Programming Language History and Evolution

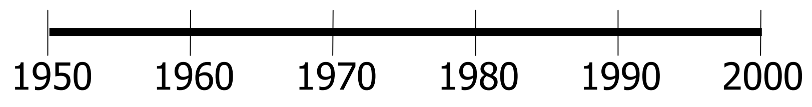
In Text: Chapter 2

1

Brief Overview of Paradigms

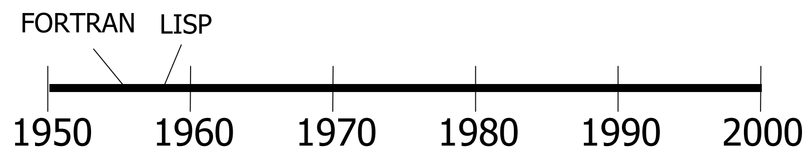
- Procedural/Imperative
- Functional/Applicative
- Logic
- Object-oriented (closely related to imperative)
- Problem-oriented/application-specific

An Overview of PL History

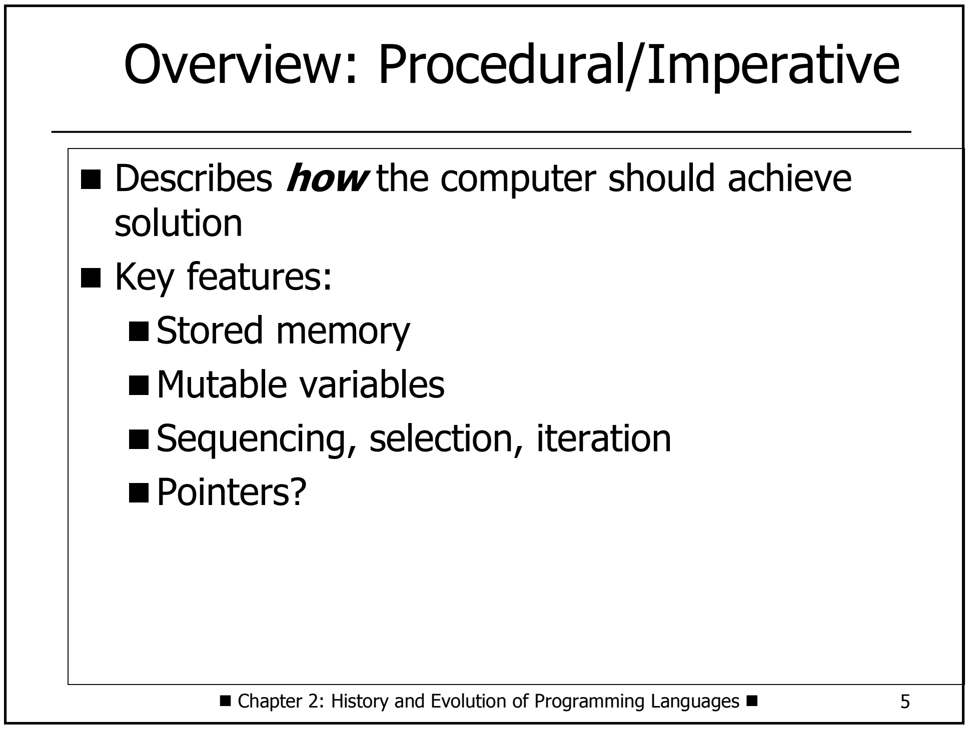


- 1950’s: Discovery and description
- 1960’s: Elaboration and analysis
- 1970’s: Technology
- 1980’s: New paradigms
- 1990’s: Internet influences

1950’s: Discovery and Description

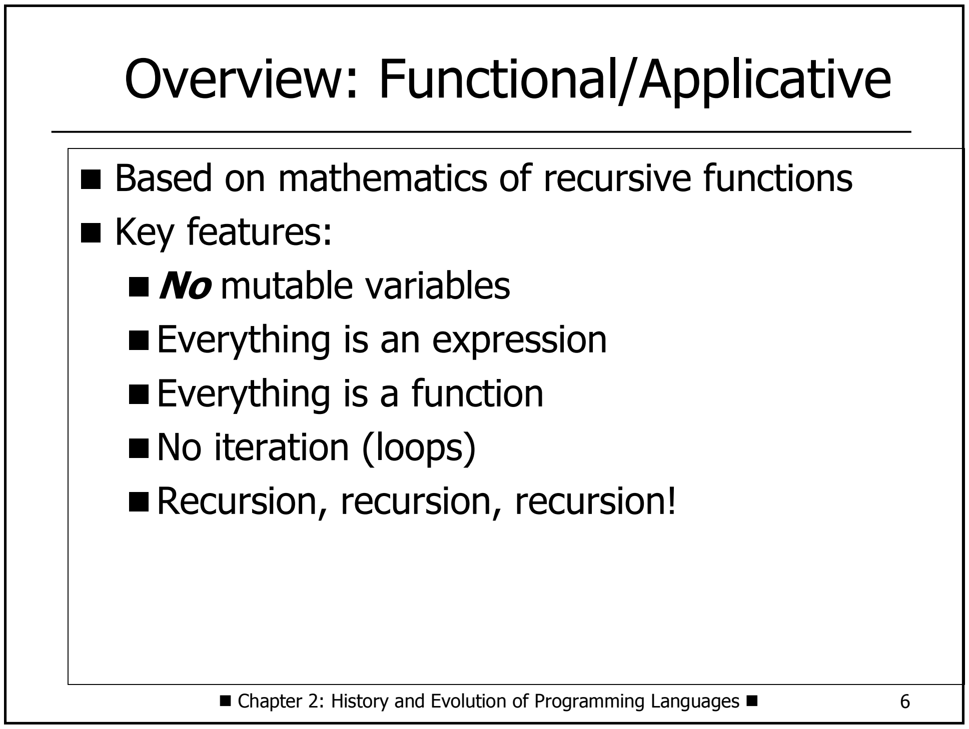


- FORTRAN (54-57, and on and on):
  - First widely used compiled language
  - Relatively efficient
- LISP (56-62):
  - First functional language, first support for recursion, activation records, run-time stack
  - First garbage collector, implicit dynamic memory mgmt.
  - Interpreter-based



Overview: Procedural/Imperative

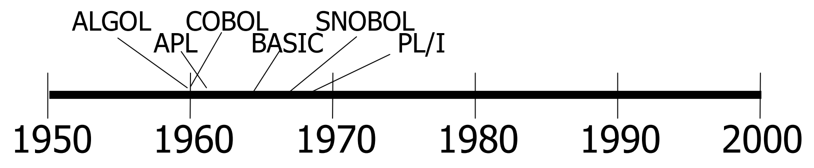
- Describes ***how*** the computer should achieve solution
- Key features:
  - Stored memory
  - Mutable variables
  - Sequencing, selection, iteration
  - Pointers?



Overview: Functional/Applicative

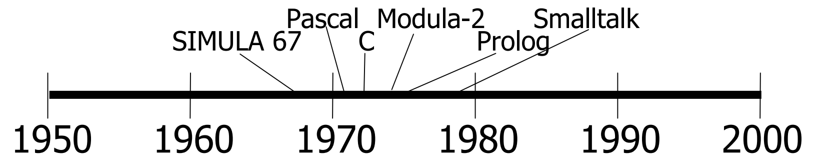
- Based on mathematics of recursive functions
- Key features:
  - ***No*** mutable variables
  - Everything is an expression
  - Everything is a function
  - No iteration (loops)
  - Recursion, recursion, recursion!

1960's: Elaboration and Analysis

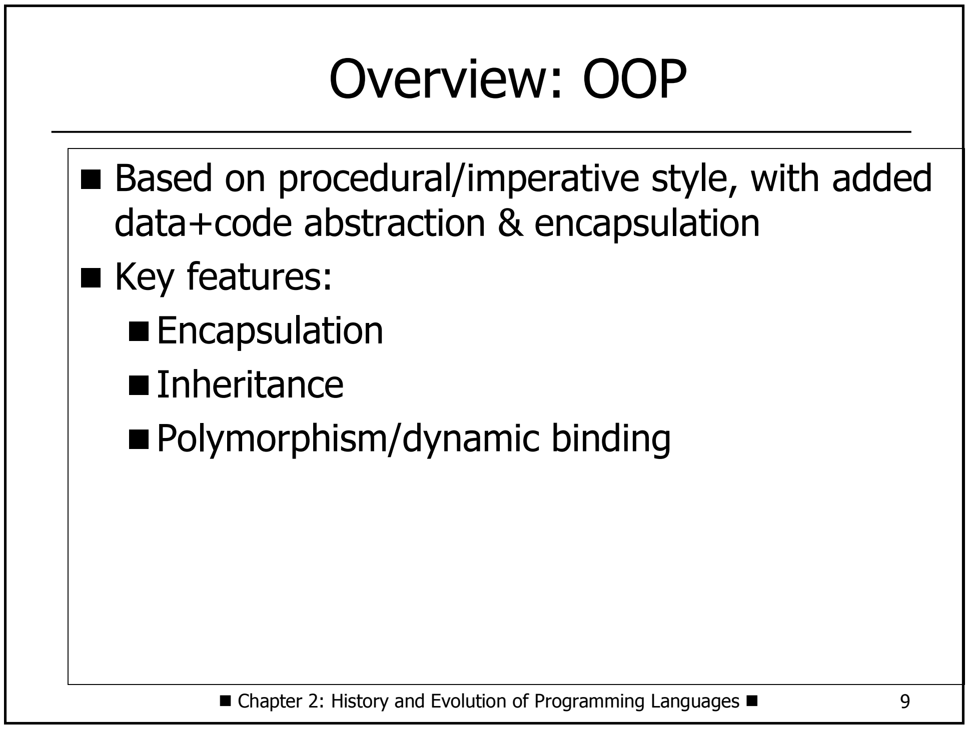


- ALGOL 58, 60: first universal language. NEW: BNF, block structure, call-by-value, stack-based evaluation, stack-based arrays
- APL: applicative, no precedence, interpreted
- COBOL:English-style syntax, records in files
- BASIC: interactive time-sharing terminals
- SNOBOL: pattern matching
- PL/I: the kitchen sink

1970's: Technology

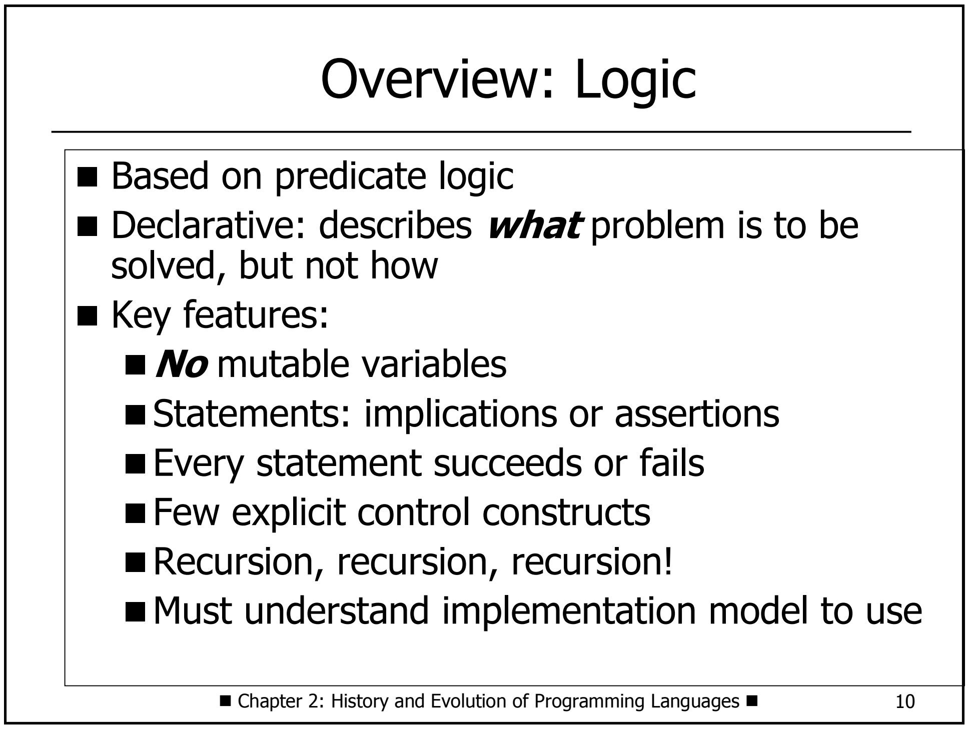


- SIMULA 67: classes, inheritance, data abstraction
- Pascal: small, elegant, structured programming, teaching
- C: systems programming, efficiency
- Modula-2: Pascal + modules, better for systems programming
- Prolog: first logic language, AI-oriented
- Smalltalk: pure OO, interpreted, entire system



Overview: OOP

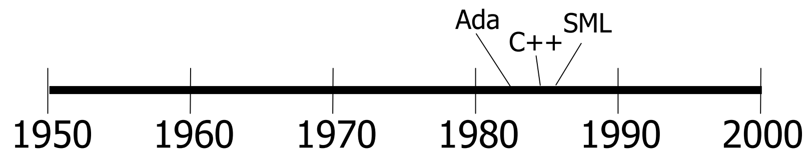
- Based on procedural/imperative style, with added data+code abstraction & encapsulation
- Key features:
  - Encapsulation
  - Inheritance
  - Polymorphism/dynamic binding



Overview: Logic

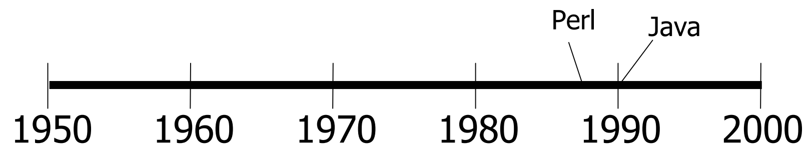
- Based on predicate logic
- Declarative: describes ***what*** problem is to be solved, but not how
- Key features:
  - ***No*** mutable variables
  - Statements: implications or assertions
  - Every statement succeeds or fails
  - Few explicit control constructs
  - Recursion, recursion, recursion!
  - Must understand implementation model to use

1980's: New Paradigms



- Ada: DoD, long committee-based development, large & complex, packages, tasks, generics, exceptions, from real-time to payroll apps.
- C++: OOP in a popular, widespread language, often seen as a "hybrid"
- Standard ML, Hope, Miranda, Haskell: functional languages

1990's: Internet Influences



- Scripting: Perl, TCL, Visual Basic, JavaScript, Python, …
- Java: designed for portable binaries and internet use, "clean" OO compared to C++, garbage collection, compiled/interpreted hybrid

Recap of Paradigms

- Procedural/Imperative
- Functional/Applicative
- Logic
- Object-oriented (closely related to imperative)
- Problem-oriented/application-specific

Paradigms: Key Differentiating Factors

- What distinguishes one paradigm from another?

Languages: Key Differentiating Factors

- What distinguishes one language from another?