CSE 325/CSE 425: Concepts of Programming Language

Introduction to Programming Language Concepts

Dr. Kamruddin Nur

Associate Professor, Computer Science

kamruddin.nur@gmail.com

January, 2018

Contents

Textbook and Ref. Books

Textbook:

- **Concepts of Programming Languages**, 10th Edition, By Robert W. Sebesta, Pearson

Reference books:

- **Comparative Programming Languages** by Leslie B. Wilson, Robert G. Clark, Addison-Wesley
- **Programming Language Concepts and Paradigms** by David Watt, Prentice Hall

Programming Languages

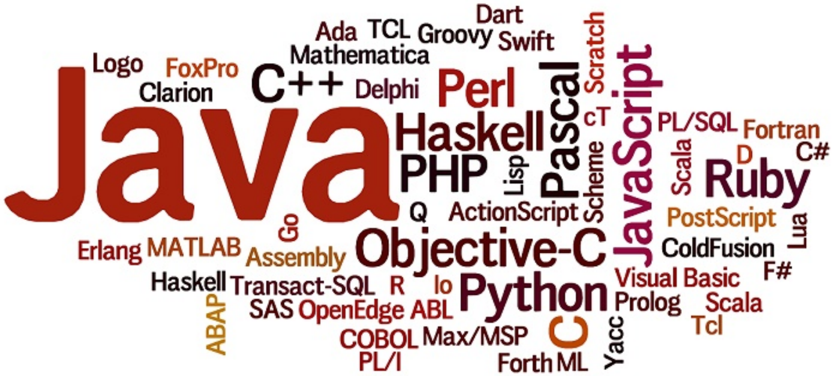


Figure 1: **Programming Languages**
(Figure source: What Programming Skills and Languages Are Needed? ernestdempsey.com)

Programming Domains

Ⓐ **Scientific Applications**

- first scientifc applications were developed in late 1940s and early 1950s
- first language for scientific applications - Fortran
- And later ALGOL 60

Ⓑ **Business Applications**

- began in the 1950s
- first successful high-level lanaguge - COBOL (appeared in 1960)
- Currently dominant languages are - C++, Java, C#

Ⓒ **Artificial Intelligence**

- LISP (prior to 1990)
- Prolog (Appeared in 2003)
- some AI applications were written in C also!

Ⓓ **Systems Programming**

- PL/S, a dialect of PL/I used in IBM Mainframe (in 1960s and 70s)
- However, most system softwares are now written in C, C++
- UNIX - written almost entirely in C (ISO, 1999)

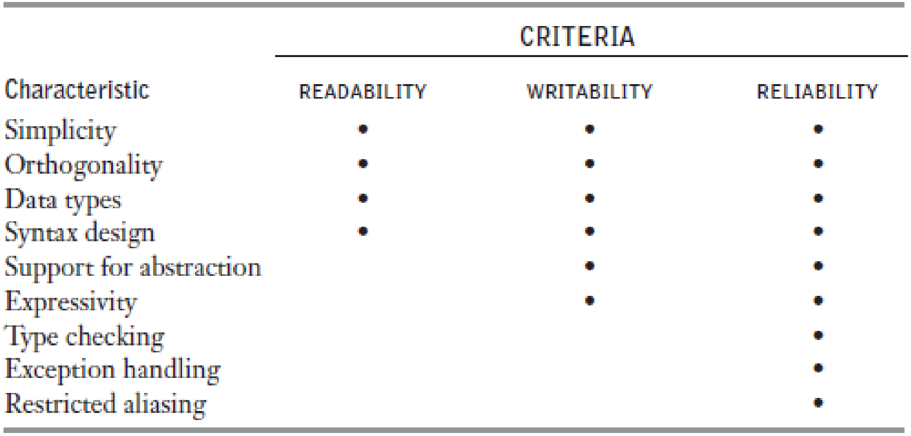
Programming Domains *(Cont'd)*

- **E** **Web Applications**
  - HTML
  - JavaScript, PHP

Why Studying Concepts of Programming Languages

- Increased capacity to express ideas
- Improved background for choosing appropriate languages
- Increased ability to learn new languages
- Better understanding of the significance of implementation
- Better use of languages that are already known
- Overall advancement of computing

Language Evaluation Criteria and Characteristics



	CRITERIA		
Characteristic	READABILITY	WRITABILITY	RELIABILITY
Simplicity	•	•	•
Orthogonality	•	•	•
Data types	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

Figure 2: **Language evaluation criteria and the characteristics that affect them**
(Figure source: Concepts of Programming Languages by Robert W. Sebesta, 10th Edition, Pearson, p. 8)

Influences on Language Design

In addition to factors described in last slide, the most influential factors are -

1. Computer Architecture
   - Most popular languages in past 50 years are meant to run on prevalent **von Neumann architecture**
   - In a von Neumann architecture, both data and programs are stored in the same memory
   - both data and instructions are fetched from memory to CPU for processing and results are stored back to memory after processing
   - languages those are designed to work with von Neumann Architecture are called **Imperative (procedural) languages**
   - imperative programming changes state with commands in the source code
   - sequence of statements to reach a certain goal
   - central features of imperative languages are -
     - variable $\rightarrow$ **memory cells**
     - assignment statement $\rightarrow$ **piping operation**
     - machine code execution $\rightarrow$ **fetch-execute cycle**

Influences on Language Design *(Cont'd)*

- **Functional Languages**
  - mathematical functions as a basic building block in the language, for example, LISP, Earlang, Haskell, F#
  - same value for an argument x produces the same result $f(x)$ each time
  - largely been emphasized in academia rather than in commercial software development
- most mainstream languages such as C++, Java, C# are primarily designed to support imperative (procedural) programming

2. **Programming Design Methodologies**
   - **Top-down design** and **stepwise refinement** (emerged in 1970s)
   - shift from procedure-oriented to data-oriented pro- gram design (emphasis on data design, focusing on abstract data types) (in late 1970s)
   - visual languages such as .NET languages (emerged in early 2000s)
   - emergence of markup languages - HTML, XML
   - emergence of scripting languages - Perl, JavaScript, and Ruby
   - emergence of Java Server Pages Standard Tag Library (JSTL) and eXtensible Stylesheet Language Transformations (XSLT) etc

Language Implementation Methods

1. **Compilation**
   - programs can be **translated into machine language**, which can be executed directly on the CPU
   - e.g. C, COBOL, C++, and Ada
2. **Pure Interpretation**
   - programs are **interpreted by an interpreter** (a program), with no translation
   - e.g. LISP, JavaScript, PHP
3. **Hybrid Implementation**
   - **translates** high-level language programs to an **intermediate language** that can be **interpreted**
   - e.g. Java, Perl

Language Implementation Methods *(Cont'd)*

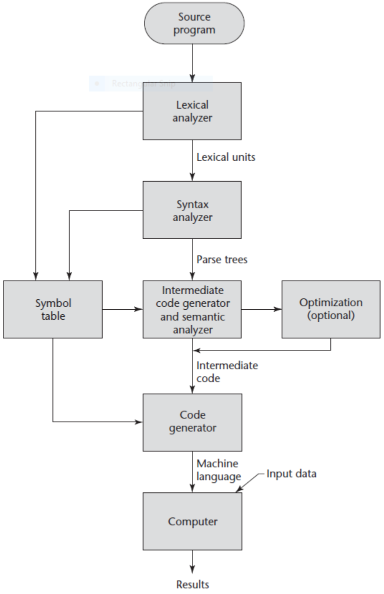


Figure 3: **Compilation Process**
(Figure source: Concepts of Programming Languages by Robert W. Sebesta, 11th Edition, Addison-Wesley, p. 26)

Language Implementation Methods *(Cont'd)*

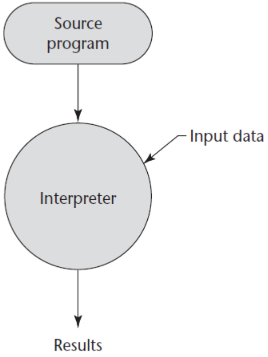


Figure 4: **Pure Interpretation Process** *(Figure source: Concepts of Programming Languages by Robert W. Sebesta, 11th Edition, Addison-Wesley, p. 28)*

Language Implementation Methods *(Cont'd)*

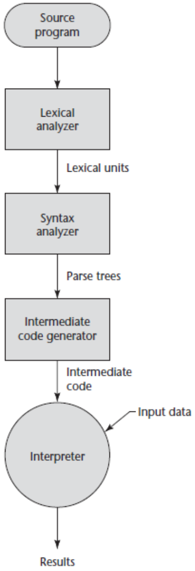


Figure 5: **Hybrid Implementation System** *(Figure source: Concepts of Programming Languages by Robert W. Sebesta, 11th Edition, Addison-Wesley, p. 29)*

Thanks

Thanks for your time and attention!

kamruddin.nur@gmail.com
researchgate.net/profile/Kamruddin_Nur