Polytechnic University of the Philippines
COLLEGE OF COMPUTER AND INFORMATION SCIENCES
DEPARTMENT OF COMPUTER SCIENCE
Sta. Mesa, Manila

C++ Programming Language
COMPILER

In partial fulfillment of the requirements for the subject

Compiler Design (COSC 4063)

CALVADORES, ANGELICA M.
CRISPINO, JEROME SAMUEL P.
ENRIQUE, MARIELLA B.
FANDINO, MARJORIE KATE P.
FONTARUM, JAICA IRISH F.
LOTO, RAMZEL RENZ L.
ROMAN, ROSE ANNE L.
SANTOS, RONNIE BOY A.

BSCS 3 – 1N

October 06 2016

PROF. RIA A. SAGUM

SEMANTIC ANALYZER

I. SEMANTIC RULES

PRODUCTION RULE	SEMANTIC RULE
<C++> => <declaration_stmt><stmt_list>	**<C++>.value** => <declaration_stmt.value><stmt_list.value>
<stmt_list> => <stmt>.<stmt_list> | <stmt> | (<stmt_list>);	**<stmt_list.value>** => <stmt>.<stmt_list.value> | <stmt.value> | (<stmt_list.value>);
<stmt> => <assignment_stmt> | <declaration_stmt> | <input_stmt> | <output_stmt> | <condition_stmt> | <iteration_stmt> | <comment> ;	**<stmt.value> =>** <assignment_stmt.value> | <declaration_stmt.value> | <input_stmt.value> | <output_stmt.value> | <condition_stmt.value> | <iteration_stmt.value> | <comment.value> ;
<declaration_stmt> => <data_type><id>[=(<value>)];	**<declaration_stmt.value> =>** <data_type.value><id.lexical>[=(<value.value>)];
<data_type> => int | char | String | float | double | bool	**<data_type.value> =>** int.lexical | char.lexical | String.lexical | float.lexical | double.lexical | bool.lexical
EXPRESSION	EXPRESSION
<expr> => <(expr)> | <id> | <arithmetic_expr> | <relational_expr> | <logical_expr> | <digits>+ | "<char>+"	**<expr> =>** <(expr).value> | <id>.value | <arithmetic_expr>.value | <relational_expr>.value | <logical_expr>.value | <digits>.value + | "<char>.value+"
<expr> => (<expr><arithmetic_oprt><expr>)	**<expr> =>** (<expr>.value<arithmetic_oprt>.value<expr>.value)
<arithmetic_oprt> => + | - | * | / | % | pow	**<arithmetic_oprt>** => +.lexical | -.lexical | *.lexical | /.lexical | %.lexical | pow.lexical
<arithmetic_unary_oprt> => ++	**<arithmetic_unary_oprt> =>** ++.lexical

<relational_expr> => (<expr><relational_oprt><expr>)	**<relational_expr> =>** (<expr>.value<relational_oprt>.value<expr>.value)
<relational_oprt> => == | > | < | != | <= | >=	**<relational_oprt> =>** ==.lexical | >.lexical | <.lexical | !=.lexical | <=.lexical | >=.lexical
<logical_expr> => (<expr><logical_oprt><expr>) **<logical_oprt> =>** && | || | !	**<logical_expr> =>** (<expr>.value<logical_oprt>.value<expr>.value) **<logical_oprt> =>** &&.lexical | ||.lexical | !.lexical
<condition_stmt> => if (<expr>) do (<stmt_list>) [else (<expr>) do (<stmt_list>) (<stmt_list>)]	**<condition_stmt> =>** if (<expr>.value) do (<stmt_list>.value) [else (<expr>.value) do (<stmt_list>.value) (<stmt_list>.value)]
<literals> ::== <char>* <digits>*	**<literals>** ::== <char>*.lexical <digits>*.lexical
• **Input Statement** **<input_stmt>** ::== cin >> id;	• **Input Statement** **<input_stmt>** ::== cin.lexical >> id.lexical;
• **Output Statement** **<output_stmt>** ::== cout << literals;	• **Output Statement** **<output_stmt>** ::== cout << literals.lexical;
• **Assignment Statement** **<assignment_stmt>** ::== <id><assignment_oprt>(<value> | <expr>))	• **Assignment Statement** **<assignment_stmt>** ::== <id>.lexical<assignment_oprt>.value(<value>.value | <expr>.value))
• **Conditional Statement** **<condition_stmt>** ::== <if_stmt> | <else_stmt> | <elseif_stmt>	• **Conditional Statement** **<condition_stmt>** ::== <if_stmt>.value | <else_stmt>.value | <elseif_stmt>.value

<if_stmt> ::== if(<expression>) {<stmt>} | if(<expression>) <stmt> | if(<expression>) {<stmt>} if(<expression>) | if(<expression>) <stmt> if(<expression>)	**<if_stmt>.value ::==** if.lexical(<expression>.value) {<stmt>}.value | if.lexical(<expression>.value) <stmt>.value | if.lexical(<expression>.value) {<stmt>}.value if.lexical(<expression>.value) | if.lexical(<expression>).value <stmt>.value if.lexical(<expression>.value)
<elseif_stmt> ::== else if (<expression>) {<stmt>} | else if (<expression>) <stmt> | else if (<expression>) {<stmt>} if(<expression>) | else if (<expression>) <stmt> if(<expression>)	**<elseif_stmt>.value** ::== else if.lexical (<expression>).value {<stmt>}.value | else if.lexical(<expression>).value <stmt>.value | else if.lexical(<expression>).value {<stmt>}.value else if.lexical(<expression>).value | else if.lexical(<expression>).value <stmt>.value if(<expression>).value
<else_stmt> ::== else {<stmt>} | else <stmt>	**<else_stmt>.value** ::== else.lexical {<stmt>}.value | else.lexical <stmt>.value
• **Iterative Statement** **<iteration_stmt>** ::== <for_stmt> | <while_stmt> | <do_stmt> **<while_stmt>** ::== while (<expression>) { <stmt> } | while (<expression>) <stmt> **<for_stmt>** ::== for (<assignment_stmt>;<expr>;<increment>) do (<stmt_list>) | for (<assignment_stmt>;<expr>;<increment>) do <stmt_list>	• **Iterative Statement** **<iteration_stmt>.value** ::== <for_stmt>.value | <while_stmt>.value | <do_stmt>.value **<while_stmt>.value** ::== while.lexical (<expression>).value {<stmt>}.value | while.lexical (<expression>).value <stmt>.value **<for_stmt>.value** ::== for.lexical (<assignment_stmt>.value;<expr>.value;<increment>.value) do.lexical (<stmt_list>).value | for.lexicall (<assignment_stmt>.value;<expr>.value;<increment>.value) do.lexical <stmt_list>.value

II. POSSIBLE SEMANTIC ERRORS

INPUT STATEMENT

C++ input statements accept a **"cin"** keyword then the **identifier** followed by a semicolon. No need for the format specifier unlike the C programming language.

Example: cin >> i;

Checking Possible Semantic Errors:

if(statement is complete)

compare data types of input and declared variable if assignable:

Equivalent:
int <-> int
float <-> float
bool <-> bool
…

Assignable:
int -> float (only in one direction)

else if(identifier is not in the symbol table within the scope)

message: **Input Statement Error: Semantic Error. Variable not declared.**

else if(semicolon missing after identifier)

message: **Input Statement Error: Syntax Error. Missing ';'.**

else if(">>" incomplete or mistaken as "<<" after 'cin' keyword)

message: **Input Statement Error: Syntax Error. Missing '>>'.**

else if(identifier missing after ">>" symbol)

message: **Input Statement Error: Syntax Error. Missing 'identifier'.**

OUTPUT STATEMENTS

C++ output statement starts with the keyword **"cout"** then the string to output. We can also concatenate a value from a variable by using **'<<'** and the variable name.

Example:

cout << “Hello World”;
This statement will display the string value - Hello World.
cout << “ Your age is ” << x;
This statement displays a string value - “Your age is” and it was concatenated by the value from the variable x.
cout << num;
This statement displays the value that was carried by the identifier num.

Checking Possible Semantic Errors:

if(statement is complete and passed the grammar rule)

prints the value of a variable or displays a string value or both

else if(identifier is not in the symbol table within the scope)

message: **Output Statement Error: Semantic Error. Variable not declared.**

else if(semicolon missing after identifier or string value)

message: **Output Statement Error: Syntax Error. ‘;’ Missing.**

else if(“<<” symbol missing before it concats a string value or identifier that holds a value)

message: **Output Statement Error: Syntax Error. ‘<<’ Missing.**

ASSIGNMENT STATEMENT

C++ assignment statements only accept the variable on the left side of the assignment operator. The corresponding value is at the right side of the operator.

Example:

a=8;
a=b;
a=1;
a=c+b;

Checking Possible Semantic Errors:

push identifier
push = (equal sign)
top of stack: = (equal sign)
next item supposed to be the value
push value
push ; (semicolon)

if(value matches identifier (equivalent/assignable)*)
store value to identifier
else if(identifier is not in the symbol table within the scope)
message: **Output Statement Error: Semantic Error. Variable not declared.**

else if(semicolon missing after value)

message: **Assignment Statement Error: Syntax Error. ';' Missing.**

else
message: **Assignment Statement Error: Semantic Error. Data type mismatched**

CONDITIONAL STATEMENT

C++ conditional statement starts with the keyword "**if"** then, after that is a logical/relational expression enclosed with the parenthesis. After the conditional statement if the expression is true then the following statement will be executed. If keyword **"if"** is present together with the valid expression it can be followed by a keyword **"else if" or "else"** so when the condition on the keyword **"if"** becomes false it will go to the other condition statement/s.

Example:

```
if (a==b)
   cout << "Pak"
else
  cout << "Ganern"
```

Checking Possible Semantic Errors:

push if
push (

check condition expression

push)

push stmt_list

push else

push smt_list

if (condition is true)

perform if_stmt

else

perform else_stmt

if(if_stmt missing before else_smt or else_if smt)

message: **Condition Error: IF statement expected**

else if(oParen missing before expression)

message: **Condition Error: Syntax Error. Missing "(".**

else if(identifier is not in the symbol table within the scope)

message: **Output Statement Error: Semantic Error. Variable not declared.**

else if(cParen missing after expression)

message: **Condition Error: Syntax Error. Missing ")".**

else if("<<" symbol missing before it concats a string value or identifier that holds a value)

message: **Output Statement Error: Syntax Error. '<<' Missing.**

else if(expression missing after if keyword)

message: **Condition Error: Syntax Error. Missing "expression".**

ITERATIVE STATEMENT

C++ iterative statements use **for loop, while loop and do-while loop**. In **for loop**, the compiler will accept iterative statements that starts with the keyword **"for"** after that the initialization, condition and iterator are to be executed by the program itself.

In initialization, it starts with the variable and the value of it serves as the initial value of the loop then semicolon. While in the condition, a variable is used to set the limit of the loop. The variable is followed by a relational operator and a value followed by a semicolon. Lastly on increment / decrement, a variable is followed by a unary operator.

Example:

```
for(a=0; a<5;a++)
    cout << a
```

Checking Possible Semantic Errors:

initialize variable to value

LOOP:

if(variable doesn't satisfy condition)

goto EXIT

else

perform variable/expression

increment variable

goto LOOP

EXIT:

terminate loop

if(condition/initial/iterator one of them are missing)

message: **Iterative Error: Syntax Error. Incomplete Loop Statement.**

else if(misses semi-colon between initial and condition or between condition and iterator)

message: **Iterative Error: Syntax Error. Missing ";".**

else if(oParen missing after keyword 'for')

message: **Condition Error: Syntax Error. Missing "(".**

else if(identifier is not in the symbol table within the scope)

message: **Output Statement Error: Semantic Error. Variable not declared.**

else if(cParen missing after iterator)

message: **Condition Error: Syntax Error. Missing ")".**

DECLARATION STATEMENT

C++ declaration statement accepts six kinds of data types:

- int - data type used to hold integer values.
- bool – data type than can hold the value true or false.
- float - data type used for floating point numbers.
- double - type used for floating point numbers.
- char - data type that can hold a single character.
- String – data type that can hold a sequence of characters.

Example:

```
int x = 3;
double b;
float d;
char c;
```

```
bool a;
char a='a';
char strn = "string";
bool fl = TRUE;
bool tr = FALSE;
```

Checking Possible Semantic Errors:

if(data_type missing at the head)

message: **Variable Declaration Error: Syntax Error**

else if(identifier already in the symbol table and within scope)

message: **Variable Declaration Error: Semantic Error: Double Declaration.**

else if(identifier was never seen again in the symbol table within intended scope)

message: **Warning: Variable was never used.**

else if(multiple data_type was recognized)

message: **Variable Declaration Error: Syntax Error.**

else if(identifier include spaces and special characters)

message: **Variable Declaration Error: Syntax Error.**

else if(identifier has an underscore or number in the first index)

message: **Variable Declaration Error: Syntax Error.**

else if(reserved words or keywords used as a identifier)

message: **Variable Declaration Error: Syntax Error.**

else if(identifier was declare by two data types)

message: **Variable Declaration Error: Syntax Error.**

else if (identifier missing before semicolon)

message: **Variable Declaration Error: Syntax Error**

else

store identifier with corresponding data_type