Analysis of Semantic Rules

-Symbol table creation phase

1. A new table is created at the beginning of the program for the global scope.

(Done) Created right away, on method call

2. A new entry is created in the global table for each class declared in the program. These entries should contain links to local tables for these classes.

(Done) Done the program tree branches off into class, functions and main, and will create an appropriate entry for all nodes in these subtrees.

3. An entry in the appropriate table is created for each variable defined in the program, i.e. a class’ data members or a function’s local variables.

(Done) During symbol table creation

4. An entry in the appropriate table is created for each function definition (free functions and member functions). These entries should be links to local tables for these functions.

(Done) During symbol table creation

5. During symbol table creation, there are some semantic errors that are detected and reported, such as multiply declared identifiers in the same scope, as well warnings such as for shadowed inherited members.

(Done) Executed in analysis of symbol table

6. All declared member functions should have a corresponding function definition, and inversely. A member function that is declared but not defined constitutes an “no definition for declared member function” semantic error. If a member function is defined but not declared, it constitutes an “definition provided for undeclared member function” semantic error.

7. The content of the symbol tables should be output into a file in order to demonstrate their correctness/completeness.

(Done) Created proper to string methods for tables and all value, and sub tables… Tested!

8. Class and variable identifiers cannot be declared twice in the same scope. In such a case, a “multiply declared class”, “multiply declared data member”, or multiply declared local variable” semantic error message is issued.

9. Function overloading (i.e. two functions with the same name but with different parameter lists) should be allowed and reported as a semantic warning. This applies to member functions and free functions.

(Done) Executed in analysis of symbol table

-Semantic checking phase – binding and type checking

10. Type checking is applied on expressions (i.e. the type of sub-expressions should be inferred). Type checking should also be done for assignment (the type of the left and right hand side of the assignment operator must be the same) and return statements (the type of the returned value must be the same as the return type of the function, as declared in its function header).

(Done) for expressions and function return types in semantic analysis method

11. Any identifier referred to must be defined in the scope where it is used (failure should result in the following error messages: “use of undeclared local variable”, “use of undeclared free function”, “use of undeclared class”).

(Done) in semantic analysis method

12. Function calls are made with the right number and type of parameters. Expressions passed as parameters in a function call must be of the same type as declared in the function declaration.

(Done) in semantic analysis method

13. Referring to an array variable should be made using the same number of dimensions as declared in the variable declaration. Expressions used as an index must be of integer type. When passing an array as a parameter, the passed array must be of compatible dimensionality compared to the parameter declaration.

(Done) Inside type checking for expression and return types, in semantic analysis method

14. Circular class dependencies (through data members\inheritance) should be reported as semantic errors.

(Done) Executed in analysis of symbol table

15. The “.” operator should be used only on variables of a class type. If so, its right operand must be a member of that class. If not, a “undeclared data member” or “undeclared member function” semantic error should be issued.

(Done) in semantic analysis method

Design

Two passes down the abstract syntax tree from assignment 2.

First pass creating the symbol tables.

Then there is an analysis of the symbol table, to report possible error and warnings.

Second pass executing the semantic analysis on the program using the tree and symbol tables.