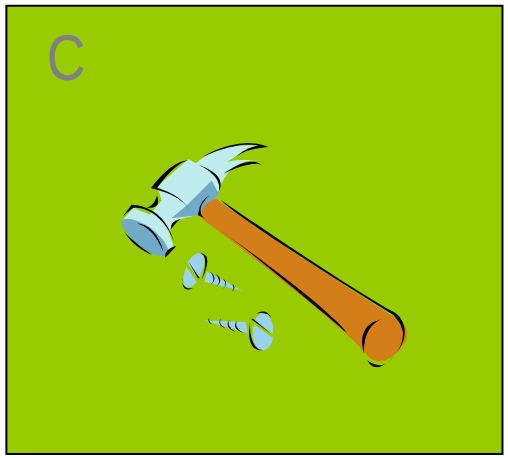
Programming Language Design Principles

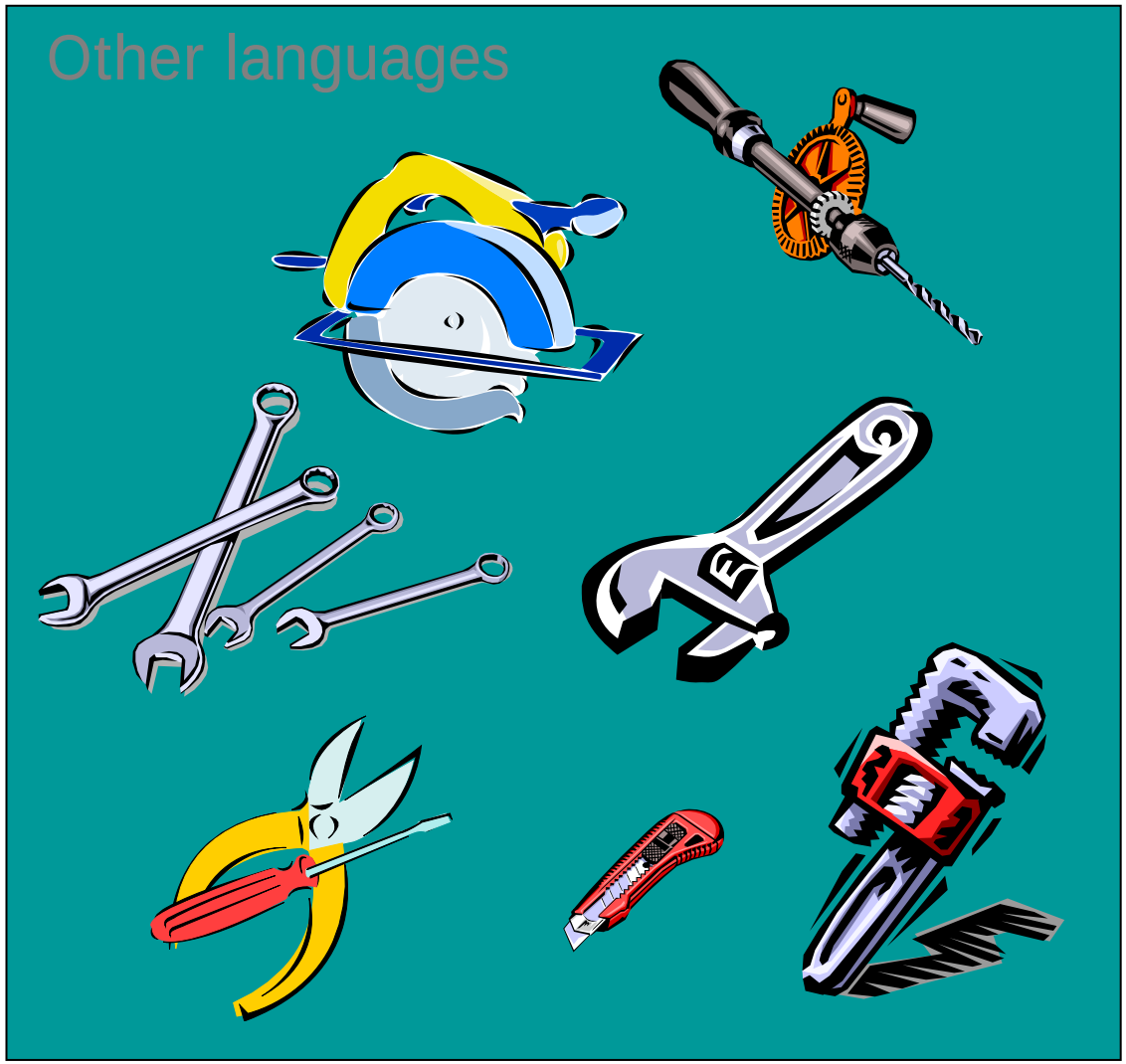
Panel Discussion

Different Programming language Design Philosophies



If all you have is a hammer,

then everything looks like a nail.



Criteria in a good language design

- **Readability**
  - understand and comprehend a computation easily and accurately
- **Write-ability**
  - express a computation clearly, correctly, concisely, and quickly
- **Reliability**
  - assures a program will not behave in unexpected or disastrous ways
- **Orthogonality**
  - A relatively small set of primitive constructs can be combined in a relatively small number of ways
  - Every possible combination is legal
  - Lack of orthogonality leads to exceptions to rules

Criteria (Continued)

- **Uniformity**
  - similar features should look similar and behave similar
- **Maintainability**
  - errors can be found and corrected and new features added easily
- **Generality**
  - avoid special cases in the availability or use of constructs and by combining closely related constructs into a single more general one
- **Extensibility**
  - provide some general mechanism for the user to add new constructs to a language
- **Standardability**
  - allow programs to be transported from one computer to another without significant change in language structure
- **Implementability**
  - ensure a translator or interpreter can be written

Tennent's Language Design principles

- The Principle of Abstraction
  - All major syntactic categories should have abstractions defined over them. For example, functions are abstractions over expressions
- The Principle of Correspondence
  - Declarations ≈ Parameters
- The Principle of Data Type Completeness
  - All data types should be first class without arbitrary restriction on their use

–Originally defined by R.D.Tennent

Principles of Programming Language Design (taken to the extreme)

Anthony A. Aaby
Walla Walla College
204 S. College Ave.
College Place, WA 99324
E-mail: aabyan@wwc.edu

Syntax

- 1 Principle of Simplicity:
  - The language should be based upon as few “basic concepts” as possible.
- 2 Principle of Orthogonality:
  - Independent functions should be controlled by independent mechanisms.
- 3 Principle of Regularity:
  - A set of objects is said to be regular with respect to some condition if, and only if, the condition is applicable to each element of the set. The basic concepts of the language should be applied consistently and universally.
- 4 Principle of Type Completeness:
  - There should be no arbitrary restriction on the use of the types of values. All types have equal status. For example, functions and procedures should be able to have any type as parameter and result. This is also called the principle of regularity.

Syntax

- 5 Principle of Parameterization:
  - A formal parameter to an abstract may be from any syntactic class.
- 6 Principle of Analogy:
  - An analogy is a conformation in pattern between unrelated objects. Analogies are generalizations which are formed when constants are replaced with variables resulting in similarities in structure. Analogous operations should be performed by the same code parameterized by the type of the objects.
- 7 Principle of Correspondence:
  - For each form of definition there exists a corresponding parameter mechanism and vice versa.

Semantics

- 8 Principle of Clarity:
  - The mechanisms used by the language should be well defined, and the outcome of a particular section of code easily predicted.
- 9 Principle of Referential Transparency:
  - Any part of a syntactic class may be replaced with an equal part without changing the meaning of the syntactic class (substitutivity of equals for equals).
- 10 Principle of Sub-types:
  - A sub-type may appear wherever an element of the super-type is expected.

Applicability

- 11 Principle of Expressivity:
  - The language should allow us to construct as wide a variety of programs as possible.
- 12 Principle of Extensibility:
  - New objects of each syntactic class may be constructed (defined) from the basic and defined constructs in a systematic way.

Safety

- 13 Principle of Safety:
  - Mechanisms should be available to allow errors to be detected.
- 14 Principle of the Data Invariant:
  - A data invariant is a property of an object that holds whenever control is not in the object. Objects should be designed around a data invariant.
- 15 Principle of Information Hiding:
  - Each “basic program unit” should only have access to the information that it requires.
- 16 Principle of Explicit Interfaces:
  - Interfaces between basic program units should be stated explicitly.
- 17 Principle of Privacy:
  - The private members of a class are inaccessible from code outside the class.

Abstraction

- 18 Principle of Abstraction:
  - Abstraction is an emphasis on the idea, qualities and properties rather than the particulars (a suppression of detail). An abstract is a named syntactic construct which may be invoked by mentioning the name. Each syntactic class may be referenced as an abstraction. Functions and procedures are abstractions of expressions and commands respectively and there should be abstractions over declarations (generics) and types (parameterized types). Abstractions permit the suppression of detail by encapsulation or naming. Mechanisms should be available to allow recurring patterns in the code to be factored out.
- 19 Principle of Qualification:
  - A block may be introduced in each syntactic class for the purpose of admitting local declarations. For example, block commands, block expressions, block definitions.
- 20 Principle of Representation Independence:
  - A program should be designed so that the representation of an object can be changed without affecting the rest of the program.

Generalization

- 21 Principle of Generalization:
  - Generalization is a broadening of application to encompass a larger domain of objects of the same or different type. Each syntactic class may be generalized by replacing a constituent element with a variable. The idea is to enlarge of domain of applicability of a construct. Mechanisms should be available to allow analogous operations to be performed by the same code.

Implementation

- 22 Principle of Efficiency:
  - The language should not preclude the production of efficient code. It should allow the programmer to provide the compiler with any information which may improve the resultant code.
- 23 Principle of Modularity:
  - Objects of each syntactic class may be compiled separately from the rest of the program.