Alexandru Bara 40132235 Assignment 4 Comp 442

Part 1

Green means done,

Red means not done,

Yellow means I tried to do it

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.

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).

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.

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.

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 rightoperandmustbeamemberofthatclass.
If not, a “undeclared data member” or “undeclared member function” semantic error should be issued.

Part 2

Phase 1:
The general design consists of 2 visitors. I have a class called semanticAnalyzer and it is called the Parser. It then gets in return the head node of the AST created. After that the head node is used to call the visitors. In order to get the visitors setup, I made the class node to contain all information needed at a node. Then each of the semantic concepts inherit the node and change the “toString” representation to indicate what is being held at that node. This way we have a bunch of different nodes and we can enable the visitor pattern. I also made a SymbolTable class that holds different tables and a SymbolEntry class that holds entries to be put in the table. Each node will keep the table it currently sees so it can search for declared variables in it. There are also

different types of entries based on if it is a class, function, variable, and so on. They all inherit from the main SymbolEntries class.

Phase 2:
In order to build the SymbolTable, I have a visitor for it. I traversed the nodes from the AST in a preorder style. Then at some nodes I have special logic for some nodes that enable us to either create tables or to create entries in our table. I also assign the local tables at each node. The last thing I do is check for some semantic errors and I report them in a file.
In order to do my type checking and bind variables I have another visitor that runs after the symbol table creation. This visitor also traverses the AST nodes but for a different purpose and the logic is at different places. This time, we are more concerned for when we actually use the variables declared and all the statements. All the types errors will also be reported in the same file as before.

Part 3

I used AtoCC in order to validate if my grammar is in LL1 format.
I used Ucalgary smlweb in order to create the parsing table and to get the first and follow sets. I tried to use the Ucalgary tool in order to fix my language, but I find that doing it by hand worked better.
I used google sheets to take the table from Ucalgary and make it in a csv file.
I used Joey Paquet's grammar tool to remove some ambiguities and left recursion.
Finally, I used the built in tools from Java like arraylist, hashmap and stacks in order to store my code.