SRES's Sanjivani College of Engineering, Kopargaon
(*An Autonomous Institute*)
Department of Computer Engineering

SPOS Lab Manual

Assignment No. 07

AIM:
Write a program using YACC specifications to implement syntax analysis phase of compiler to validate type and syntax of variable declaration in Java.

PROBLEM DEFINITION:
Write a program using YACC specifications to implement syntax analysis phase of compiler to validate type and syntax of variable declaration in Java.

OBJECTIVES:
1. To understand syntax analyzer
2. To use the lex and yacc tool

INPUT:
Declarative statement

OUTPUT:
Statement syntactically valid or invalid

THEORY:

- YACC stands for **Yet Another Compiler Compiler**.
- YACC provides a tool to produce a parser for a given grammar.
- YACC is a program designed to compile a LALR (1) grammar.
- It is used to produce the source code of the syntactic analyzer of the language produced by LALR (1) grammar.
- The input of YACC is the rule or grammar and the output is a C program.

Structure of a YACC source program
A YACC source program is structurally similar to a LEX one.

declarations
%%
rules
%%
routines

- The declaration section may be empty. Moreover, if the routines section is omitted, the second %% mark may be omitted also.
- Blanks, tabs, and newlines are ignored except that they may not appear in names.

THE DECLARATIONS SECTION may contain the following items.

- Declarations of tokens. Yacc requires token names to be declared as such using the keyword %token.
- Declaration of the start symbol using the keyword %start
- C declarations: included files, global variables, types.
- C code between %{ and %}.

RULES SECTION.

A rule has the form:

nonterminal : sentential form(string of terminal and nonterminal symbols)
| sentential form
.....
| sentential form
;

Actions may be associated with rules and are executed when the associated sentential form is matched.

ROUTINES SECTION:

In the routines section required functions can be defined. Standard yacc functions like yyparse(), yyerror(), yywrap() are defined in this section. yyparse is called in the main() function. Also required user defined functions can be defined in this section, which may be called through the actions part of rules section.

LEX-YACC INTERACTION

yyparse() calls yylex() when it needs a new token.

LEX	YACC
return(TOKEN)	%token TOKEN
	TOKEN is used in production

The external variable yylval

- is used in a LEX source program to return values of lexemes,
- yylval is assumed to be integer if you take no other action.
- Changes related to yylval must be made in the definitions section of YACC specification by adding new types in the following way

```
%union {
   (type fieldname)
   (type fieldname)
   .....
}
```

and defining which token and non-terminals will use these types

%token <fieldname> token
%type <fieldname> non-terminal

- in LEX specification by using the fieldnames in the assignment as follows

  yylval.fieldname = ...

- If you need a record type, then add it in the union. Example:

```
%union {
  struct s {
    double fvalue;
    int ivalue;
  } t;
}
```

- in the LEX specification use the record name and record field in assignments:

  yylval.t.ivalue = ...

- in the YACC rules specification use the record field only in the assignment:

  $1.ivalue = ...

  assuming that $1 has the appropriate type, whatever it denotes.

Functions Used in Yacc Program

yyparse() : This function is called in the third section within the main() function of yacc program. yyparse internally calls yylex for demanding the next tokens from lexical analyzer. These tokens are used by yacc during parsing process**.**

yywrap(): yywrap returns value 1 when scanning of whole input is finished that is processing of input file is done. Else yywrap function returns value 0.

yyerror(): yyerror function is called by yacc when input sentence is not following the grammar rules of the language. Input sentence is syntactically incorrect then yyerror will give error message

COMPILING AND EXECUTING LEX AND YACC PROGRAMS

Compile LEX Program example.l to generate c program lex.yy.c
lex example.l -→ lex.yy.c

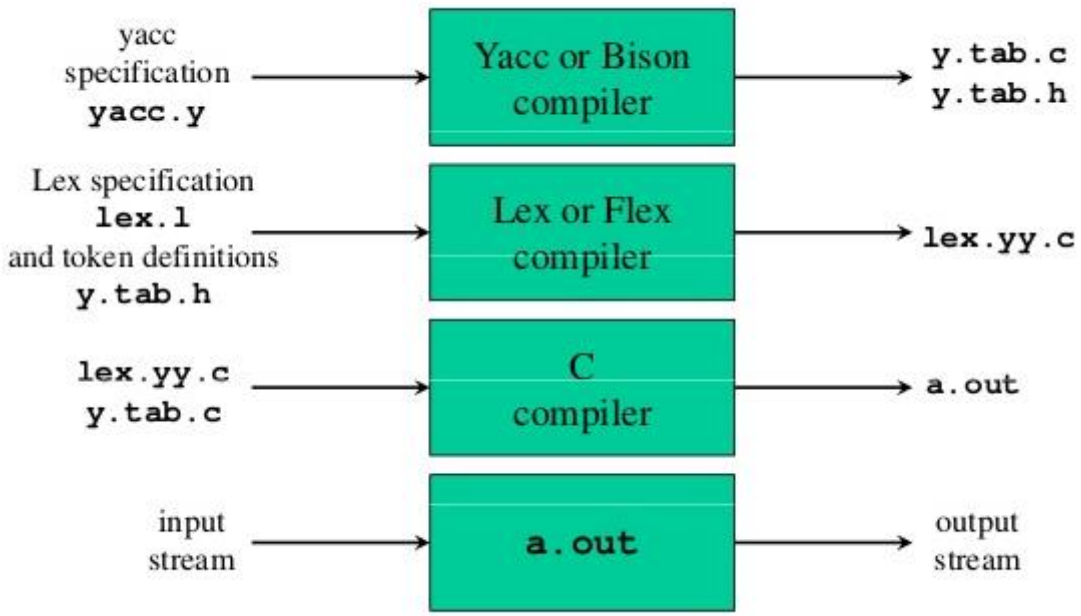
Compile YACC Program example.y to generate c program y.tab.c and -d will generate y.tab.h which holds unique identifiers for tokens
yacc -d example.y --→ y.tab.c, y.tab.h

Compile LEX 'C' program and YACC 'C' program to generate default object program a.out which can be renamed by using -o as below

cc -o example lex.yy.c y.tab.c

Execute Object Program

./example



ALGORITHM:

1. Write the lex program which recognise necessary tokens extension
2. Write yacc program with grammar for declarative statement
3. compile lex and yacc program to generate lex.yy.c and y.tab.c programs
4. Compile c program to generate the object program a.out
5. a.out is the generated syntax analyzer
6. Provide the input declarative statement to syntax analyzer to generate the output

Sample Program

LEX Program (example.l)

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
int|float|char          return TYPE;
```

```
","                    return COMMA;
";"                    return SC;
[a-zA-Z]+[a-zA-Z0-9]*  return ID;
"n"                    return NL;
%%
```

YACC Program (example.y)

```
%{
#include<stdio.h>
%}
%token ID TYPE SC NL COMMA
%%
start : TYPE varlist SC NL {printf("valid");}

varlist : varlist COMMA ID
        | ID;
%%
void yyerror(const char *str)
        {       printf("error");      }
int yywrap()
        {       return 1;              }
main()
        {       yyparse();             }
```

INPUT: int a, b, c;

OUTPUT: valid

OBSERVATION:

It is observed that above parser parse the declarative statements

CONCLUSION:

LEX and YACC tool is used to parse the declarative statement

References:

Lex & Yacc Book by Doug Brown, John R. Levine, and Tony Mason

Prepared by
Prof.N.G.Pardeshi
Subject Teacher

Approved by
Dr. D.B.Kshirsagar
HOD