G22.2210

Programming Languages: PL I

B. Mishra
New York University.

Lecture # 1

Room 801, Warren Weaver Hall
251 Mercer Street
Tel.# 212.998.3464 (Voice Mail)
E-mail:`mishra@cs.nyu.edu`
WWW: See `http://cs.nyu.edu/`

—Slide 1—

Text Books

- RAVI SETHI, *Programming Languages—Concepts and Constructs*, (2nd Edition), Addison Wesley. ISBN 0-201-59065-4.
- **Other Recommended Books**
  - RYAN STANSIFER, *The study of Programming Languages*, Prentice hall, Inc, Engelwood Cliff. ISBN 0-13-726936-6.
  - BJARNE STROUSTROUP, `C++`*: Programming Language*, Addison Wesley Publishers.ISBN 0-201-12078-X.
  - JAMES GOSLING, *Java Language Specification*, Addison Wesley. ISBN 0-201-63451-1.
  - J. BARNES, *Programming in ADA 95* Addison Wesley. ISBN 0-201-87700-7.
  - GUY STEELE, *Common Lisp Manual*, Digital Press. ISBN 0-932376-41-X.
  - STEPHEN WOLFRAM *MATHEMATICA: A System for Doing Mathematics*, Addison Wesley. ISBN 0-201-51507-5.

—Slide 2—

Administrivia

- **Office Hours**
  12:00 noon–1:00pm, Thursday
  Room 801: Warren Weaver Hall
  (Also, by appointment)
- **Tests and Assignments**
  1. Home Works (Some programming)
     About 4 Assignments
     40% of the total Grade
  2. Mid-Term (1 Hr Quiz, Closed Book)
     20% of the total Grade
  3. Final (2 Hrs, Closed Book)
     40% of the total Grade

—Slide 3—

Who is interested in Programming Languages?

- **Users**—*Issues:*
  - **Ease of Programming**
  - **Productivity**

    i) Prototyping, ii) Optimization, iii) Correctness, iv) Flexibility v) Reusability
  - **Readability**

    i) Self-documenting, ii) Syntax, iii) Semantics
  - **Interface**

    i) Libraries, ii) Other languages

—Slide 4—

Language Designers

- *Issues:*
  - **Lexical Structure**
  - **Syntactic Structure**
  - **Semantic Structure**

    i) Every feature has a well-defined meaning.

    ii) Interaction among the features
  - **Operations**

    i) Bit Operations, ii) Typed First-Class Objects:

    `int`, `Boolean`, `float`, `double`, ...,

    iii) Objects, iv) S-expressions, v) Logical expressions vi) Rules and Patterns, vii) Sets

    —Low level vs. High level—
  - **Interactive/Dynamic vs. Static**
  - **Imperative vs. Declarative**

—Slide 5—

Compiler Writers

- *Issues:*
  - **Separation between Language & Machine-dependent Features** —Portability—
  - **Compile Time vs. Run Time**
  - **System Interface**

    i) Operating System, ii) File System, iii) Separate Compilation, iv) Programming Environment (Version Management, Editor, Debugger, Profiler)
  - **User Interface**

    i) Error reporting/Recovery, ii) Assertions & Checks, iii) Options Control, iv) Preprocessing
  - **Performance**

    i) Compiler (Size/Speed), ii) Generated Code (Size/Speed)
  - **Compiler-Compiler**

—Slide 6—

Why are we Interested in Programming Languages?

- Mostly as a <u>**USER**</u>
  Recognize and understand common features

  1. Makes it easy to learn
  2. Understand common idioms/phrases
     (Stylized and Readable Code)
  3. Improve Productivity

—Slide 7—

A Brief History of Programming Languages

Early Languages

- *Algebraic Interpreter*: MIT for the Whirlwind computer (1950).
- *A-2*: Grace Murray Hopper. For UNIVAC. (1950)
- **FORTRAN**: FORmula TRANslator. John Backus & his team at IBM. (1953)

  ◦ Simple compiler capable of producing efficient code. First Fortran compiler construction: 1955–1957.

  ◦ Standardized by ANSI (American National Standards Institute) in 1966 and by ISO (International Standards Organization) in 1978. Latest standard—ANSI X3.198–1992, September 1992.

—Slide 8—

History (Contd)

- **ALGOL**: ALGOrithmic Language. Committee consisting of GAMM (Gesellschaft für angewandte Matheamtik und Mecahnik) and ACM. 1958–1960. ALGOL 60.
  - Block structure
  - Recursive Procedures
  - Data Types
  - Scope Rules (Lexical)
  - ALGOL 68—Parallel computing, Semaphores, Coercion, Overloading, Flexible Arrays
  - Hard to implement.

—Slide 9—

History (Contd)

- *LISP*: LISt Processing Language. John McCarthy & his colleagues in MIT. Late 1950s

  ○ A list of symbols: E.g., word in a sentence, attributes, algebraic expressions
  ○ List represents both data and expressions (Cambridge Polish Notation.)—S-expressions
  ○ Functional Programming

  ○ Dialects: MacLisp, FranzLisp, BBNLisp...
  ○ Standardization: Common LISP and Scheme. 1975–1985.

- *SIMULA*: Royal Norwegian Council. Kristen Nygaard & Ole-Johan Dahl, Early 60s.

  ○ Simulation of Physical entities
  ○ Objects with independent existence–OOPS. SIMULA 67 ○ Beta: successor of Simula in the 90's

—Slide 10—

History (Contd)

- **COBOL**: CODASYL (Committee on Data Systems Language) executive committee, sponsored by US DoD. May 1959

  ∘ Mainly for business Data processing applications (e.g., payroll)

- **APL**: A Programming Language. Ken Iverson. Early 60s.

  ∘ Succinct representation obtained by compositional application of operations on high-level structures such as arrays and matrices.

- *SNOBOL*: David Farber, Ralph Griswold & Ivan Polonsky at Bell Labs. Mid 60s.

  ∘ Processing of string data
  ∘ Concepts of generators. Pattern matching by unification
  ∘ Icon: successor of SNOBOL in the mid 80's

—Slide 11—

History (Contd)

Recent Languages

- **PL/I**: Advanced Language Development Committee sponsored by IBM. Mid 60s.

  ◦ Goal was to replace FORTRAN and COBOL by a more powerful language.
  ◦ ANSI standard in 1976. "General Purpose Subset" in 1981.
  ◦ As a language it failed, mostly due to its complexity.

- **BASIC**: Beginner's All-Purpose Symbolic Instruction Code. John Kemeny and Thomas Kurtz of Dartmouth College. Early 70s.

  ◦ Easy to learn, code and debug.
  ◦ ANSI Standard in 1978. True BASIC in 1985.

—Slide 12—

History (Contd)

- `C`: Dennis Ritchie at Bell Labs. 1972. Influenced by `B` and `BCPL` (Ken Thompson).

  ◦ Low level language. With greater access to hardware environment (necessary in applications such as graphics, networking, etc.)
  ◦ Good interface with operating system (`UNIX` and the libraries)
  ◦ ANSI standard in 1989.

- **BLISS**: Bill Wulf and Nico Habermann of Carnegie-Mellon. Mid 70s.

  ◦ Low level abilities.
  ◦ Optimizing compiler for BLISS 11 to run on PDP 11.

—Slide 13—

History (Contd)

Structured Languages

- **Pascal**: Niklaus Wirth, 1971.
  - ◦ Influenced by ALGOL 68 (ALGOL W).
  - ◦ Simple, 1 Pass compiler. First compiler written in FORTRAN by a single graduate student.
  - ◦ Pedagogic language. Used for teaching structured program development via *stepwise refinement*.
  - ◦ User-defined types.
  - ◦ Standard Pascal lacks dynamic arrays. Makes its hard to use for many real applications.
- **Modula**: Niklaus Wirth. Early 80s.
  - ◦ Allows programs to be organized in *modules*
  - ◦ Some amount of *encapsulation*

—Slide 14—

History (Contd)

Object Oriented Languages

- **Smalltalk**: Alan Kay, Xerox Parc (KiddiKomp project). Summer of 71.

  ◦ Object Oriented Programming (OOP).
  ◦ Concept of *classes*. Object is an instance of a class.
  ◦ Classes are organized in a *class hierarchy*. Properties can be inherited from super/base classes.

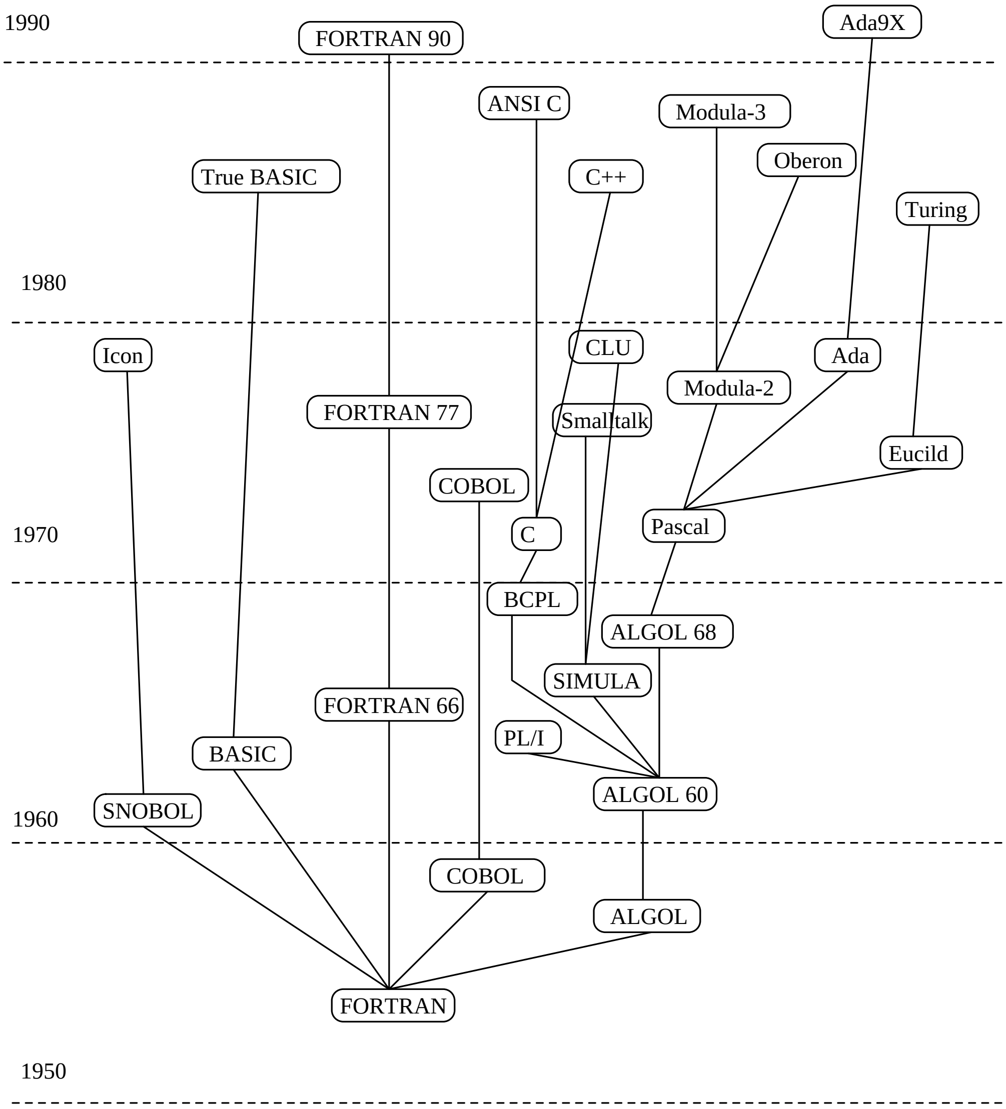
- `C++`: Bajrne Stroustroup. 1982–1985.

  ◦ Influenced by SIMULA. Evolved from an earlier version called "`C` with classes."
  ◦ Class mechanisms, Inheritance, Friend-mechanisms. Template (polymorphisms).
  ◦ Powerful, yet efficient.

- **Ada**: Jean Ichbiah of Honeywell of France, 1983.

—Slide 15—

Imperative Languages



—Slide 16—

Turing Awards

Or, Why you may want to design a Programming Language.

Name	Language	Year
John McCarthy	LISP	1971
John Backus	Fortran	1977
Ken Iverson	APL	1979
Dennis Ritchie	`C`	1983
Ken Thompson	`B` & `C`	1983
Niklaus Wirth	Pascal	1984
Robin Milner	ML	1992

—Slide 17—

The INTERCAL Programming Language

- Donald R. Woods & James M. Lyons
- The full name of the compiler is "Compiler Language With No Pronounceable Acronym," which is, for obvious reasons, abbreviated to "INTERCAL."