Chapter 1

Programming in Haskell, 2nd ed.

Connor Baker

Compiled on December 21, 2019 at 9:04pm

1.1 Functions

> In Haskell, a function is a mapping that takes one or more arguments and produces a single result, and is defined using an equation that gives a name for the function, a name for each of its arguments, and a body that specifies how the result can be calculated in terms of the arguments.
>
> Excerpt From: Graham Hutton. “Programming in Haskell.”

1.2 Functional Programming

> … functional programming can be viewed as a style of programming in which the basic method of computation is the application of functions to arguments. In turn, a functional programming language is one that supports and encourages the functional style. … In general, programming languages such as Java in which the basic method of computation is changing stored values are called imperative languages, because programs in such languages are constructed from imperative instructions that specify precisely how the computation should proceed. Most imperative languages provide some form of support for programming with functions… However, many imperative languages do not encourage programming in the functional style. For example, many such languages discourage or prohibit functions from being stored in data structures such as lists, from constructing intermediate structures such as the list of numbers in the above example, from taking functions as arguments or producing functions as results, or from being defined in terms of themselves. In contrast, Haskell imposes no such restrictions on how functions can be used, and provides a range of features to make programming with functions both simple and powerful.
>
> Excerpt From: Graham Hutton. “Programming in Haskell.”

1.3 Features of Haskell

- Concise programs
- Powerful type system
- List comprehensions
- Recursive functions
- Higher-order functions
- Effectful functions
- Generic functions
- Lazy evaluation
- Equational reasoning