Jason Lunder
4/20/21
CPSC 326
Semester Project Update

Module extension

At this point in the project, I have most of the functionality implemented for the module extension, as described in the first update. All that is left is to add support in the Interpreter, as well as preform some tests to confirm that the other parts are working.

Some modifications:

When the parser encounters an import statement, it lexes and then parses the file referenced. Then it adds the list of declarations from the parsed file to the ast tree, so that there is only one ast tree.

For type checking, when it is possible for a module name to be used in an idr value or assignment statement (such as module.function_name()), we need to know if the first part before the . Is a module or not, so that it can be ignored in type checking if it is a module. To do this, I have modified the symbol table to keep track of a list of namespaces, which can be searched through to compare against the id before the period. This way, I can simply drop the module name and type check the rest of the expression, since it doesnt matter what module a function or type exists in for type checking purposes, as they all are a part of the same AST and symbol table.

In interpreter, it will matter what module each function or type belongs to (including ones in the current file, which are a part of the “main” module), so I have changed the type and function maps in interpreter to be nested, with the module names being used as a key for an unordered map of functions or types in that module.

What I have left to finish:

Ensure that errors reference the file that they occurred in for a better stack trace.

Add functionality for interpreter to track function and type names correctly, as well as call them correctly.

Add more tests.

Account for collisions of function/type names when using the from keyword and referencing a specific set of types to import, as these are added under the main namespace.

To build and run:

Simply run cmake CmakeLists.txt, and make. I have copied the testing files/programs from previous homeworks into my repository, so you will be able to run things like ./hw3 <test_file>. Unfortunately, I ran into an issue where the parser cant build because I am looping through a list of decls, and treating them as if they are the subclasses, (which they are), that all have the variable id. C++ doesn’t really like it and I don’t have a workaround yet for that.