Number Representation
3/2/01 Lecture #11 16.070

- How are numbers represented in a computer?
- Data come in two basic types
  - ➢ Numbers
    - – Whole/Integer (1, -3, 0)
    - – Natural, Positive (1, 2, 3)
    - – Real/Floating-point (32.6, 3.14)
  - ➢ Letters
    - – Character (a, *, /): typographic symbols
    - – Boolean **(`TRUE, FALSE`)**

Data Representation in C

- Numbers
  - ➢ Whole/Integer defined in C with `int, long, short, unsigned`
  - ➢ Natural, Positive not specified in C. Can be specified in other languages such as Ada
  - ➢ Real/Floating-point defined in C with `float, double`
- Letters
  - ➢ Characters defined in C with `char`
  - ➢ Boolean not specified in C. Can be specified in other languages such as Ada
- Refer to C5.9, p. 175 for table of all C data types, type specifiers, number of bits used for internal representation

Numeric Data Types - Integers

- Integer type used for things that humans perceive as whole numbers or discrete items; e.g., 1, 16, 42, 365

    binary num e.g., 0100100100001110

    - ➢ Precision is one digit
    - ➢ Range can be set by number of bits used. For n bits
        - – Unsigned: $0 \rightarrow 2^n - 1$
        - – Signed: $-2^{(n-1)} - 1 \rightarrow 2^{(n-1)}$
    - ➢ In C, representation is defined in declaration statement
        - – `int` (signed): Uses one machine-word -- 8 bits, 16 bits, 32 bits, etc.
        - – `short` (signed): Typically uses 16 bits
        - – `long` (signed): Typically uses 32 bits
        - – Use `INT_MIN, INT_MAX`, `SHRT_MAX, LONG_MAX` to determine range (a library is needed to use these -- refer to C5.9, p. 177)

Machine Representation of Integers

- Integer represented as one machine word.
  - If one machine word is 8 bits, integer represented with 8 bits

    10000010

  - Actual value will depend on representation
    - Unsigned: 130_{10}
    - Sign Magnitude: -2_{10}
    - One's Compelement: -2_{10}
    - Two's Complement: -126_{10}
  - Use *sizeof* to determine how many bytes are used for representation

Numeric Data Types - Floating Point

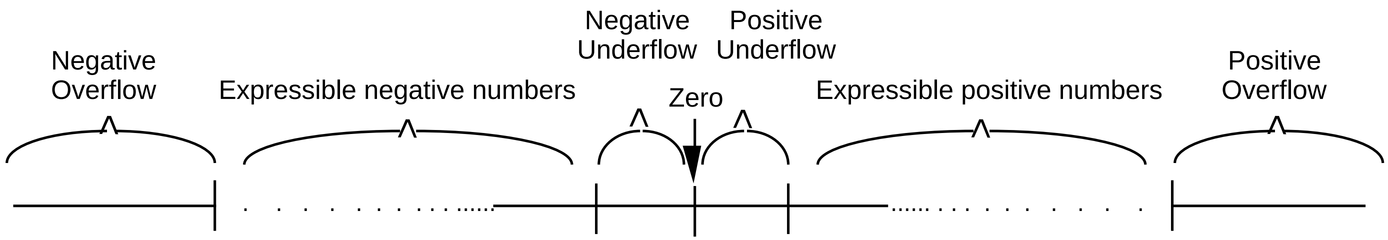
- Floating point used for things which humans perceive as continuous variables -- speed, temperature, etc.
- Floating point used to represent numbers that have fractional part
- Floating point numbers expressed as a mantissa (signed fraction) and an exponent (signed integer):

  `mantissa | exp` → `mantissa x 2` ^{exp}

  - ➢ Number of digits in mantissa specify precision
  - ➢ Number of digits in exponent specify range
- In C, representation is defined in declaration statement
  - `float`: Typically uses 32 bits
  - `double`: Typically uses 64 bits
  - `long double`: Typically uses 64 bits
  - Use `FLT_MIN, FLT_MAX`, and `FLT_DIG` to determine range & precision (a library is needed to use these -- refer to C5.9, p. 177)

Floating Point - Machine Representation

- In actuality, digital computers represent floating points as discrete values, NOT continuous values
- Due to finite nature of computers, Floating point numbers cannot always be represented exactly.



- This can create a number of errors in representing Floating Points
  - Overflow - number too large to be represented
  - Underflow - number too small to be represented
  - Rounding - due to insufficient precision
  - Relative Error - Spacing not constant between representable numbers

Rounding

- If result of a calculation cannot be expressed, use nearest number that can be expressed
- If result is halfway between two numbers that can be expressed, round away from 0

Relative Error

- Spacing not constant between representable numbers
- When spacing is represented as a percentage, the relative error that is introduced by rounding is approximately the same.

Encountering Floating Point Representation Inaccuracies

- How might you encounter problems with floating point numbers?
  - Overflow – multiply two large numbers
  - Underflow – multiply two small numbers
  - Representation Errors -- E.g., 100/3
  - “Swamping” Errors – When manipulating large and small numbers, large number may “overpower” small number. Eg., 100000.0 + 0.000001 = 100000.0 → Carefully select order of computation
- When coding, recommend not checking that a floating point number is equal to a value; instead, check that it is within a certain range

```
if (f == 0.0)    /* may not work */
     vs
if ((f > -0.001) && (f < 0.001))  /* safer */
```

Floating Point Representation

- Changing number of digits in fraction or exponent shifts boundaries of regions 2 and 6 and changes number of expressible points in them
- Increasing number of digits in fraction increases density of points → improves accuracy of approximations
- Increase number of digits in exponent increases size of regions 2 and 6 by shrinking regions 1, 3, 5, 7

Using Integers vs Floats

- If concerned about speed and/or accuracy use integers

Integer	Float = mantissa x 2^{exp}
Faster	
Less Storage Space	
Precise	May be some loss of accuracy
	Wider Range of Values
	Do not form a continuum

Real-world Signals - Number Representation

- Real-world signals are usually analog outputs of sensors (e.g., strain gauge, potentiometer)
- Analog to digital conversion (A/D) converts the analog signal to a prescribed digital format with some precision and range, at some sampling rate
- In spacecraft, measurements performed on-board and telemetered to ground using integer representation
  - ➢ Measurements telemetered as a series of binary digits

Data Conversion for Human Understanding

- Translate telemetered data to make it readable to humans by converting to appropriate float/integer value, a.k.a. engineering units
- Conversion based on calibration on test articles (for volume production) or prior service (for one-off)
- Simplest conversion involves application of linear multiplicative scale factor
  - ➢ For telemetered value x, $y = ax + b$, where a and b are constants based on measurements performed prior to launch
  - ➢ For example, solar panel position data may be telemetered as an 8-bit "word" with possible values of 0 -> 255
  - ➢ Temperature data often requires second-order conversions

Character Data Type

- A character literal is a single printable character in single quotes
- The computer maps characters on to integers
- Each has its own unique numeric code:

'A'	=> 65	(41H)	[$0100\ 0001_2$]
'D'	=> 68	(44H)	[$0100\ 0100_2$]

- The binary form is stored in a memory cell that is of type "character"

Character Data Type (cont.)

- Characters can be used in expressions similar to integers
  - C stores integer values in one byte for character data

‘A’ => 65 (41H)

‘D’ => 68 (44H)

('D' - 'A') will be evaluated to a character whose value is 3H

Non-printable Control Characters

- Control Characters control an output device to perform a special operation
  - Linefeed
  - Bell
  - Carriage return

Review

- Material covered today - refer to C5.9-5.13
  - ➢ How numbers are represented in the computer
    - – Integers - fast, less storage space, precise
    - – Floats - Not always precise, wider range of values
  - ➢ Know the capabilities as well as the limitations of the computer
- Enhanced Style Guide will be posted on the Web this weekend
- New Homework Policy
  - ➢ Hand in by 2:05 Wednesdays, else dock 1/3 for every 24 hours
  - ➢ Arrange late turn in >24 hours in advance for possible full credit
- Exam #1 on Monday - Study Guide on the Web (Announcements)
- Handyboards to be handed out in Lab Session on Monday/Tuesday, 3/5 and 3/6