Emerald Documentation

Oleks Shturmov, Eric Bartley Jul

Feb 24, 2021

CONTENTS:

CHAPTER

ONE

CLASSICAL EMERALD SYNTAX

Emerald was conceived in an academic setting, as an object-based language for distributed computing in the early 1980s. As such, its classical syntax is inspired by some of the academic giants of the time, namely Pascal, Simula, and S.

An Emerald program is a sequence of constant declarations. These constants are initialized in the order that they appear in. There is no "main method" as such (as you must have in C, C#, Java, etc.). The in-order initialization of the constants constitutes the entire execution of an Emerald program. Constants therefore, are of paramount importance in classical Emerald.

1.1 Constant Declarations

A constant declaration declares a name to correspond to the evaluation of an expression. Optionally, you can specify the type that you expect the resulting value to conform to. Formally, the syntax is as follows:

constDecl ::= **const** *name* [**:** *type*] **<-** *expr*

The Emerald compiler will analyse the expression to infer its type. If the actual type does not conform to the expected type, a compile-time type error results. Hence, classical Emerald is a **statically-typed language**, with (limited) type inference. This is akin to C#.

The correspondence between a *name* and the value that *expr* evaluates to remains constant throughout the lifetime of a program, and has global scope. The value itself however, is not constant—it is mutable, and subject to change throughout the lifetime of a program. Hence, classical Emerald is an **emperative language**, just like modern-day C# and Java.

1.2 Identifiers

An Emerald identifier is a non-empty sequence of letters, digits, and the character _, beginning with a letter or _. Identifiers are **case-insensitive**. Some identifiers are reserved as literals and keywords; the rest name constants, variables, operations, parameters, and objects.

The reserved identifiers are:

- *Literals*: **false**, **nil**, **true**
- *Keywords*: **const**, **end**, **object**, **var**

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search