CS 350: Programming Language Design

Chapter 1: Why Programming Languages

Why study programming languages?

Adds to your arsenal of idea expression tools.

Understanding the power of programming languages lets you understand the many different ways you can build a piece of software and what tools you can use; this can break the common limitations we face.

Even in environments unhospitable to certain design methods, you can simulate friendly features through 3rd party or self-made expansions.

Allows you to select the appropriate language for a project with greater efficacy, this can be seen simply as keeping your skills up-to-date since new languages pop up with new features that can be better suited to any project.

It will become easier to learn new languages, understanding the basic structure of writing languages allows this knowledge to be attained with greater ease since the structures will show themselves in other languages.

Allows for clever and efficient implementation of a programming language as well as serves as a study point for the implementation for compilers and languages design.

Studying languages allows you to make better use of language features

It lastly allows you to advance computing as a whole, since the most efficient and powerful languages will have the opportunity to be used.

Domains of Programming

Science applications, ALGOL was developed for mathematics, allowing for mathematical coding and coding algorithms as they were written in standard notation.

Business Applications, COBOL was designed as the de facto business language, most of the languages in this domain are designed with the capability to produce reports, storing and describing number and character data with decimal arithmetic.

AI, AI relates to using symbols versus numeric computation, in this case computations with symbolic names rather than numbers. Flexibility is required in these cases, and Lisp was one of the first languages developed and used for these purposes. Python and Prolog are two others, with Prolog being suited for logic programming.

Web Software uses markup and processing to display a web page and modify it on the fly to generate a UX.

Evaluating Languages

We evaluate languages based on three criteria, Readability, Writability, and Reliability

Readability, writing a program for Web purposes in Java doesn’t lend well and may be heavy handed.

Simplicity is an important factor for readability. Get a lot done with a little, is a good marker of simplicity. Don’t write a language with 10 different ways to do a single thing, it becomes hard to understand, while you may not use all of these methods the readers might use different parts making it more difficult to deduce or break down. This numerous ways to do a single operation is feature multiplicity.

Operator Overloading, while useful can make it difficult to understand a program since you are changing the meaning of operators, despite it’s usefulness. It is important to use operator overloading wisely, in cases where it matters but not in confusing manners.

Simplicity can be over-done, making it far more hard to derive all of the meaning hidden in the code.

Orthogonality means that a small set of primitive constructs can be combined to build data structures and control structures in the program. So, we can only combine specific components of our programming language in a specific set of ways that is manageable in different permutations. Too much orthogonality makes a program very difficulty to understand and hard to learn, while too little does the same.

Data types also influence the readability, since Booleans may not be used, we may use an integer isolated to two values, 1 and 0.

Keywords, the keywords used in your language are important for writing clear code. In C languages, braces are used to end code groups, making them somewhat difficult to understand at times or determine which group ends. While Ada and Fortran use “end if” to terminate those groups.

Form and Meaning, statements should indicate exactly what they are supposed to do and what they mean. Static is a great example of this having gone bad, since in C it has different meanings based on the context.

Writability, measures how easy it is to create programs in a language in a specific domain.

Simplicity and Orthogonality, apply similarly here as they do in reading, since the numerous ways to do things can make it cumbersome to write even the simplest programs.

Expressivity, is the usage or inclusion of features that make certain domain related operations or computations easy

Reliability, a key attribute, is the ability of our program or language to perform to it’s specification under all conditions.

Type checking is checking for type errors, using a float when a Boolean should be used etc. However, compile-time type checking is the most desired since it is lightweight compared to run-time.

Exception Handling, this allows you to deal with errors gracefully, stopping the program from falling apart due to one small error.

Aliasing, is providing a distinct name for a program construct or variable and even providing two names for that same thing. This is a very dangerous feature

Readability and Writability both influence the reliability, since being able to write things clearly and read them clearly will lead to more sensible and intuitive solutions or methods.

Cost, the total cost that lies in training and writing in a programming language, the cost of running a program in a specific language, the cost of failures and maintenance.

The most important though are development, maintenance, and reliability.

Portability, Generality, and Well-definedness are other important criteria.

Influences on Language Design

Architecture, the way our computer processes commands influences the way our compiler will run commands and the speed with which they run.

Von Neumann Architecture; the ALU, Control Unit and memory of the CPU.

Fetch-Execute cycle is used to get instructions from the program counter then increment the counter, decode and execute the instructions.

Data vs. Procedure orientation

80s – OOP

Types of languages – Imperative, Functional, Logic, Object-oriented

Rule-based – rules not described in a particular order, system chooses how to use rules in what order etc.

Scripting are imperative

Markup languages, HTML and XML.

Language Trade offs

Reliability vs. cost

Ensuring maximum reliability will increase the cost greatly, in terms of running the program.

C++ Pointers are an example of a reliability issue, while easy to manipulate in a variety of ways they prove to be dangerous.

Implementation Methods

Use low-level languages for using hardware, this provides a flexible and easy to implement system.