COSC 2P90
Programming Languages
& Object-Orientation

Hi!

Textbooks

- **Main Text**
  - Comparative Programming Languages
    3rd ed.; Wilson, LB & Clark, RG; Addison-Wesley (2001); ISBN 0-201-71012-9
- **Supplemental Texts**
  - The Unified Modeling Language User Guide, 2nd edition, Booch, G, Rumbaugh, J & Jacobson, I; Addison-Wesley (2005); ISBN 0-321-26797-4
  - Design Patterns: Elements of Reusable Object-Oriented Software
    Gamma, E, Helm, R, Johnson, R, Vlissides, J; Addison-Wesley (1995); ISBN 0-201-63361-2

Course Work

- **Marking Scheme**
  - Programming Assignments (4x7%) 28%
  - Midterm Exam: Oct 21, 15:30-17:00, TH259 (lecture time) 22%
  - Final Exam 50%

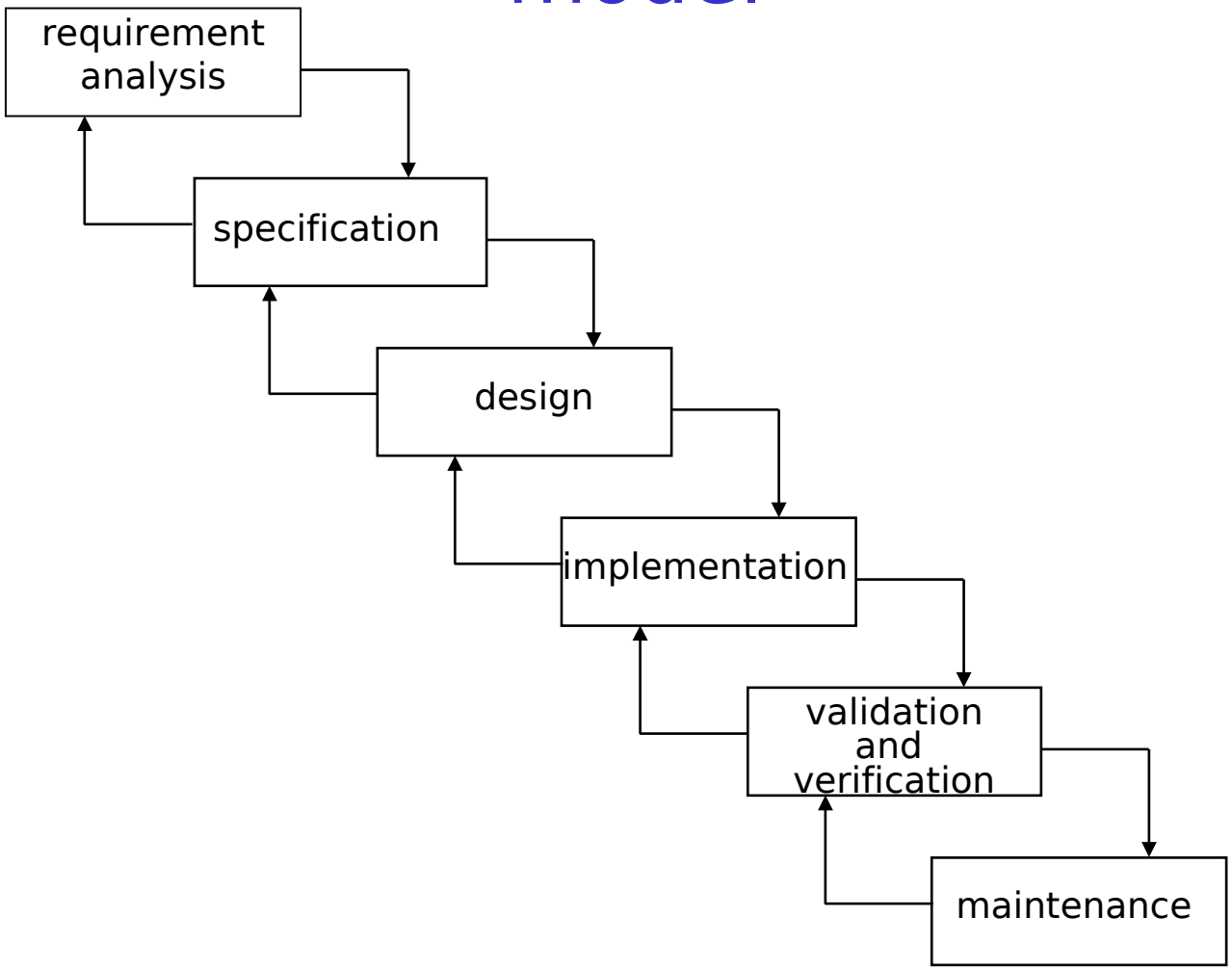
Assignments and Final Exam

- Assignments will be available online. Assignments will be returned via room J333 at the times posted.
- Due dates/times for assignments will be included on assignment specification pages and will be accepted late until the indicated time, subject to a penalty of 25%. After the late period, assignments will **not** be accepted.
- Assignments will be carefully examined regarding plagiarism. Cases of suspected plagiarism will be dealt with according to the University regulations and Departmental procedures.
- A mark of at least 40% on the final exam is required to achieve a passing grade in this course. No electronic devices and especially no calculators will be allowed in the examination room.
- The final examination will not be directly supervised or written in a gymnasium. Rather, it will be a “take-home” exam. The procedures and schedule for the exam will be posted at a later date. The allotted time will be a single weekend (receiving on Friday, and submitting on Monday). You will be permitted to conduct research during your examination, but everything you submit must be your own work. All submitted work will be scrutinized carefully and plagiarism on a final exam will be taken extremely seriously. Your work may be submitted to Turnitin.com, or subjected to other third-party integrity devices.
  - (also, I do know how to use Google, too. Plus I'm psychic)
- Consideration regarding illness for assignment submission or test dates will **only** be considered if accompanied with the completed Departmental Medical Excuse form, or proper advance notification is given.

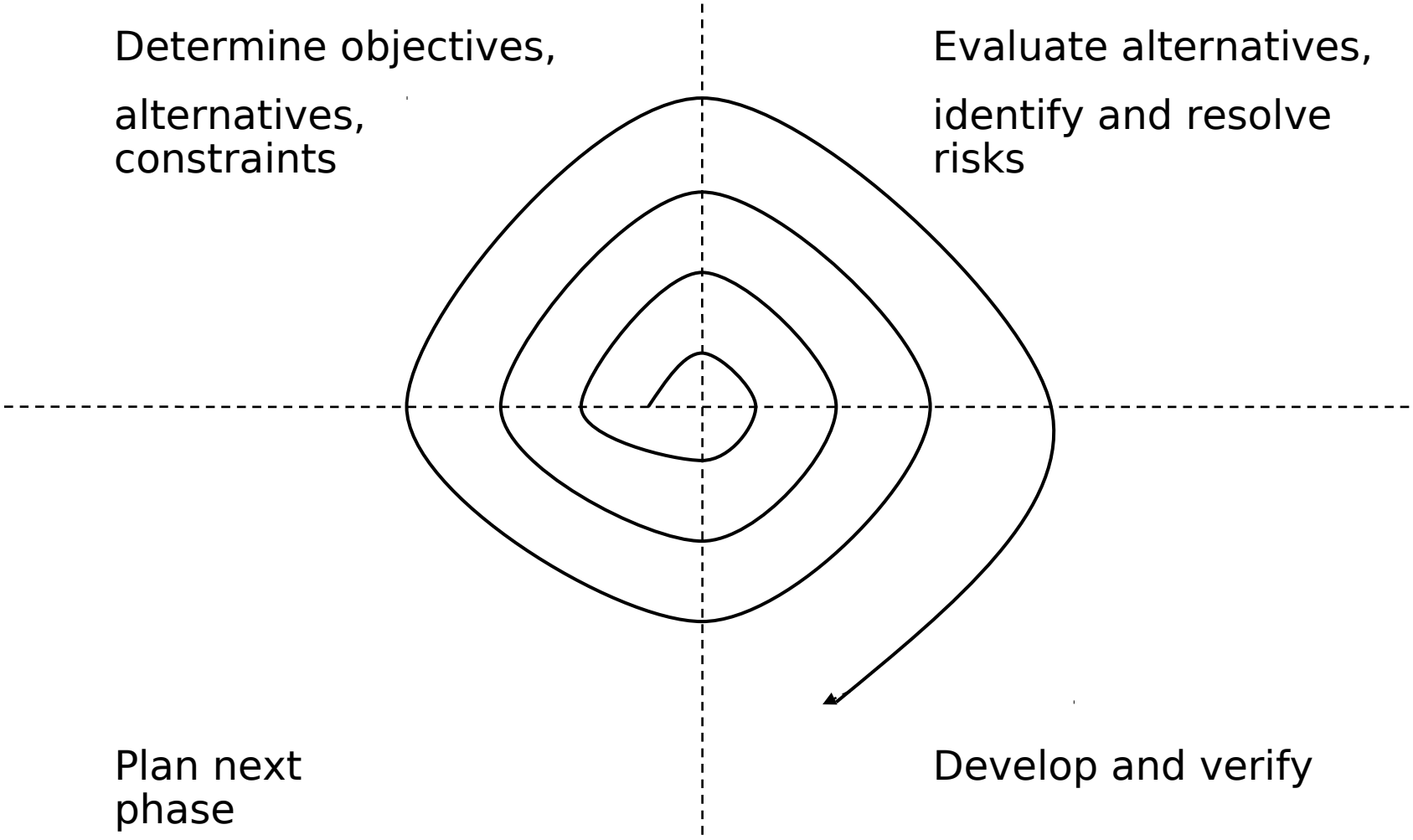
Languages & the Development Process

- diversity of languages
  - similarities & differences
- development process
  - waterfall model
    - language influence
  - iterative development
    - spiral model
  - modeling languages
    - UML
  - impact of software development on language design
    - structured programming
    - modules

Waterfall model



Boehm spiral model



Programming Paradigms

- imperative
  - C, FORTRAN, COBOL, ADA, ALGOL, PASCAL, ...
- functional
  - pure-LISP, ML, HASKELL, GOFER, FP, SCHEME,...
- logic
  - PROLOG,...
- object-oriented
  - with imperative base language: C++, JAVA, DELPHI, POLYTOIL,...
  - with functional base language: FOOPS, OBJECT-GOFER,...
  - pure object-oriented: SMALLTALK,...

Language Design

- expressive power
  - high-level
  - abstract data types & libraries
  - readability
    - readable vs writeable
- simplicity & orthogonality
  - simplicity
    - avoiding complexity vs handling complexity
    - Algol 68 vs Pascal
  - orthogonality
    - combination of features without interference
  - consistency
    - “law of least surprise”

- implementation
  - imperative → compiled
    - efficient execution
  - functional & logic → interpreter
    - more flexible
  - virtual machine
    - implemented by interpreter
    - Java virtual machine
      - “compile once, run anywhere”
  - Pascal vs Algol 68
  - C++ vs Ada

- error detection & correction
  - compile-time preferred
    - C vs Java
  - redundancy
    - variable declarations
- program correctness & standards
  - program proving
  - structured programming & goto’s
  - specification of syntax & semantics
    - Backus Normal Form (BNF)
  - official standards
    - Ada

Languages vs Systems

- what makes the language?
- feature glut vs extensible libraries
  - PL/I vs Java
- OS dependence
  - Java vs Visual Basic
  - efficiency
- event driven programming
  - GUIs
  - control loop vs language support

Lexical Elements

- character set
  - hardware limitation vs appropriate set
    - FORTRAN vs ALGOL
  - ASCII
    - composite symbols
  - Unicode
- identifiers
  - names for declared entities
  - keywords vs reserved words
  - predefined identifiers
- comments
  - comment lines
  - comment terminators
    - unclosed comments
- spaces and line terminators

History of Languages

- evolution
  - machine codes
  - assembly language
  - high-level languages (HLL)
- software support
  - assembler
  - compiler
  - interactive development environment (IDE)
  - rapid application development (RAD)

FORTRAN

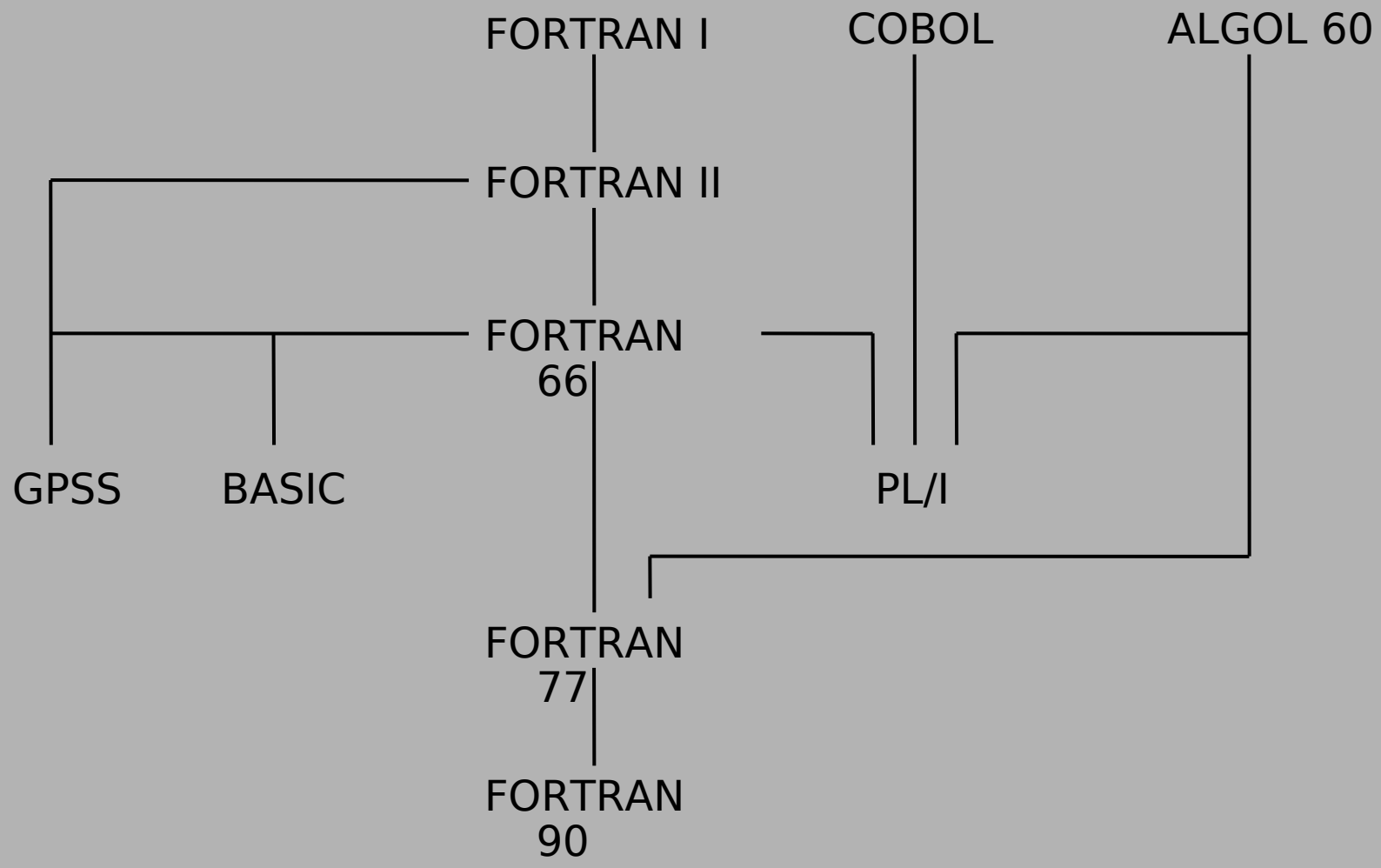
- FORmula TRANslating system
- first generally available HLL
- goals
  - reduce development & debugging costs
  - efficient compilation
- features
  - comments
  - mathematical notation
  - looping statement (`DO`)
  - subroutines & functions
  - I/O formatting
  - machine independence
    - but no standard

- reasons for success of FORTRAN
  - easy to learn vs assembly
  - makes efficient use of programmers' time
  - supported by IBM
  - most users and applications at the time were scientific
  - simplified tedious tasks, e.g. I/O

A FORTRAN Program

```
C FORTRAN PROGRAM TO FIND MEAN OF N NUMBERS AND
C NUMBER OF VALUES GREATER THAN THE MEAN
      DIMENSION A(99)
      REAL MEAN
      READ(1,5) N
 5    FORMAT(I2)
      READ(1,10)(A(I), I=1, N)
10    FORMAT(6F10.5)
      SUM=0.0
      DO 15 I=1, N
15    SUM=SUM+A(I)
      MEAN=SUM/FLOAT(N)
      NUMBER=0
      DO 20 I=1, N
      IF (A(I) .LE. MEAN) GOTO 20
      NUMBER=NUMBER+1
20    CONTINUE
      WRITE(2,25) MEAN, NUMBER
25    FORMAT(8H MEAN = ,F10.5, 5X, 20H NUMBER OVER MEAN = ,I5)
      STOP
      END
```

The FORTRAN Family



ALGOL 60

- ALGOrithmic Language
- joint European-US Committee
- goals
  - standard mathematical notation
  - use to describe computing processes
  - machine translatable
- ALGOL 60 report
  - BNF
- features
  - formal language definition
  - block structure
  - arrays with variable bounds
  - structured control statements
  - recursion

- successful in Europe
- reasons not widely used in North America
  - 3 years after FORTRAN
  - more features, harder to learn
  - compilers more complex, less efficient
  - no standard I/O

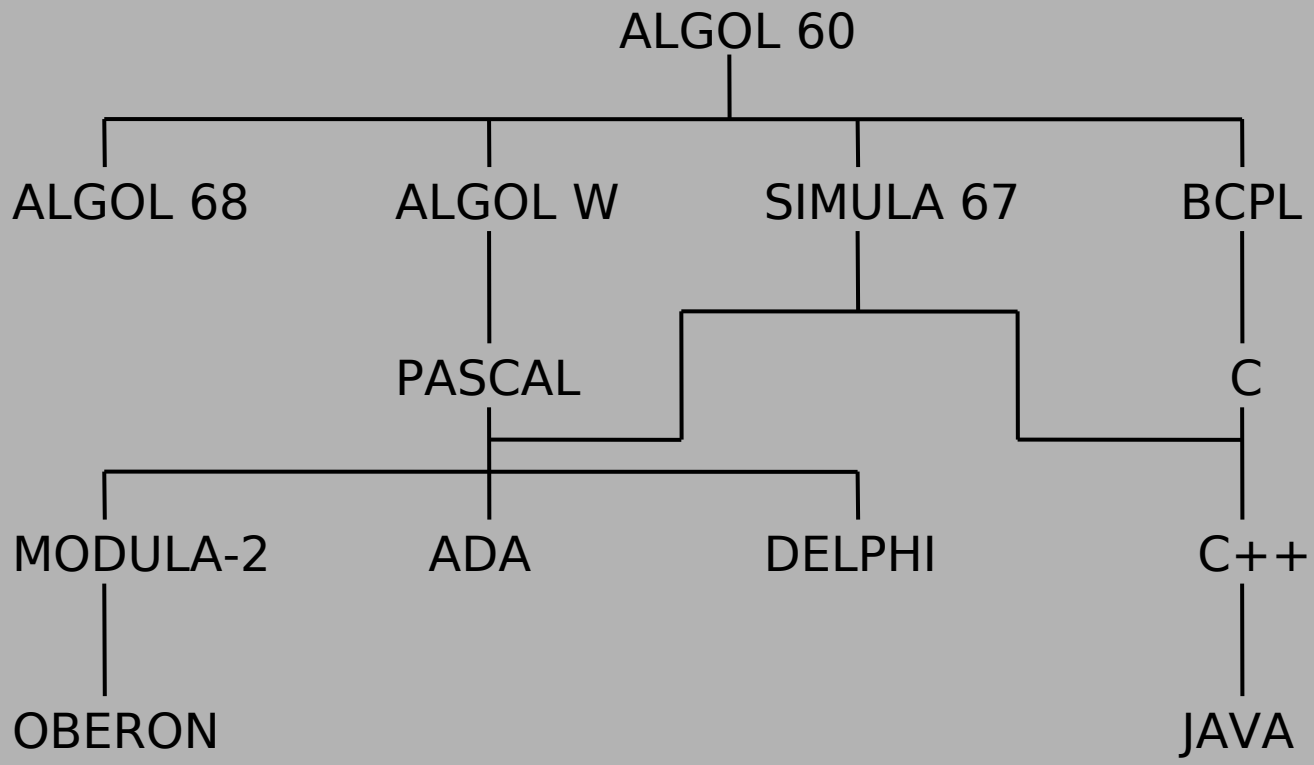
An ALGOL 60 Program

```
begin
  comment this program finds the mean of n numbers
           and the number of values greater than the mean;
  integer n;
  read(n);
  begin
   real array a[1:n];
   integer i, number;
   real sum, mean;
   for i := 1 step 1 until n do
     read(a[i]);
   sum := 0.0;
   for i := 1 step 1 until n do
     sum := sum + a[i];
   mean := sum / n;
   number := 0;
   for i := 1 step 1 until n do
     if a[i] > mean then number := number + 1;
   write("MEAN = ", mean, "NUMBER OVER MEAN = ", number)
  end
end
```

The Algol Family

- ALGOL 60 very influential on language design
- ALGOL W
  - tidied up ALGOL 60
  - introduced
    - records and references (linked structures)
    - case statement
    - multiple looping structures
    - parameter passing: call-by-value, call-by-result, call-by-value/result, call-by-name
    - string support
    - assert statement
- Algol 68
  - IFIP working group
  - highly orthogonal
    - small number of features used in combination
      - vs PL/I
  - highly-formal specification - abstruse

The ALGOL Family



Pascal

- built on ALGOL W
- goals
  - efficient implementation
  - for teaching good programming style
- widely used in universities and became common in industry
  - DEC
- features
  - restrictions on goto’s
  - user-defined data types
  - strings
- extensions
  - modules (units)

COBOL

- COmmon Business Oriented Language
- business data processing
- no common notation
  - English?
- DoD requirements
- features
  - English-like syntax
    - verbose
  - emphasis on file processing
  - 4 divisions: identification, environment, data, procedure
- tight language control
  - relatively unchanged in 25 years
  - legacy systems

4th Generation Languages

- problem-oriented vs implementation-oriented
  - specify problem not solution
- report generators
  - RPG (Report Program Generator)
- application specific software
  - spreadsheets
  - SQL (Structured Query Language)
- visual programming
  - VisualBasic
  - Rapid Application Development (RAD)

General-Purpose Languages

- scientific vs commercial programming & languages
- multi-purpose language
- PL/I
  - IBM to replace FORTRAN & COBOL
  - features
    - mathematical notation, parameters, separate compilation formatted I/O from FORTRAN
    - block structure, structured statements from ALGOL 60
    - record I/O, PICTURE type, heterogeneous records from COBOL
    - list processing, exception handing
  - large feature set
    - vs Algol 68 with small set & orthogonality vs O-O with small set & extensibility

A PL/I Program

```
EXAMPLE : PROCEDURE OPTIONS (MAIN);
   /* Program to find the mean of n numbers and the number
          of values greater than the mean */
   GET LIST (N);
   IF N > 0 THEN BEGIN;
      DECLARE MEAN, A(N) DECIMAL FLOAT,
      SUM DEC FLOAT INITIAL(0), NUMBER FIXED INITIAL(0);
      GET LIST (A);
      DO I = 1 TO N;
        SUM = SUM + A(I);
      END;
      MEAN = SUM/N;
      DO I = 1 TO N;
    IF A(I) > MEAN THEN NUMBER = NUMBER + 1;
      END;
      PUT LIST ('MEAN=', MEAN,'NUMBER GREATER THAN MEAN=',NUMBER);
END EXAMPLE;
```

Interactive Languages

- introduction of timesharing terminals
- edit and run seamlessly
- rapid development and prototyping
- programming for non-programmers
- JOSS
- BASIC (Beginner's All Purpose Symbolic Instruction Code)
  - derived from FORTRAN
- APL (A Programming Language)
  - numerical algorithms
  - large operator set, few control structures
  - no operator precedence

A BASIC Program

```
10  REM THIS IS A BASIC PROGRAM FOR FINDING THE MEAN
20  DIM A(99)
30  INPUT N
40  FOR I = 1 TO N
50  INPUT A(I)
60  LET S =  S + A(I)
70  NEXT I
80  LET M = M/S
90  LET K = 0
100 FOR I = 1 TO N
110 IF A(I) < M THEN 130
120 LET K = K + 1
130 NEXT I
140 PRINT "MEAN IS", M
150 PRINT "NUMBER GREATER THAN MEAN IS", K
160 STOP
170 END
```

Special-Purpose Languages

- higher-level – closer to application domain
- string manipulation
  - pattern matching
  - SNOBOL
- list processing
  - lists as primary data structure
  - non-numeric computation
  - garbage collection
  - recursion is primary control
  - self-modification
  - AI
  - LISP
    - pure LISP is functional

- simulation
  - state changes to entities in model
  - GPSS
  - Simula67
    - based on ALGOL
    - introduced class concept
      - objects represent entities in system
      - independent lifetimes
      - basis for object-orientation
- scripting
  - processing textual information
  - pattern matching
    - awk, grep, Perl
    - CGI programming
    - JavaScript

Systems Programming Languages

- HLLs for low level systems programming (OS, compilers)
- access to low level concepts (bits, addresses)
- efficient code
- BCPL, Coral
- C
  - loosely-typed
  - small language, portable
  - combination of ALGOL & FORTRAN
  - UNIX
  - many pitfalls
    - for advanced programmers not novices
  - C++ - O-O extension to C
  - C# - Micorsoft’s response to Java
- occam
  - transputers – parallel programming

A C Program

```
main() {
   /* this is the C version of the program to find the
      mean and the number of those greater than the mean */
   float a[100], mean, sum;
   /* the array a has 100 elements – a[0], .. a[99] */
   int n, i, number;
   scanf("%d", &n);
   for (i = 0; i < n; i++)
      scanf("%f",&a[i]);
   sum = 0.0;
   for (i = 0; i < n; i++)
      sum += a[i];
   mean = sum / n;
   number = 0;
   for (i = 0; i < n; i++) {
      if (a[i] > mean)
         number++;
   }
   printf("MEAN = %f\n", mean);
   printf("NUMBER OVER MEAN = %d\n", number);
}
```