00000000000000000000001_2 = 0000\ 0001_{16} = 2^{-126} \times 2^{-23} = 2^{-149} \approx 1.4012984643 \times 10^{-45}$
(smallest positive subnormal number)

$0\ 00000000\ 11111111111111111111111_2 = 007f\ ffff_{16} = 2^{-126} \times (1 - 2^{-23}) \approx 1.1754942107 \times 10^{-38}$
(largest subnormal number)

$0\ 00000001\ 00000000000000000000000_2 = 0080\ 0000_{16} = 2^{-126} \approx 1.1754943508 \times 10^{-38}$
(smallest positive normal number)

$0\ 11111110\ 11111111111111111111111_2 = 7f7f\ ffff_{16} = 2^{127} \times (2 - 2^{-23}) \approx 3.4028234664 \times 10^{38}$
(largest normal number)

$0\ 01111110\ 11111111111111111111111_2 = 3f7f\ ffff_{16} = 1 - 2^{-24} \approx 0.999999940395355225$
(largest number less than one)

$0\ 01111111\ 00000000000000000000000_2 = 3f80\ 0000_{16} = 1$ (one)

$0\ 01111111\ 00000000000000000000001_2 = 3f80\ 0001_{16} = 1 + 2^{-23} \approx 1.00000011920928955$
(smallest number larger than one)

$1\ 10000000\ 00000000000000000000000_2 = c000\ 0000_{16} = -2$
$0\ 00000000\ 00000000000000000000000_2 = 0000\ 0000_{16} = 0$
$1\ 00000000\ 00000000000000000000000_2 = 8000\ 0000_{16} = -0$

$0\ 11111111\ 00000000000000000000000_2 = 7f80\ 0000_{16} =$ infinity
$1\ 11111111\ 00000000000000000000000_2 = ff80\ 0000_{16} =$ -infinity

$0\ 10000000\ 10010010000111111011011_2 = 4049\ 0fdb_{16} \approx 3.14159274101257324 \approx \pi$ (pi)
$0\ 01111101\ 01010101010101010101011_2 = 3eaa\ aaab_{16} \approx 0.333333343267440796 \approx 1/3$

$x\ 11111111\ 10000000000000000000001_2 = ffc0\ 0001_{16} =$ qNaN (on x86 and ARM processors)
$x\ 11111111\ 00000000000000000000001_2 = ff80\ 0001_{16} =$ sNaN (on x86 and ARM processors)