Andre Ibrahim
40132881

Section 1: List of semantic rules implemented

Legend

- 🟩 Completed
- 🟨 Partially completed
- 🟥 Not done

1. A new table is created at the beginning of the AST traversal for the global scope.
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.
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.
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.
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.
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.
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.
• 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).

    10.1 |X| Type error in expression
    10.2 |X| Type error in assignment statement
    10.3 |X| Type error in return statement

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 variable”, “use of undeclared member function”, “use of undeclared free function”, “use of undeclared class”).

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.

12.1 |X| function call with wrong number of parameters
12.2 |X| function call with wrong type of parameters

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.

13.1 |X| Use of array with wrong number of dimensions
13.2 |X| Array index is not an integer
13.3 |X| Array parameter using wrong number of dimensions

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

Section 2: Design

I am using the visitor pattern as described in class. There are two visitors the SymbTableVisitor and the TypeCheckingVisitor the symbTableVisitor is in charge of creating the global symbol table and the associated class and function tables. During the table creation we go through an initial phase of error and warning checking. We then go through the tables to see if there are declared functions that are undefined. Finally we use the TypeCheckingVisitor to go through the last phase of error checking. It is important to run the SymbolTableVisitor before the TypeCheckingVisitor since the type checking visitor uses the SymbolTables stored in the nodes to run its operations.

Section 3: Tools

Tools used in grammar transformation:

Tools used in code:

1. The only tool used in the code is the Lexer and Parser that I had built in assignment 1,2,3 everything else is vanilla Typescript.
2. Design Patterns: Elements of Reusable Object-Oriented Software and the notes to understand the visitor pattern
3. I did some modification to the attribute grammar to facilitate the the parsing of the tree

3.1 To get the parsing table and the first and follow sets I translated the grammar by hand from atocc to ucal format and used the university of calgary tool https://smlweb.cpsc.ucalgary.ca/start.html to get the parsing table and the first and follow sets

3.2 The ucal tool generates the parsing table as html and we can get that table and put it through this tool https://www.convertcsv.com/html-table-to-csv.htm to convert the tables into csv format.