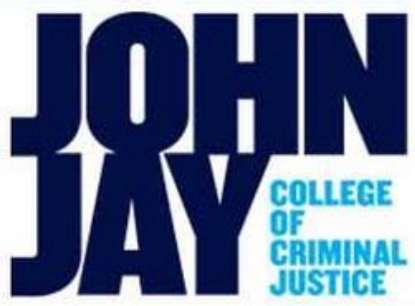


Programming Languages:

Lecture 2

Chapter 2: Evolution of the Major Programming Languages

Jinwoo Kim

jwkim@jjay.cuny.edu

Chapter 2 Topics

- History of Computers
- Zuse's Plankalkul
- Minimal Hardware Programming: Pseudocodes
- The IBM 704 and Fortran
- Functional Programming: LISP
- The First Step Toward Sophistication: ALGOL 60
- Computerizing Business Records: COBOL
- The Beginnings of Timesharing: BASIC

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Chapter 2 Topics (continued)

- Everything for Everybody: PL/I
- Two Early Dynamic Languages: APL and SNOBOL
- The Beginnings of Data Abstraction: SIMULA 67
- Orthogonal Design: ALGOL 68
- Some Early Descendants of the ALGOLs
- Programming Based on Logic: Prolog
- History's Largest Design Effort: Ada

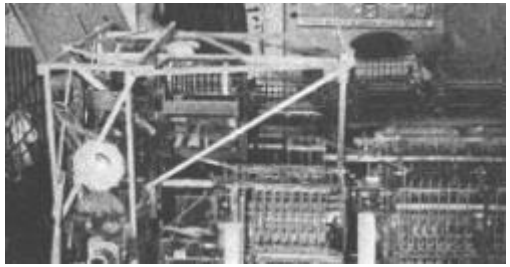
Chapter 2 Topics (continued)

- Object-Oriented Programming: Smalltalk
- Combining Imperative ad Object-Oriented Features: C++
- An Imperative-Based Object-Oriented Language: Java
- Scripting Languages: JavaScript, PHP, Python, and Ruby
- A C-Based Language for the New Millennium: C#
- Markup/Programming Hybrid Languages

History of Computers

- Who invented computers?
  - Contribution from many inventors
  - A computer is a complex piece of machinery made up of many parts, each of which can be considered a separate invention.

- In 1936, Konrad Zuse made a mechanical calculator using three basic elements: a control, a memory, and a calculator for the arithmetic and called it Z1, the first binary computer
  - First freely programmable computer
  - Konrad Zuse wrote the first algorithmic programming language called 'Plankalkül' in 1946, which he used to program his computers
  - He wrote the world's first chess-playing program using Plankalkül



Konrad Zuse's Z1 Circa 1938

History of Computers (Continued)

- Professor John Atanasoff and his graduate student Clifford Berry built the world's first electronic-digital computer at Iowa State University between 1939 and 1942
  - ABC computer
  - Several innovations in computing, including a binary system of arithmetic, parallel processing, and a separation of memory and computing functions

- In 1946, John Mauchly and J Presper Eckert developed the ENIAC I (**E**lectrical **N**umerical **I**ntegrator **A**nd **C**alculator)
  - By support from U.S. military
  - Considered as first modern computer
  - The ENIAC contained 17,468 vacuum tubes, along with 70,000 resistors, 10,000 capacitors, 1,500 relays, 6,000 manual switches and 5 million soldered joints
  - It covered 1800 square feet (167 square meters) of floor space, weighed 30 tons, consumed 160 kilowatts of electrical power

History of Computers (Continued)

- In 1951, John Presper Eckert & John W. Mauchly built first commercial computer called "UNIVAC"
  - By doing the research for their customer, United States Census Bureau
    - **The Bureau needed a new computer to deal with the exploding U.S. population (the beginning of the famous baby boom)**
  - Able to pick presidential winners (Eisenhower vs. Stevenson)

- In 1953, IBM enters into "The History of Computers" with IBM 701 EDPM Computer
  - In 1954, the first successful high level programming language, Fortran, was developed by John Backus & IBM

- In 1958, The Integrated Circuit, otherwise known as "The Chip", was developed by Jack Kilby & Robert Noyce

History of Computers (Continued)

- In 1964, IBM unveiled first "mainframe computer" with the System/360
  - Cost to develop: $5 billion ($30 billion in today's dollars)
    - **But the gamble paid off: company's revenue jumped from $3.2 billion the year it was introduced to $7.5 billion in 1970**
  - Major breakthrough in the technology and business worlds
  - Allowed companies to perform multiple tasks at the same time on a single machine
    - **Before then, a user would have to schedule time on the company computer to do a specific task, whether to process payroll or analyze business expenses**
  - Dominated computing industry until PC revolution in the 1980s

- In 1969, "ARPAnet", the origin of the Internet, was constructed
  - Packet-switching development
  - ARPA introduces network for defense and develops e-mail and US universities join network in 1970

- In 1971, Faggin, Hoff & Mazor made "Intel 4004 Computer Microprocessor"
  - The first microprocessor

History of Computers (Continued)

- In 1981, IBM introduced “IBM PC - Home Computer” and Microsoft revealed its “MS-DOS” Operating System
- In 1983, The first home computer with a GUI, graphical user interface, was developed by Apple
- In 1985, Microsoft begins the friendly war with Apple with its launch of Microsoft Windows operating system
- By the early 1990s, sales of mainframes, then IBM's main product, were dropping dramatically in the face of stiff competition from rivals such as Sun Microsystems
  - Also, instead of big boxes in the back room, companies turned to servers that connected PCs in a network
  - People predicted it will extinct in decades

History of Computers (Continued)

- In 1991, “Mosaic”, the first properly developed web-browser, takes Internet by storm

- We live in the age of “embedded computer” world now
  - Downshift of “center of gravity of computing”
  - But at the same time, IBM sells record number of mainframe computers
    - **IBM sold $4.2 billion worth of mainframes in 2003, up 6 percent from the previous year**

- So, we have a history of computers about 70 years!

Genealogy of Common Languages

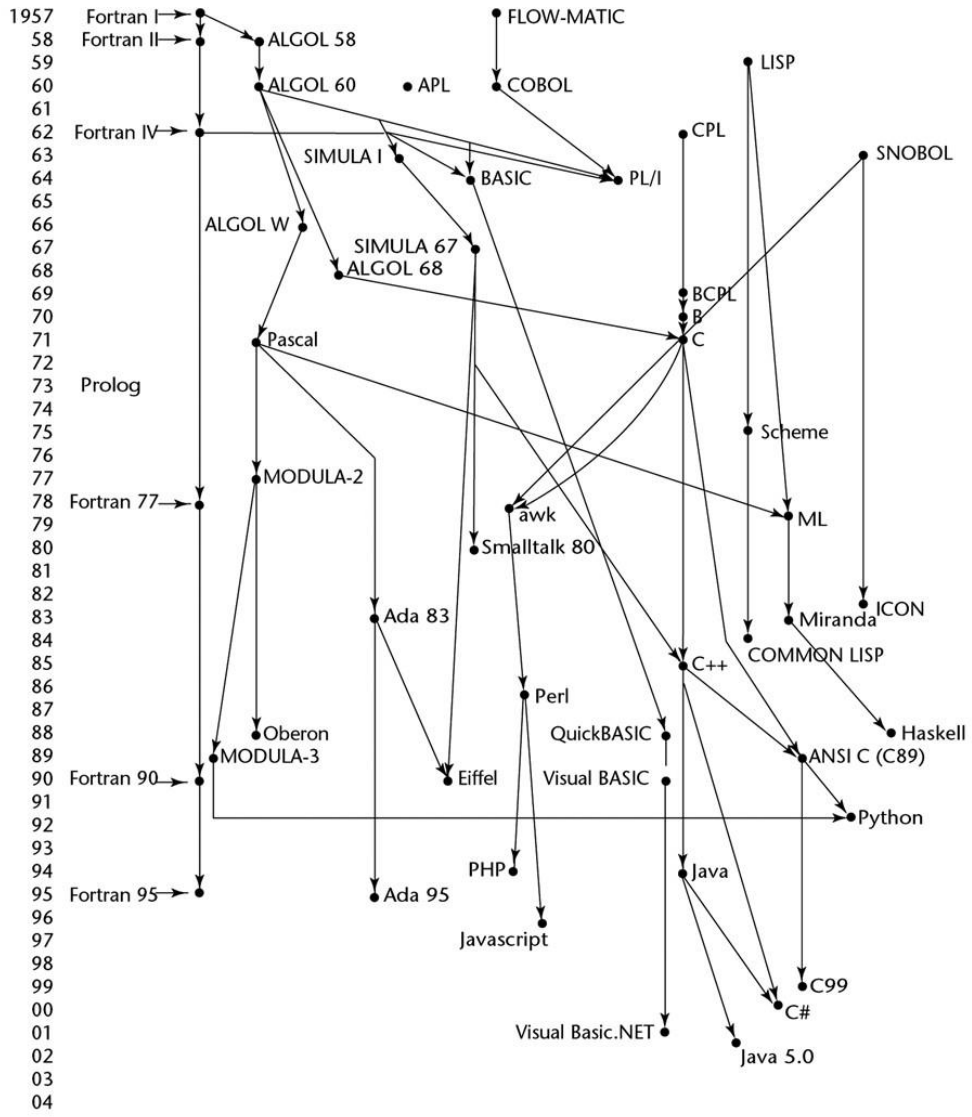


Figure 2.1 Genealogy of common high-level programming languages

Zuse's Plankalkül

- Never implemented
- Advanced data structures
  - floating point, arrays, records

Plankalkül Syntax

- An assignment statement to assign the expression A[4] + 1 to A[5]

```
     |   A + 1 => A
V    |   4          5          (subscripts)
S    |   1.n        1.n        (data types)
```

Minimal Hardware Programming: Pseudocodes

- What was wrong with using machine code?
  - Poor readability
  - Poor modifiability
  - Expression coding was tedious
  - Machine deficiencies--no indexing or floating point

Pseudocodes: Short Code

- Short Code developed by Mauchly in 1949 for BINAC computers
  - Expressions were coded, left to right
  - Example of operations:

```
01 –    06 abs value     1n (n+2)nd power
02 )    07 +             2n (n+2)nd root
03 =    08 pause         4n if <= n
04 /    09 (             58 print and tab
```

(Example)

X0 = SQRT(ABS(Y0))

would be coded as

00 X0 03 20 06 Y0

Pseudocodes: Speedcoding

- Speedcoding developed by Backus in 1954 for IBM 701
- Pseudo ops for arithmetic and math functions
  - Conditional and unconditional branching
  - Auto-increment registers for array access
  - Slow!
  - Only 700 words left for user program

Pseudocodes: Related Systems

- The UNIVAC Compiling System
  - Developed by a team led by Grace Hopper
  - Pseudocode expanded into machine code

- David J. Wheeler (Cambridge University)
  - developed a method of using blocks of re-locatable addresses to solve the problem of absolute addressing

IBM 704 and Fortran

- Fortran 0: 1954 - not implemented
  - John Backus and his IBM group wrote a report
    - **"IBM Mathematical Formula Translating System: Fortran"**
- Fortran I:1957
  - Designed for the new IBM 704, which had index registers and floating point hardware
  - Environment of development
    - **Computers were small and unreliable**
    - **Primary applications were scientific**
    - **No programming methodology or tools**
    - **Machine efficiency was most important**
      - **Speed of object code was the key**

Design Process of Fortran

- Impact of environment on design of Fortran I
  - No need for dynamic storage
  - Need good array handling and counting loops
  - No string handling, decimal arithmetic, or powerful input/output (commercial stuff)

Fortran I Overview

- First implemented version of Fortran I
  - Names could have up to six characters
    - **Up to just 2 characters in Fortran 0**
  - Post-test counting loop (`DO`)
    - **Do N1 Variable = first_value, last_value**
      - **Ex) nfac = 1**
      - **do 100 n=2, 10, 1**
      - **100 nfac = nfac*n**
  - Formatted I/O
  - User-defined subprograms
  - Three-way selection statement (arithmetic `IF`)
    - **If (arithmetic expression) N1, N2, N3**
  - No data typing statements
    - **Variables whose name starts with I, J, K, L, M, N were implicitly integer type**
      - **All others were implicitly floating point**

Fortran I Overview (continued)

- First implemented version of FORTRAN
  - Compiler released in April 1957, after 18 worker-years of effort
  - Code was very fast
  - Quickly became widely used
  - No separate compilation
    - **Any change in a program requires that entire program be recompiled**
  - Programs larger than 400 lines rarely compiled correctly
    - **Mainly due to poor reliability of 704**

Fortran II

- Distributed in 1958
  - Independent compilation
  - Fixed the bugs in Fortan I compilation system

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Fortran IV

- One of the most widely used programming language of its time
- Evolved during 1960-62
  - ANSI standard in 1966
    - **“Fortran 66”**
- Improvement over Fortran II
  - Explicit type declarations
  - Logical If construct
  - Subprogram names could be parameters
    - **You can pass subprorams as parameters to other subprograms**

Fortran 77

- Became the new standard in 1978
  - ANSI 1978

- Extra features
  - Character string handling
  - Logical loop control statement
  - **`IF-THEN-ELSE`** statement

Fortran 90

- ANSI, 1992

- Most significant changes from Fortran 77
  - Modules
  - Dynamic arrays
  - Pointers
  - Recursion
  - **`CASE`** statement
  - Parameter type checking

Latest versions of Fortran

- Fortran 95
  - relatively minor additions, plus some deletions
- Fortran 2003
  - ditto

99 Bottles of Beer in Fortran

```
      program ninetyninebottles
      integer bottles
      bottles = 99
 1    format (I2, A)
 2    format (A)
 3    format (I2, A, /)
 4    format (A, /)
 10   write (*,1) bottles, ' bottles of beer on the wall,'
      write (*,1) bottles, ' bottles of beer.'
      write (*,2) 'Take one down, pass it around...'
      if (bottles - 1 .gt. 1) then
         write (*,3) bottles - 1, ' bottles of beer on the wall.'
      else
         write (*,3) bottles - 1, ' bottle of beer on the wall.'
      end if
      bottles = bottles - 1
      if (bottles - 1) 30, 20, 10
*     Last verse
 20   write (*,1) bottles, ' bottle of beer on the wall,'
      write (*,1) bottles, ' bottle of beer.'
      write (*,2) 'Take one down, pass it around...'
      write (*,4) 'No bottles of beer on the wall.'
 30   stop
      end
```

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

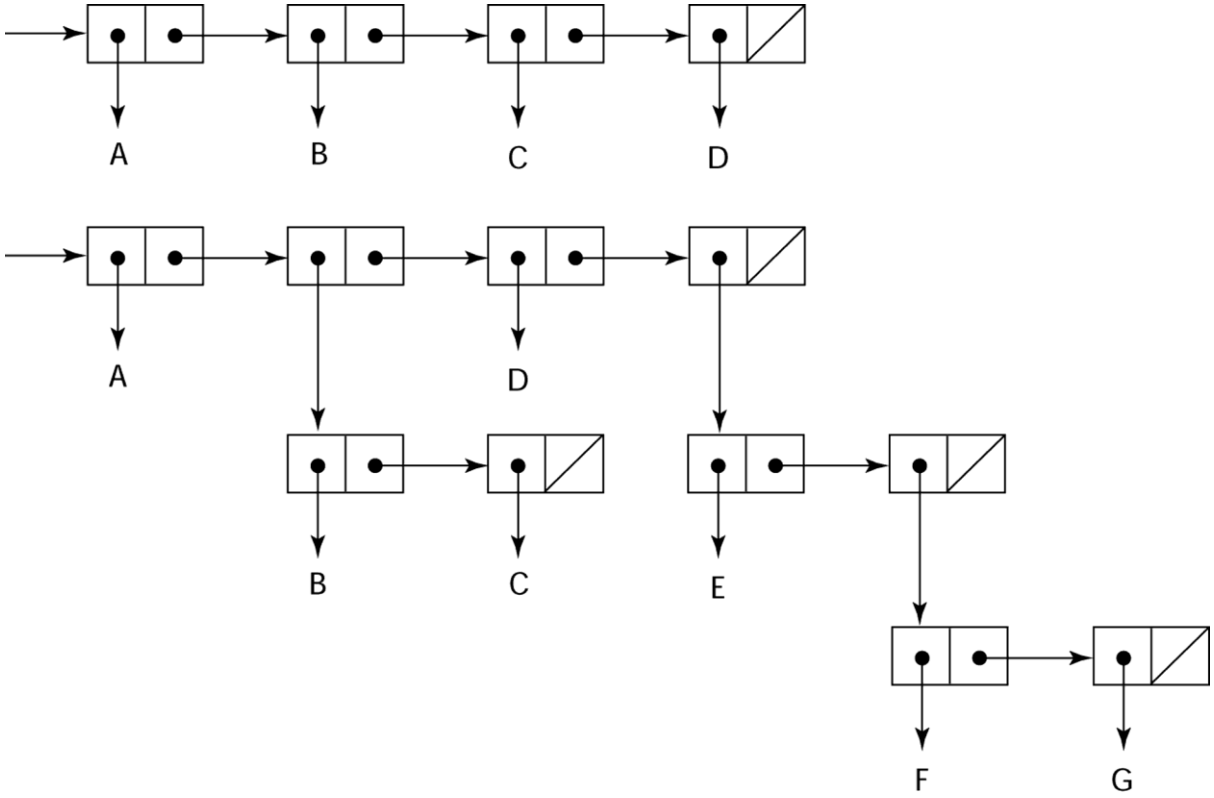
Fortran Evaluation

- Highly optimizing compilers (all versions before 90)
  - Types and storage of all variables are fixed before run time
    - **No new variables or space can be allocated dynamically**
  - Sacrifice of flexibility to simplicity and efficiency
- Dramatically changed forever the way computers are used
  - First widely used high level language!
- Characterized as the *lingua franca* of the computing world

Functional Programming: LISP

- LISt Processing language
  - Designed at MIT by McCarthy
- AI research needed a language to
  - Process data in lists (rather than arrays)
  - Symbolic computation (rather than numeric)
- Only two data types: atoms and lists
- Syntax is based on *lambda calculus*

Representation of Two LISP Lists



Representing the lists `(A B C D)` and `(A (B C) D (E (F G)))`

99 Bottles of Beer in LISP

```
(defun bottles-of-bier (n)
  (case n
   (0
    '(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 w
   (1
    '(1 bottle of beer on the wall 1 bottle of beer.
        Take one down and pass it around no more bottles of beer on th
        ,@(bottles-of-bier 0)))
   (2
    '(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.
       ,@(bottles-of-bier 1)))
   (t
    '(,n bottles of beer on the wall ,n bottles of beer.
         Take one down and pass it around
         ,(1- n) bottles of beer on the wall.
         ,@(bottles-of-bier (1- n))))))
```

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

LISP Evaluation

- Pioneered functional programming
  - No need for variables or assignment
  - Control via recursion and conditional expressions
- Still the dominant language for AI
- COMMON LISP and Scheme are contemporary dialects of LISP
- ML, Miranda, and Haskell are related languages

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Scheme

- Developed at MIT in mid 1970s
- Small
- Extensive use of static scoping
- Functions as first-class entities
- Simple syntax (and small size) make it ideal for educational applications

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

COMMON LISP

- An effort to combine features of several dialects of LISP into a single language

- Large, complex

The First Step Toward Sophistication: ALGOL 60

- Environment of development
  - FORTRAN had (barely) arrived for IBM 70x
  - Many other languages were being developed, all for specific machines
  - No portable language; all were machine-dependent
  - No universal language for communicating algorithms

- ALGOL 60 was the result of efforts to design a universal language

Early Design Process

- ACM and GAMM met for four days for design (May 27 to June 1, 1958)

- Goals of the language
  - Close to mathematical notation
  - Good for describing algorithms
  - Must be translatable to machine code

ALGOL 58

- Concept of type was formalized
- Names could be any length
- Arrays could have any number of subscripts
- Parameters were separated by mode (in & out)
- Subscripts were placed in brackets
- Compound statements (**`begin ... end`**)
- Semicolon as a statement separator
- Assignment operator was :=
- **`if`** had an **`else-if`** clause
- No I/O - “would make it machine dependent”

ALGOL 58 Implementation

- Not meant to be implemented, but variations of it were (MAD, JOVIAL)
- Although IBM was initially enthusiastic, all support was dropped by mid 1959

ALGOL 60 Overview

- Modified ALGOL 58 at 6-day meeting in Paris

- New features
  - Block structure (local scope)
  - Two parameter passing methods
  - Subprogram recursion
  - Stack-dynamic arrays
  - Still no I/O and no string handling

ALGOL 60 Evaluation

- Successes
  - It was the standard way to publish algorithms for over 20 years
  - All subsequent imperative languages are based on it
  - First machine-independent language
  - First language whose syntax was formally defined (BNF)

ALGOL 60 Evaluation (continued)

- Failure
  - Never widely used, especially in U.S.
  - Reasons
    - **Lack of I/O and the character set made programs non-portable**
    - **Too flexible--hard to implement**
    - **Entrenchment of Fortran**
    - **Formal syntax description**
    - **Lack of support from IBM**

Computerizing Business Records: COBOL

- Environment of development
  - UNIVAC was beginning to use FLOW-MATIC
  - USAF was beginning to use AIMACO
  - IBM was developing COMTRAN

COBOL Historical Background

- Based on FLOW-MATIC

- FLOW-MATIC features
  - Names up to 12 characters, with embedded hyphens
  - English names for arithmetic operators (no arithmetic expressions)
  - Data and code were completely separate
  - Verbs were first word in every statement

COBOL Design Process

- First Design Meeting (Pentagon) - May 1959
- Design goals
  - Must look like simple English
  - Must be easy to use, even if that means it will be less powerful
  - Must broaden the base of computer users
  - Must not be biased by current compiler problems
- Design committee members were all from computer manufacturers and DoD branches
- Design Problems: arithmetic expressions? subscripts? Fights among manufacturers

COBOL Evaluation

- Contributions
  - First macro facility in a high-level language
  - Hierarchical data structures (records)
  - Nested selection statements
  - Long names (up to 30 characters), with hyphens
  - Separate data division

COBOL Evaluation (Continued)

- Added type declarations, record types, file manipulation

```
data division.
file section.
*    describe the input file
fd   employee-file-in
               label records standard
               block contains 5 records
               record contains 31 characters
               data record is employee-record-in.
01   employee-record-in.
     02   employee-name-in   pic x(20).
     02   employee-rate-in   pic 9(3)v99.
     02   employee-hours-in  pic 9(3)v99.
     02   line-feed-in       pic x(1).
```

COBOL: DoD Influence

- First language required by DoD
  - would have failed without DoD
- Still the most widely used business applications language

The Beginning of Timesharing: BASIC

- Designed by Kemeny & Kurtz at Dartmouth
- Design Goals:
  - Easy to learn and use for non-science students
  - Must be “pleasant and friendly”
  - Fast turnaround for homework
  - Free and private access
  - User time is more important than computer time
- Current popular dialect: Visual BASIC
- First widely used language with time sharing

Everything for Everybody: PL/I

- Designed by IBM and SHARE

- Computing situation in 1964 (IBM's point of view)
  - Scientific computing
    - **IBM 1620 and 7090 computers**
    - **FORTRAN**
    - **SHARE user group**
  - Business computing
    - **IBM 1401, 7080 computers**
    - **COBOL**
    - **GUIDE user group**

PL/I: Background

- By 1963
  - Scientific users began to need more elaborate I/O, like COBOL had; business users began to need floating point and arrays
  - It looked like many shops would begin to need two kinds of computers, languages, and support staff--too costly
- The obvious solution
  - Build a new computer to do both kinds of applications
  - Design a new language to do both kinds of applications

PL/I: Design Process

- Designed in five months by the 3 X 3 Committee
  - Three members from IBM, three members from SHARE
- Initial concept
  - An extension of Fortran IV
- Initially called NPL (New Programming Language)
- Name changed to PL/I in 1965

PL/I: Evaluation

- PL/I contributions
  - First unit-level concurrency
  - First exception handling
  - Switch-selectable recursion
  - First pointer data type
  - First array cross sections
- Concerns
  - Many new features were poorly designed
  - Too large and too complex

Two Early Dynamic Languages: APL and SNOBOL

- Characterized by dynamic typing and dynamic storage allocation
- Variables are untyped
  - A variable acquires a type when it is assigned a value
- Storage is allocated to a variable when it is assigned a value

APL: A Programming Language

- Designed as a hardware description language at IBM by Ken Iverson around 1960
  - Highly expressive (many operators, for both scalars and arrays of various dimensions)
  - Programs are very difficult to read
- Still in use; minimal changes

APL: Powerful Operators, interactive language, custom character set

```
[0]    Z←GAUSSRAND N;B;F;M;P;Q;R
[1]    ⍝Returns ⍵ random numbers having a Gaussian normal distribution
[2]    ⍝ (with mean 0 and variance 1)  Uses the Box-Muller method.
[3]    ⍝ See Numerical Recipes in C, pg. 289.
[4]    ⍝
[5]    Z←⍳0
[6]    M←¯1+2*31          ⍝ largest integer
[7]   L1:Q←N-⍴Z           ⍝ how many more we need
[8]    →(Q≤0)/L2          ⍝ quit if none
[9]    Q←⌈1.3×Q÷2         ⍝ approx num points needed
[10]   P←¯1+(2÷M-1)×¯1+?(Q,2)⍴M  ⍝ random points in -1 to 1 square
[11]   R←+/P×P            ⍝ distance from origin squared
[12]   B←(R≠0)∧R<1
[13]   R←B/R ⋄ P←B⌿P      ⍝ points within unit circle
[14]   F←(¯2×(⊛R)÷R)*.5
[15]   Z←Z,,P×F,[1.5]F
[16]   →L1
[17]  L2:Z←N↑Z
[18]   ⍝ ArchDate: 12/16/1997 16:20:23.170
```

"Emoticons for Mathematicians"

Source: Jim Weigang, http://www.chilton.com/~jimw/gsrand.html

At right: Datamedia APL Keyboard

99 Bottles of Beer in APL

```
⍝ APL (A Programming Language)

⍝ Program written by JT. Taylor, www.jttaylor.net

T1←98↑[1]⍬⍴1 99⍴⍳99

T4←⍬⍴1 98⍴⍳98

T1,(98 30⍴' BOTTLES OF BEER ON THE WALL, '),T1,
(98 47⍴'BOTTLES OF BEER, TAKE ONE DOWN, PASS IT
AROUND,'),T4,(98 28⍴'BOTTLES OF BEER ON THE
WALL ,')

'1 BOTTLE OF BEER ON THE WALL, 1 BOTTLE OF BEER,
TAKE IT DOWN, PASS IT AROUND, NO BOTTLES OF BEER
ON THE WALL.'
```

SNOBOL

- Designed as a string manipulation language at Bell Labs by Farber, Griswold, and Polensky
- Powerful operators for string pattern matching
- Slower than alternative languages (and thus no longer used for writing editors)
- Stilled used for certain text processing tasks

The Beginning of Data Abstraction: SIMULA 67

- Designed primarily for system simulation in Norway by Nygaard and Dahl
- Based on ALGOL 60 and SIMULA I
- Primary Contributions
  - Co-routines - a kind of subprogram
  - Implemented in a structure called a class
  - Classes are the basis for data abstraction
  - Classes are structures that include both local data and functionality

Orthogonal Design: ALGOL 68

- From the continued development of ALGOL 60 but not a superset of that language
- Source of several new ideas (even though the language itself never achieved widespread use)
- Design is based on the concept of orthogonality
  - A few principle concepts, few combining mechanisms

ALGOL 68 Evaluation

- Contributions
  - User-defined data structures
  - Reference types
  - Dynamic arrays (called flex arrays)

- Comments
  - Less usage than ALGOL 60
  - Had strong influence on subsequent languages, especially Pascal, C, and Ada

Early Descendants of ALGOLs

- ALGOL languages impacted all imperative languages
  - Pascal
  - C
  - Modula/Modula 2
  - Ada
  - Oberon
  - C++/Java
  - Perl (to some extent)

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Pascal - 1971

- Developed by Wirth (a member of the ALGOL 68 committee)
- Designed for teaching structured programming
- Small, simple, nothing really new
- Largest impact on teaching programming
  - From mid-1970s until the late 1990s, it was the most widely used language for teaching programming

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

C - 1972

- Designed for systems programming (at Bell Labs by Dennis Richie)
- Evolved primarily from BCLP, B, but also ALGOL 68
- Powerful set of operators, but poor type checking
- Initially spread through UNIX
- Many areas of application

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

99 Bottles of Beer in C

```
#define MAXBEER 99
void chug(int beers);

int main()
{
  int beers;
  for(beers = MAXBEER; beers; chug(beers--)) ;
  puts("\nTime to buy more beer!\n");
  return 0;
}

void chug(int beers)
{
  char howmany[8], *s;
  s = beers != 1 ? "s" : "";
  printf("%d bottle%s of beer on the wall,\n", beers, s);
  printf("%d bottle%s of beeeeer . . . ,\n", beers, s);
  printf("Take one down, pass it around,\n");
  if (--beers) sprintf(howmany, "%d", beers);
  else strcpy(howmany, "No more");
  s = beers != 1 ? "s" : "";
  printf("%s bottle%s of beer on the wall.\n", howmany, s);
}
```

Perl

- Related to ALGOL only through C
- A scripting language
  - A *script* (file) contains instructions to be executed
  - Other examples: sh, awk, tcl/tk
- Developed by Larry Wall
- Perl variables are statically typed and implicitly declared
  - Three distinctive namespaces, denoted by the first character of a variable's name
- Powerful but somewhat dangerous
- Widely used as a general purpose language

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Programming Based on Logic: Prolog

- Developed, by Comerauer and Roussel (University of Aix-Marseille), with help from Kowalski (University of Edinburgh)
- Based on formal logic
- Non-procedural
- Can be summarized as being an intelligent database system that uses an inferencing process to infer the truth of given queries
- Highly inefficient, small application areas

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

99 Bottles of Beer in Prolog

```
bottles :-
    bottles(99).

bottles(1) :-
    write('1 bottle of beer on the wall, 1 bottle of beer,'), nl,
    write('Take one down, and pass it around,'), nl,
    write('Now they are all gone.'), nl,!.
bottles(X) :-
    write(X), write(' bottles of beer on the wall,'), nl,
    write(X), write(' bottles of beer,'), nl,
    write('Take one down and pass it around,'), nl,
    NX is X - 1,
    write(NX), write(' bottles of beer on the wall.'), nl, nl,
    bottles(NX).
```

History's Largest Design Effort: Ada

- Huge design effort, involving hundreds of people, much money, and about eight years
  - Strawman requirements (April 1975)
  - Woodman requirements (August 1975)
  - Tinman requirements (1976)
  - Ironman equipments (1977)
  - Steelman requirements (1978)
- Named Ada after Augusta Ada Byron, known as being the first programmer

Ada Evaluation

- Contributions
  - Packages - support for data abstraction
  - Exception handling - elaborate
  - Generic program units
  - Concurrency - through the tasking model

- Comments
  - Competitive design
  - Included all that was then known about software engineering and language design
  - First compilers were very difficult; the first really usable compiler came nearly five years after the language design was completed

Ada 95

- Ada 95 (began in 1988)
  - Support for OOP through type derivation
  - Better control mechanisms for shared data
  - New concurrency features
  - More flexible libraries

- Popularity suffered because the DoD no longer requires its use but also because of popularity of C++

Object-Oriented Programming: Smalltalk

- Developed at Xerox PARC, initially by Alan Kay, later by Adele Goldberg
- First full implementation of an object-oriented language (data abstraction, inheritance, and dynamic type binding)
- Pioneered the graphical user interface design
- Promoted OOP

Combining Imperative and Object-Oriented Programming: C++

- Developed at Bell Labs by Stroustrup in 1980
- Evolved from C and SIMULA 67
- Facilities for object-oriented programming, taken partially from SIMULA 67
- Provides exception handling
- A large and complex language, in part because it supports both procedural and OO programming
- Rapidly grew in popularity, along with OOP
- ANSI standard approved in November 1997
- Microsoft's version (released with .NET in 2002): Managed C++
  - delegates, interfaces, no multiple inheritance

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Related OOP Languages

- Eiffel (designed by Bertrand Meyer - 1992)
  - Not directly derived from any other language
  - Smaller and simpler than C++, but still has most of the power
  - Lacked popularity of C++ because many C++ enthusiasts were already C programmers
- Delphi (Borland)
  - Pascal plus features to support OOP
  - More elegant and safer than C++

An Imperative-Based Object-Oriented Language: Java

- Developed at Sun in the early 1990s
  - C and C++ were not satisfactory for embedded electronic devices
- Based on C++
  - Significantly simplified (does not include **`struct`**, **`union`**, **`enum`**, pointer arithmetic, and half of the assignment coercions of C++)
  - Supports *only* OOP
  - Has references, but not pointers
  - Includes support for applets and a form of concurrency

99 Bottles of Beer in Java

```
class Bottles {
  public static void main(String args[]) {
    String s = "s";
    for (int beers=99; beers>-1;) {
      System.out.print(beers+" bottle"+s+" of beer on the wall, ");
      System.out.println(beers + " bottle" + s + " of beer, ");
      if (beers==0) {
        System.out.print("Go to the store, buy some more, ");
        System.out.println("99 bottles of beer on the wall.\n");
        System.exit(0);
      } else
        System.out.print("Take one down, pass it around, ");
      s = (--beers == 1)?"":"s";
      System.out.println(beers+" bottle"+s+" of beer on the wall.\n");
    }
  }
}
```

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Java Evaluation

- Eliminated unsafe features of C++
- Concurrency features
- Libraries for applets, GUIs, database access
- Portable: Java Virtual Machine concept, JIT compilers
- Widely used for WWW pages
- Use for other areas increased faster than any other language
- Most recent version, 5.0, released in 2004

Scripting Languages for the Web

- JavaScript
  - Began at Netscape, but later became a joint venture of Netscape and Sun Microsystems
  - Used in Web programming (client side) to create dynamic HTML documents
  - Purely interpreted
  - Related to Java only through similar syntax
- PHP
  - PHP: Hypertext Preprocessor
  - A server-side HTML-embedded scripting language, often used for form processing and database access through the Web
  - Purely interpreted
- Python
  - An OO interpreted scripting language
  - Type checked but dynamically typed
  - Used for CGI programming and form processing
  - Dynamically typed, but type checked
  - Supports lists, tuples, and hashes

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Scripting Languages for the Web

- Ruby
  - Designed in Japan by Yukihiro Matsumoto (a.k.a, “Matz”)
  - Began as a replacement for Perl and Python
  - A pure object-oriented scripting language
    - **All data are objects**
  - Most operators are implemented as methods, which can be redefined by user code
  - Purely interpreted

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

99 Bottles of Beer in Python

```
for quant in range(99, 0, -1):
    if quant > 1:
        print quant, "bottles of beer on the wall,", \
              quant, "bottles of beer."
        if quant > 2:
            suffix = str(quant - 1) + " bottles of beer on the wall."
        else:
            suffix = "1 bottle of beer on the wall."
    elif quant == 1:
        print "1 bottle of beer on the wall, 1 bottle of beer."
        suffix = "no more beer on the wall!"
    print "Take one down, pass it around,", suffix
    print ""
```

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

A C-Based Language for the New Millennium: C#

- Part of the .NET development platform
- Based on C++ , Java, and Delphi
- Provides a language for component-based software development
- All .NET languages (C#, Visual BASIC.NET, Managed C++, J#.NET, and Jscript.NET) use Common Type System (CTS), which provides a common class library
- Likely to become widely used

Markup/Programming Hybrid Languages

- XSLT
  - eXtensible Markup Language (XML): a metamarkup language
  - eXtensible Stylesheet Language Transformation (XSTL) transforms XML documents for display
  - Programming constructs (e.g., looping)

- JSP
  - Java Server Pages: a collection of technologies to support dynamic Web documents
  - servlet: a Java program that resides on a Web server; servlet's output is displayed by the browser

JOHN JAY COLLEGE OF CRIMINAL JUSTICE

Summary

- Development, development environment, and evaluation of a number of important programming languages
- Perspective into current issues in language design

Homework #1

- Read articles introduced in this lecture
  - On the design of programming languages
    - http://jjcweb.jjay.cuny.edu/~jwkim/class/csci374-fall-24/PLHistoryGoodDesign.pdf

- Problem Solving (Homework 1)
  - Use problems on class homepage
  - No late homework will be accepted
  - How to hand in?
    - **By email attachment**
    - **Use text editor** (**exception: pictures, etc)**
    - **CC yourself with the proper subject line**
      - **Ex: CSCI 374-01, hw1**