Emerald Design Documentation

Bryce Davis*[1] and Nick Diguido†[1]

[1]Datum Unlimited

July 22, 2014

*me@bryceadavis.com
†nick@diguido.com

Notes

Emerald is currently pre-alpha. Things *will* change drastically. Be prepared for things to just stop working randomly.

Part I
Language Design

1 Paradigms

1.1 Imperative

Emerald includes most of C's basic control structures:

- if
- if-else
- while
- do-while

For-loops are not included since they can be implemented using while-loops, compound Boolean statements, and accumulators. Emerald also includes some interesting control structures from Ruby, including:

- unless
- until

This are the opposites of if and while, and their use can be determined from English usage.

1.2 Object-Oriented

Emerald is unique in the idea that while data and functions are both considered first-class, they are separate. Data are handled using `types` while functions are their own first class objects. Higher-level data types are combinations of built-in and user-defined types in a class-like format, like so:

```
type
    int data
    string name
 sometype
```

1.3 Functional