Chapter 2 - History

Thought questions

• When (decade) was first compiler written for a high level machine?
• What were the first languages like?
• Why do we have so many languages?
• What makes a language popular?
• How is the importance of a language judged?
Example of Babylonian “Programming” (p. 35 of text) to make an underground cistern to hold water:

A cistern.
The length equals the height - we want the width.
A certain volume of dirt has been excavated.
The cross-sectional area plus this volume comes to 120.
The length is 5. What is the width?
Add 1 to 5, getting 6.
Divide 6 into 120, obtaining 20.
Divide 5 into 20, obtaining the width, 4.
This is the procedure.

Notes on Babylonian “Program”:

• Despite the numbers, it is expressing an algebraic solution to equations:
  – Area + Volume = \( T \) (120 in the example)
  – Area = length \( \times \) width
  – Volume = length \( \times \) width \( \times \) height
  – height = length
  – In code: \( T = \text{length} \times \text{width} \times \text{length} \)
  – Solving for the width:
    \( w = \frac{T}{(\text{length} + \text{length} \times \text{length})} = \frac{T}{(1 + \text{length}) \div \text{length}} \)
Jacquard Loom (Early 1800’s)

• Machines for weaving cloth
• Weaving pattern was programmed by cards / paper tape

Louden, Chapter 2/Scott, Chapter 1

Babbage’s Analytical Engine (Design)

• Mechanical digital computer (cogs, levers…)
• Programmed by a sequence of data and operation cards
• Never fully built, but some programs were written by Ada Lovelace (first programmer)
• Charles Babbage (1792-1871) was an eminent figure in his day,
• He was a close acquaintance of Charles Darwin, Sir John Herschel, Laplace, and Alexander Humboldt, and was author of more than eighty papers and books on a broad variety of topics.
• His vision of a massive brass, steam-powered, general-purpose, mechanical computer inspired some of the great minds of the nineteenth century but failed to persuade any backer to provide the funds to actually construct it.
• It was only after the first computers had been built that designers discovered the extent to which Babbage had anticipated almost every aspect of their work.

Louden, Chapter 2/Scott, Chapter 1
1940’s: Languages without Machines

- Lambda Calculus by Alonzo Church (1941)
  - The basis for functional programming languages
- Konrad Zuse, a German engineer working alone while hiding out in the Bavarian Alps, develops Plankalkul (1945). Means “Plan calculus”.
  Different machine architecture (typical was: program variables, control statements, assignments, expressions)
  It includes assignment statements, subroutines, conditional statements, iteration, floating point arithmetic, arrays, hierarchical record structures, assertions, exception handling, and other advanced features such as goal-directed execution.

Assembly Languages

- Assembly languages were originally designed 1-1 correspondence with machine language.
- Translating from a high level language to assembly language requires a compiler.
1950’s: first Implemented high level languages

• Early 1950’s: First few stored program computers working.
  – Mark 1, EDSAC, ACE, EDVAC, UNIVAC 1
• Small memory machines
• Programmed in machine code
• The good old days!!

Rear Admiral Grace Mary Hopper

• Programmer on Mark I, Mark II, and Mark III computers and UNIVAC I, the first large-scale electronic digital computer
• 1949 began work on first compiler A-0
  – Translated symbolic mathematical code into machine code
• Then came B-0, later called FLOW-MATIC.
  – automatic billing and payroll calculation
• She conceptualized and led the development of COBOL (1959)
• Popularized the term “debugging”
Grace Hopper quotes …

• It’s easier to ask forgiveness than it is to get permission

• A ship in port is safe, but that is not what ships are for. Sail out to sea and do new things

• the most damaging phrase in the language is ‘We've always done it this way’

FORTRAN - the first language

• John Backus: leader of the group at IBM that developed FORTRAN and its compiler
• A tool for scientific calculation (FORmula TRANslation).
• Execution efficiency the main goal.
• Still very much in use today (Fortran I, II, III, IV, Fortran66, Fortran77, Fortran90, Fortran95).
Overview of FORTRAN IV

- Column 1 (punched cards) used to indicate comment lines
- Column 2-5 used for line numbers (optional, dropping deck was serious)
- Data: integer, real, arrays (no chars, records or pointers!)
- Variable declarations are optional (variables starting with I..N are integer, others are real)

Overview of FORTRAN IV...

- Has a three-way if test, goto statements and computed gotos (like a switch), but no recursion
- EQUIVALENCE declaration causes variables to be aliased (dangerous!)
Other languages followed quickly:


COBOL (1959-1960)

- Common Business-Oriented Language
- Developed in 1959 by a group of computer professionals called the Conference on Data Systems Languages (CODASYL).
- COBOL was the first programming language whose use was mandated by the US Department of Defense
COBOL...

• English – like verbose syntax (Goal: Human readability – but didn’t really help)

• Largely ignored by the academic community

• And if you thought COBOL was dead…

Think again..

Object-oriented COBOL is a subset of COBOL 97, which is the fourth edition in the continuing evolution of ANSI/ISO standard COBOL

Louden, Chapter 2/Scott, Chapter 1

000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300
000400*
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
010000 DATA DIVISION.
010100 FILE SECTION.
010200 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400 DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500 DISPLAY "Hello world!" LINE 15 POSITION 10.
100600 STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800 EXIT.

Louden, Chapter 2/Scott, Chapter 1
ALGOL 60 (1958-1960)

- **ALGO**rithmic Language: general expressive language for describing algorithms
- Used widely in Europe and academia in USA
- Modern syntax: defined using BNF, free format, structure statements, with `begin/end` pairs
- Type declarations required for all variables

ALGOL60...

- Introduced recursion, call-by-name and call-by-value
- Required stack-based runtime environment
- Huge influence on later languages: Pascal, C, Module-2, Ada etc
Call-by-name

- Page 321 Louden
- The argument is not evaluated until its actual use (as a parameter) in the called procedure
- The name of the argument replaces the name of the parameter it corresponds to

LISP (1956-1962)

- The first functional language
- The first language to include garbage collection
- Intended for list processing and symbolic manipulation
- Syntax was radically different – lots of parentheses
- Efficiency not a huge concern. Ideas more important
- Still heavily used today for AI research and applications
- IDE
Reverse a list

(defun reverse (L)
  (if (null L) nil
       (list-append (reverse (rest L))
                   (list (first L)))))

Many new languages followed:

- **Algol68**: a theoretical triumph. A practical disaster. Still a very interesting language.
- **BASIC**: 1964. Distilled the most elementary ideas for the simplest computers. Still alive.
- **Simula67**: the first OO language. Way ahead of its time. But inefficient.
The 70s: simplicity and abstraction.

- C (Ritchie, 1972), Pascal (Wirth, 1971). Much simpler than 60s languages, but they add data structures a la COBOL and structured control a la Algol60. Added no new concepts to language design.
- C sometimes called middle layer (rather than high level) it is so close to underlying architecture.
- “ADT” languages CLU, Euclid, Mesa, examined the ways programs can be decomposed into independent units (1974-79).
- Scheme (Sussman, Steele, 1975): a “regularized” LISP.

The 80s: the rise of modularity and object-orientation.

- Modula-2 (Wirth, 1982). A not-so-successful sequel to Pascal. (After its time.)
- Smalltalk80 (Alan Kay). Advances the cause of OO programming.
- C++ (Stroustrup, 1980++). Shows that OO programming can be efficient.
The 90s: technology takes off

- Increasing need for big libraries, APIs.
- Java (Gosling, 1995). The first language to come with an API already developed.
- System programming becomes huge (Perl, Tcl, Javascript, VBScript, Python, PHP)
- Scripting languages tie together utilities, library components, and operating system commands into complete programs.

What’s next?

- C# (2000). Will it really replace Java?
- Where will Java be in 5 or 10 years? (Most platform-specific applications are still written in C++ or C.)
- Will a “new” language come along?
- What happened to Prolog (Colmerauer creator, 1972-1982)?
- Does it make any sense to try to predict?
Why are there so many programming languages?

- Evolution – we are constantly finding better ways to do things
- Special Purpose – many languages designed for a particular goal. Ex: Ruby less code to write, support for regular expressions
- Personal Preference
- Expressive power – ability to write clear, concise, maintainable code

Louden, Chapter 2/Scott, Chapter 1 29

Why are there so many programming languages? (cont)

- Ease of use for novice
- Ease of implementation. Pascal became popular largely because of simple, portable, free implementation
- Standardization. Pascal was abandoned because of missing several features which were implemented in different ways. such features as separate compilation, strings, static initialization, I/O)

Louden, Chapter 2/Scott, Chapter 1 30
Why are there so many programming languages? (cont)

• Excellent compilers and IDE. Fortran owes much to the excellent compilers for Fortran.

• Economics patronage and Inertia. Cobol is still used due to the economic investment in existing code.