Sprint 3 Plan

Go Compiler

April 18 - April 25

Kyle Remmert (Scrum Master), Petar Zaninovich, Trevor Ching, Vincent Kim (Product Owner)

Task listing, organized by user story

Sprint 3

(18)User Story 1: As a user, I would like to be able to use a struct and the fields within a struct.

Assigned: Vince, Trevor

Tasks:

- **(5)Vince:** Create the struct type node with a symbol table.
- **(3)Trevor:**Allow composite lits to either assign variables by name or by order
- **(5)Trevor:**Selector expressions read from the frameslot and access the struct variable in its symbol table.
- **(5)Vince:**Allow structs to define themselves in the lexical scope at the top level so that they can add themselves as a field

(22)User Story 2: As a user, I would like my Go program to typecheck correctly to keep language consistency and provide with meaningful error messages.

Assigned: Kyle, Petar

Tasks:

- (8)**Kyle:** Single assignments check the type of both the variable and the value before assigning
    - Throw GoException if the types do not match
    - Modify before assignment to check types
        - In the parser, get the identifier node for the left and right hand side then check types there.
- (9)**Petar:** Function type checking checks parameters and values.
    - Multiple assignments: check the type and length of every variable before assigning the values
    - Function calls are type checked in the parameters with the function being called.
    - Return types and length match function signature checked inside a function
- (5)**Petar:** Non primitive types check before accessing and writing to the type

(11)User Story 3: As a user, I would like the float32 and float64 types for various math operations.

Assigned: Petar

Tasks:

- (3)**Petar:** Make floats work with the composite lit node
    - Case 1: It is initially assigned an int value, but declared as float
    - Case 2: Declared and initialized as float
- (1)**Petar:** Add unit tests for following types
    - Float32, Float64
        - Various math operations using the two values
- (1)**Petar:** Add specializations when floats are used
    - Binary/Unary expressions
        - Add corresponding functions
- (5)**Petar:** Type check
    - Left and right hand side match in type and size
    - Throw corresponding GoException with dialogues similar to Go's error
- (1)**Petar:** Prints floats in correct notation
    - Modified println builtin