Sami Blevens

CPSC 326

Project – Final Writeup

The extension created for this project is constant variables. This is in the scope of constant variables within functions and types, and function arguments. This includes creating a constant variable of a new type. However, while this is supported, it doesn't make the most sense as that type will have to keep the default values. This also includes being able to make a type field a constant variable. In terms of function arguments, this was defined in the following way: if a function is expecting or allowing a constant parameter within the function, that function must explicitly define that parameter as constant. Thus, if a parameter doesn't have the explicit definition of constant, there will be an error if a constant argument is passed in. Along with this, it is allowed to pass a non-constant argument into a constant parameter, as that value is guaranteed to not be mutated so it does not matter to the argument.

In terms of what was accomplished from the progress report, I finished that and more. The progress report stated that I still had to update the way the function arguments are statically checked, and input into the symbolTable and typeInfo table that those parameters can be constant. As well as update callExpr to check for this constant at the "same time" (right next to each other) as checking the parameter types. This was completed. After completing this, I thought about user defined types and realized I had not mentioned or considered that in any part of my previous thinking, partially because I was not sure if I would have time to get to it, but I had also not thought of it as a reasonable action. But looking further into it I came to realize that it is something that should be supported and added that. In terms of that, I added the functionality to create constant fields within user defined types and to have a constant variable initialized to a new user defined type. From that, there wasn't anything that I considered adding that I didn't accomplish.

I created multiple tests for this project. Starting with the basics, I added a simple test to the LexerTest. This test checked to make sure that the string "const" was marked as a Token type of CONST. After that I added some basic tests to the parser tests. These were along the lines of checking that the grammar was correct, that it would pass when const was used in a variable declaration, or in a function parameter, but not after a while statement or with incorrect variable declaration grammar. I wrote the most tests for the StaticCheckerTest. These started with the basic variable declarations, making sure it declared a constant variable correctly, and then that it errored when trying to mutate a constant variable. Then I moved on to the function parameters, testing that it worked with the syntax in the parameter location, and that it was able to distinguish when sending in a constant variable, or non constant variable and that it only errored when a constant variable was sent into a non constant parameter. After that, I added three tests for the user defined types. These checked that it passed when the correct syntax was used to create constant variables, whether within the type or of the type, and then also that it failed both when trying to mutate a constant field within a type, and trying to mutate the constant type.

To build and run this program on your own machine, I maintained the same syntax and instructions as the homeworks.

To build: bazel build //…

To run tests: bazel test –test_output=errors //…

To run examples: bazel-bin/mypl .\examples\(name of file)