Alfred V. Aho
aho@cs.columbia.edu

Computational Thinking in Programming Language Design

NEC Labs, Princeton, NJ
January 17, 2013
Software, software, everywhere

How much software does the world use today?

Guesstimate: around one trillion lines of source code

What is the sunk cost of the legacy software base?

$100 per line of finished, tested source code

How many bugs are there in the legacy base?

10 to 10,000 defects per million lines of source code
## Evolution of programming languages

<table>
<thead>
<tr>
<th>1970</th>
<th>2013</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran</td>
<td>C</td>
<td>Java</td>
</tr>
<tr>
<td>Lisp</td>
<td>Java</td>
<td>PHP</td>
</tr>
<tr>
<td>Cobol</td>
<td>Objective-C</td>
<td>C#</td>
</tr>
<tr>
<td>Algol 60</td>
<td>C++</td>
<td>C#</td>
</tr>
<tr>
<td>APL</td>
<td>C#</td>
<td>C++</td>
</tr>
<tr>
<td>Snobol 4</td>
<td>PHP</td>
<td>C</td>
</tr>
<tr>
<td>Simula 67</td>
<td>Visual Basic</td>
<td>Python</td>
</tr>
<tr>
<td>Basic</td>
<td>Python</td>
<td>JavaScript</td>
</tr>
<tr>
<td>PL/1</td>
<td>Perl</td>
<td>Visual Basic</td>
</tr>
<tr>
<td>Pascal</td>
<td>Ruby</td>
<td>Ruby</td>
</tr>
</tbody>
</table>

[http://www.tiobe.com, January 2013] [PyPL Index, January 2013]
Programming languages today

Today there are thousands of programming languages.

The website http://www.99-bottles-of-beer.net has programs in over 1,500 different programming languages and variations to print the lyrics to the song “99 Bottles of Beer.”
“99 Bottles of Beer”

99 bottles of beer on the wall, 99 bottles of beer.  
Take one down and pass it around, 98 bottles of beer on the wall.  

98 bottles of beer on the wall, 98 bottles of beer.  
Take one down and pass it around, 97 bottles of beer on the wall.  

.  
.  
.  

2 bottles of beer on the wall, 2 bottles of beer.  
Take one down and pass it around, 1 bottle of beer on the wall.  

1 bottle of beer on the wall, 1 bottle of beer.  
Take one down and pass it around, no more bottles of beer on the wall.  

No more bottles of beer on the wall, no more bottles of beer.  
Go to the store and buy some more, 99 bottles of beer on the wall.
"99 Bottles of Beer” in AWK

BEGIN {
for(i = 99; i >= 0; i--) {
print ubottle(i), "on the wall," , lbottle(i) "."
print action(i), lbottle(inext(i)), "on the wall."
print
}
}
function ubottle(n) {
return sprintf("%s bottle%s of beer", n ? n : "No more", n - 1 ? "s" : ")
}
function lbottle(n) {
return sprintf("%s bottle%s of beer", n ? n : "no more", n - 1 ? "s" : "")
}
function action(n) {
return sprintf("%s", n ? "Take one down and pass it around," : \
    "Go to the store and buy some more,"")
}
function inext(n) {
return n ? n - 1 : 99
}

“99 Bottles of Beer” in Perl

[Andrew Savage, http://search.cpan.org/dist/Acme-EyeDrops/lib/Acme/EyeDrops.pm]
“99 Bottles of Beer” in the Whitespace language

[Andrew Kemp, http://compsoc.dur.ac.uk/whitespace/]
Evolutionary forces on languages

Increasing diversity of applications

Stress on increasing programmer productivity and shortening time to market

Need to improve software security, reliability and maintainability

Emphasis on mobility and distribution

Support for parallelism and concurrency

New mechanisms for modularity

Trend toward multi-paradigm programming
Case study 1: Scala

- Scala is a multi-paradigm programming language designed by Martin Odersky at EPFL starting in 2001
- Intended as a “better Java”
- Integrates functional, imperative and object-oriented programming in a statically typed language
- Functional constructs used for parallelism and distributed computing
- Generates Java byte code
- Used to implement Twitter
  – Lady Gaga has 32 million followers
  – Barack Obama has 25 million followers
Case study 2: Ruby

• Ruby is a dynamic scripting language designed by Yukihiro Matsumoto in Japan in the mid 1990s

• Influenced by Perl and Smalltalk

• Supports multiple programming paradigms including functional, object oriented, imperative, and reflective

• The three pillars of Ruby
  – everything is an object
  – every operation is a method call
  – all programming is metaprogramming

• Made famous by the web application framework Rails
Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child’s analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incestuous about this vision is that computing and computers facilitate the spread of computational thinking.
What is computational thinking?

The thought processes involved in **formulating problems** so their solutions can be represented as computation steps and algorithms.

*Alfred V. Aho*

*Computation and Computational Thinking*

What is computational thinking?

The thought processes involved in formulating a problem and expressing its solution in a way that a computer – human or machine – can effectively carry out.
Models of computation in languages

Underlying most programming languages is a model of computation:

Procedural: Fortran (1957)

Functional: Lisp (1958)

Object oriented: Simula (1967)

Logic: Prolog (1972)

Relational algebra: SQL (1974)
Computational model of AWK

AWK is a scripting language designed to perform routine data-processing tasks on strings and numbers.

Use case: given a list of name-value pairs, print the total value associated with each name.

alice 10
eve 20
bob 15
alice 30

\{ total[$1] += $2 \}
END \{ for (x in total) print x, total[x] \}

An AWK program is a sequence of pattern-action statements.

alice 10
eve 20
bob 15
alice 40
Theory in practice: regular expression pattern matching in Perl, Python, Ruby vs. AWK

Time to check whether $a^n a^n$ matches $a^n$

Russ Cox, *Regular expression matching can be simple and fast (but is slow in Java, Perl, PHP, Python, Ruby, ...)* [http://swtch.com/~rsc/regexp/regexp1.html, 2007]
A good way to learn computational thinking

Design and implement your own programming language!
The programming languages and compilers course at Columbia

1. Theory
   • principles of modern programming languages
   • fundamentals of compilers
   • fundamental models of computation

2. Practice
   • a semester-long programming project in which students work in small teams to create and implement an innovative little language of their own design. This project teaches computational thinking as well as project management, teamwork, and communication skills that are useful in all aspects of any career.
# The project schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Form a <strong>team</strong> and design an innovative new language</td>
</tr>
<tr>
<td>4</td>
<td>Write a <strong>whitepaper</strong> on your proposed language modeled after the Java whitepaper</td>
</tr>
<tr>
<td>8</td>
<td>Write a <strong>tutorial</strong> patterned after Chapter 1 and a <strong>language reference manual</strong> patterned after Appendix A of Kernighan and Ritchie’s book, <em>The C Programming Language</em></td>
</tr>
<tr>
<td>14</td>
<td>Give a ten-minute <strong>presentation</strong> of your language to the class</td>
</tr>
<tr>
<td>15</td>
<td>Give a 30-minute <strong>working demo</strong> of your compiler to the teaching staff</td>
</tr>
<tr>
<td>15</td>
<td>Hand in the <strong>final project report</strong></td>
</tr>
</tbody>
</table>
Some of the languages created

- Producing applications for an Android cell phone
- Configuring a wireless sensor network
- Turning data into music
- Giving advice on what to wear
- Generating code for a quantum computer
W2W
(WHAT TO WEAR)

TEAM MEMBERS:
Jasleen Lamba(jl3809) - PROJECT MANAGER
Afreen Azad(aa3165)- LANGUAGE GURU
Cyril Joshi(ckj2108) - SYSTEM ARCHITECT
Kunal Mudgal(kum2104) - SYSTEM INTEGRATOR

PROFESSOR: Prof. Alfred V. Aho
TA GUIDE: Shuai Sun
What is W2W?

W2W is a programming language that allows you to store information about the garments in your wardrobe and write programs to generate outfits depending on the weather conditions of that day.
Why W2W?
Have you spent hours everyday wondering WHAT TO WEAR ?!
MOTIVATION

Has this happened to you?

It rained but you didn’t have an umbrella!

It was extremely cold but you didn’t have a jacket!

It was sunny and you came out wearing a jacket!
TARGET AUDIENCE

• YOU?

• Any user from any age group, any phase of life.

• Clothing retailers like Macys, Forever 21 etc.
W2W - WHAT TO WEAR

LANGUAGE BUZZ WORDS

Domain-specific

Declarative

Simple

Robust

User and situation oriented

Easy to use

Easy to learn

Intuitive

Interactive

Portable
It’s summer break and Ethan is going out for a 2 day trip. He wants to make sure that he carries along clothes appropriate for the weather accounting for the fact that the clothes should be light or blue (his favourite color)!
SYNTACTIC CONSTRUCTS

create wardrobe ethanwardrobe;
{
    { item = jacket; }
    { item = tshirt;
        color = blue;
        shade = light;
    }
    { item = trousers;
        maker = prada;
        color = black;
    }
    { item = sweater;
        weight = heavy;
    }
}
SYNTACTIC CONSTRUCTS

```java
use wardrobe mywardrobe;
void main()
{
    date start = 04/25;
    date end = 04/26;

    for each garment in wardrobe
    {
        if(garment.shade == "light")
            { include; }
        else
            { 
                if(garment.color == "blue")
                   { include; } 
        }
    }

    generateOutfit(start,end);
}
SYSTEM ARCHITECTURE

Front End
- Lexer & Parser
- Syntax Tree
- Symbol Table

Back End
- Target Java Program
- Wardrobe Selection Algorithm
- Yahoo Weather API
- Database

User Input
- Suggested Outfit

Java Compiler
- Runnable Java Program
- JVM
DEVELOPMENT TOOLS USED

ANTLR

eclipse

Google

code

MySQL

Yahoo
Telling lessons learned by students

• “During this course we realized how naïve and overambitious we were, and we all gained a newfound respect for the work and good decisions that went into languages like C and Java which we’ve taken for granted for years.”

• “Designing a language is hard and designing a simple language is extremely hard!”
Quantum computing: What the physicists are saying

“Quantum information is a radical departure in information technology, more fundamentally different from current technology than the digital computer is from the abacus.”

William D. Phillips
1997 Nobel Prize Winner in Physics
Shor’s integer factorization algorithm

Problem: Given a composite $n$-bit integer, find a nontrivial factor.

Best-known deterministic algorithm on a classical computer has time complexity $\exp(O(n^{1/3} \log^{2/3} n))$.

A quantum computer can solve this problem in $O(n^3)$ operations.

Peter Shor

Algorithms for Quantum Computation: Discrete Logarithms and Factoring
Integer factorization: estimated times

Classical: number field sieve

- Time complexity: $\exp(O(n^{1/3} \log^{2/3} n))$
- Time for 512-bit number: 8400 MIPS years
- Time for 1024-bit number: 1.6 billion times longer

Quantum: Shor’s algorithm

- Time complexity: $O(n^3)$
- Time for 512-bit number: 3.5 hours
- Time for 1024-bit number: 31 hours
  (assuming a 1 GHz quantum device)
Shor’s integer factorization algorithm

Input: A composite number $N$
Output: A nontrivial factor of $N$

if $N$ is even then return 2;
if $N = a^b$ for integers $a \geq 1$, $b \geq 2$ then
  return $a$;
x = rand(1, N-1);
if gcd($x$, $N$) $> 1$ then return gcd($x$, $N$);
r = order($x \mod N$); // only quantum step
if $r$ is even and $x^{r/2} \neq (-1) \mod N$ then
  \{f1 = gcd($x^{r/2}-1$, $N$);  f2 = gcd($x^{r/2}+1$, $N$)\};
if $f1$ is a nontrivial factor then return $f1$;
else if $f2$ is a nontrivial factor then return $f2$;
else return fail;

M. A. Nielsen and I. L. Chuang
Quantum Computation and Quantum Information
Cambridge University Press, 2000
The order-finding problem

Given positive integers \( x \) and \( N \), \( x < N \), such that \( \gcd(x, N) = 1 \), the order of \( x \) (mod \( N \)) is the smallest positive integer \( r \) such that \( x^r \equiv 1 \) (mod \( N \)).

E.g., the order of 5 (mod 21) is 6. \[ 5^6 = 15625 = 744 \times 21 + 1 \]

The order-finding problem is, given two relatively prime integers \( x \) and \( N \), to find the order of \( x \) (mod \( N \)).

All known classical algorithms for order finding are superpolynomial in the number of bits in \( N \).
Quantum order finding

Order finding can be done with a quantum circuit containing

\[ O((\log N)^2 \log \log (N) \log \log \log (N)) \]

elementary quantum gates.

Best known classical algorithm requires

\[ \exp(O((\log N)^{1/2} (\log \log N)^{1/2})) \]

time.
Towards a model of computation for quantum programming languages

- Physical System
- Mathematical Abstractions
- Basic Data Types and Operations
- Model of Computation
Towards a model of computation for quantum programming languages

The Four Postulates of Quantum Mechanics

M. A. Nielsen and I. L. Chuang
Quantum Computation and Quantum Information
Cambridge University Press, 2000
State-space postulate

Postulate 1

The state of an isolated quantum system can be described by a unit vector in a complex Hilbert space.
Qubit: quantum bit

• The state of a quantum bit can be described by a unit vector in a 2-dimensional complex Hilbert space (in Dirac notation)

\[ |\psi\rangle = \alpha |0\rangle + \beta |1\rangle \]

where \( \alpha \) and \( \beta \) are complex coefficients called the amplitudes of the basis states \( |0\rangle \) and \( |1\rangle \), and

\[ |\alpha|^2 + |\beta|^2 = 1 \]

• In linear algebra

\[
\begin{pmatrix}
\psi_0 \\
\psi_1
\end{pmatrix}
= \alpha \begin{pmatrix} 1 \\
0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\
1 \end{pmatrix}
\]
Postulate 2

The evolution of a closed quantum system can be described by a **unitary operator** $U$. (An operator $U$ is unitary if $U^\dagger U = I$.)

\[ |\psi\rangle \rightarrow U \rightarrow U|\psi\rangle \]

- state of the system at time $t_1$
- state of the system at time $t_2$
Useful quantum operators: Hadamard

The **Hadamard operator** has the matrix representation

\[
H = \frac{1}{\sqrt{2}} \begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\]

\(H\) maps the computational basis states as follows

\[
H|0\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)
\]

\[
H|1\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)
\]

Note that \(HH = I\).
Composition-of-systems postulate

Postulate 3

The state space of a combined physical system is the tensor product space of the state spaces of the component subsystems.

If one system is in the state $\ket{\psi_1}$ and another is in the state $\ket{\psi_2}$, then the combined system is in the state $\ket{\psi_1} \otimes \ket{\psi_2}$.

$\ket{\psi_1} \otimes \ket{\psi_2}$ is often written as $\ket{\psi_1}\ket{\psi_2}$ or as $\ket{\psi_1\psi_2}$. 
Useful quantum operators: CNOT

The two-qubit CNOT (controlled-NOT) operator has the matrix representation:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

CNOT flips the target bit \( t \) iff the control bit \( c \) has the value 1:

The CNOT gate maps

\[
|00\rangle \mapsto |00\rangle, \quad |01\rangle \mapsto |01\rangle, \quad |10\rangle \mapsto |11\rangle, \quad |11\rangle \mapsto |10\rangle
\]
Measurement postulate

Postulate 4

Quantum measurements can be described by a collection \( \{M_m\} \) of operators acting on the state space of the system being measured. If the state of the system is \( |\psi\rangle \) before the measurement, then the probability that the result \( m \) occurs is

\[
p(m) = \langle \psi | M_m^\dagger M_m | \psi \rangle
\]

and the state of the system after measurement is

\[
\frac{M_m | \psi \rangle}{\sqrt{\langle \psi | M_m^\dagger M_m | \psi \rangle}}
\]
Properties of measurement operators

The measurement operators satisfy the completeness equation:

\[ \sum_{m} M_{m}^{\dagger} M_{m} = I \]

The completeness equation says the probabilities sum to one:

\[ \sum_{m} p(m) = \sum_{m} \langle \psi | M_{m}^{\dagger} M | \psi \rangle = 1 \]
Computational model: Quantum Circuits

Quantum circuit to create Bell (Einstein-Podulsky-Rosen) states:

Circuit maps

\[
|00\rangle \mapsto \frac{|00\rangle + |11\rangle}{\sqrt{2}},
|01\rangle \mapsto \frac{|01\rangle + |10\rangle}{\sqrt{2}},
|10\rangle \mapsto \frac{|00\rangle - |11\rangle}{\sqrt{2}},
|11\rangle \mapsto \frac{|01\rangle - |10\rangle}{\sqrt{2}}
\]

Output is an entangled state, one that cannot be written in a product form. (Einstein: “Spooky action at a distance.”)
Quantum computer compiler

QIR: quantum intermediate representation
QASM: quantum assembly language
QPOL: quantum physical operations language

Computational abstractions

K. Svore, A. Aho, A. Cross, I. Chuang, I. Markov
A Layered Software Architecture for Quantum Computing Design Tools
MIT ion trap simulator
Design flow with fault tolerance and error correction

Mathematical Model: Quantum mechanics, unitary operators, tensor products

Computational Formulation: Quantum bits, gates, and circuits

QCC: QIR, QASM

Software: QPOL

Physical System: Laser pulses applied to ions in traps

EPR Pair Creation

Quantum Circuit Model

QIR → QASM

QIR:
- qubit x,y;
- gate h;
- gate cx;
- h x;
- cx x,y;

QASM:
- <layout>
- grid (3,1)
- empty (1,1) - (3,1)
- <ion x, "data", (1,1)>
- <ion y, "data", (3,1)>
- </layout>
- <qpol>
- gate "H", (x)
- move x, (2,1)

Machine Instructions

Physical Device

Fault Tolerance and Error Correction (QEC)

K. Svore
PhD Thesis
Columbia
Topological quantum computer

**Theorem:** In any topological quantum computer, all computations can be performed by moving only a single quasiparticle!
Topological robustness
Topological robustness

\[
\begin{array}{c}
\text{time} \\
\end{array} = \begin{array}{c}
\text{time} \\
\end{array}
\]
Quantum computation by braiding

Quantum Circuit

Braid

L. Hormozi, G. Zikos, N. Bonesteel, S. Simon
Topological quantum compiling
1. Degenerate ground states (in punctured system) act as the qubits.

2. Unitary operations (gates) are performed on ground state by braiding punctures (quasiparticles) around each other.

   Particular braids correspond to particular computations.

3. State can be initialized by “pulling” pairs from vacuum.
   State can be measured by trying to return pairs to vacuum.

4. Variants of schemes 2, 3 are possible.

Advantages:

• Topological quantum “memory” highly protected from noise
• The operations (gates) are also topologically robust
Universal set of topologically robust gates

Single qubit rotations: $|\psi\rangle \xrightarrow{U_\phi} U_\phi |\psi\rangle$

Controlled NOT:

Bonesteel, Hormozi, Simon, 2005, 2006
Target language code braid for CNOT gate with Solovay-Kitaev optimization

Steve Simon, Oxford
http://www-thphys.physics.ox.ac.uk/people/SteveSimon/overview.html
Recent work: Synthesis and simulation of quantum circuits

Synthesis of efficient quantum circuits

- depth-optimal single-qubit circuits [Bocharov & Svore, 2012]
- fault-tolerant single-qubit rotations [Duclos-Cianci & Svore, 2012]
- approximating single-qubit unitaries with Clifford and T- gates
  [Kliuchnikov, Maslov & Mosca, 2012]
- fast synthesis of depth-optimal quantum circuits
  [Amy, Maslov, Mosca & Roetteler, 2012]
- exact synthesis of multi-qubit Clifford and T- circuits
  [Giles & Sellinger, 2012]

Efficient simulation of quantum circuits

- QuIDDPro quantum circuit simulator [Viamontes, Markov & Hayes, University of Michigan, 2009]
- LIQUi|> software architecture and toolsuite [Wecker, Microsoft Research, ongoing]
Why quantum computing is challenging - Physical constraints

• States are superpositions
• Operators are unitary transforms
• States of qubits can become entangled
• Measurements are destructive
• No-cloning theorem: you cannot copy an unknown quantum state!
Why quantum computing is challenging - Nontraditional programming patterns

• Phase estimation
• Quantum Fourier transform
• Period finding
• Eigenvalue estimation
• Grover search
• Amplitude amplification
Quantum computing research challenges

More qubits

Scalable, fault-tolerant architectures

Suggestive programming languages

Efficient compilation techniques

More good algorithms!
Open question: Is computational thinking innate?