Basic variable types and binary representation

- Basic element : a bit   1/0 , on/off , true/false , ...
- All variable types based on using bits and the binary (base 2) number system to encode information

- Logicals / booleans
  - Single bit
  - 1/0 $\Longleftrightarrow$ true/false

- Integers

Binary	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
10000	16

  - Series of logicals (bits) corresponding to different powers of 2
    $\hookrightarrow$ so just natural numbers in base 2
  - Can be unsigned or signed (need one sign bit)
  - Ex : 137 in base 10 and base 2 : $(137)_{10} = (100010001)_2$

$$(137)_{10} = \begin{array}{ccc} 10^2 & 10^1 & 10^0 \\ \hline 1 & 3 & 7 \end{array} = (1\times10^2)+(3\times10^1)+(7\times10^0)$$
$$= 100 + 30 + 7$$

$$(100010001)_2 = \begin{array}{cccccccc} 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} = (1\times2^7)+(0\times2^6)+(0\times2^5)+(0\times2^4)+(1\times2^3)+(0\times2^2)+(0\times2^1)+(1\times2^0)$$
$$= 128 + 0 + 0 + 0 + 8 + 0 + 0 + 1$$

  - short integer (15+1 bits) or long integer (31+1 bits)

- Strings / characters
  - "cat", "Mercury", "compilation error"
  - Each character ⟷ a number ⟷ a set of bits
  - Standard ASCII : each character encoded using 7 bits
    → $2^7 = 128$ possible characters
    (but typically stored in 8 bits = 1 byte)

    Ex:

Char.	Decimal	Binary
"x"	120	01111000

- Pointers
  - A memory address saved as a variable
  - Used to refer to (point to) other variables, functions, objects, ...

- References
  - Similar to a pointer, but works (and looks) more like an alias for the original variable

- Arrays, structures, classes, ...
  - Bundles of other variables
  - Array : many variables of same type ordered sequentially in memory
  - Classes typically have defined additional operations ("methods", "member functions")

- "Floating point" numbers

  - How to represent real numbers ($\mathbb{R}$) in binary?

  - Effectively need three pieces of info

    1) The sign
    2) The digits appearing in the number
    3) The location of the point

    Ex: -235713.66401

  - Strategy: use scientific notation in base 2

  - Scientific notation in decimal (base 10) system:

    $$-9.90625 \times 10^{0}$$

  - Normalized scientific notation:

    $$-0.990625 \times 10^{1}$$

    In general: $\pm\left[\text{number between } \frac{1}{10} \text{ and } 1\right] \times 10^{[\text{integer exp.}]}$

  - Normalized scientific notation in base 2:

    $$\pm\left[\text{number between } \frac{1}{2} \text{ and } 1\right] \times 2^{[\text{integer exp.}]}$$

    $$[\text{sign}] \times [\text{mantissa}] \times 2^{[\text{exponent}]}$$

  - Representation of mantissa:

    $$(0.1001\ldots)_2 = \underset{\uparrow \text{ implicit}}{(0 \times 2^{0})} + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4}) + \ldots$$

- Single precision number, 32 bits

  Sign : 1 bit

  Exponent : 8 bits

  Mantissa : 23 bits

- Ex : $-3.25 = (-1)^1 \times (0.8125) \times 2^2$

  32-bit representation :

sign bit (1)	8-bit exponent (2)	23-bit mantissa (0.8125)
1	0 0 0 0 0 0 1 0	0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

- Double precision, 64 bits

  Sign : 1 bit

  Exponent : 11 bits

  Mantissa : 52 bits

- Extended precision, 128 bits (1 + 15 + 112)

- Different standards for floating point representation

  e.g.

  $$(-1)^s \times [\text{number between } 1 \text{ and } 2] \times 2^{[\text{exp.} - \text{bias}]}$$

  $\longrightarrow$ different string of 0's and 1's ...

Source of problems

- Limited number of bits for exponent $\Rightarrow$ limited range of $\mathbb{R}$ can be represented

- Limited number of bits for mantissa $\Rightarrow$ limited precision ("resolution") in representation of the continuous $\mathbb{R}$

- We'll get back to this...