Comp 520: Milestone #2

The Heapsters

1 Design Decisions

- Implemented weeder to handle the errors that are not captured by our scanner and parser. Example includes balancing ids and expressions for short assign, ensuring that break and continue only appear in a loop, etc.
- Implemented symbol table building and type checker in SableCC3
- Symbol table is built using a two-pass approach. The first pass collects the type information associated with all top-level declarations, as they can appear in any order. The second pass traverses the entire tree while typechecking and building upon the symbol table as needed.
- Symbol table is build using the standard cactus stack of frames, which are represented by hash tables. However instead of creating a separate symbol class, our symbol table simply maps an identifier directly to its corresponding AST node, which turns out to be an extremely poor design decision (we regret this deeply). Later during the type checking phase, whenever we need to retrieve the type information of an id, we have to traverse a subtree of the AST where that id comes from. As a result, we wrote a lot of helper methods. And some parts of the type checker implementation can be a little messy.
- The type information of expressions is stored in an auxiliary hash table that maps from AST nodes to each type class, given that SableCC3 does not provide straight-forward extension for adding extra fields such as type to the AST node classes.
- Type checking is done through a single bottom-up traversal of the AST. As the type checker is traversing through the AST, the type information of each expression is gradually added to the type hash table.
- The hash table contained expression types is later used by TypePrettyPrinter to annotate the type information for each expression.
- Discuss special cases of nodes that cannot be handled simply through bottom-up traversal (if any)

2 Scoping Rules

Our type checker obeys standard scoping rules:

- Blocks (e.g. in if-statements, in a for loop) define a scope.
- Referring to a variable that was previously declared in the current scope or one of the outer scopes is legal.
- Declarations using an already-defined identifier in the current scope is illegal (but those from outer scopes is legal).

3 Type Checks

Here, we enumerate the type checks and give the correspondence of each type check to the test programs used to demonstrate the type error.

3.1 Declaration

- Undeclared variable: `print_undeclared_variable.go`
- Redeclare variable: `re_declare_variable_in_same_scope.go`, `short_assign_with_same_ids.go`, `use_variable_before_declaration.go`, `redeclare_type.go`

3.1.1 Function declaration

- Incorrect return type: `func_return_wrong_type.go`, `func_with_return_type_return_void.go`
- Returning expression in a non-void function: `func_without_return_type_return_string.go`, `void_func_return_int.go`

3.2 Statements

3.2.1 Short declaration

- Mixed types: `short_assign_with_new_id_wrong_expr_type.go`

3.2.2 Op-assignment

- L.H.S. and R.H.S. are incompatible: `minus_assign_int_to_rune.go`, `plus_assign_float_int.go`, `rshift_assign_int_with_rune.go`

3.2.3 For loop

(Same tests for If statements apply here.)

3.2.4 If Statement

- Non-boolean condition: `if_with_int_condition.go`, `if_with_rune_condition.go`, `invalid_condition_for_while_loop.go`

3.2.5 Switch Statement

- Expression type and case type don't match: `switch_expression_not_match_case_expression.go`
- No expression: `switch_without_expression_with_int_case.go`

3.2.6 Increment/Decrement

- Incorrect type: `incr_string.go`

3.3 Expressions

3.3.1 Unary expressions

- Incorrect operand: `bit_complement_float_type.go`, `boolean_not_rune_type.go`, `unary_minus_bool_type.go`, `unary_minus_string_type.go`, `unary_plus_bool_type.go`, `unary_plus_string_type.go`

3.3.2 Binary expressions

- Incorrect operands: `add_bool_and_bool.go`, `add_string_and_rune.go`, `add_int_and_float.go`, `conditional_and_int_type.go`, `float_less_than_int.go`, `float_mod_int.go`, `int_compared_to_rune.go`, `array_comparable_1.go`, `array_comparable_2.go`, `array_comparable_3.go`, `array_comparable_4.go`, `slice_comparable_1.go`, `slice_comparable_2.go`, `slice_comparable_3.go`, `slice_comparable_4.go`, `slice_comparable_5.go`

3.3.3 Function call

- Incorrect argument types: `func_1.go`, `func_2.go`, `func_3.go`, `func_4.go`, `func_5.go`, `func_6.go`, `func_7.go`, `func_8.go`, `func_9.go`

3.3.4 Indexing

- Non-integer index: `non-int_in_array_var_dec.go`

3.3.5 Field selection

- Incompatible types: `field_struct_bad_assign_1.go`, `field_struct_bad_assign_2.go`, `field_struct_bad_assign_3.go`, `field_struct_prim_1.go`, `field_struct_prim_2.go`, `field_struct_prim_3.go`, `field_struct_prim_4.go`, `field_struct_prim_5.go`, `field_struct_prim_6.go`

3.3.6 `append`

- Incompatible types: `slice_append_single_to_tiple.go`, `slice_append_wrong_struct.go`

3.3.7 Type cast

- Uncastable: `array_index_not_prim_param.go`
- Too many arguments: `too_many_arguments_to_type_cast.go`

4 Contributions

The bulk of the workload for this milestone was carried out by Ethan and Leo, most of which was dedicated to the planning, design, implementation, and testing of the type checking rules. They held a total of **7** meetings over the course of the milestone, with the first few dedicated to designing the symbol table and type checker. Hardik was absent during most of these meetings, only participating in the last few scrum-like meeting. He primarily worked independently and constributed to a new grammar, the weeder, and the typed pretty printer.

Ethan

- Implemented the first pass for the weeder.
- Brainstormed, planned, and designed the symbol table and type checking rules with Leo.
- Worked with Leo to implement the symbol table.
- Worked with Leo to implement the type checker.
- Implemented a lot of type checking tests (both valid and invalid).

Leo

- Brainstormed, planned, and designed the symbol table and type checking rules with Ethan.
- Worked with Ethan to implement the symbol table.
- Worked with Ethan to implement the type checker.
- Implemented a lot of type checking tests (both valid and invalid).
- Made some fixes to the typed pretty printer and helped with testing.

Hardik

- Implemented an automatic test suite generator for tests on syntax.
- Catalogued failed parsing tests from other groups and corrected the grammar accordingly.

- Modified the pretty printer to accomodate the new grammar.
- Built on Ethan's weeder and added more weeding rules.
- Implemented weeder tests.
- Implemented and tests typed pretty printer.

Everyone

- Contributed to this document.