Design Document

Milestone 2

COMP 520: Compiler design

Maxence Frenette (260 685 124)
Maxime Plante (260 685 695)
Mathieu Lapointe (260 685 906)

Mar. 13, 2018

Scoping Rules

```
type scope =
  {
    bindings: (string * typesDef) list;
    types: (string * typesDef) list;
    functions: (string * typesDef list * typesDef option) list;
    parent: scope option;
    children: scope list
  }
```

The symbol table of the compiler is in a tree form. Each node of the tree is a scope. Each scope has a reference to its parent scope and a list of references to its child scopes. The root scope is the scope englobing the top-level scope. It is where the base types are defined. Functions are defined in the top-level scope (the top-level scope is the only child scope of the root scope). Each time a new scope is created, it is added as a child of the current scope.

For each scope, variable bindings are stored in the `bindings` member as a list of type definitions associated with the variable's identifier. Types are stored in the `types` member as a list of type definition with the associated identifier. Functions are stored in the `functions` member as a list of function type definition. Each function type definition is a string (the function identifier), a list of type definition (the arguments) and an optional type definition (for the return type).

Events triggering the creation of a scope:

- Function declaration (the argument bindings are added to the new scope).
- Declaration of a block (including the body of `if`, `else`, `for`, `switch`, `case`).
- Init statement of a `switch`, `if`, `for`[1]

Event triggering the close of the scope opened by an init statement

- End of `if` body if there is no `else`
- End of `else`, `for`, `switch` body

Design of Test Program

Talk about the design of your Q1 programs to test the typechecker (including typing rule).

[1] This means that, in a `switch`, `if`, `for` with an init statement, two scopes are opened. The scope of the init scope is opened first, then the scope of the body is opened.

Problems and Solutions

Explanation of problems and/or solutions is sufficient to implement a fully functional GoLite compiler.

Root Scope

We needed that the base types (int, float64, string, rune, bool) to be already defined in the top-level scope, but that could also be shadowed in the root scope. To do this, we decided to add a scope that would be the root of our symbol table and also the parent of the top-level scope. In this root scope, only the base types are defined.

Team Organization

For this milestone, Maxence had a few personal problems that caused him to have less time to put into school work. Mathieu and Maxime put more work in this milestone to counter-balance it.

Maxence

- Removed tests from past teams from the GitHub repository.

Maxime

- Wrote the design document.
- Created the typechecking tests.
- Implemented the symbol table printer.
- Implemented the typechecking along with Mathieu.

Mathieu

- Created most of the typechecking code.