Compiler Construction (CS402)

Phase 2
Syntax Analyzer

Assigned on: February 24, 2016
Due Date: March 13, 2016

Instructions

All the problems related to Lexical Phase should be resolved first before starting this phase.

- Input file to this phase is the output file from the lexical phase that is the file with the tokens. However the input to your project would be a source code file to the lexical phase.
- Output in this phase is the symbol table.
- Suggested format of the symbol table is:

Lexeme	Type	Scope	Array Size	Function
a	char	global		
A	char	global		
b	char	global		
b	char	global		
main	int	global		Y
d	char	main		
c	int	main		
a	int	main		
e	int	main	10	
g	int	main	0	
f	floatptr	main		

- The lexeme in this table will be those identifiers that are declared in the code, that is, all the identifiers appearing in the declaration parts and it also includes the function names/ids in the program.
- The type field of the table is the respective type of the declared identifier and in case of function name the type will be the return type.
- The value in the Function will be **Y** if the identifier is a function name.
- The scope field in case of identifiers declared globally in the main program would be **Global** and in case of the declarations inside the function would be the respective function name. In case of function ids the scope will be **Global**. For example if a part of code is as follows, its symbol table will be as above

```
      char a, A, b, b;
```

```
void main (char d)
{
     int c, a, e[10], g[ ];
     float *f;
}
```

Thus the identifiers, which are not declared in the declaration part, anywhere, will not be stored in the symbol table at all. No entry in the symbol table for undeclared identifiers.

- The parsing should be done in single pass, that is, entries in the symbol table and syntax checking should be done in one flow, no two times input reads.
- There can be multiple entries in the symbol table if a particular identifier is declared more than one time. However the error such as duplicate declaration or undeclared identifier would be reported in the next phase.
- If there is a syntax error in the file, the output would be the respective error along with its line number
- The compiler should skip all the tokens on the line where the error was detected. The rest of error recovery strategy is up to you.
- The parsing method you will use should be Top Down Parsing (Predictive parsing) unless a parser generator such as YACC/BISON is used, in which case it will automatically generate bottom-up parser.
- Some of the common syntax errors are:
    - Semicolon expected
    - Operator expected
    - Opening/closing parenthesis missing
    - { or } missing
    - Identifier or integer constant expected
    - Keyword expected
    - Left/right bracket expected
    - Do without while

These are just few of the errors. Your parser should report all syntax errors that can occur considering the grammar given. Try to give specific errors not general error and report the position of the error as close to the actual error as possible. Also error message displayed should be meaningful for the user.