Basic and Complex Objects in C

David E. Culler
CS61CL
Feb 4, 2009
Lecture 2
Computers manipulate finite representations of things
Basic Data Types in C

- **char** – 1 byte
  - Sufficient to represent the local character set
  - Typically ASCII

- **int** – signed integer values
  - Range dictated by natural word width of the machine

- **float** – single precision floating point number
  - A lot like real numbers
  - Specific representation defined by IEEE

- **double** – double precision floating point number
  - Even more like real numbers...
What is this?

• 101
• 0101
Objects are represented as collections of symbols.

Their meaning is derived from what you do with them.
Another mapping function

### Regular ASCII Chart (character codes 0 - 127)

<table>
<thead>
<tr>
<th>Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>\r</td>
</tr>
<tr>
<td>0001</td>
<td>\n</td>
</tr>
<tr>
<td>0010</td>
<td>\t</td>
</tr>
<tr>
<td>0011</td>
<td>\b</td>
</tr>
<tr>
<td>0100</td>
<td>\f</td>
</tr>
<tr>
<td>0101</td>
<td>\d</td>
</tr>
<tr>
<td>0110</td>
<td>\n</td>
</tr>
<tr>
<td>0111</td>
<td>\n</td>
</tr>
</tbody>
</table>

### Extended ASCII Chart (character codes 128 - 255; Codepage 850)

<table>
<thead>
<tr>
<th>Code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>\x</td>
</tr>
<tr>
<td>129</td>
<td>\x</td>
</tr>
<tr>
<td>130</td>
<td>\x</td>
</tr>
<tr>
<td>131</td>
<td>\x</td>
</tr>
<tr>
<td>132</td>
<td>\x</td>
</tr>
<tr>
<td>133</td>
<td>\x</td>
</tr>
<tr>
<td>134</td>
<td>\x</td>
</tr>
<tr>
<td>135</td>
<td>\x</td>
</tr>
<tr>
<td>136</td>
<td>\x</td>
</tr>
<tr>
<td>137</td>
<td>\x</td>
</tr>
<tr>
<td>138</td>
<td>\x</td>
</tr>
<tr>
<td>139</td>
<td>\x</td>
</tr>
<tr>
<td>140</td>
<td>\x</td>
</tr>
<tr>
<td>141</td>
<td>\x</td>
</tr>
<tr>
<td>142</td>
<td>\x</td>
</tr>
<tr>
<td>143</td>
<td>\x</td>
</tr>
</tbody>
</table>

### Hexadecimal to Binary

<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>0001</td>
<td>0001</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
</tr>
<tr>
<td>0011</td>
<td>0011</td>
</tr>
<tr>
<td>0100</td>
<td>0100</td>
</tr>
<tr>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>0110</td>
<td>0110</td>
</tr>
<tr>
<td>0111</td>
<td>0111</td>
</tr>
</tbody>
</table>

### Groups of ASCII-Code in Binary

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Control Characters</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Digits and Punctuation</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Upper Case and Special</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Lower Case and Special</td>
</tr>
</tbody>
</table>
What’s this?

Cal™
And this?
And this?
Coding / Decoding

Representation (symbols, bits, …)

Meaning

Coding

Decoding

828250d3
Trivial Coding/Decoding

Diagram showing a binary to hexadecimal conversion process.
The Number of Numbers (Symbols)

• Finite Representations
• N digits in base 10 can represent $10^N$ values
  – 000, 001, ..., 999
• N digits in base 2 can represent $2^N$ values
  – 0000, 0001, ..., 1111
• N digits in base B can represent $B^N$ values
• N symbols in an alphabet of k characters can represent $k^N$ distinct “words”
Which ones?

- char: 1 byte, 0..255
- unsigned int
  - 32 bits => 0 .. $2^{32} - 1$ (about 4 billion, $2^{10} \sim 10^3$)
  - 64 bits => 0 .. $2^{64} - 1$ (about 16 quintillion)
- int
  - 32 bits => $-2^{31} \ldots 0 \ldots 2^{31} - 1$ (about -2 billion to +billion)
  - 64 bits => $-2^{63} \ldots 0 \ldots 2^{63} - 1$
  - long long int x;
Operations (on the representation)

- Computers provide direct hardware support for manipulating certain basic objects
- *Word* of bits

{&, |, ~, ^, <<, >>, ...}

Logical Operations (on bits)

{+, -, ...}

Arithmetic Operations

a + 0 = a
a – a = 0
a + b = b + a
a ≠ b ⇔ a – b ≠ 0 (???)
Manipulating bits

• In C we can work with what the object represents
  - $x = x + y$;
  - $Z = x \times 2$;
  - $Q = x \div 2$;
  - `putchar(c);`
  - `If (A && !B) { ... }`

• Or with its underlying representation
  - $Z = x << 1$;
  - $Q = x >> 1$;
  - $D = x \& 1$;
  - $E = x \& 0xFFFFFFFFE$;
  - $G = x \& \sim 1$
  - $M = (y \& 0x74) >> 3$
Where do Objects live and work?

Processor

Memory

register

load
operate
store

word

000..0:

n:

FFF..F:
Finite Representation

• When the limitations of the representation are exceeded,
• the illusion that the “thing” and “its representation” the same
• breaks down
  – x = x << 32;
  – x = x * x;
Name versus address

- The White House … vs…
- 1600 Pennsylvania Avenue NW Washington, DC 20500
- berkeley.edu .vs. 169.229.131.81
- What operations are defined on names?
- On addresses?
- In C we refer to objects by variable name
  - int x = y + 3;
Administration

• All future CS61CL lectures will be held on wed 3-4 in 10 Evans
  – They will be webcast
  – Great for review, but still better to be there

• HW2 will be submitted using the inst tools
  – See announcement, due before section Tu/W
  – See http://inst.eecs.berkeley.edu/cgi-bin/pub.cgi?file=submit.help
  – Please resubmit your hw1 file (ala bspace) as a test

• The waitlist has been accepted into the course
  – We plan to expand existing sections with LAs and laptops
  – TA’s will be finalizing the section allocation and ensuring that UCWISE section ID is correct
Bears in the news (…oops)

After the Transistor, a Leap Into the Microcosm

YORKTOWN HEIGHTS, N.Y. — Gaze into the electron microscope display in Frances Ross’s laboratory here and it is possible to persuade yourself that Dr. Ross, a 21st-century materials scientist, is actually a farmer in some Lilliputian silicon world.

By JOHN MARKOFF
Published: August 31, 2006

Science

Chenming Calvin Hu

FinFET
What about complex objects?

- Strings – sequences of characters
- Vectors – sequences of numbers
- Matrixes – 2D collections of numbers
- Records – finite sets of strings and numbers
- Lists
- Tables
- Sounds
- Images
- Graphs
- …
Arrays in C

- Ordered collection of objects of homogeneous type
  - “this string is also an array of chars”
  - {1, 2, 437, 61}

- Can be declared and named
  - char class[7] = “cs61cl”;
  - char class[] = “cs61cl”;
  - char classname[8];
  - int somenums[4] = {1, 2, 437, 61};
  - int morenums[] = {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};
  - int numbuf[32];

- Can be indexed to refer to an element
  - char x = class[0]; /* access or selection */
  - int y = somenums[2];
  - somenums[1] = morenums[3]; /* assignment */

- Elements have a static size determined by type
Where do complex objects reside?

- Arrays are stored in memory
- The variable (i.e., name) is associated with the location (i.e., address) of the collection
  - Just like variables of basic type
- Elements are stored consecutively
  - Can locate each of the elements
- Can operate on the indexed object just like an object of that type
  - \( A[2] = x + Y[i] - 3; \)
What can be done with a complex object?

- Access its elements
  - A[i]
- Pass it around
  - Sort(A)
  - x = max(A, n)
All objects have a size

- The size of their representation
- The size of static objects is given by sizeof operator

```c
#include <stdio.h>
int main() {
    char c = 'a';
    int x = 34;
    int y[4];
    printf("sizeof(c)=%d\n",   sizeof(c) );
    printf("sizeof(char)=%d\n",sizeof(char));
    printf("sizeof(x)=%d\n",   sizeof(x) );
    printf("sizeof(int)=%d\n", sizeof(int) );
    printf("sizeof(y)=%d\n",   sizeof(y) );
    printf("sizeof(7)=%d\n",   sizeof(7) );
}
```
Complex Objects - really

• In many cases we want to treat objects truly as an object
  – Hide its internal representation
    » Don’t know its size, …
  – Invoke methods that work on the object
  – Store it and retrieve it from other objects

• In C this is explicit in the type – pointer
  – char *text = “this is a string”;
  – printf(“print %s\n”, text);
An object and its value...

\[ X = X + 1; \]

The value of variable \( X \)

The storage that holds the value \( X \)
With complex objects the distinction matters

```c
char *s;
s = "abc";
```

s is a reference to the string “abc”
s is a pointer to the string “abc”
Array variables are also a reference for the object

```c
int main() {
    char *c = "abc";
    char ac[4] = "def";
    printf("c[1]=%c\n",c[1] );
    printf("ac[1]=%c\n",ac[1] );
}
```

- Array name is essentially the address of (pointer to) the zero\textsuperscript{th} object in the array
- There are a few subtle differences
  - Can change what \texttt{c} refers to, but not what \texttt{ac} refers to
Big Ideas

• Computers manipulate finite representations of things.
• A bunch of bits can represent anything, it is all a matter of what you do with it.
• Finite representations have limitations.

• An object, its value, its location, its reference

• Pointers are THE most subtle concept in C
  – Very powerful
  – Easy to misuse
  – Completely hidden in Java