PRINCIPLES OF PROGRAMMING LANGUAGES

Prepared By:

Dr. Shanmugasundaram

Assistant Professor

CS Department

Jazan University

❖ *Reasons for Studying Concepts of Programming Languages*

- 1 Increased ability to express ideas.
- 2 Improved background for choosing appropriate languages.
- 3 Increased ability to learn new languages.
- 4 Better understanding of significance of implementation.
- 5 Better use of languages that are already known.
- 6 Overall advancement of computing.

❖ *Programming Domains*

1 - ***Scientific Applications***
  - Large numbers of floating point computations; use of arrays.
  - Example:Fortran.
2 - ***Business Applications***
  - Produce reports, use decimal numbers and characters.
  - Example:COBOL.
3 - ***Artificial intelligence***
  - Symbols rather than numbers manipulated; use of linked lists.
  - Example:LISP.

❖ ***Programming Domains***

4 - ***System programming***
  - *Need effieciency because of continous use.*
  - *Example:C*

5 - ***Web Software***
  - *Eclectic collection of languages: markup(example:XHTML),scripting(example:PHP), general-purpose(example:JAVA).*

❖ *Language Evaluation Criteria*

1 - ***Readability***:
   - ➢ The ease with which programs can be read and understood.

2 - ***Writability***:
   - ➢ The ease with which a language can be used to create programs.

3 - ***Reliability***:
   - ➢ Conformance to specifications (i.e., performs to its specifications).

4 - **Cost**:
   - ➢ The ultimate total cost.

❖ Evaluation Criteria: Readability

1 ➔ ***Overall simplicity***

- ◆ *A manageable set of features and constructs.*
- ◆ *Minimal feature multiplicity .*
- ◆ *Minimal operator overloading.*

2 ➔ ***Orthogonality***

- ◆ *A relatively small set of primitive constructs can be combined in a relatively small number of ways*
- ◆ *Every possible combination is legal*

3 ➔ ***Data types***

- ◆ *Adequate predefined data types.*

❖ ***Evaluation Criteria:Readability***

4 ➔ ***Syntax considerations***

-*Identifier forms:flexible composition.*

-*Special words and methods of forming compound statements.*

-*Form and meaning:self-descriptive constructs,meaningful keywords.*

❖ *Evaluation Criteria: Writability*

1- **_Simplicity and orthogonality_**
  - Few constructs, a small number of primitives, a small set of rules for combining them.

2- **_Support for abstraction_**

-The ability to define and use complex structures or operations in ways that allow details to be ignored.

3- **_Expressivity_**
  - A set of relatively convenient ways of specifying operations.
  - Strength and number of operators and predefined functions.

❖ *Evaluation Criteria: Reliability*

1. • ***Type checking***
   – Testing for type errors.
2. • ***Exception handling***
   – Intercept run-time errors and take corrective measures.
3. • ***Aliasing***
   – Presence of two or more distinct referencing methods for the same memory location.
4. • ***Readability and writability***
   – A language that does not support "natural" ways of expressing an algorithm will require the use of "unnatural" approaches, and hence reduced reliability.

❖ *Evaluation Criteria: Cost*

- 1 Training programmers to use the language
- 2 Writing programs (closeness to particular applications)
- 3 Compiling programs
- 4 Executing programs
- 5 Language implementation system: availability of free compilers
- 6 Reliability: poor reliability leads to high costs
- 7 Maintaining programs

❖ *Language Categories*

- **Imperative**
  - Central features are variables, assignment statements, and iteration
  - Include languages that support object-oriented programming
  - Include scripting languages
  - Include the visual languages
  - Examples: C, Java, Perl, JavaScript, Visual BASIC .NET, C++
- **Functional**
  - Main means of making computations is by applying functions to given parameters
  - Examples: LISP, Scheme
- **Logic**
  - Rule-based (rules are specified in no particular order)
  - Example: Prolog
- **Markup/programming hybrid**
  - Markup languages extended to support some programming
  - Examples: JSTL, XSLT

❖ *Language Design Trade-Offs*

- Reliability vs. cost of execution
  - Example: Java demands all references to array elements be checked for proper indexing, which leads to increased execution costs

- Readability vs. writability

  Example: APL provides many powerful operators (and a large number of new symbols), allowing complex computations to be written in a compact program but at the cost of poor readability

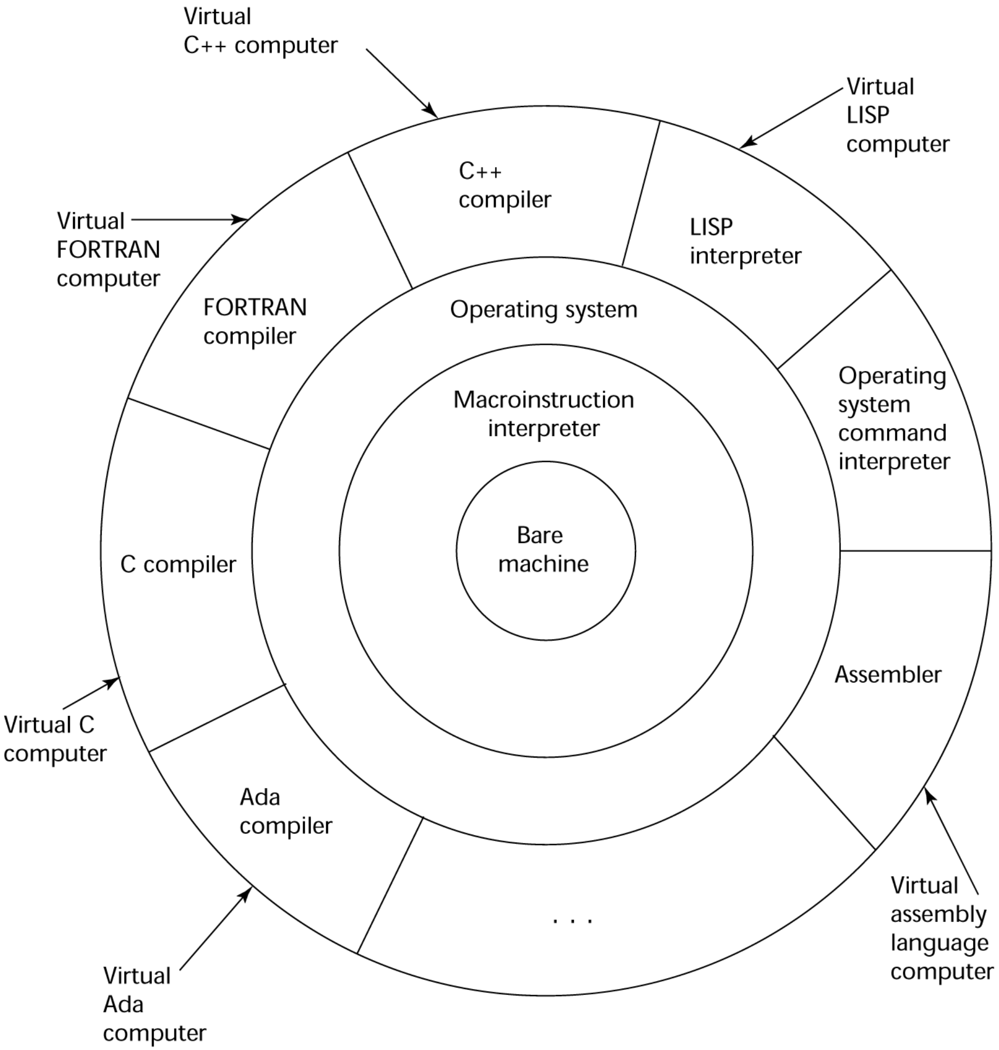
- Writability (flexibility) vs. reliability
  - Example: C++ pointers are powerful and very flexible but are unreliable

❖ *Implementation Methods*

- ***Compilation***
  - Programs are translated into machine language
- ***Pure Interpretation***
  - Programs are interpreted by another program known as an interpreter
- ***Hybrid Implementation Systems***
  - A compromise between compilers and pure interpreters

❖ Layered View of Computer

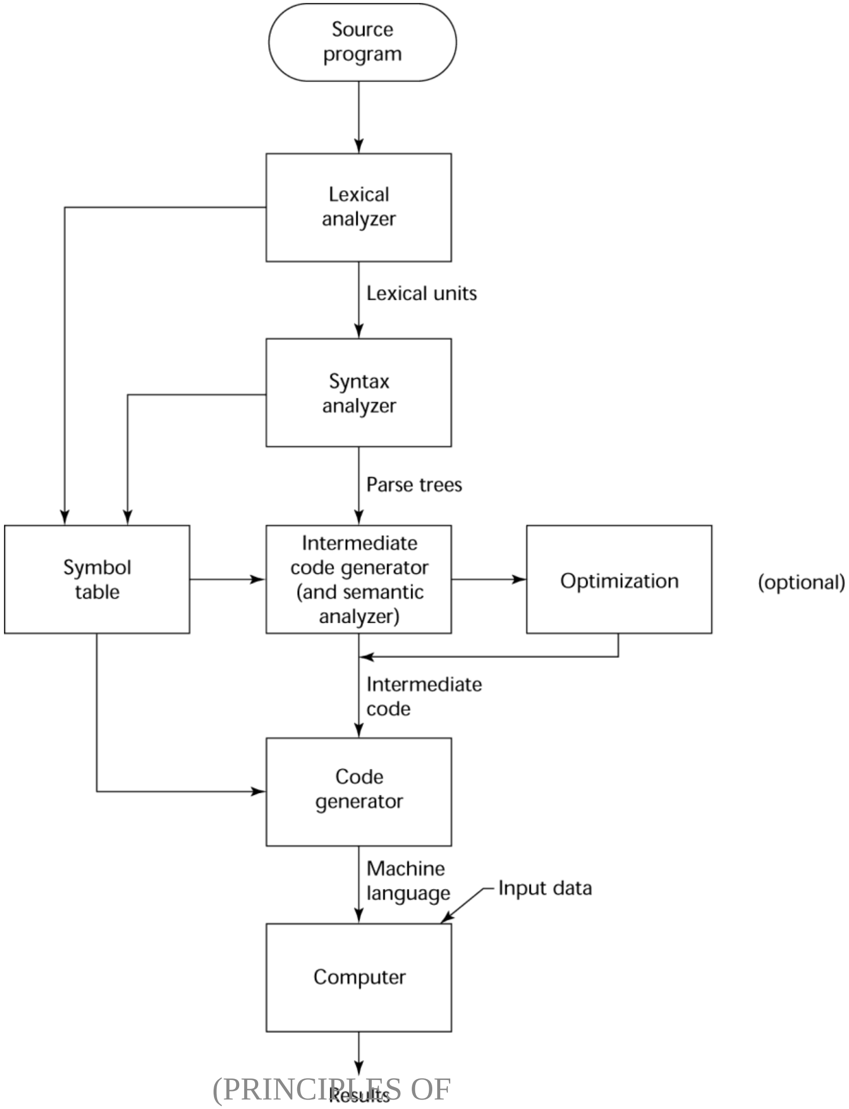
The operating system and language implementation are layered over machine interface of a computer



Compilation

- Translate high-level program (source language) into machine code (machine language)
- Slow translation, fast execution
- Compilation process has several phases:
  1 – lexical analysis: converts characters in the source program into lexical units
  2 – syntax analysis: transforms lexical units into *parse trees* which represent the syntactic structure of program
  3 – Semantics analysis: generate intermediate code
  4 – code generation: machine code is generated

The Compilation Process



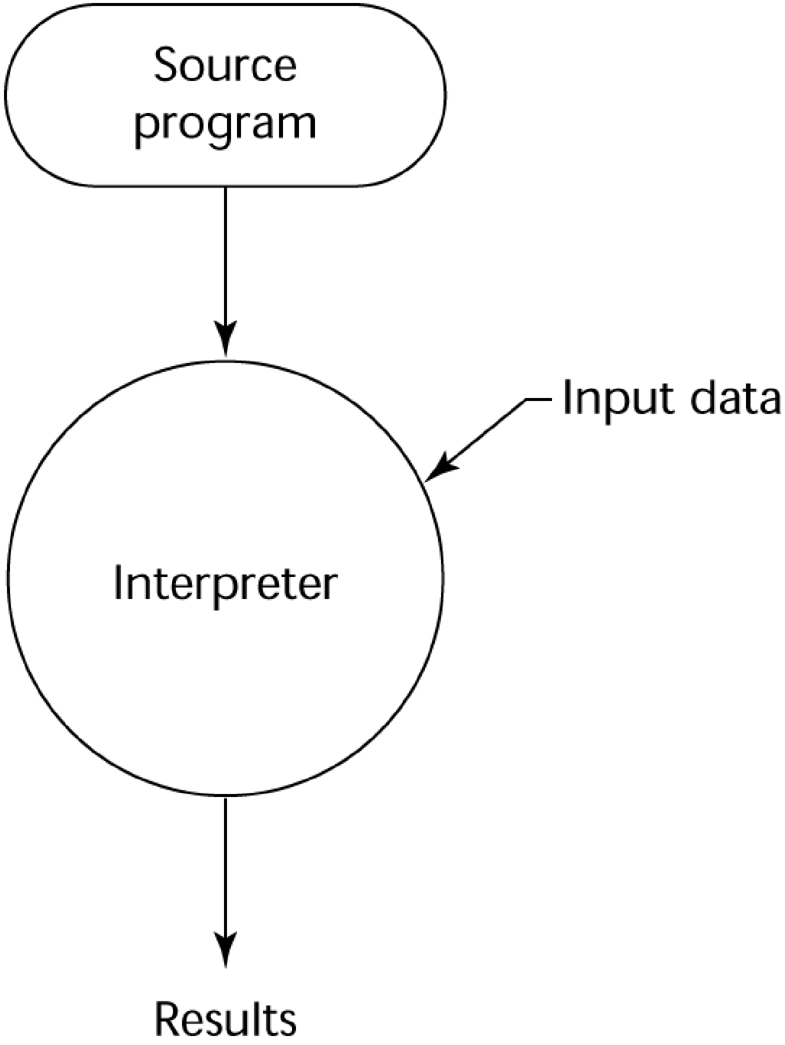
Additional Compilation Terminologies

- **Load module** (executable image): the user and system code together
- **Linking and loading**: the process of collecting system program units and linking them to a user program

Pure Interpretation

- No translation
- Easier implementation of programs (run-time errors can easily and immediately be displayed)
- Slower execution (10 to 100 times slower than compiled programs)
- Often requires more space
- Now rare for traditional high-level languages
- Significant comeback with some Web scripting languages (e.g., JavaScript, PHP)

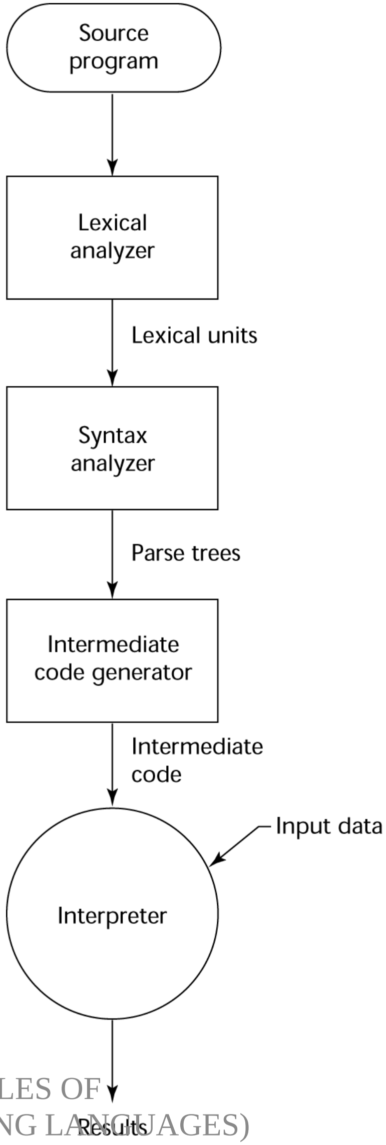
Pure Interpretation Process



Hybrid Implementation Systems

- A compromise between compilers and pure interpreters
- A high-level language program is translated to an intermediate language that allows easy interpretation
- Faster than pure interpretation
- Examples
  - Perl programs are partially compiled to detect errors before interpretation
  - Initial implementations of Java were hybrid; the intermediate form, *byte code*, provides portability to any machine that has a byte code interpreter and a run-time system (together, these are called *Java Virtual Machine*)

Hybrid Implementation Process



Just-in-Time Implementation Systems

- 1 Initially translate programs to an intermediate language
- 2 Then compile the intermediate language of the subprograms into machine code when they are called
- 3 Machine code version is kept for subsequent calls
- JIT systems are widely used for Java programs
- .NET languages are implemented with a JIT system

Preprocessors

- Preprocessor macros (instructions) are commonly used to specify that code from another file is to be included
- A preprocessor processes a program immediately before the program is compiled to expand embedded preprocessor macros
- A well-known example: C preprocessor
  - expands `#include`, `#define`, and similar macros

Programming Environments

- A collection of tools used in software development
- UNIX
  - An older operating system and tool collection
  - Nowadays often used through a GUI (e.g., CDE, KDE, or GNOME) that runs on top of UNIX
- Microsoft Visual Studio.NET
  - A large, complex visual environment
- Used to build Web applications and non-Web applications in any .NET language
- NetBeans
  - Related to Visual Studio .NET, except for Web applications in Java