CSE 301
History of Computing
History of Programming Languages (through 1980)
Theoretical Foundations of Computation

- So far we’ve primarily talked about computer engineering
- Computer programming’s foundations lie in theory
  - Kurt Gödel
  - Alan Turing
Kurt Gödel

- Incompleteness Theorem
  - Provided arithmetic is consistent, mathematics is incomplete in that there exist propositions which cannot be proved.
  - Landmark in 20th century mathematics
  - Implies that a computer can never be programmed to answer all mathematical questions

- Fled Nazi occupied Austria in 1940 to U.S. because:
  - He thought he might be conscripted to fight in German army
  - Though he was Christian, people thought he was Jewish
    - On one occasion he and his wife were attacked by a gang in the street

- Became a chair at Princeton University
- Close friend of Einstein

- [http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Godel.html](http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Godel.html)
Alan Mathison Turing

- Born in 1912 in London, England
- Began studying the work of John von Neumann in quantum mechanics in 1932
- Wrote “On Computable Numbers, with an application to the Entscheidungsproblem” in 1936
  - Defines the notion of an algorithm
  - Introduces the fundamental concept of a computing model commonly referred to as a “Turing machine”
  - Introduces the notion of a universal algorithmic automaton, more commonly referred to later as a universal Turing machine
  - Considered to be one of the major contributions to the logical foundations of Computer Science
Turing Machine

- An abstract mathematical device capable of reading and writing units of information on a tape that is partitioned into a succession of squares and potentially infinite in length.
  - A tape (of infinite length) divided into “cells” horizontally along its surface
  - A finite set of symbols that can be stored on the tape
  - A reading and writing device capable of reading data from a single cell, erasing data in a single cell, writing data to a single cell or moving left or right along the tape one cell at a time
  - A control unit that determines its operation
Turing Machine

- The machine has a number of states that it can be in.
- The behavior of the machine is determined by the current state of the machine and the symbol at the cell on the tape where the read/write device is situated.
- Given this information, the machine may
  - change to another state or remain in the same state
  - move left or right on the tape
  - write data to the tape or erase data on the tape
Above: a fanciful mechanical Turing machine's TAPE and HEAD (TABLE not shown).

Below: Usually Turing machines are drawn in a much simpler way -- as just a row of squares with symbols in them, the "head" drawn over the "square being scanned" (drawing after Kleene (1952) p. 375).
Universal Turing Machine

- A Turing machine that is capable of simulating any other Turing machine.
  - One tape holds the "program" (machine to be simulated).
  - A second tape holds the current "state" of the Turing machine that we are emulating.
  - A third tape holds the "input" and will, upon completion, hold the "output."

- Turing was describing a modern computer in 1936 before it was realistically possible to construct one.

- No one has come up with a more general model for computation to this day.
Universal Turing Machine

The diagram illustrates a Turing machine with a tape, a head, and a control unit. The tape contains the code number of the Turing machine \( M \), input to \( M \), and output. The control unit includes a table of \( U \) specifying the actions for each symbol (Scanned symbol) when in a particular state (Current state). The table entries include the action to write, move, and the next state for symbols 1, X, Y, and L, D, E, etc. The diagram also shows the directions for printing, marking, left, or right. The tape head moves left or right according to the actions in the table.
The Halting Problem

- Turing’s work gave rise to the notion that computers could not solve every theoretically possible problem.
  - A universal program U cannot exist that is capable of answering, within a finite number of steps, the following question relative to any program P:
    - *Will P terminate after a finite number of steps?*
  - If such a program U existed, it would generate a logical contradiction simply by being applied to itself.
- Many scientists claimed that a Turing machine could compute any intellectual process, and Turing proves them wrong.
Turing’s Work Continues

- Worked on the Enigma problem during WWII at Bletchley Park
- Developed the Bombe in 1940 to help decode encrypted Enigma messages by the Germans
  - Based on earlier work by Polish mathematicians Rejewski, Rozycki, Zygalski
- Worked in 1941 to help break more difficult Enigma codes using statistical analysis
- Joins the National Physical Laboratory in 1946 and works on ACE (Automatic Computing Engine)
- Works on MADAM (Manchester Automatic Digital Machine) project in 1948 at Manchester University
Final contribution to computing

- Writes *Computing Machinery and Intelligence* in 1950 and describes his famous Turing Test:
  - If a computer and a person are placed behind a wall and answer questions such that we cannot tell which is answering, the computer exhibits artificial intelligence.

- Arrested in 1952 for violating British homosexuality statutes
  - Found guilty and sentenced to injections of estrogen for a year rather than prison
  - Loses his security clearance and is deemed a security risk

- Dies of potassium cyanide poisoning in 1954 at the age of 41. Officials rule this a suicide.
Hardware vs. Software

- **Computer Hardware**
  - the physical components of a computer
  - ex: CPU, RAM, Hard Drive, etc …

- **Computer Software**
  - programs that run on a computer
  - ex: Operating System, Word Processor, Internet Browser, etc …
  - makes use of *algorithms* to solve problems
    - algorithms are step-by-step procedures for solving a problem in a finite number of steps
    - based on mathematical principles
  - software is defined using programming languages

- In the old days, money was spent primarily on hardware
  - By 1953, half the cost of running a computer was already spent on programming

- These days, it’s not even close, software is more than 90% of computer costs
  - many more CS jobs than CE
  - more research in CS than CE

- One thing is always for sure, software is beholden to hardware
  - software can only use real hardware technologies that exist
The first assembler

- Assembler - a computer program for translating assembly language into executable machine code
  - Example: ADD R1, R2, R3 0110000100100011

- The EDSAC programming system was based on a subroutine library
  - commonly used functions that could be used to build all sorts of more complex programs
  - the first version, Initial Orders 1, was devised by David Wheeler, then a research student, in 1949

- Team published “The Preparation of Programs for an Electronic Digital Computer”
  - the only programming textbook then available
  - computers today still use Cambridge model for subroutines library
The first compiler

- A compiler is a computer program that translates a computer program written in one computer language (the *source* language) into a program written in another computer language (the *target* language).
  - Typically, the target language is assembly language
  - Assembler may then translate assembly language into machine code
    - Machine code are directions a computer can understand on the lowest hardware level (1s & 0s)
- A-0 is a programming language for the UNIVAC I or II, using three-address code instructions for solving mathematical problems.
- A-0 was the first language for which a compiler was developed.
  - It was produced by Grace Hopper's team at Remington Rand in 1952
  - Grace Hopper had previously been a programmer for the Harvard Mark machines
    - One of U.S.’s first programmers
    - She found a moth in the Mark I, which was causing errors, and called it a computer “bug”
FORTRAN (1957)

- First successful high-level programming language
  - Code more readable and understandable by humans
- Developed by John Bachus at IBM
  - Stands for: FORmula TRANslatin
  - Started development in 1954
  - Released in 1957, is still in use today (how many technologies can say that?)
- A key goal of FORTRAN was efficiency, although portability was also a key issue
  - Automatic programming that would be as good as human programming of assembly code
  - Resulted in making programs 90% as good as humans
- Programs that took weeks to write could now take hours
- 1961 – First FORTRAN programming textbook
  - Universities began teaching it in undergrad programs
- Provided standard exchange of programs despite different computers
- Became the standard for scientific applications
REAL SUM6, SUM7, SUM8, DIF6, DIF7, DIF8, SUMINF
OPEN(6, FILE='PRN')
SUM6=.9*(1.0 - 0.1**6)/0.9
SUM7=.9*(1.0 - 0.1**7)/0.9
SUM8=.9*(1.0 - 0.1**8)/0.9
******COMPUTER SUM OF INFINITE TERMS
SUMINF=0.9/(1.0 - 0.1)
******COMPUTE DIFFERENCES BETWEEN FINITE & INFINITE SUMS
DIF6 = SUMINF - SUM6
DIF7 = SUMINF - SUM7
DIF8 = SUMINF - SUM8
WRITE(6,*) 'INFINITE SUM = ', SUMINF
WRITE(6,*) 'SUM6 = ', SUM6, ' INFINITE SUM - SUM6 = ', DIF6
WRITE(6,*) 'SUM7 = ', SUM7, ' INFINITE SUM - SUM7 = ', DIF7
WRITE(6,*) 'SUM8 = ', SUM8, ' INFINITE SUM - SUM8 = ', DIF8
STOP
END
COBOL (1960)

- Stands for: COmmon Business-Oriented Language
- COBOL was initially created in 1959 (and released in 1960 as Cobol 60) by a group of computer manufacturers and government agencies
  - US Government wanted a standard for its computers
- One goal of COBOL's design was for it to be readable by managers, so the syntax had very much of an English-like flavor.
  - The specifications were to a great extent inspired by the FLOW-MATIC language invented by Grace Hopper
    - She then promoted COBOL’s use
- Became the standard for business applications
  - Still used in business applications today.
- 90% of applications over next 20 years were written in either COBOL or FORTRAN
  - Old programmers came out of hiding for Y2K
000100 ID DIVISION.
000200 PROGRAM-ID. ACCEPT1.
000300 DATA DIVISION.
000400 WORKING-STORAGE SECTION.
000500 01 WS-FIRST-NUMBER PIC 9(3).
000600 01 WS-SECOND-NUMBER PIC 9(3).
000700 01 WS-TOTAL PIC ZZZ9.
000800*
000900 PROCEDURE DIVISION.
001000 0000-MAINLINE.
001100     DISPLAY 'ENTER A NUMBER: '.
001200     ACCEPT WS-FIRST-NUMBER.
001300*  
001400     DISPLAY 'ANOTHER NUMBER: '.
001500     ACCEPT WS-SECOND-NUMBER.
001600*  
001700     COMPUTE WS-TOTAL = WS-FIRST-NUMBER + WS-SECOND-NUMBER.
001800     DISPLAY 'THE TOTAL IS: ', WS-TOTAL.
001900     STOP RUN.
Living & Dead Languages

- Hundreds of programming languages popped up in the 1960s, most quickly disappeared
- Some dead:
  - JOVIAL, SNOBOL, Simula-67, RPG, ALGOL, PL/1, and many, many more
- Some still kicking:
  - LISP (1957)
  - BASIC (1964)
  - Pascal (1970)
  - Prolog (1972)
  - And of course, C (1973)
ALGOL-60 (1960)

- Created mainly in Europe by a committee of computer scientists
  - John Backus and Peter Naur both served on the committee which created it
  - Desired an IBM-independent standard
- Stands for: ALGOrithmic Language
- Primarily intended to provide a mechanism for expressing algorithms uniformly regardless of hardware
- The first report on Algol was issued in 1958,
  - Specifications revised in 1959 and 1960 (and later in 1968)
- The language itself was not a success, but it was an influence on other successful languages
  - A primary ancestor of Pascal and C.
- It introduced block structure, compound statements, recursive procedure calls, nested if statements, loops, and arbitrary length identifiers
LISP (1958)

- Developed by John McCarthy at MIT
- Stands for: LISt Processing
  - Designed for symbolic processing
  - Introduced symbolic computation and automatic memory management
- Used extensively for Artificial Intelligence applications
BASIC (1964)

- Created by John Kemeny and Thomas Kurtz at Dartmouth College
- Stands for: Beginner's All-purpose Symbolic Instruction Code
  - one of the first languages designed for use on a time-sharing system
  - one of the first languages designed for beginners
- Variants like Visual BASIC still used today by Microsoft.
Pascal (1970)

- Developed by Niklaus Wirth in an effort to make structured programming easier for a compiler to process.
- Based on Algol
  - Named in honor of mathematician and philosopher Blaise Pascal
- Wirth also developed Modula-2 and Oberon, languages similar to Pascal which also support object-oriented programming.
- Pascal was the most popular programming language for teaching computer programming in the 1970s and 1980s (now it’s very, very ill)
Prolog (1972)

- Created by Alain Colmerauer and Phillipe Roussel of the University of Aix-Marseille and Robert Kowalski of the University of Edinburgh.
- Stands for: PROgramming in LOGic.
- Prolog is the leading *logical* programming language.
  - used in artificial intelligence programs, computer linguistics, and theorem proving.
parents(william, diana, charles).
parents(henry, diana, charles).
parents(charles, elizabeth, philip).
parents(diana, frances, edward).
parents(anne, elizabeth, philip).
parents(andrew, elizabeth, philip).
parents(edwardW, elizabeth, philip).
marrried(diana, charles).
marrried(elizabeth, philip).
marrried(frances, edward).
marrried(anne, mark).
parent(C,M) <= parents(C,M,D).
parent(C,D) <= parents(C,M,D).
sibling(X,Y) <= parents(X,M,D) and parents(Y,M,D).
C (1973)

- Developed by Ken Thompson and Dennis Ritchie at AT&T Bell Labs for use on the UNIX operating system.
  - now used on practically every operating system
  - popular language for writing system software
- Features:
  - An extremely simple core language, with non-essential functionality provided by a standardized set of library routines.
  - Low-level access to computer memory via the use of pointers.
- C descendants: C++, C#, Java
- We’ll see more when we talk more about the PC & Internet
Simula 67

- First Object Oriented Programming languages
- Developed in Norway in 1960s by Kristen Nygaard and Ole-Johan Dahl
  - got idea making ship simulations
  - they won Turing Award in 2001
- Later Object Oriented Programming languages?
  - Smalltalk, C++, Visual Basic, Java, C#