

## **ASSIGNMENT NUMBER: B-04**

**TITLE :** Yacc program to validate variable declarations

### **PROBLEM STATEMENT:**

Working of a compiler with their phases specifically syntactic analysis  
Basics of Java programming  
Basics of grammar analysis and regular expressions

### **OBJECTIVES:**

- Analyze source code for variables
- Identify data type and syntax of a variable

### **OUTCOMES:**

The students will be able to

- Match the variables and identify the data type.

### **THEORY:**

Lex recognizes regular expressions, whereas YACC recognizes entire grammar. Lex divides the input stream into tokens, while YACC uses these tokens and groups them together logically.

#### **The syntax of a YACC file:**

```
%{  
    declaration section  
}%  
    rules section  
%%  
    user defined functions
```

#### **Declaration section:**

Here, the definition section is same as that of Lex, where we can define all tokens and include header files. The declarations section is used to define the symbols used to define the target language and their relationship with each other. In particular, much of the additional information required to resolve ambiguities in the context-free grammar for the target language is provided here.

#### **Grammar Rules in Yacc:**

The rules section defines the context-free grammar to be accepted by the function Yacc generates, and associates with those rules C-language actions and additional precedence information. The grammar is described below, and a formal definition follows.

The rules section is comprised of one or more grammar rules. A grammar rule has the form:

A : BODY ;

The symbol A represents a non-terminal name, and BODY represents a sequence of zero or more names, literals, and semantic actions that can then be followed by optional precedence rules. Only the names and literals participate in the formation of the grammar; the semantic actions and precedence rules are used in other ways. The colon and the semicolon are Yacc punctuation.

If there are several successive grammar rules with the same left-hand side, the vertical bar '|' can be used to avoid rewriting the left-hand side; in this case the semicolon appears only after the last rule. The BODY part can be empty.

### **Programs Section**

The programs section can include the definition of the lexical analyzer yylex(), and any other functions; for example, those used in the actions specified in the grammar rules. It is unspecified whether the programs section precedes or follows the semantic actions in the output file; therefore, if the application contains any macro definitions and declarations intended to apply to the code in the semantic actions, it shall place them within "%{ ... %}" in the declarations section.

### **Interface to the Lexical Analyzer**

The yylex() function is an integer-valued function that returns a token number representing the kind of token read. If there is a value associated with the token returned by yylex() (see the discussion of tag above), it shall be assigned to the external variable yylval.

**CONCLUSION :** By performing this experiment we have successfully validate variable declaration .