Boa - Intermediate

- Level 2 -

Data Structure Types

Array - array of **Type**

Built-ins: new(**array**, int, **Type**), sort(**array**)

Map and Stack have very similar builtins to their Java counterparts

Map - map[**Type**] of **Type**

Stack - stack of **Type** (LIFO)

Queue - queue of **Type** (FIFO)

Special Built-ins: offer(**queue**, **Type**), poll(**queue**)

Set - set of **Type**

Special Built-ins: add(**set**, **Type**), union(**set**, **set**), intersect(**set**, **set**)

Other Types

For an Enum, each of the named values should be the same type

Enum - type name = enum { name1 = **Type**,name2 = **Type**, …}

Ex. type compass = enum{ N = “north”, S = “south”, W = “west”, E = “east”}

A Tuple is much like a struct

Tuple - type name = { name1: **Type**, name2: **Type**, …}

Ex. type worker = { num: int, name: string, onLeave: boolean}

steve : worker = { 100, “Steven Even”, false}

steve._1 = “Steven Odd” will change the second entry by position

steve.onLeave = true will change the name of the entry given

Output and Aggregators

- Boa has specific variable for output: **`output type`**
- Process:
  - Boa code pulls data & sends it to output variables
  - All projects processed in parallel

Components

1. Parameters
    a. “Formal” variables
2. Indices
    a. 1 or more indices
    b. Allows for grouping
    c. NOT the same as parameters
3. Weights
    a. Max weight of 10

Output Aggregator Examples

```
# Gets average number of programming languages used in a project
p: Project = input;
counts: output mean of int;
counts << len(p.programming_languages);
```

Alternate Example

```
# False average number of programming languages in a project
p: Project = input;
counts: output mean of int;
foreach (i: int; def(p.programming_languages[i]))
        counts << 1;
```

Bottom Example

```
# Shows 5 least used programming languages that are used with at least 2 other languages
p: Project = input;
counts: output bottom(5) of string weight int;

if(len(p.programming_languages) > 2)
    foreach (i: int; def(p.programming_languages[i]))
            counts << p.programming_languages[i] weight 1;
```