Imperative Asteroid with Objects

- The “Hello World” program…

Load system module

Sentence terminator

```
1  load system "io".
2
3  println("Hello World!").
```

Iteration

- ‘while’, ‘for’, ‘loop’ constructs are all supported

```
-- compute the factorial

load system "io".
load system "type".

function fact with n:%integer do
  let val = 1.
  while n > 1 do
    let val = val*n.
    let n = n-1.
  end
  return val.
end

let x = tointeger(input("Enter a positive integer: ")).
println ("The factorial of "+x+" is "+(fact x)).
```

Constraint pattern

Iteration

Assignment

Type conversion

Data Structures

- Built-in lists
  - [1,2,3]
- Built-in tuples
  - (x,y)
- Element access
  - a@i

```
-- the bubble sort
load system "io".

function bubblesort with a:%list do

  loop
    let i = 0.
    let swapped = false.
    for i in 0 to len(a)-2 do
      if a@(i+1) <= a@i do
        let (a@i,a@(i+1)) = (a@(i+1),a@i).
        let swapped = true.
      end
    end
    if not swapped do
      break.
    end
  end

  return a.
end

let l = [6,5,3,1,8,7,2,4].
println("unsorted array: "+l).
println("sorted array: "+(bubblesort l)).
```

Structures & Objects

- Asteroid is object-based
- Bundle operations with data
- No inheritance
  - Construct new objects from other objects via object composition (and traits in the case of Rust)
- New languages with a full object-oriented type system are waning
  - Of the three “big” new languages (Rust, Go, Swift) only Swift supports OO, the others are object-based.

Structures

```
-- rectangle structure
load system "io".

structure Rectangle with
  data xdim.
  data ydim.
end

let r = Rectangle(4,2). -- default constructor
println ("Rectangle with x="+r@xdim+" and y="+r@ydim).
```

- Structures consist of ‘data’ fields and are associated with a ‘default constructor
- Member access is via the ‘@’ operator

Structure

```
-- rectangle structure
load system "io".
load system "type".

structure Rectangle with
  data xdim.
  data ydim.

  -- member function
  function area with none do
    return this@xdim*this@ydim.
  end

  -- constructor
  function __init__ with (x %if isscalar(x), y %if isscalar(y)) do
    let this@xdim = x.
    let this@ydim = y.
  end
end

let r = Rectangle(4,2).
println ("The area of Rectangle ("+r@xdim+","+r@ydim+") is "+r@area()).
```

- Member functions
- Object identity is given with the ‘this’ keyword
- Member functions are called on objects with the ‘@’ operator
  - E.g., r@area()

Structures: Rust & Go

```rust
#[derive(Debug)]
struct Rectangle {
    width: u32,
    height: u32,
}

impl Rectangle {
    fn area(&self) -> u32 {
        self.width * self.height
    }
}
```

Rust

```go
type rect struct {
    width int
    height int
}

func (r *rect) area() int {
    return r.width * r.height
}
```

Go

Types in Asteroid

- Asteroid has a set of **primitive data types**:
  - integer
  - real
  - string
  - boolean
- Asteroid arranges these data types in a **type hierarchy** in order to facilitate automatic type promotion:
  - boolean < integer < real < string

Types in Asteroid

- Asteroid has two more built-in data types:
  - list
  - tuple
- These are **structured data types** in that they can contain entities of other data types.
- Lists and tuples themselves are also embedded in type hierarchies, although very simple ones:
  - list < string
  - tuple < string
- That is, any list or tuple can be viewed as a string. This is very convenient for printing lists and tuples.

Types in Asteroid

- Using the ‘structure’ keyword Asteroid also supports user defined types.
  - The name of the structure becomes a new type available in the program.

```
-- user defined types
structure Person with
  data name.
  data profession.
end

let p:%Person = Person("Fred","Carpenter").
```

Types in Asteroid

- Finally, Asteroid supports one more type, namely the **none** type.
  - The none type has a constant named conveniently ’none’.
  - The empty pair of parentheses () can often be used as a convenient short-hand for the constant none.
  - The none data type does not belong to any type hierarchy.

Running Asteroid

- Install the interpreter on your machine
  - See https://github.com/lutzhamel/asteroid
- Run Asteroid in the cloud
  - https://replit.com/@lutzhamel/asteroid

Assignments

- Explore the links in the github readme
- Do Assignment #1 – see BrightSpace