Chapter 1

Expressions

1.1 What Is An Expression?

In mathematics, an expression is an evaluable collection of symbols constructed with the concept of correctness within a context. In other words, an expression is something which can be evaluated given the right set of symbols and values, which will result in a new meaningful result.

Although this is a rather abstract way of thinking about an expression, it is the basis for how computer science built its own idea of an expression. Even with the great generality which comes with a mathematical definition, we can actually apply this even to everyday programming fairly directly.

If we consider an expression in computer programming, it lines up quite nicely with our mathematical counterpart: a composition of values, operators, constants and functions which is evaluated at run time, and returns a value.

The important idea to consider when we talk about the combination of values, operators, constants and functions, we really are aligning ourselves with the mathematical idea of symbols and evaluable construction. In other words, any expression in computer programming will have some notion of how to correctly assemble language constructs into a new idea which can be compiled or interpreted and executed to give us a result.

Although an expression may return a result and that result will always come from the combination of parts in our expression, there is no assumed notion that our result will be meaningful or useful to us. We will worry about that a little later, though.

In many modern languages, not every line of code is an expression. If we consider flow control structures like if/else and looping blocks, the construct is not, itself, and expression. Although having language structures which are

not expressions is perfectly acceptable, it introduces challenges when writing functional code.