Chapter 2

Evolution of
the Major Programming Languages

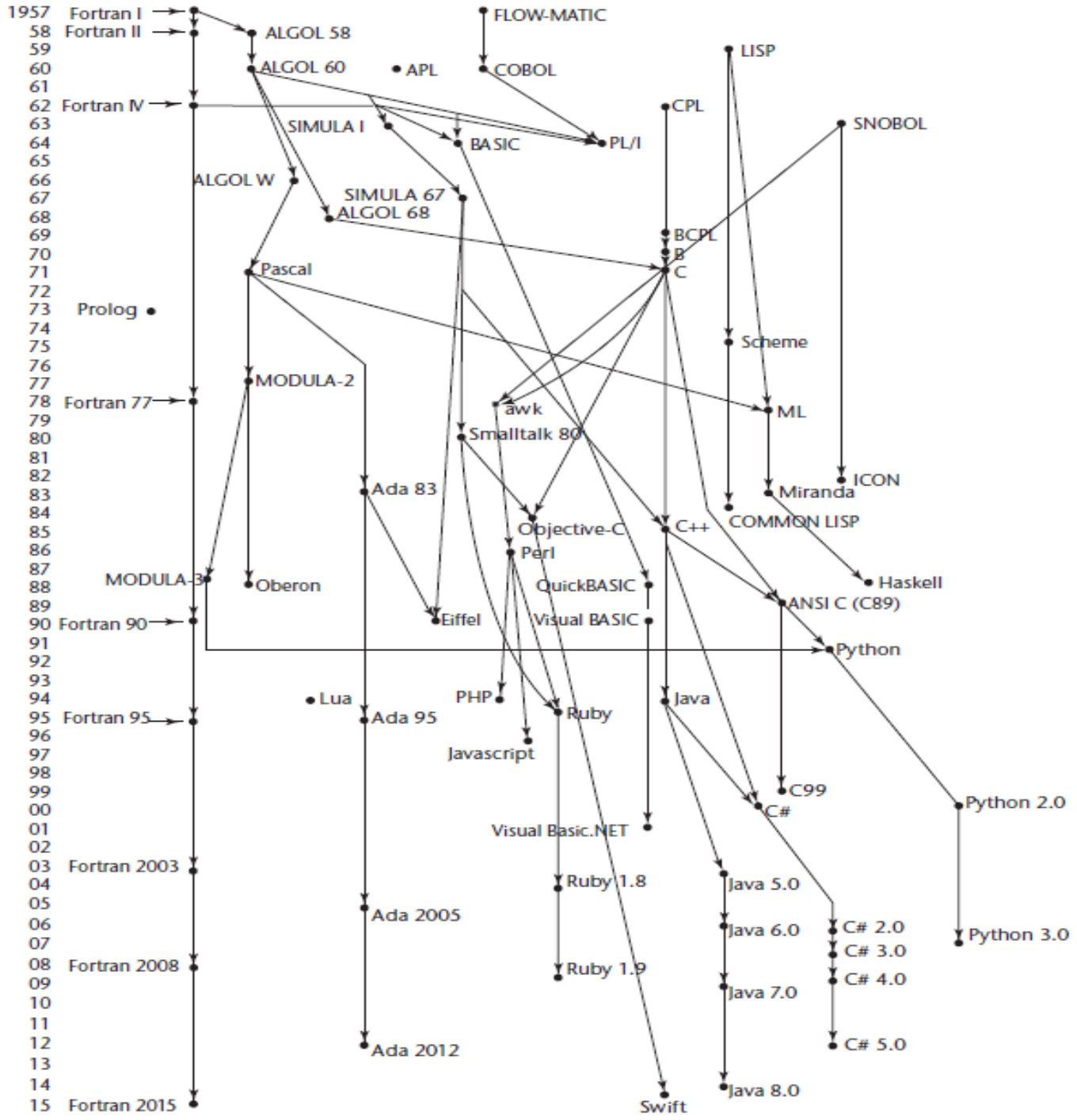
Topics

- Zuse's Plankalkül
- 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
- Everything for Everybody: PL/I
- Two Early Dynamic Languages: APL and SNOBOL
- The Beginnings of Data Abstraction: SIMULA 67
- Orthogonal Design: ALGOL 68

Topics (cont.)

- Some Early Descendants of the ALGOLs
- Programming Based on Logic: Prolog
- History's Largest Design Effort: Ada
- Object-Oriented Programming: Smalltalk
- Combining Imperative ad Object-Oriented Features: C++
- An Imperative-Based Object-Oriented Language: Java
- Scripting Languages
- The Flagship .NET Language: C#
- Markup/Programming Hybrid Languages

Genealogy of Common Languages



1. Zuse's Plankalkül

- Plankalkül means program calculus.
- a PL defined by German scientist Konrad Zuse ("Tsoo-zuh").
- Designed in 1945, but not published until 1972
- Never implemented
- Advanced data structures
  - the single bit → integer, floating-point types
  - floating point, arrays, records (≈ `structs` in C)
  - An iterative statement (≈ `for` in Ada), etc.
- Invariants
  - mathematical expressions that would be ture during execution (≈ Assertions of Java and in Axiomatic semantics)

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
                              – an integer of n bits)
```

2. Minimal Hardware Programming: Pseudocodes

- Difficulty of using machine code -- #18@chap.1
  - Poor readability
  - Poor modifiability

    e.g.) absolute addressing makes program modification tedious and error-prone.
  - Expression coding was tedious
  - Machine deficiencies -- no indexing or floating-point

Pseudocodes: Short Code

- Short Code developed by Mauchly in 1949 for BINAC computers

  → UNIVAC I computer (1 word = 72 bits = 12 6-bit bytes)

  – Expressions were coded, left to right

  – Example of operation codes:

```
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: the statement

  `X0 = SQRT(ABS(Y0))` where `X0,Y0` are variables.

  would be coded in a word 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 the 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

3. IBM 704 and Fortran

- Fortran 0: 1954 - not implemented
- Fortran I: 1957
  - Designed for the new IBM 704, which had index registers and floating-point hardware.

    - This led to the idea of compiled programming languages because there was no place to hide the cost of interpretation (no floating-point software)
  - Environment of development
    - Computers were small and unreliable
    - Applications were scientific
    - No programming methodology or tools
    - Machine efficiency was the most important concern

Design Process of Fortran

- Impact of environment on the 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 (for business software)

Fortran I Overview

- First implemented version of Fortran
  - Names could have up to six characters
  - Post-test counting loop (**DO**)
  - Formatted I/O
  - User-defined subprograms
  - Three-way selection statement (arithmetic **IF**)
  - No data typing statements
  - No separate compilation
  - Compiler released in April 1957, after 18 worker-years of effort
  - Programs larger than 400 lines rarely compiled correctly, mainly due to poor reliability of 704
  - Code was very fast
  - Quickly became widely used -- in High Performance Computing

Fortran II

- Distributed in 1958
  - Independent compilation
  - Fixed the bugs

Fortran IV

- Evolved during 1960-62
- Explicit type declarations
- Logical selection statement
- Subprogram names could be parameters
- ANSI standard in 1966

Fortran 77

- Became the new *standard* in 1978
  - Character string handling
  - Logical loop control statement
  - `IF-THEN-ELSE` statement

Fortran 90

- 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:
  - support for OOP, procedure pointers, interoperability with C
- Fortran 2008:
  - blocks for local scopes, co-arrays, `Do Concurrent`
- Fortran 2015:
- Fortran 2018
- Fortran 2023 – the latest version

Example: Fortran 95

```
! Fortran 95 Example program
!  Input:   An integer, List_Len, where List_Len is less
!           than 100, followed by List_Len-Integer values
!  Output: The number of input values that are greater
!          than the average of all input values
Implicit none
Integer Dimension(99) :: Int_List
Integer :: List_Len, Counter, Sum, Average, Result
Result= 0
Sum = 0
Read *, List_Len
If ((List_Len > 0) .AND. (List_Len < 100)) Then
! Read input data into an array and compute its sum
   Do Counter = 1, List_Len
      Read *, Int_List(Counter)
      Sum = Sum + Int_List(Counter)
   End Do

 ! Compute the average
    Average = Sum / List_Len
 ! Count the values that are greater than the average
   Do Counter = 1, List_Len
      If (Int_List(Counter) > Average) Then
         Result = Result + 1
      End If
   End Do
 ! Print the result
    Print *, 'Number of values > Average is:', Result
 Else
    Print *, 'Error - list length value is not legal'
 End If
 End Program Example
```

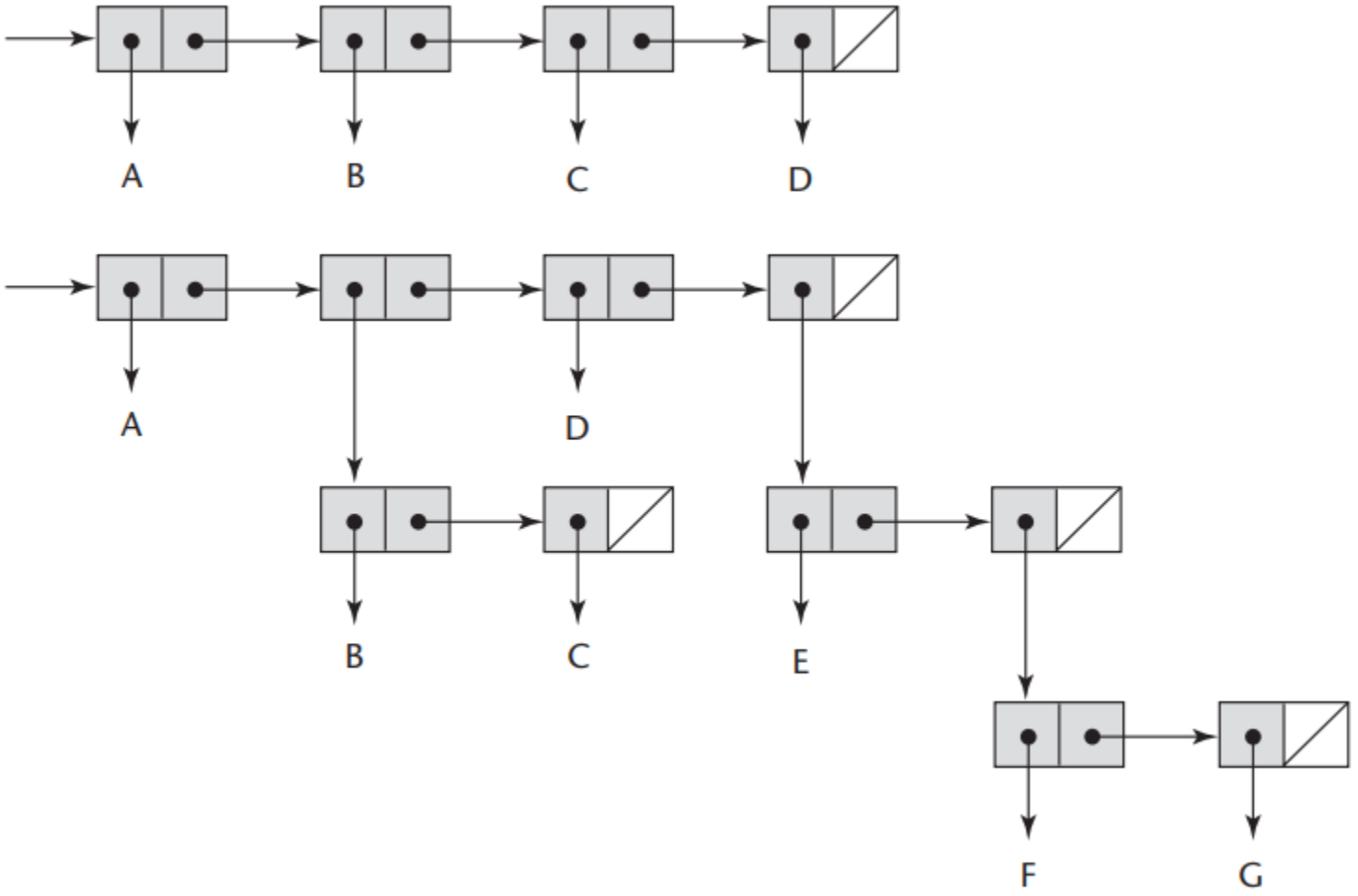
Fortran Evaluation

- Highly optimizing compilers (all versions before 90)
  - Types and storage of all variables are fixed before run time.
- Dramatically changed the way computers are used.

4. Functional Programming (FP): 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)))**

Example: Lisp program

```
; Lisp Example function
; The following code defines a Lisp predicate function
; that takes two lists as arguments and returns True
; if the two lists are equal, and NIL (false) otherwise
   (DEFUN equal_lists (lis1 lis2)
    (COND
      ((ATOM lis1) (EQ lis1 lis2))
      ((ATOM lis2) NIL)
      (equal_lists (CAR lis1) (CAR lis2))
                   (equal_lists (CDR lis1) (CDR lis2)))
      (T NIL)
    )
)
```

condition

If lis1 is ATOM then return EQ(lis1, lis2)

action

(function parameter-list)

```
defun equal_lists(lis1, lis2):
    if ATOM(lis1):
        return lis = lis2              // return true/false
    if ATOM(lis2):
        return Nil
    if equal_lists(lis1[0], lis2[0]):
        equal_lists(lis1[1:], lis2[1:])
    else:
        return Nil
```

Lisp Evaluation

- Pioneered functional programming
  - No need for variables or assignment
  - Control via recursion and conditional expressions
- Still the dominant language for (classic) AI.
- Common Lisp, Scheme : contemporary dialects of Lisp.
- Other FP Languages: ML, Haskell, and F#

  -- use very different syntax from Lisp.

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*

Common Lisp

- An effort to combine features of several dialects of Lisp into a single language
- Large, complex, used in industry for some large applications

5. 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)
  - ACM: Association for Computing Machinery
  - GAMM: German Association for Applied Mathematics & Mechanics
- 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
- 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)
- 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

Example: ALGOL 60

```
comment ALGOL 60 Example Program
 Input:  An integer, listlen, where listlen is less than
         100, followed by listlen-integer values
 Output: The number of input values that are greater than
         the average of all the input values ;
begin
  integer array intlist [1:99];
  integer listlen, counter, sum, average, result;
   sum := 0;
   result := 0;
   readint (listlen);
   if (listlen > 0) ^ (listlen < 100) then
     begin
 comment Read input into an array and compute the average;
     for counter := 1 step 1 until listlen do
       begin
       readint (intlist[counter]);
       sum := sum + intlist[counter]
       end;
 comment Compute the average;
     average := sum / listlen;
 comment Count the input values that are > average;
     for counter := 1 step 1 until listlen do
       if intlist[counter] > average
         then result := result + 1;
 comment Print result;
     printstring("The number of values > average is:");
     printint (result)
     end
   else
     printstring ("Error-input list length is not legal";
 end
```

6. Computerizing Business Records: COBOL

- Environment of development
  - UNIVAC was beginning to use FLOW-MATIC
    - 1st English-like data processing language
  - USAF was beginning to use AIMACO
  - IBM was developing COMTRAN
- 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
    - The first word in every statement was a verb

```
ADD 1 TO x
ADD 1, a, b TO x ROUNDED, y, z ROUNDED

ADD a, b TO c
    ON SIZE ERROR
        DISPLAY "Error"
END-ADD

ADD a TO b
    NOT SIZE ERROR
        DISPLAY "No error"
    ON SIZE ERROR
        DISPLAY "Error"
```

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: DoD Influence

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

Example: COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRODUCE-REORDER-LISTING.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. DEC-VAX.
OBJECT-COMPUTER. DEC-VAX.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT BAL-FWD-FILE ASSIGN TO READER.
    SELECT REORDER-LISTING ASSIGN TO LOCAL-PRINTER.

DATA DIVISION.
FILE SECTION.
FD  BAL-FWD-FILE
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 80 CHARACTERS.

01  BAL-FWD-CARD.
    02 BAL-ITEM-NO          PICTURE IS 9(5).
    02 BAL-ITEM-DESC        PICTURE IS X(20).
    02 FILLER               PICTURE IS X(5).
    02 BAL-UNIT-PRICE       PICTURE IS 999V99.
    02 BAL-REORDER-POINT    PICTURE IS 9(5).
    02 BAL-ON-HAND          PICTURE IS 9(5).
    02 BAL-ON-ORDER         PICTURE IS 9(5).
    02 FILLER               PICTURE IS X(30).
FD  REORDER-LISTING
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 132 CHARACTERS.

01  REORDER-LINE.
    02 RL-ITEM-NO           PICTURE IS Z(5).
    02 FILLER               PICTURE IS X(5).
    02 RL-ITEM-DESC         PICTURE IS X(20).
    02 FILLER               PICTURE IS X(5).
    02 RL-UNIT-PRICE        PICTURE IS ZZZ.99.
    02 FILLER               PICTURE IS X(5).
    02 RL-AVAILABLE-STOCK   PICTURE IS Z(5).
    02 FILLER               PICTURE IS X(5).
    02 RL-REORDER-POINT     PICTURE IS Z(5).
    02 FILLER               PICTURE IS X(71).

WORKING-STORAGE SECTION.
01  SWITCHES.
    02 CARD-EOF-SWITCH      PICTURE IS X.
01  WORK-FIELDS.
    02 AVAILABLE-STOCK      PICTURE IS 9(5).

PROCEDURE DIVISION.
000-PRODUCE-REORDER-LISTING.
    OPEN INPUT BAL-FWD-FILE.
    OPEN OUTPUT REORDER-LISTING.
    MOVE "N" TO CARD-EOF-SWITCH.
    PERFORM 100-PRODUCE-REORDER-LINE
        UNTIL CARD-EOF-SWITCH IS EQUAL TO "Y".
    CLOSE BAL-FWD-File.
    CLOSE REORDER-LISTING.
    STOP RUN.

100-PRODUCE-REORDER-LINE.
    PERFORM 110-READ-INVENTORY-RECORD.
    IF CARD-EOF-SWITCH IS NOT EQUAL TO "Y"]
        PERFORM 120-CALCULATE-AVAILABLE-STOCK
        IF AVAILABLE-STOCK IS LESS THAN BAL-REORDER-POINT
            PERFORM 130-PRINT-REORDER-LINE.

110-READ-INVENTORY-RECORD.
    READ BAL-FWD-FILE RECORD
         AT END
             MOVE "Y" TO CARD-EOF-SWITCH.

120-CALCULATE-AVAILABLE-STOCK.
ADD BAL-ON-HAND BAL-ON-ORDER
    GIVING AVAILABLE-STOCK.

130-PRINT-REORDER-LINE.
    MOVE SPACE              TO REORDER-LINE.
    MOVE BAL-ITEM-NO        TO RL-ITEM-NO.
    MOVE BAL-ITEM-DESC      TO RL-ITEM-DESC.
    MOVE BAL-UNIT-PRICE     TO RL-UNIT-PRICE.
    MOVE AVAILABLE-STOCK    TO RL-AVAILABLE-STOCK.
    MOVE BAL-REORDER-POINT  TO RL-REORDER-POINT.
    WRITE REORDER-LINE.
```

7. 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

Example: Basic

```
REM  Basic Example Program
REM  Input:  An integer, listlen, where listlen is less
REM          than 100, followed by listlen-integer values
REM  Output: The number of input values that are greater
REM          than the average of all input values
  DIM intlist(99)
  result = 0
  sum = 0
  INPUT listlen
  IF listlen > 0 AND listlen < 100 THEN
REM  Read input into an array and compute the sum
   FOR counter = 1 TO listlen
     INPUT intlist(counter)
     sum = sum + intlist(counter)
   NEXT counter
REM  Compute the average
   average = sum / listlen
REM  Count the number of input values that are > average
   FOR counter = 1 TO listlen
     IF intlist(counter) > average
        THEN result = result + 1
   NEXT counter
REM  Print the result
    PRINT "The number of values that are > average is:";
          result
  ELSE
   PRINT "Error-input list length is not legal"
  END IF
END
```

8. 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: a user group for IBM mainframe computers
  - 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 for MIS.
  - 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

Example: PL/I

```
 /* PL/I PROGRAM EXAMPLE
  INPUT:   AN INTEGER, LISTLEN, WHERE LISTLEN IS LESS THAN
           100, FOLLOWED BY LISTLEN-INTEGER VALUES
  OUTPUT:  THE NUMBER OF INPUT VALUES THAT ARE GREATER THAN
           THE AVERAGE OF ALL INPUT VALUES */
 PLIEX: PROCEDURE OPTIONS (MAIN);
   DECLARE INTLIST (1:99) FIXED.
   DECLARE (LISTLEN, COUNTER, SUM, AVERAGE, RESULT) FIXED;
   SUM = 0;
   RESULT = 0;
   GET LIST (LISTLEN);
   IF (LISTLEN > 0) & (LISTLEN < 100) THEN
    DO;
/* READ INPUT DATA INTO AN ARRAY AND COMPUTE THE SUM */
    DO COUNTER = 1 TO LISTLEN;
      GET LIST (INTLIST (COUNTER));
      SUM = SUM + INTLIST (COUNTER);
    END;
/* COMPUTE THE AVERAGE */
    AVERAGE = SUM / LISTLEN;
/* COUNT THE NUMBER OF VALUES THAT ARE > AVERAGE */
    DO COUNTER = 1 TO LISTLEN;
      IF INTLIST (COUNTER) > AVERAGE THEN
        RESULT = RESULT + 1;
    END;
/* PRINT RESULT */
    PUT SKIP LIST ('THE NUMBER OF VALUES > AVERAGE IS:');
    PUT LIST (RESULT);
    END;
ELSE
    PUT SKIP LIST ('ERROR-INPUT LIST LENGTH IS ILLEGAL');
END PLIEX;
```

9. 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 – *dynamic typing*
- Storage is allocated to a variable when it is assigned a value – dynamic storage allocation

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

SNOBOL

- Designed as a *string manipulation* language at Bell Labs by Farber, Griswold, and Polensky in 1964
- Powerful operators for *string pattern matching.*
- Supports a number of built-in data types: integers, real, strings, patterns, arrays, tables (associative arrays)
- Slower than alternative languages (and thus no longer used for writing editors)
- Still used for certain text-processing tasks

10. The Beginning of Data Abstraction: SIMULA 67

- Designed primarily for *system simulation* in Norway by Nygaard and Dahl.
- Based on ALGOL 60 - for block structure & control statement - and SIMULA I
- Primary Contributions
  - *Coroutines* - a kind of subprogram
  - Classes → Object-Oriented Programming

11. *Orthogonal* Design: ALGOL 68

- The continued development of ALGOL 60,
  but *not a superset* of ALGOL 60.
- Source of several new ideas (even though the language itself never achieved widespread use)
- Design is based on the concept of *orthogonality*
  - A few basic concepts, plus a 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

12. Early Descendants of the ALGOLs

Pascal - 1971

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

Example: Pascal

```
{Pascal Example Program
 Input:   An integer, listlen, where listlen is less than
          100, followed by listlen-integer values
 Output:  The number of input values that are greater than
          the average of all input values }
program pasex (input, output);
  type intlisttype = array [1..99] of integer;
  var
    intlist : intlisttype;
    listlen, counter, sum, average, result : integer;
  begin
  result := 0;
  sum := 0;
  readln (listlen);
  if ((listlen > 0) and (listlen < 100)) then
    begin
{ Read input into an array and compute the sum }
  for counter := 1 to listlen do
    begin
    readln (intlist[counter]);
    sum := sum + intlist[counter]
    end;
{ Compute the average }
    average := sum / listlen;

{ Count the number of input values that are > average }
      for counter := 1 to listlen do
        if (intlist[counter] > average) then
          result := result + 1;
{ Print the result }
    writeln ('The number of values > average is:',
             result)
    end  { of the then clause of if (( listlen > 0 ... }
  else
    writeln ('Error-input list length is not legal')
end.
```

C - 1972

- Designed for *systems programming* (at Bell Labs by Dennis Richie)
- Evolved primarily from BCPL and B, but also ALGOL 68
  - BCPL: Basic Computer Programming Language
- Powerful set of operators, but poor type checking
- Initially spread through UNIX
- Though designed as a systems language, it has been used in many application areas

Example: C

```
/* C Example Program
 Input:  An integer, listlen, where listlen is less than
         100, followed by listlen-integer values
 Output: The number of input values that are greater than
         the average of all input values */
int main (){
  int intlist[99], listlen, counter, sum, average, result;
  sum = 0;
  result = 0;
  scanf("%d", &listlen);
  if ((listlen > 0) && (listlen < 100)) {
/* Read input into an array and compute the sum */
    for (counter = 0; counter < listlen; counter++) {
     scanf("%d", &intlist[counter]);
     sum += intlist[counter];
    }
/* Compute the average */
    average = sum / listlen;

/* Count the input values that are > average */
     for  (counter = 0; counter < listlen; counter++)
       if  (intlist[counter] > average) result++;
/* Print result */
    printf("Number of values > average is:%d\n", result);
  }
  else
    printf("Error-input list length is not legal\n");
}
```

13. 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.*: *Logical Programming Language*
- Non-procedural.
- Can be summarized as being an intelligent database system that uses an *inferencing process* to infer the truth of given queries.
  - E.g.) Query: D? DBS/KBS: A, B, A & B → C, C → D $\Rightarrow$ D.
- Comparatively inefficient
- Few application areas

Example: Prolog

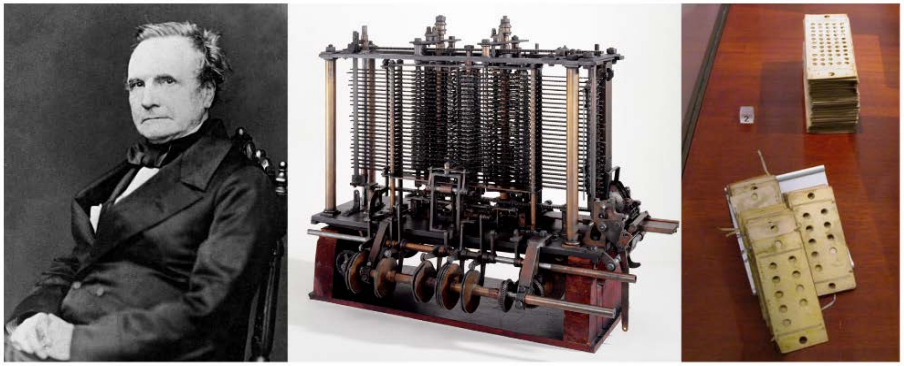
- Fact statements:
  - `mother(joanne, jake).`
  - `father(vern, joanne).`
- Rule statements:
  - `parent(X,Y):- mother(X,Y).`
  - `parent(X,Y):- father(X,Y).`
  - `grandparent(X,Z):- parent(X,Y), parent(Y,Z).`
- Query:
  - `father(bob, darcie)` → `false`
  - `parent(vern, joanne)` → `true`
  - `grandparent(X, Y)` → `X = vern, Y = jake`

14. History's Largest Design Effort: Ada

- Huge design effort, involving hundreds of people, much money, and about eight years.
- Designed for and used by DoD.
- Sequence of requirements (1975-1978)
  - (Strawman, Woodman, Tinman, Ironman, Steelman)
- Named Ada after Augusta Ada Byron (countess of Lovelace, 1815 - 1852), the first programmer.
  - To translate a document about Babbage's Analytical Engine from French to English, she augmented the document with her own notes (named A - G), tripling the size of the original document. Note G detailed the steps to compute the Bernoulli numbers - the 1st computer program.

14. History's Largest Design Effort: Ada (cont.)

- Analytical Engine:
  - *Charles Babbage*'s 2nd mechanical computer ← Difference Engine.
    - A British mathematician, philosopher, and mechanical engineer in the mid-1800s.
  - A *programmable* computing engine
  - The components:
    - Arithmetic logic unit (ALU): to perform arithmetic. Called a "mill".
    - Conditional logic: to perform conditional branching and loops.
    - Memory: it could hold > 1,000 numbers containing 40 decimal digits.
    - Punched card reader: to input programs and data via punched cards.



Sources: Charles Babbage - Public domain via Wikimedia Commons;
Analytical Engine - Mrjohncummings at Science Museum London / CC BY-SA 2.0 via Wikimedia Commons;
Punched cards - Karoly Lorentey at Science Museum London / CC BY 2.0 via Wikimedia Commons

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
- Ada 2005
  - Interfaces and synchronizing interfaces
- Popularity suffered because the DoD no longer requires its use but also because of popularity of C++

Example: Ada

```
-- Ada Example Program
-- Input:   An integer, List_Len, where List_Len is less
--          than 100, followed by List_Len-integer values
-- Output:  The number of input values that are greater
--          than the average of all input values
with Ada.Text_IO, Ada.Integer.Text_IO;
use Ada.Text_IO, Ada.Integer.Text_IO;
procedure Ada_Ex is
  type Int_List_Type is array (1..99) of Integer;
  Int_List : Int_List_Type;
  List_Len, Sum, Average, Result : Integer;
begin
   Result:= 0;
   Sum := 0;
   Get (List_Len);
   if (List_Len > 0) and (List_Len < 100) then
-- Read input data into an array and compute the sum
     for Counter := 1 .. List_Len loop
       Get (Int_List(Counter));
       Sum := Sum + Int_List(Counter);
     end loop;
-- Compute the average
     Average := Sum / List_Len;
-- Count the number of values that are > average
     for Counter := 1 .. List_Len loop
       if Int_List(Counter) > Average then
         Result:= Result+ 1;
       end if;
     end loop;
-- Print result
     Put ("The number of values > average is:");
     Put (Result);
     New_Line;
   else
     Put_Line ("Error-input list length is not legal");
   end if;
end Ada_Ex;
```

15. 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 binding)
- Pioneered the graphical user interface (GUI) design
- Promoted OOP -- 1st pure OOP

Example: Smalltalk

```
"Smalltalk Example Program"
"The following is a class definition, instantiations of
which can draw equilateral polygons of any number of sides"
class name                    Polygon
superclass                    Object
instance variable names       ourPen
numSides
sideLength
"Class methods"
  "Create an instance"
  new
     ^ super new getPen

  "Get a pen for drawing polygons"
  getPen
     ourPen <- Pen new defaultNib: 2

  "Instance methods"
  "Draw a polygon"
  draw
     numSides timesRepeat: [ourPen go: sideLength;
                            turn: 360 // numSides]

  "Set length of sides"
  length: len
  sideLength <- len

  "Set number of sides"
  sides: num
     numSides <- num
```

16. Combining Imperative and Object-Oriented Programming (OOP): C++

- Developed at Bell Labs by Stroustrup in 1984.
- C with *classes* (1983) → C++
- Evolved from C and SIMULA 67.
- Facilities for OOP, taken partially from SIMULA 67.
- Design goals of C with Classes:
  - Program could be organized, similar to SIMULA 67.
  - Little/no performance penalty – e.g.) no array index range checking
  - For every application for which C could be used.
- A *large and complex* language, in part because it supports ***both*** procedural and OO programming.

Combining Imperative and Object-Oriented Programming (OOP): C++ (cont.)

- Operators and methods can be overloaded for user-defined type and methods, respectively.
- Rapidly grew in popularity, along with OOP.
  - Backward compatible with C; i.e. C programs, with few changes, can be compiled as C++ programs.
  - Mostly, possible to link C++ code with C code.
- ANSI standard approved in November 1997.
- Microsoft's version: MC++ (Managed C++)
  - Used .NET platform
  - Properties, delegates, interfaces, no multiple inheritances.

A Related OOP Language

- Swift : a replacement for Objective-C (for MAC OS X)
  - Released in 2014 by Apple
  - Tuple data type, option type, protocol, generic type, etc.
  - Two categories of types, classes (reference type) and struct (value type), like C#.
  - Used by Apple for systems programs.
- Delphi : another related language
  - A hybrid language, like C++, by Boland in 1995.
  - Began as an object-oriented version of Pascal: Object Pascal
  - Designed by Anders Hejlsberg, who also designed Turbo Pascal and C#

17. 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++
  - Smaller, much simpler, flexible and safer than C++:
    - removed `struct`, `union`, `enum`, pointer arithmetic, and half of the assignment coercions of C++
  - Supports *only* OOP. Cf) C++ supports both procedural & OOP.
  - Has *references*, but *not pointers*
  - Supports only single inheritance of classes.
  - Includes support for applets and a form of concurrency.

Java Evaluation

- Eliminated many unsafe features of C++:
- Supports concurrency: `thread`
- Libraries for applets, GUIs, database access.
- *Portable*: Java Virtual Machine concept, JIT compilers
- Widely used for Web programming
  - Used to create *applets* on the front-end and servlets, JavaServer Pages, and web APIs on the back-end.
  - Still a popular server-side language, but most developers now use Java primarily for creating *web APIs*.
- Use increased faster than any previous language
- The latest version, Java 22, was released on 3/2024.

Java Applet, Application, Servlet

- Applet (cf. standalone Java application)
  - Java program that runs in a Web browser – designed to be embedded within an HTML page.
  - When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
  - A JVM (Java interpreter) was required to view an applet
  - An applet can be a fully functional Java application because it has the entire Java API at its disposal.
  - 1st version in 1995; however, deprecated by Java 9 in 2017.
- Servlet:
  - Java programs that run on the Java-enabled web *server* or application server.
  - Used to handle the complex requests obtained from the web server, process the request, produce the response, then send a response back to the web server.

Example: Java

```
// Java Example Program
// Input:   An integer, listlen, where listlen is less
//          than 100, followed by length-integer values
// Output:  The number of input data that are greater than
//          the average of all input values
import java.io.*;
class IntSort {
public static void main(String args[]) throws IOException {
  DataInputStream in = new DataInputStream(System.in);
  int listlen,
      counter,
      sum = 0,
      average,
      result = 0;
  int[] intlist =  new int[99];
  listlen = Integer.parseInt(in.readLine());
  if ((listlen > 0) && (listlen < 100)) {

/* Read input into an array and compute the sum */
    for (counter = 0; counter < listlen; counter++) {
      intlist[counter] =
             Integer.valueOf(in.readLine()).intValue();
      sum += intlist[counter];
    }
/* Compute the average */
    average = sum / listlen;
/* Count the input values that are > average */
    for (counter = 0; counter < listlen; counter++)
      if (intlist[counter] > average) result++;
/* Print result */
      System.out.println(
          "\nNumber of values > average is:" + result);
    } //** end of then clause of if ((listlen > 0) ...
    else System.out.println(
             "Error-input list length is not legal\n");
  } //** end of method main
} //** end of class IntSort
```

18. Scripting Languages for the Web

- Scripting Language:
  - *script*: a list of commands in a file to be *interpreted*.
  - a language where instructions are written for a *run-time environment*.
  - Designed for integrating and communicating with other programming languages. -- Initially, `sh` (for shell), `ksh`, `awk`.

- Perl
  - Designed by Larry Wall—first released in 1987.
  - Initially used as a UNIX utility for processing text files.
  - Similar to imperative language – compiled.

Scripting Languages for the Web (cont.)

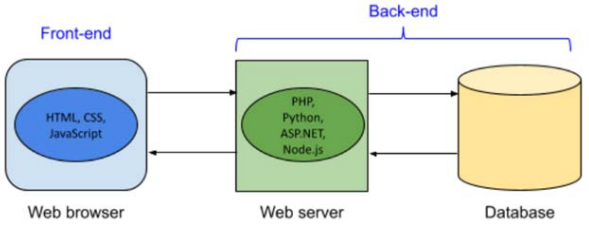
- Perl (cont.)
  - Variables are *statically typed* but implicitly declared.
  - Three distinctive namespaces, denoted by the first character of a variable's name.
  - *Dynamic length array*: grow/shrink during execution
  - can be sparse: gaps b/t elements
    - gaps do not take space in memory
    - Iteration, **`foreach`**, iterates over the missing elements.
  - Associative array - hash
  - Powerful, but somewhat dangerous.
  - Gained widespread use for CGI programming on the Web.
    - Common Gateway Interface b/t HTTP server and programs generating dynamic content on the webpage.
  - Also used for a replacement for UNIX system administration language.

Example: Perl

```
# Perl Example Program
# Input:   An integer, $listlen, where $listlen is less
#          than 100, followed by $listlen-integer values.
# Output:  The number of input values that are greater than
#          the average of all input values.
($sum, $result) = (0, 0);
$listlen = <STDIN>;
if (($listlen > 0) && ($listlen < 100)) {
# Read input into an array and compute the sum
  for ($counter = 0; $counter < $listlen; $counter++) {
    $intlist[$counter] = <STDIN>;
  } #- end of for (counter ...
# Compute the average
  $average = $sum / $listlen;
# Count the input values that are > average
  foreach  $num (@intlist) {
    if ($num > $average) { $result++; }
  } #- end of foreach $num ...
# Print result
  print "Number of values > average is: $result \n";
} #- end of if (($listlen ...
else {
  print "Error--input list length is not legal \n";
}
```

Scripting Languages for the Web (cont.)

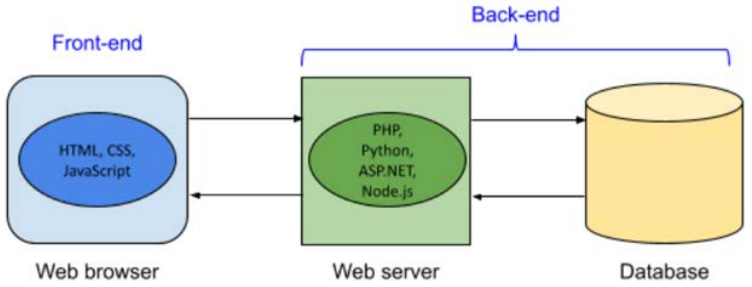
- JavaScript
  - Began at Netscape, but later became a joint venture of Netscape and Sun Microsystems
    - Java, Solaris OS, NFS, SPARC microprocessor – contribution to Unix, RISC processor, etc.
  - A client-side (i.e. front-end) HTML-embedded scripting language, often used to create dynamic HTML documents
  - Purely interpreted
  - Related to Java only through similar syntax



- PHP
  - PHP: Hypertext Preprocessor, designed by Rasmus Lerdorf in 1994.
  - A server-side (i.e. back-end) HTML-embedded scripting language, often used for form processing and database access through the Web
  - Purely interpreted

Scripting Languages for the Web (cont.)

- JavaScript
  - Began at Netscape, but later became a joint venture of Netscape and Sun Microsystems
    - Java, Solaris OS, NFS, SPARC microprocessor – contribution to Unix, RISC processor, etc.
  - A client-side (i.e. front-end) HTML-embedded scripting language, often used to create dynamic HTML documents.
    - It runs in a browser, enabling web pages that support actions like responding to a button click.
  - Purely interpreted
  - Related to Java only through similar syntax



Scripting Languages for the Web (cont.)

- PHP
  - PHP: Hypertext Preprocessor, designed by Rasmus Lerdorf in 1994.
  - A server-side (i.e. back-end) HTML-embedded scripting language, often used for form processing and database access through the Web
  - Purely interpreted

Cf) Web languages

- HTML (Hyper-Text Markup Language):
  - a textual language for creating web pages.
  - extension of the HTML file: .html or .htm
- CSS (Cascading Style Sheets):
  - a language that can be used to style a web page via changes in colors, sizes, spacing, fonts, and more.
- JavaScript:
  - a language that runs in a browser, enabling web pages that support actions like responding to a button click.

Example: JavaScript -- also see pg.95 of the textbook.

```
<!DOCTYPE html>
<html lang="en">
   <meta charset="UTF-8">
   <script>
      function changeTextColor(newColor) {
         let x = document.getElementById("Colorable");
         x.style.color = newColor;
      }
   </script>
   <style>
   h1 {
      color: green;
      background-color: lightgray;
      font-size: 16pt;
   }
   p {
      font-family: arial;
      margin-left: 10px;
   }
   strong {
      background-color: lightgreen;
      padding: 5px;
   }
   </style>
   <title>For sale: 2012 Ducati Streetfighter</title>
   <body>
      <h1 id="Colorable">Ducati Streetfighter - $9000</h1>
      <p>year: <strong>2012</strong></p>
      <p>make and model: <strong>Ducati Streetfighter 848</strong></p>
      <p>condition: <strong>excellent</strong></p>
      <p>odometer: <strong>9500</strong></p>

      <p>Change heading color to:
         <button type="button" onclick="changeTextColor('white')">White</button>
         <button type="button" onclick="changeTextColor('green')">Green</button>
      </p>
   </body>
</html>
```

In JavaScript

In CSS

In HTML

Example: JavaScript (cont.).

color.html

```
<!DOCTYPE html>
<html lang="en">
   <meta charset="UTF-8">

   <script src="color.js">
   </script>

   <style>
      <link href="color.css" rel="stylesheet">
   </style>

   <title>For sale: 2012 Ducati Streetfighter</title>

   <body>
      <h1 id="Colorable">Ducati Streetfighter - $9000</h1>
      <p>year: <strong>2012</strong></p>
      <p>make and model: <strong>Ducati Streetfighter 848</strong></p>
      <p>condition: <strong>excellent</strong></p>
      <p>odometer: <strong>9500</strong></p>

      <p>Change heading color to:
         <button type="button" onclick="changeTextColor('white')">White</button>
         <button type="button" onclick="changeTextColor('green')">Green</button>
      </p>
   </body>

</html>
```

color.css

```
h1 {
      color: green;
      background-color: lightgray;
      font-size: 16pt;
   }
   p {
      font-family: arial;
      margin-left: 10px;
   }
   strong {
      background-color: lightgreen;
      padding: 5px;
   }
```

color.js

```
function changeTextColor(newColor) {
        let x = document.getElementById("Colorable");
        x.style.color = newColor;
```

Click a button to change this text's color.

Red Blue

Scripting Languages for the Web (cont.)

- Python
  - An object-oriented interpreted script language.
  - A general-purpose OO interpreted scripting language by Guido van Rossum in 1990.
  - Used for system administration, form processing, etc.
  - Type-checked but dynamically typed.
  - Supports lists, tuples, and hashes.
  - Google's language of choice.
- 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

Server-side programming languages & *platforms*

- Server-side programming: server platform, tool support, developer experience, library support.
- https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites
- ASP.NET:
  - A powerful platform generally used on Windows servers.
  - Collection of web development technologies
  - first released in 2002 by Microsoft that uses the C# or VB.NET programming languages.
- Ruby on Rails
  - Web application framework written in Ruby and created by David Heinemeier Hansson in 2004.
  - Twitter was originally created with Rails.
- Node.*js*
  - Runtime environment that uses modules written in JavaScript. Originally created in 2009 by Ryan Dahl.
  - relatively new, but many notable companies like Walmart and LinkedIn spout Node's ability to scale-up efficiently.

19. The Flagship .NET Language: C#

- Part of the .NET development platform (2000)
- Based on C++ , Java, and Delphi.
- Purpose: to provide a language for component-based SW development in .NET framework.
- Includes pointers, delegates, properties, enumeration types, a limited kind of dynamic typing, and anonymous types.
- Is evolving rapidly.

Example: C#

```
// C# Example Program
// Input:  An integer, listlen, where listlen is less than
//         100, followed by listlen-integer values.
// Output: The number of input values that are greater
//         than the average of all input values.
using System;
public class Ch2example {
  static void Main() {
    int[] intlist;
    int listlen,
        counter,
        sum = 0,
        average,
        result = 0;
    intList = new int[99];
    listlen = Int32.Parse(Console.readLine());
    if ((listlen > 0) && (listlen < 100)) {
// Read input into an array and compute the sum
      for (counter = 0; counter < listlen; counter++) {
        intList[counter] =
                      Int32.Parse(Console.readLine());
        sum += intList[counter];
      } //- end of for (counter ...
// Compute the average
      average = sum / listlen;
// Count the input values that are > average
      foreach (int num in intList)
        if (num > average) result++;
// Print result
      Console.WriteLine(
         "Number of values > average is:" + result);
    } //- end of if ((listlen ...
    else
      Console.WriteLine(
         "Error--input list length is not legal");
  } //- end of method Main
} //- end of class Ch2example
```

20. Markup Language

- HTML: HyperText Markup Language
  - The standard markup language for web documents.
- Hypertext:
  - text that has links to other text, (images, videos, and more).
- Markup:
  - Document markup is unique markings in the document that provide additional information about links, formatting, and images, etc.
  - E.g.)

```
<img src="https://resources.zybooks.com/WebProgramming/kyotov1.jpg">
<a href=https://www.und.edu> <strong> UND </strong> </a>
```

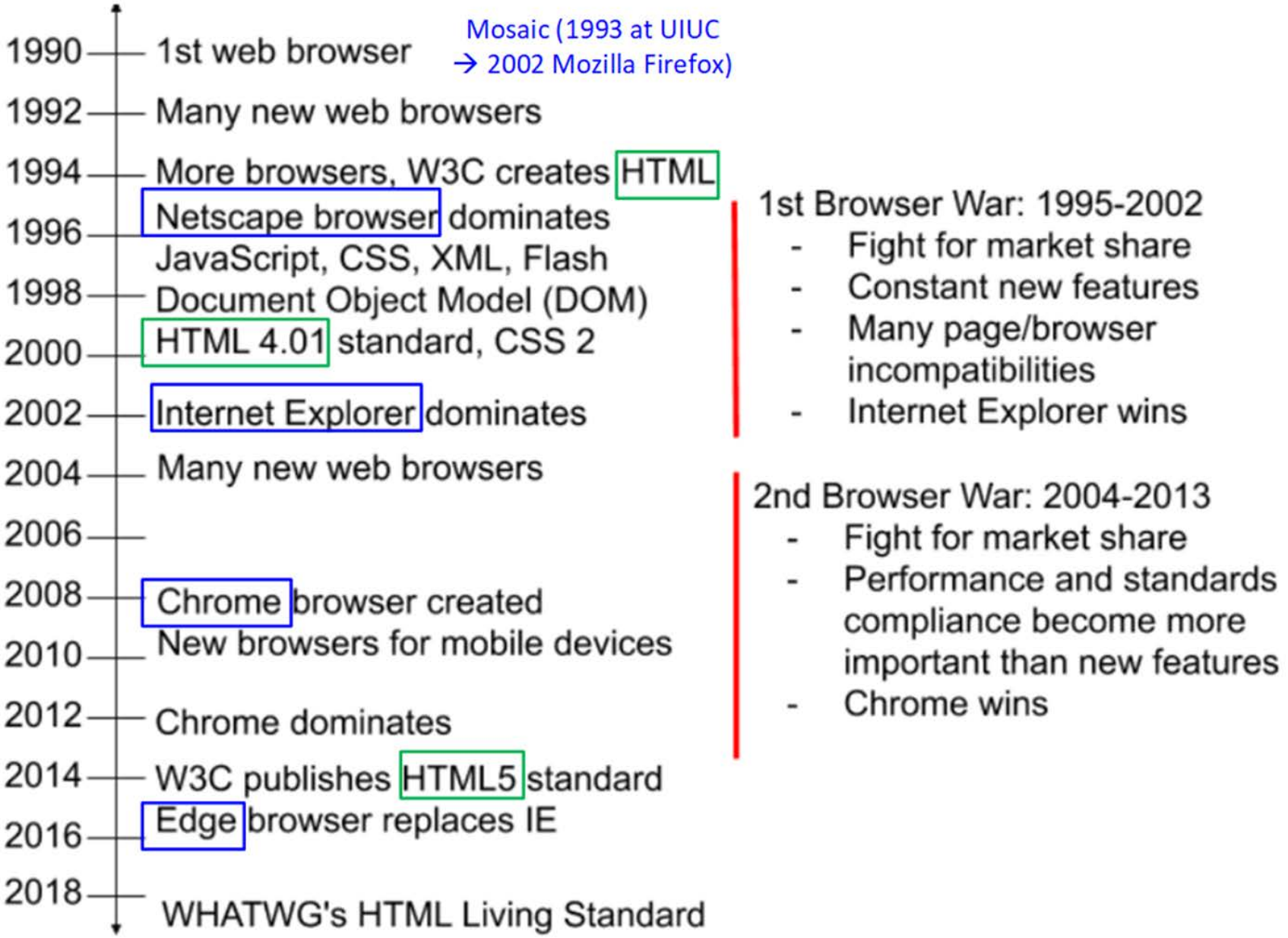
HTML Standardization

- World Wide Web Consortium (W3C):
  - the international standards organization that has controlled a number of web standards, due to Incompatibility between the browser and the web page.
  - HTML5: the latest HTML standard by W3C in 2014.
- Web Hypertext Application Technology Working Group (WHATWG):
  - an organization that develops a variety of web standards.
  - WHATWG produces the HTML Living Standard, a continually evolving standard without version numbers that replaces HTML5.
- Separation of document

→ Modern Web Page is composed of HTML, CSS, and JavaScript.

  - Document Structure: in HTML for the structure and content of a web page.
  - Document Presentation: in CSS for the layout and visible appearance.
  - Webpage Interaction: in JavaScript for the dynamic behaviors and actions of a web page.

Web Browser History



Markup/Programming Hybrid Languages

- XSLT
  - eXtensible Markup Language (XML): a meta-markup 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
  - JSTL, a JSP library, includes programming constructs in the form of HTML elements

Summary

- Development, development environment, and evaluation of a number of important programming languages

- Perspective into current issues in language design