Chapter One

Preliminaries, including
- –Why study PL concepts?
- –Programming domains
- –PL evaluation criteria
- –What influences PL design?
- –Tradeoffs faced by programming languages
- –Implementation methods
- –Programming environments

Why study Programming Language Concepts?

- Increased capacity to express programming concepts
- Improved background for choosing appropriate languages
- Increased ability to learn new languages
- Understanding the significance of implementation
- Increased ability to design new languages
- Overall advancement of computing

Programming Domains

- Scientific applications
- Business applications
- Artificial intelligence
- Systems programming
- Scripting languages
- Special purpose languages

Language Evaluation Criteria

- Readability
- Writability
- Reliability
- Cost
- Etc…

Evaluation Criteria: Readability

How is it for one to read and understand programs written in the PL?

Arguably the most important criterion!

Factors effecting readability include:

- Overall simplicity
  - » Too many features is bad as is a multiplicity of features
- Orthogonality
  - » Makes the language easy to learn and read
  - » Meaning is context independent
- Control statements
- Data type and structures
- Syntax considerations

Evaluation Criteria: Writability

How easy is it to write programs in the language?

Factors effecting writability:

- Simplicity and orthogonality
- Support for abstraction
- Expressivity
- Fit for the domain and problem

Evaluation Criteria: Reliability

Factors:

- Type checking
- Exception handling
- Aliasing
- Readability and writability

Evaluation Criteria: Cost

Categories:

- Programmer training
- Software creation
- Compilation
- Execution
- Compiler cost
- Poor reliability
- Maintenance

Evaluation Criteria: others

Portability
Generality
Well-definedness
Etc…

Language Design Influences
Computer architecture

- We use imperative languages, at least in part, because we use von Neumann machines
  - John von Neuman is generally considered to be the inventor of the "stored program" machines - the class to which most of today's computers belong.
  - CPU+memory which contains both program and data
- Focus on moving data and program instructions between registers in CPU to memory locations

Language Design Influences:
Programming methodologies

- *50s and early 60s:* Simple applications; worry about machine efficiency
- *Late 60s:* People efficiency became important; readability, better control structures. maintainability
- *Late 70s:* Data abstraction
- *Middle 80s:* Object-oriented programming
- *95-today:* distributed programs, the web

Language Categories

The big four:
- Imperative or procedural (e.g., Fortran, C)
- Functional (e.g., Lisp, ML)
- Rule based (e.g. Prolog)
- Object-oriented (e.g. Smalltalk, Java)

Others:
- Scripting (e.g., Perl, Tcl/Tk)
- Constraint (e.g., Eclipse)

Language Design Trade-offs

Reliability versus cost of execution

Ada, unlike C, checks all array indices to ensure proper range.

Writability versus readability

(2 = 0 +.= T o.| T) / T <- iN is an APL one liner that produces a list of the prime numbers from 1 to N inclusive.

Flexibility versus safety

C, unlike Java, allows one to do arithmetic on pointers.

Implementation methods

- Direct execution by hardware
  - E.g., machine language
- Compilation to another language
  - e.g., C
- Interpretation
  - Direct execution by software
  - E.g., csh, Lisp (traditionally)
- Hybrid
  - Compilation to another language (aka bytecode) which is then interpreted
  - e.g., Java, Perl

Implementation issues

- Complexity of compiler/interpreter
- Speed of translation
- Speed of execution
- Portability of translated code
- Compactness of translated code
- Debugging ease

compile hybrid interpret

Programming Environments

The collection of tools used in software development, often including an integrated editor, debugger, compiler, collaboration tool, etc.

Examples:

- UNIX -- Operating system with tool collection
- EMACS – a highly programmable text editor
- Borland C++ -- A PC environment for C and C++
- Smalltalk -- A language processor/environment
- Microsoft Visual C++ -- A large, complex visual environment
- Your favorite Java environment: Jbuilder, J++, …

Summary