Are you sure?

Let’s check…
(see sizeck.c)

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>1 byte</td>
<td>-128 to 127</td>
</tr>
<tr>
<td>unsigned char</td>
<td>1 byte</td>
<td>0 to 255</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes</td>
<td>-32768 to 32767</td>
</tr>
<tr>
<td>unsigned short</td>
<td>2 bytes</td>
<td>0 to 65535</td>
</tr>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2147483648 to 2147483647</td>
</tr>
<tr>
<td>unsigned int</td>
<td>4 bytes</td>
<td>0 to 4294967295</td>
</tr>
<tr>
<td>long</td>
<td>4 bytes</td>
<td>-2147483648 to 2147483647</td>
</tr>
<tr>
<td>unsigned long</td>
<td>4 bytes</td>
<td>0 to 4294967295</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes</td>
<td>1.175494e-38 to 3.402823e+38</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes</td>
<td>2.225074e-308 to 1.797693e+38</td>
</tr>
</tbody>
</table>

ANSI rules

- Variables of type char are guaranteed to always be one byte.
- There is no maximum size for a type, but the following relationships must hold:
  - sizeof(short) <= sizeof(int) <= sizeof(long)
  - sizeof(float) <= sizeof(double) <= sizeof(long double)
What does this mean?
- abc
- 123
- 3.14
- 0x61

Representation (encode?)
- ASCII
- Simple Binary*
- One’s complement
- Two’s complement*
- Binary Coded Decimal
- Floating-point*

Limited number of bits to encode a value

Will there be a time that the value we want to encode does not fit? Yes! OVERFLOW

Need to be aware of the range of values that each limited number of bits will hold

Inaccuracies exist...

(see overflw.c)
Floating point

- Google
  - “what every computer scientist should know about floating point”
  - Squeezing infinitely many real numbers into a finite number of bits requires an approximate representation (rounding)

- Overflow $\rightarrow +\infty$

- Not associative
  - Due to finite precision of the representation
    - A float has roughly seven decimal digits of precision
    - see floatpt.c
The machine level program generated has no information about data types.

- It’s the C compiler that maintains this type of information.

The different mathematical properties of integers vs floating-point arithmetic stem from the difference in how they handle the finite-ness of their representations.

- Integers – smaller values but more precise.
- Real – wide range of values, but only approximately.

Defines ranges of values.

Computer security vulnerabilities.

Need to know/understand before progress to machine level programming (chp.3).
Information Storage (details)

- Byte = smallest addressable unit of memory
- Virtual memory = very large array of bytes
- Address = how byte of memory is uniquely identified
- Virtual address space = the set of all possible addresses

Reminder: no data typing at this level
Hexadecimal Notation (Hex)

- Base 16
- Useful in describing bit patterns
- Digits 0-9 and A-F
- In C
  - 0x or 0X prefix interpreted as hex value
  - Not case sensitive
  - Example
    - FA1D37B₁₆ --> 0Xfa1d37b, 0xFA1D37B, 0xFA1d37B
- Easy to convert to/from hex, octal and binary

<table>
<thead>
<tr>
<th>DEC</th>
<th>HEX</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2^0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2^1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2^2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2^2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>2^3</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
Binary to Hex to Octal

FYI: “bit” stands for “binary digit”

Fact: $2^4 = 16$ and $2^3 = 8$
- The power is the # of bits per hex/octet digit

Binary to Hex
- Every 4 bits = 1 hex digit

Octal – base 8
- Digits 0-7

Binary to Octal
- Every 3 bits = 1 octal digit

Example: FA1D37B_{16}

Practice problem 2.1 (pg. 35)
When value x is a power of 2

- x = 2^n for some non-negative integer n, then if
  - x = 16_{10} = 2^4 = 10000_2 = 10_{16}
- Binary rep of x → 1 followed by n zeroes
- Hex rep of x is → i + 4j
  - Reminder: hex digit 0 in binary is 0000
  - Leading hex digit where 0 <= i <=3
    - 1 (i=0)
    - 2 (i=1)
    - 4 (i=2)
    - 8 (i=3)
  - Followed by j hex 0s
- Examples:
  - n=9 → so i = 1 and j = 2 → x = (2)(00) hex i.e. x = 200_{16}
  - n=6 → so i = 2 and j = 1 → x = (4)(0) hex i.e. x = 40_{16}
  - See more practice problems 2.2 pg. 35
Convert to Decimal... why?

- Base ten (decimal): digits 0-9
  - E.g., $316_{10} = \text{ }$

- Base eight (octal): digits 0-7
  - E.g., $474_8 = \text{ }$

- Base 16 (hexadecimal): digits 0-9 and A-F.
  - $13C_{16} = \text{ }$

- Base 2 (binary): digits 0, 1
  - $100110_2 = \text{ }$

In general, radix $r$ representations use the first $r$ chars in \{0...9, A...Z\} and have the form $d_{n-1}d_{n-2}...d_1d_0$.

  Summing $d_{n-1} \times r^{n-1} + d_{n-2} \times r^{n-2} + ... + d_0 \times r^0$ converts to base 10.
Yes, really!

There are 10 rocks.
Oh, you must be using base 4. See, I use base 10.

No. I use base 10. What is base 4?

Every base is base 10.

Explanations:
In general, $10_x = X$
- $10_2 = 2$
- $10_3 = 3$
- $10_4 = 4$
- $10_5 = 5$
- $10_6 = 6$
- $10_7 = 7$
- $10_8 = 8$
- $10_9 = 9$
- $10_{10} = 10$

http://cowbirdsinlove.com/43