# Type and Scope checking

We have partially implemented type and scope checking. Scope checking is fully functional, but type checking is only partially functional.

Run the scope/type checker with the QC3 class. It’s output is very verbose at this time and let’s the examiner see which parts have been implemented.

Here is an example output on testfile 1. We can see saving and retrieving from the symbol table, correct scoping, and some type checking on assignments.

Parsing completed

Function declaration found. Checking symbol table...

Function has not been declared before. Saving and continuing happily

Saving param

Saving param

j : inti : intintBegan new scope

return i + j>>Incorrect typein Plus expression

;

Exited current scope

Function declaration found. Checking symbol table...

>>Variable/Function sum has already been declared

Saving param

Saving param

j : floati : floatfloatBegan new scope

return i + j>>Incorrect typein Plus expression

;

Exited current scope

Began new scope

Saving new variable s1 to current scope in symbol table as ast.VarDeclNode@67a9b034

s1 : int = Function sum called. Checking if exists in scope

has been defined. Continuing happily

-1020Saving new variable s2 to current scope in symbol table as ast.VarDeclNode@356f5b17

s2 : float = Function sum called. Checking if exists in scope

has been defined. Continuing happily

10.0-20.0Saving new variable b to current scope in symbol table as ast.VarDeclNode@21c55e69

b : bool = if (s1 < s2 || s1term is ast.AccessorNode@24b950d1 and val is ast.LessThNode@268dc2d

isBool queried. className is AccessorNode

>>Incorrect typein OR expression

== s2){

s1 + s2 / (s1 + s2>>Incorrect typein Plus expression

)>>Incorrect typein DIVIDE expression

>>Incorrect typein Plus expression

>= 30Var for assignment was found

;

} else {

}

Exited current scope