Introduction

Programming languages are fundamental tools for developers. Their design shapes the way we think about problems, express solutions, and organize solutions into maintainable systems.

Why study programming languages?

Every programmer needs a deep understanding of programming languages to learn them faster, use them more effectively, evaluate strengths and weaknesses of new ones, and design their own.

Most don't think they'll ever design a programming language. However, one tool of automation and abstraction is a programming language. You may never need to design a full-fledged, general-purpose language, but you may need to design a special-purpose programming language (also called a domain-specific language) that makes it faster and easier to solve a narrow class of problems.

Methodology

To deepen our understanding of programming languages, we are going to build some. We will start with a very simple language, and incrementally build on that language to build increasingly more complicated languages. As we do, we will explore core concepts that appear in many modern and classic programming languages.

A programming language consists of two parts: syntax and semantics. **Syntax** defines the notation we use to write programs in a language. **Semantics** defines the meaning of programs written in a language; that is, what a program does when we run it.

The syntax of a language is the programmer's interface to a language. From this perspective, syntax is important. Poor syntax makes programming an onerous task. Improving syntax can make a language less error prone, easier to type, easier to understand, and easier to maintain.

Semantics is the engine of a language. It is where the expressive power of a language is determined. It is where we decide if a language support loops, if-statements, case-statements, exception handling, goto statements, functions, procedures, closures, type checking, classes and objects, out parameters, in-out parameters, pass by reference, pass by value, continuations, parallelism, side-effects, unification, etc.

Because semantics embodies the core of a programming language, it is where we want to spend our time in this course. We are going to build many, rather ugly, languages. But these languages will support powerful programming language features!

To make this possible, we are going to use a tool: **PLCC - Programming Languages Compiler Compiler**. It will allow us to simplify the process of building languages by generating parsers for our languages, allowing us to focus our efforts on implementing the semantics of our languages.