COMS 1003 Fall 2005

Introduction to Computer Programming in C

Bits, Boolean Logic & Discrete Math

September 13th
Hello World!
Logistics

• See the website:
• Course Web-board (linked off website)
• Textbooks
• Readings
What This Class Isn't

- It is NOT a mathematics class
- I WON'T teach you how to type
- I WILL NOT waste your time
- A C++ class
- A Java class
Expectations

- Email address
- Participate in class
- Have fun
- This is a class in problem solving, not merely a C tutorial

- Open a dialog, simple transfer of information is boring
- I move fast. Slow me down.
First Assignment

• Recipe
• Put in dropbox 'Assignment_1' in Courseworks
• Read:
  - The Command Line
  - The Tao of Programming
Review

• There is a review problem set after each lecture
• About 10 problems, 1 or 2 challenging ones
• Answers provided
What Is Computer Science, Anyway?

- Study of information technology
- Study of computer machinery
  - A combination of math and physics
- A philosophy of problem solving and abstraction
- The study of algorithmic thinking
Computer Science Requires:

- Asking questions
- Thinking **abstractly**
- Making clarifications
- Being precise
- Testing requirements and results
Computing Devices

- PC's & laptops
- Embedded devices
- Mainframes
- Calculators
- Game consoles
What is a Computer?

• A device capable of performing computation
• Computers process data by following a set of instructions call a computer program.
• Computer programs are software.
• The devices are hardware.
• The CPU is the “brain” of the computer.
• More on computer organization later this week.
The Language of Computers

• Computers understand bits
• A bit is: 1/0, true/false, on/off, binary digit value
• Certain sequences of bits are meaningful, others are nonsense, so:
  - Need to organize bits (provide structure and context) as well as define bit operations
• A byte is a sequence of 8 bits
• Binary is a number system based on 2 digits
Binary Numbers

- Decimal Number system: base 10
- Binary Number system: base 2
- Hexadecimal Number system: base 16
- How to convert?
- Modular math (mod 2): remainder after div

- $345 = 3*10^2 + 4*10^1 + 5*10^0$
- $101011001 = 1*2^8 + 1*2^6 + 1*2^4 + 1*2^3 + 1*2^0 = ??$
- Place * base ^ pos
Counting in Binary

• 0,1,0,1,0,1 ... what's wrong here?
• Counting to 3: 00,01,10,11
• Note: Computer Science geeks always start counting from zero, whatever the number system
• Most geeks memorize (or at least fake it) the powers of 2: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, etc.
Math in Binary

• Review: closure of operations
  - For each pair of numbers $i$ and $j$, $i \, op \, j$ yields a like number

• Addition: (starting at rightmost bit)
  - $0 + 0 = 0$
  - $0 + 1 = 1$
  - $1 + 0 = 1$
  - $1 + 1 = 0$, carry 1
Negative Binary Numbers

- Four schemes:
  - Signed magnitude (leftmost bit is sign)
  - One's complement (leftmost sign bit, flip bit values)
  - Two's complement (form one's complement, add 1)
  - Excess 128

- Two's complement
  - Throw away leftmost carry
Floating Point

• Binary numbers are a great representation for the integer numbers, but how do you represent something like 3486545.121234339 or 3.14159?
  − Limited space, limited numbers?

• The IEEE Floating Point Standard
  − Circa 1980
  − A binary number is interpreted differently
  − (more on this after set and string theory)
A Different Kind of Math: Boolean Logic

- Boolean math is named after George Boole, an English mathematician who proposed much of the basis of two-valued logic.
- Boolean logic is a system built around the values TRUE and FALSE.
- A detour: truth tables
  - Define $p$ and $q$ to be boolean variables (statements).
  - Relate $p$ and $q$ in known ways:
    - And, or, not, xor, implication.
Boolean Operations

• Negation: NOT, ~, ! - returns the opposite value
  - “not” is a unary operator
• Conjunction: AND, && - returns TRUE if both values are TRUE
• Disjunction: OR, || - returns TRUE if either value is TRUE
• Implication: -> returns FALSE only if a TRUE value leads to a FALSE conclusion
Boolean Operations (cont.)

- Exclusive OR: XOR, returns TRUE if either but not both of its operands are TRUE
- Equality: Iff, if and only if $<=$, returns TRUE if both operands have the same value
- AND and NOT are complete: that is, combinations of these two operators can perform the same operation as all the other boolean operators.
Standard Two-Variable Truth Tables

- **AND**
  
<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p and q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

- **OR**
  
<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p or q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
### More Truth Tables

- **Implies (if \( p \) then \( q \))**

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \implies q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

- **Equality (\( p \iff q \))**

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \iff q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Logic for Binary Numbers

- What if I replaced TRUE with 1 and FALSE with 0?
- What if I could build an electronic device that implements the behaviour of Boolean operators?
- You'd have a logic gate.
- Logic gates provide a computer the ability to compute boolean functions. Complex combinations of boolean functions allow you to play Quake, write email, or chat with friends.
What Do Gates Look Like?

• Combining these gates in different (but structured!) ways gives a computer the power to calculate a great number of things, as we'll see later this week.

• The Digital Logic Level
  - Combinations of gates form memory, circuits that can compute boolean and mathematical functions
Some More Basics: Discrete Math

• Discrete Math forms the basis of theoretical (and practical) computer science by providing objects, tools, and notations to formalize our thoughts.

• Discrete math is almost never concerned with continuous data (real numbers, etc.)
Sets

- A set is one of the basic units of discrete math
  - A set is an unordered collection of *elements*
  - A set cannot contain duplicate elements, so repetition is meaningless
  - A set is usually written as a comma-separated list in curly braces: e.g. \{1,3,5,7,9\} is the set of odd integers less than 10
  - The null set, infinite sets
  - Subset, proper subset
Sets (cont.)

• Sets can also be described in words (sort of)
  – \{ n \mid n \text{ is even for } n \text{ element of the natural numbers} \}

• Set Operations:
  – Union (like OR)
  – Intersection (like AND)
  – Complement (like NOT)

• Venn diagrams
Sequences

• A sequence is a list with order. In a sequence, *order matters*. Repetition is permitted.

• A sequence is usually written as a comma-separated list in parentheses:
  
  - \((0,2,4,6,8)\) is a sequence.

• Finite sequences can also be called *tuples*
Relations, Mappings, Functions

- A function is a method of associating some input value with some output value. We can say that a function maps a certain value to another value.
- Just like real math: $f(x) = x + 3$
- Domain and range: sets
  - One to one: 1 D value to 1 R value
  - Onto: uses all values of the range
Multiple Argument Functions

- Sometimes a function operates on more than one domain value at once:
  - $f(a,b,c,d,...,x,y,z)$
  - Function arguments and argument lists are a central mechanism in many programming languages
Back to Boolean Logic

• A predicate is a function that has a set \{TRUE, FALSE\} as its range

• We can more precisely define Boolean algebra now:
  
  − AND is a function that takes two arguments and returns the intersection of them
  
  − OR is a function that takes two arguments and returns the union
  
  − NOT is a function that takes one argument and returns the complement
Graphs and Trees

- A graph, G, is a combination of two sets:
  - A set of *points*, V (called *nodes* or *vertices*)
  - A set of *edges*, E, that contains pairs from V. Each pair defines a path or line (an edge) between the two nodes.
  - The *degree* of any node is the number of edges at that node
  - G = (V,E)
  - Graphs represent data and relationships
Graphs (cont.)

• A path is a sequence of vertices
• A simple path does not repeat any vertices
• A cycle is a path that starts and ends at the same vertex
• A simple cycle is a path that does not repeat nodes except for the start and end node
• A graph is a tree if it is connected and has no cycles
Directed Graphs & Subgraphs

• A graph $G$ is directed when the set $E$ contains ordered pairs

• Directed graphs are drawn with arrows

• A graph $G$ is a subgraph of graph $J$ if the nodes of $G$ are a subset of the nodes of $J$
Strings and Languages

- Languages are fundamental to computer science. Higher level languages build on lower ones via abstraction and encapsulation to provide more complex expression and operating power.
- Binary numbers are one example of strings.
- An alphabet is any finite set of symbols.
- A string of an alphabet is a finite sequence (order!) of alphabet symbols.
Strings (cont.)

- The **length** of a string $s$ is $|s|$ and is the number of symbols in $s$
- The **empty string** has length zero
- A string $w$ is a **substring** of $s$ if $w$ appears in order in $s$
- Two strings can be **concatenated** (roughly equivalent to adding, more like splicing)
- A **language** is a set of strings (usually related in some way)
• backup
Lab: The DOS & Unix Command Line

- Learn about the command-argument paradigm
- Analogous to function-name(arglist)
- Logging in (via SSH)
- Typing basic commands
- Invoking 'java' VM and 'javac' compiler
Odds and Sods: Abstraction and Interfaces

- Abstraction is one of the key components of computer science
- To this end, we often talk about the notion of an interface
- Acronyms are a type of naming abstraction
- The Unix command line is another abstraction
- The MSDOS command line is another abstraction or interface
The Four “Themes” Of CS

- The History of Computers: the birth, life, and future of computers and computation
- The Anatomy/Biology of Computers: hardware, memory, devices, multimedia, machine internals
- The Art of Computers: algorithms, data structures, software engineering
- The Philosophy of Computers: algorithmic thinking, computational complexity, theory, AI, ethics, philosophies, methodologies