Author: Nghia Duc Hong

Student Number: 292119

Email: duc.hong@tuni.fi

Phase 4 summary:

Note: I used numpy python library to make things easier

The semantics check:

- **Level 1**: Variables have to be defined before use and no double definition allowed.
- & **Level 8**: basic scoping.

So I did create a temp symbol table to remember the parameter in the declared functions. The local variables would first storage in global, as the visiting tree function would not help us differentiate the local and global variables, but then with after_func parameter, we can make a clean-up function with a temporary symbol table containing parameters and local variables by query the definition node of functions and subroutines. This only has one drawback is that if global variables are declared, they can't be declared as local variables. But we still use parameters in both scenarios (as global and parameters).

The way checking of redefining variables is quite simple, just check if the symbol table already occurs that variables. On the other hand, checking using variables with no declaration was done by checking if the symbol table contains it or not.

- **Level 2**: Range expression must have start and end refering to the same sheet and to be horizontal or vertical sequence.

The coordinate was extracted from the phase 3, so first I check the sheet ident if they are the same, if not, they are not from the same sheet. Then the coordinate is checked if they have the same row numbers or column numbers. If the condition is not satisfied, the range is not valid.

- **Level 3**: Checking number of cells on each row

Checking the initializing list having the same number of cells on each row is quite simple. I assigned the number of cells on the first row is nOnRow, then I check if any other row has the different number of cells.

- **Level 4**: A function call syntax only call function and subroutine statement only call subroutine

This is checked by name of the node. If there is a subroutine call, I will look up to the symbol table if the node type is definition of subroutine or function. If it is a function definition, then emit an error. Checking the function call is done by the same way.

- **Level 5**: The number of actual parameters in a function call match the number for formal parameters in the function definition.

Again, by looking up the symbol table for function and subroutine definition, we can compare number of parameters in function definition and parameters when it is called.

- **Level 7**: The return only appears in function, not subroutine.

By checking subroutine definition, if there are any return statement, it has to throw an error there.

Sum: about 6 levels and half of level 7.

Interpretation:

- **Level 1**: Evaluation of arithmetic expression, decimal literal & print scalar statement
- & **Level 2**: evaluation of scalar. Extra: evaluation of scalar expression (expression with compare operator)

The decimal literal is evaluated from the tree node, and evaluating expression is done recursively in eval_node function. In level 2, scalar is obtained by examine the symbol table.

- **Level 3**: If- and while statement work.

The statement is done recursively with examining the condition expression. Examining conditon is done by using eval_node function which has been implemented, the statements inside the blocks will be executed by calling the function execute again.

- **Level 4**: sheet variable and referring cells work, print_sheet statement is also implemented. The initializing sheet is also implemented (both way : sheet initializing list or integers)

The value of sheet is constructed by a 2-dimensional numpy array in symbol table. Referring cells operations work as we are changing the value the array in symbol table. Print sheet statement is just simply iterating the columns and the rows of array

- **Level 5**: subroutine definition and subroutine call work.

In this level, first arguments are first evaluated, then it would be stored in symbol table, then also the local variables. After executing subroutine, the local variables and parameters are deleted from the symbol table.

Reflection: This phase is not so hard as all the organizing things are done in the phase 3. I did the phase 3 with a lot of optimizations and persistence. Therefore, doing phase 4 is more convenient as everything is easy to handle and care. The implementation of mine is just only about the basic things but I think it can be implemented more level if I have enough time to do it.

Testing:

My sample script:

python main.py -f level1_scalar_exprs.ss

python main.py -f level1_var_funcs.ss

python main.py -f level2_range_same_sheet.ss

python main.py -f level2_scalar_vars.ss

python main.py -f level3_if_while.ss

python main.py -f level3_sheet_row_sizes.ss

```
python main.py -f level4_function_subroutine.ss

python main.py -f level4_sheets.ss

python main.py -f level5_param_number.ss

python main.py -f level5_subroutines.ss

python main.py -f level7_return_statement.ss

python main.py -f level8_local_vars_params.ss
```