Dublin City University
Ollscoil Chathair Bhaile Átha Cliath

DCU

Declaration

Name: Ryan McDyer	**Student ID Number: 13431038**
Programme: CASE 4	
Module Code: CA4003	
Assignment Title: **Semantic Analysis and Intermediate Representation for the CCAL Language**	
Submission Date: 12 December 2016	

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying is a grave and serious offence in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion, or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study
I have read and understood the referencing guidelines found at
http://www.library.dcu.ie/citing&refguide08.pdf

and/or recommended in the assignment guidelines.

Name: Ryan McDyer
Date: 12 December 2016

If submitting on-line, please tick the following box to indicate that you have read and that you agree with the statements above ☑

Abstract Syntax Tree

To implement the abstract syntax tree I followed the guide at:
http://stackoverflow.com/questions/13902239/how-to-implement-jjtree-on-grammar

I also had to add some new Rules to create the tree, such as the #FunctionCall rule.

Symbol table

My Symbol table contains the following:

- id : the name of the entry
- token : the type of the entry (eg: Integer, Boolean etc)
- scope : what scope it belongs to
- dataType : whether this entry is a Var, a Const or a Function
- map : the values associated with this id
- Methods to get, set and print these values

My scoped symbol table was implemented using a HashMap with the scope as the key, and the associated value is another HashMap. This second HashMap has an id as the key, and the associated value is the SymbolTable itself.

I implemented the Scope with a Stack. The broadest scope is “Global”.

Semantic checks

I implemented a number of the recommended semantic checks in my Semantic Check Visitor.

Is every identifier declared within scope before its is used?

This check ensures that an id is first declared in it’s scope before it is used in the right hand side of an assignment, or as a parameter to a function. I took this to mean that a var or const declared in a parent scope would have to be re-declared in a child scope. For example,

```
var i:integer;
main
{
     i = 1;
}
```

would not be valid because *i* has not been re-declared inside main.
This isn’t a perfect check because it results in a lot of false positives.

Is no identifier declared more than once in the same scope?

This check ensures that something like

```
main
{
     var i:integer;
     i = 1;
      var i:integer;
}
```

cannot occur in the code. Once again, I took this to mean that variables in different scopes can have the same ID name.

Are the arguments of the operators of correct type?

This check prints out the type (integer or boolean) of the arguments to the binary mathematical operation, and the binary logical operation.

Is every function called?

This check would help a user to make file sizes smaller by removing un-used functions. Every time the ASTFunc_decl node is visited I add that function's name to a set (declaredFunctions), and every time the ASTFunctionCall is visited I add that function's name to another set (calledFunctions).

Is every variable & constant written to and read from?

I achieved this by appending "Written = true" into the id's metadata whenever it is used in the ASTAssignment node, and by appending "Read = true" into the id's metadata when it is used in the ASTId node, but only when the ASTId node is not called as part of an Assignment.

Does every function call have the correct number of arguments?

I implemented this simply by comparing the number of arguments passed in with the number of parameters stored in the Symbol Table.

Intermediate Code

Unfortunately I was not able to implement the 3-address code IR segment of this assignment.