Sudan University of Science and Technology

Concepts of programing Languages

Lecture1 : Introduction

Dr. Aghabi Nabil Abosaif
06/10/2021

Lecture Contents

- Why studying concepts of Programming Language(PL)?
- Programming Domains.
- Language Evaluation Criteria.

Why studying concepts of Programming Language?

- What are the benefits from the study of programming language concepts.
  - In this lecture some potential benefits of studying concepts of programming languages will be discussed.

1.Increased capacity to express ideas

- It is widely believed that *the depth at which people can* ***think is influenced by the expressive power*** *of the language in which they communicate their thoughts.*

- Study of PL concepts **build an appreciation for valuable** language features and constructs and encourages programmers to use them, even when the language they are using **does not directly support** such features and constructs.

2. Improved background for choosing appropriate languages

- Many programmers, when given **a choice of languages** for a new project, use the language with which they are **most familiar**, even if it is **poorly suited** for the project .

- Generally, It is preferable to use a **feature whose design** has been integrated into a language *than* to use a **simulation of that** feature, which is often less elegant, more cumbersome, and less safe.

3.Increased ability to learn new languages

- Design methodologies, software development tools, and PL are **still in a state of continuous evolution**.
- This makes software development an **exciting profession**, but it also means that **continuous learning** is essential.

- Once understanding of the fundamental concepts of languages, it becomes **far easier to see how these concepts are incorporated into the design** of the language being learned.
  - Ex. Object Oriented Programming (OOP) concepts.

- Also, it is essential that practicing programmers know the **vocabulary** and **fundamental** concepts , so they can ***read and understand*** PL descriptions and evaluations, as well as promotional literature for languages and compilers.

4. Better understanding of the significance of implementation

- Understanding of implementation issues **leads to**:

Increase the ability to use a language more **intelligently**, as it was designed to be used.

  - We can become better programmers by **understanding** the choices among PL **constructs and the consequences**.
  - Also, Certain kinds of **program bugs** can be found and **fixed** only by a programmer who knows some related implementation details.

5. Better use of languages that are already known

- it is **uncommon** for a programmer s to be familiar with and use **all of the features** of a language they use.
- By studying the **concepts of PLs**, programmers can learn about previously unknown and **unused parts** of the languages they already use and begin to use those features.

6. Overall advancement of computing

- Although it is usually possible to determine **why a particular PL became popular**, many believe, at least in retrospect, that the **most PLs are not always the best available.**

- In some cases, it might be concluded that a language became widely used, at least in part, because those in positions **to choose languages were not sufficiently familiar with PL concepts**.

- **In general**, if those who choose languages were well informed, **perhaps better languages would eventually squeeze out poorer ones.**

Programming Domains

- Computers have been applied into **the different areas**, from controlling nuclear power plants to providing video games in mobile phones to more and more complicated applications.

- This lecture briefly discuss a few of the areas of computer applications.

1. Scientific Applications

- The first digital computers, which appeared in the late 1940s and early 1950s, were **invented and used** for scientific applications.
- The first language for scientific applications was **Fortran**.

- Typically, the scientific applications of that time used **relatively simple data structures**, but required large numbers of floating- point arithmetic computations.
- It advanced by the time to **content more structures** and **constructs** .

- The most common **data structures** were *arrays and matrices*; the most common **control structures** were *counting loops and selections*.

2. Business Applications

- The use of computers for business applications began in the **1950s**. *Special computers were developed* for this purpose, along with special languages.

- Business languages are characterized by facilities for:
  - Producing **elaborate reports.**
  - Precise ways of describing and storing decimal numbers and character data.
  - Ability to specify **decimal arithmetic operations**.
    - Such as COBOL, RPG

3. Artificial Intelligence (AI)

- AI **is a broad area of computer applications** characterized by the use of **symbolic rather than numeric** computations. Symbolic computation means that symbols, consisting of names rather than numbers, are manipulated.

- It requires **more flexibility** than other programming domains.
- The **first widely used** PL developed for AI applications was the functional language **Lisp**, which appeared in **1959**.

- Some AI applications have been written in systems languages such as Lisp, Prolog , and Scheme.

4.Systems Programming

- Development of computer software that is part of a computer **operating system** or other **control program**, especially as used in **computer networks**.

- Systems programming covers data and program management, including operating systems, control programs, network software, and database management systems.
- Need efficiency because of continuous use
  - IBM's PL/S, Digital's BLISS, UNIX's C.

5.Web Software

- The World Wide Web is ranging from **markup languages, such as HTML**, which is not a PL, to **general-purpose** PLs, such as Java.

- This functionality can be provided by **embedding programming code in an HTML document**. Such code is often in the form of a scripting language, such as JavaScript or PHP.

- There are also some **markup-like** languages that have been **extended to include** constructs that **control document processing**.

Language Evaluation Criteria

- The set of **evaluation criteria** which needed to evaluates the PLs **features**, focusing on their impact on the software development process,
- Such a list of criteria is necessarily controversial, because it is **difficult to get even two computer scientists** to agree on the **value** of some given language characteristic relative to others.

1.Readability

- One of the most important criteria is the ease to read and understand the programs.
- Before 1970, software development was largely thought of in terms of writing code. Language constructs were designed **more from the point of view of the compute**r than of the **computer users**.
- Readability is important because **ease of maintenance** is determined in large part by the readability of programs.
- Its became an important **measure of the quality of program**s and PLs. So, there was a distinct crossover from a focus on **machine orientation to a focus on human orientation**.
- Readability must be considered in the **context of the problem domain**. For example, if a program that describes a computation is written in a language not designed for such use, the program may **be unnatural and convoluted, making it difficult to read**.

1.1 Overall Simplicity

- Overall Simplicity strongly **affects** programs readability, a language with a **large number of basic constructs** is more **difficult to learn** than one with a smaller number.

- Also, **multiplicity**— that is, *having more than one way to accomplish a particular operation* **can disserve the readability.**

- For example, in Java, a user can increment a simple integer variable in four different ways:
  - count = count + 1 , count += 1 , count++ , ++count
- Although the last two statements have slightly different meanings from each other and from the others in some contexts, all of them have the same meaning when used as stand- alone expressions.

- A third potential problem is **operator overloading**, *in which a single operator symbol has more than one meaning*.

1.2 Orthogonality

- It means that a **relatively small set of primitive constructs** can be *combined* in a relatively **small number of ways** to build the control and data structures of the language.

- For example, consider **data types**, a language has **four primitive** data types (integer, float, double, and character) and two **type operators** (array and pointer).
- If the two type operators **can be applied to themselves** and the four primitive data types, a large number of data structures can be defined.

- Orthogonality follows from a symmetry of relationships among primitives.

1.2 Orthogonality(Cons.)

- As examples of the lack of orthogonality in a high-level language, consider the following rules in C.

  - Although C has **two kinds** of structured data types, **arrays** and **records** (structs)*, records can be returned from functions but arrays cannot*.
  - A **member** of a structure can be **any data type except void or a structure** of the same type.
  - An array **element** can be any data type except **void or a function**.

1.3 Data Types

- The presence of adequate facilities for defining data types and data structures in a language is another **significant aid to readability**.

- For example, suppose a numeric type is used for an indicator flag because there is no Boolean type in the language.

- In such a language, we might have an assignment such as the following:
  - timeOut = 1 The meaning of this statement is unclear, whereas in a language that includes Boolean types,
  - timeOut = true The meaning of this statement is perfectly clear

1.4 Syntax Design

- The syntax, or form, of the elements of a language has a **significant effect** on the readability of programs.

Following are some examples of syntactic design that affect readability:

- **Special words.** For example, while, class, and for using a **brace**.

- Most languages have **diminished** readability because statement groups are always terminated in the same way, which makes it difficult to determine which **group is being ended** when an end or a **right brace appears**.

1.4 Syntax Design(Cons.)

- • **Form and meaning**. Designing statements so that their appearance at least partially indicates their purpose is an obvious aid to readability.

- In C, for example, the meaning of the **reserved word static** depends on the context of its appearance.
  - If used on the definition of a variable **inside** a function, it means the variable is **created at compile time**.
  - If used on the definition of a variable that is **outside** all functions, it means the variable is **visible only in the file** in which its definition appears; that is, it is not exported from that file.

Writability

- It is a **measure** of how easily a language can **be used to create programs for a chosen problem domain**.
  - Most of the language characteristics that affect readability also affect writability.

- This follows directly from the fact that process of writing a program requires the programmer **frequently to reread** the part of the program that is already written.

- It is not fair to compare the writability of two languages in the realm of a particular application when one was designed for that application and the other was not.

- For example, the writabilities of Visual BASIC (VB) and C are dramatically different for creating a program that has a **Graphical User Interface (GUI)**, for which **VB** is ideal.

- Their writabilities are also quite different for **writing systems programs**, such as an **operation system**, for which **C** was designed.

2.1 Simplicity and Orthogonality

- If a language has a large number of different constructs, some programmers **might not be familiar with all of them**. This situation can lead to **a misuse of some features** and a disuse of others that may be either more elegant or more efficient, or both, than those that are used.
- A programmer can design a solution to a complex problem after learning only a simple set of primitive constructs.
- On the other hand, **too much orthogonality** can be a detriment to writability. Errors in programs can go undetected when nearly any combination of primitives is legal.
- This can lead to code **absurdities** that cannot **be discovered by the compiler**

2.2 Expressivity

- It refers to a programming language's **ability to represent ideas and algorithms clearly and effectively**.

- It means that the language provides **tools and constructs that allow programmers to write code that closely reflects their intentions**, making it easier to express different solutions in various ways.

- In a language such as APL, it means that there are **very powerful operators** that allow a great deal of **computation** to be accomplished with a **very small program.**

- More commonly, it means that a language has **relatively convenient**, rather than **cumbersome**, ways of **specifying computations**.

3.Reliability

- It`s refers to the ability of a system or application to consistently perform **its intended functions without failure over time.**

- A reliable system **minimizes bugs, handles errors gracefully, and provides accurate results**, ensuring that users can trust its performance.

- Factors like thorough **testing**, **code quality**, and **proper error handling** contribute significantly to a system's reliability.

- A program is said to **be reliable** if it **performs** to its specifications **under all conditions**.

3.1 Type Checking

- It is **simply testing for type errors** in a given program, either by the **compiler** or **during program** execution.
- **Run-time type** checking is **expensive**, but **compile-time** type checking is more **desirable**.

- For example, An ***int*** type variable could be used as an actual parameter in **a call to a function** that expected a **float type** as its formal parameter, and neither the compiler nor the run-time system **would detect the inconsistency**.

3.2 Exception Handling

- The ability of a program to **intercept** run- time errors, take **corrective** measures, and then **continue** is an obvious aid to reliability.

- C++, Java, and C# include **extensive capabilities** for exception handling, but such facilities are practically nonexistent in some widely used languages, for example C.

3.3 Aliasing

- Aliasing is having **two or more distinct** names in a program that can be used **to access the same memory cell.**

- It is now generally accepted that aliasing **is a dangerous feature** in a programming language.

- For example, **two pointers** set to point to the same variable, which is possible in most languages.

The programmer must always **remember** that **changing** the value pointed to **by one** of the two changes the value referenced **by the other**.

4. Cost Criteria

- The total cost of a programming language is a function of many of its characteristics. There is the cost of

1. **Training programmers** to use the language, which is a function of the **simplicity and orthogonality** and the experience of the programmers.

2. **Writing programs** in the language. This is a function of the **writability**, which depends in part on its purpose to the particular application.

   - Both the **cost of training programmers and the cost of writing** programs in a language can be **reduced** in a good programming environment.

3. **compiling programs** in the language.

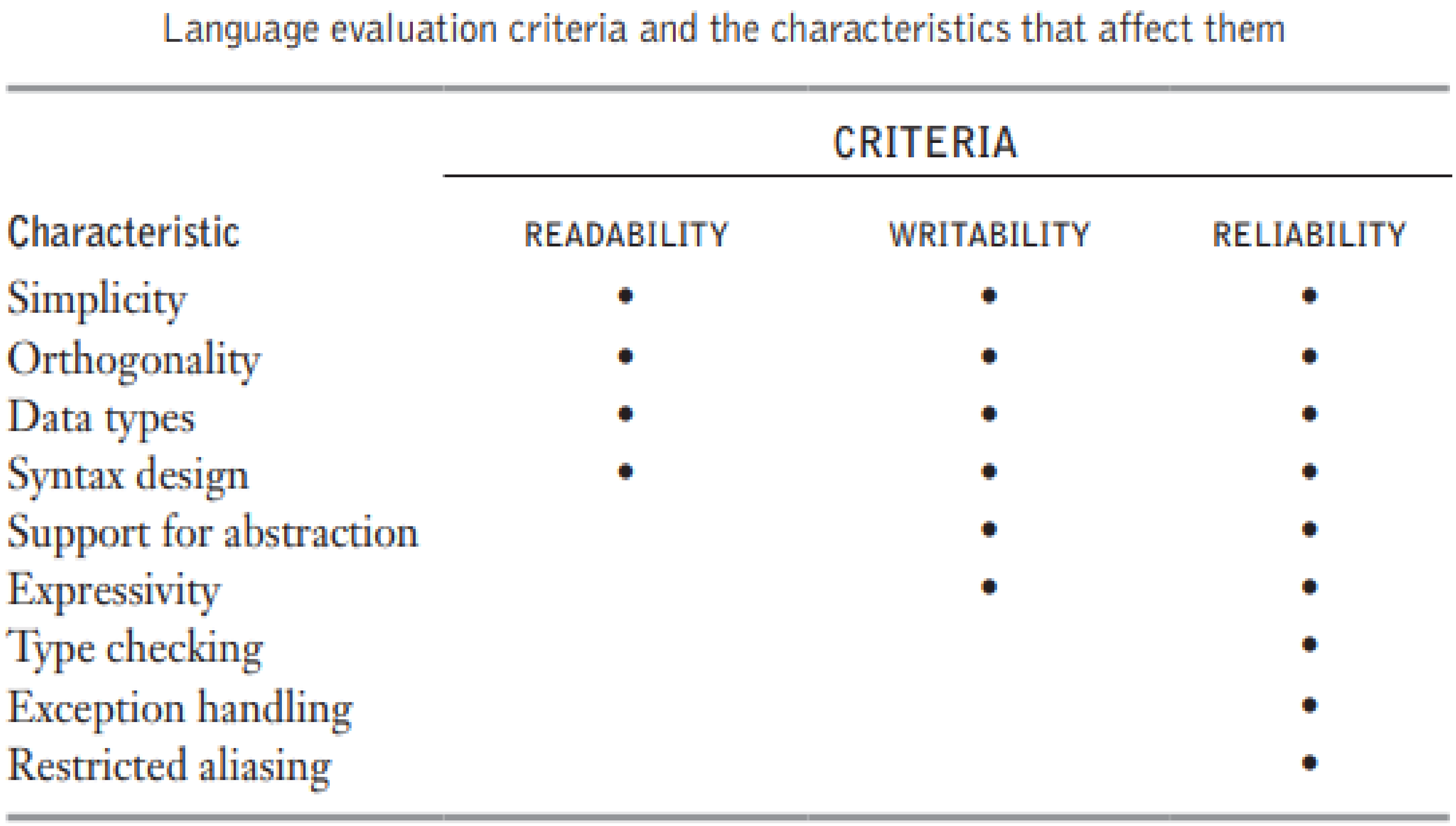
4. Cost Criteria(Cons.)

4. **Executing programs** written in a language is greatly influenced by that **language's design.** A language that requires many **run-time type checks** will prohibit **fast code execution**.

- A simple trade-off can be made between compilation cost and execution speed of the compiled code.

5. **Language implementation** system. One of the factors that explains the rapid acceptance of Java is that free compiler/interpreter systems became available for it soon after its design was released.
   - A language whose implementation system is either expensive or runs only on expensive hardware will have a much smaller chance of becoming widely used.

6. **Maintaining programs**, which includes both **corrections** and **modifications** to add new functionality.



Language evaluation criteria and the characteristics that affect them

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

Language Categories

- **Imperative**
  - Central features are variables, assignment statements, and iteration
  - Include languages that support object-oriented programming
  - Include scripting and 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

Language Categories

- **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

Programming Environments

- The **collection of tools** used in software development
- Simple – file system, text editor, compiler, interpreter or linker.
- Extensive – rich set of tools
  - •Borland JBuilder
    - An integrated development environment for Java
  - • Microsoft Visual Studio.NET
    - A large, complex visual environment
    - Used to program in C#, Visual BASIC.NET, Jscript, J#, and C++

Summary

- The study of programming languages is valuable for a number of reasons:
  - Increase our capacity to use different constructs
  - Enable us to choose languages more intelligently
  - Makes learning new languages easier
- Most important criteria for evaluating programming languages include:
- Readability, writability, reliability, cost

Thank You