Programming Paradigms

Gregory Hill

September 7, 2018

Abstract

A program is an ordered set of instructions that a computer executes and high-level programming languages clarifies this in a human readable way. Such schemes are often classified by certain properties that differentiate unique aspects of the language.

1 Achieving Results

1.1 Imperative

Statements that iteratively change a program's state - command driven.

1.1.1 Procedural

Routines or functions that contain a series of computational steps to be processed. Any given procedure might be called at any point during a program's execution, including by other procedures or itself.

Examples: Fortran, ALGOL, COBOL BASIC, Pascal, C, Ada, and Go.

1.1.2 Object-Oriented

Based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. These functions can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). Many popular languages are also class-based, meaning that objects are instances of classes, which typically also determine their type.

- **Abstraction:** hiding the details of the implementation from outside the class.
- **Encapsulation:** restricting access of some of the fields and method of a class from the outside world and binding together related data and methods.
- **Inheritance:** reuse code of existing objects, or establish a subtype from an existing object. These relationships of classes gives rise to a hierarchy.
- **Polymorphism:** multiple methods all share the same name, but offer slightly different functionality. Two types; overridding and overloading.

Examples: Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, and Smalltalk.

1.2 Declarative

Expresses the logic of computation without describing its control flow - rule based.

1.2.1 Functional

Treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data.

Examples: ML, Lisp, CLOS, Scheme, Haskell

1.2.2 Logic

Based on formal logic, a program written in a logic programming language is a set of sentences in logical form, expressing facts and rules about some problem domain.

Examples: PROLOG, GHC

1.2.3 Mathematical

The desired result is declared as the solution of an optimization problem. For example, **dynamic programming** studies the case in which the optimization strategy is based on splitting the problem into smaller subproblems. The equation that describes the relationship between these subproblems is called the Bellman equation.

2 Expressing Variables

2.1 Statically Typed

A language is statically typed if the type of a variable is known at compile time. For some languages this means that the type for each declaration must be specified, but some languages offer a form of type inference - the capability to deduce the type of a variable.

Examples: Java, C, C++

2.2 Dynamically Typed

A language is dynamically typed if the type is associated with run-time values, and not named variables/fields. Most scripting languages have this feature as there is no compiler to perform static type-checking. Many dynamically typed languages also allow you to provide type information, but do not require it.

Examples: Perl, Python, Ruby

3 Building Software

3.1 Compiled

A compiled language is one where the original program is translated into native machine instructions, which are then executed directly by the hardware.

Examples: C, COBOL, Fortran

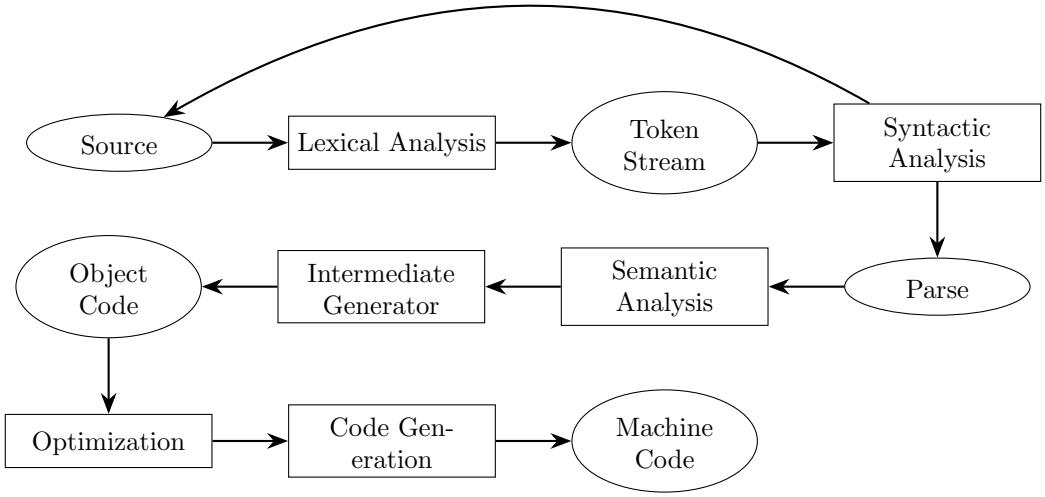

Figure 1: Compilation Process

3.2 Interpreted

An interpreted language is one where the instructions are not directly executed by the target machine, but are instead read and executed by some other program (which normally is written in the language of the native machine).

Examples: JavaScript, Perl, Python