Project work – Phase 4

Principles of programming languages

Pinja Mikkonen 99219

I've implemented the following functionalities into the phase 4 of this project work. Please note that some of the following functions have not been implemented fully, ie. function parameter passing or return passing is not implemented. I'll further explain the missing features in the relevant sections.

Semantic checks:

1. *Variables (+ functions and parameters if you've implemented them) have to be defined before being used, no double definitions allowed*

Inializing scalar variables, sheets and ranges works. The initialization happens by inserting the name of the variable and a reference to the node it references into the symbol table. Upon initialization it's confirmed that the variable's name is not already in use. If the name already exists in the symbol table, the compilator raises an error. Scopes have not been implemented.

2. *In range expression* `range SS'A1..SS'A5` *both the start and end must refer to the same sheet, and the range has to be either a vertical or horizontal sequence of cells.*

When a range expression is created, the names of the sheet variables are compared to make sure they refer to the same sheet. If this check passes, the function uses regular expressions to extract the referenced row and column values and compares them. If neither the row nor the column match, an error is raised.

3. *Each row in a sheet initialization has the same number of cell values.*

When initializing a sheet, the length of the first row is compared in a loop to the length of the other rows. If the length of the rows doesn't match an error is raised.

4. *A function call syntax can only call functions (not subroutines) and a subroutine call statement can only call subroutines (not functions).*

When calling a function or a subroutine, the type of the call and the type of the function or subroutine being called are compared. For example, if the type of the calling expression is "subroutine call" but the node it's trying to call is "function_def", an error is raised.

5. *The number of actual parameters in a function call has to match the number for formal parameters in the function definition.*

When calling a function or a subroutine, it is first confirmed if both the calling node and the definition node have parameters. If this is the case, the numbers of the parameters are compared and an error is raised if they don't match. Next it is checked if either the calling node or the definition node alone has parameters, in

which case an error is also raised. Otherwise it is assumed that neither have parameters and thus, the call is legal.

Interpretation levels:

1. *Evaluation of expressions consisting of arithmetic expressions and decimal literals, and the print_scalar statement*

When it comes to operators and arithmetic expressions, first the left and right child nodes are evaluated and the operator being used is extracted. After this there is an if-else structure that performs the appropriate procedure to the different sides of the operation. `print_scalar` works similarly simply by printing the result of the node evaluation.

2. *Additionally, using scalar variables works (i.e., definition, assignment, and reading the value)*

Scalar variables are initialized during the semantic check phase, but the interpreter handles assigning new values to the already initialized values by simply changing the node the variable's name refers to. Reading the value works by simply returning the value of the node.

3. *The if- and while-statements also work*

If-statements simply evaluate the condition of the if-statement and iterate through the statement's `then_list` if the condition passes. While-statements work with similar principle by having a while-loop that evaluates the condition-node at the start of each iteration, and upon the condition passing iterates through the do-list.

Additional implementation that doesn't fill full reguirements:

1. *Initializing sheets*

Sheets can be initialized, but referring to their values and printing them has not been implemented.

2. *Range initialization*

Ranges can be initialized, but they have no further implemented functionality

3. *Subroutine definitions and calls work (this requires that you've first implemented the first two bullets).*

Semantic checker handles defining the subroutines, and interpreter allows for subroutines to be called, after which the statements inside the subroutine are evaluated and executed. Passing parameters to the subroutine was not implemented due to time constraints.

4. *In addition to subroutines, function definitions, calls, and return value passing work.*

Function definitions and calls were implemented in a similar manner to subroutines, but return value and parameter passing were skipped due to time constraints.