Peitam Mogal TE(B)-17

Assignment No:7

LEX & YACC Program.

Aim:- Design Lex & Yacc program to validate type and Syntax of variable declaration in Java.

Problem Statement:- Write a program using Yacc Specifications to implement lexical analysis phase of compiler to validate type and Syntax of variable declaration in Java.

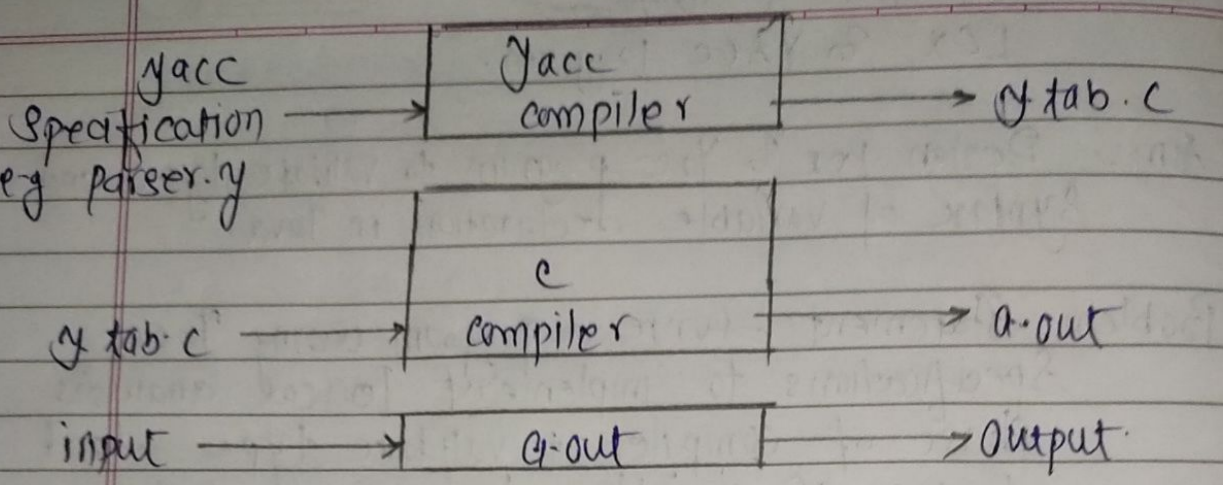
Pre-requisites:- LEX 110, LEX 120, LEX 130, LEX 140, LEX 160, 250.

Software requirements:-

S.NO	Facilities required	Quantity.
1	System	1
2	O/S	Ubuntu Kylin
3	S/w name	FLEX, YACC (LEX & YACC).

Theory:

Yacc (Yet another compiler-Compiler) is a computer program for the UNIX operating System developed by Stephen C. Johnson. It is a Look Ahead Left-to-Right (LALR) parser generator, generating a parser, the part of compiler that tries to make syntactic sense of the Source Code, Specifically a LALR Parser, based on an analysis grammer written in a notation similar to Backus-Naur Form (BNF). Yacc is Supplied as a Standard utility on BSD and AT&T UNIX. GNU based LINUX distributions include Bison, a forward-compatible Yacc replacement.



YACC Parser Generation Model.

Structure of a Yacc file:- A Yacc file looks much like a lex file:

..... definations .....
%%
..... rules .....
%%
..... code .....

Definations as with lex, all code between %{ and %} is copied to the beginning of the resulting C file. Rules As with lex, a number of combinations of pattern and location. The patterns are now those of a context-free grammer, rather than of a regular grammer as was the case with lex code. This can be very elaborate, but the main ingredient is the call to yyparse. For example, the grammer for an expression that multiplies and adds numbers is

1. E → E + E
2. E → E * E
3. E → id.

Translating, Compiling and Executing A Yacc program :-

The lex program file consists of lex specification and should be named .l and the Yacc program consists of Yacc specifications and should be named .y following command may be issued to generate the parser.

```
Lex <filename>.l
Yacc -d <filename>.y
cc lex.yy.c y.tab.c -ll
./a.out
```

Yacc reads the grammer description in .y and generates a parser function yyparse, in file y.tab.c. The -d option causes yacc to generate the definitions for tokens that are declared in the .y and place them in file y.tab.h.
Finaly, the lexer and the parser are compiled and linked (-ll) together to form the output file, a.out (by default).

Lexical Analyzer For YACC:

The user must supply a lexical analyzer to read the input stream and communicate tokens (with values if desired) to the parser.
The parser and the lexical analyzer must agree on those tokens numbers in order for communication between them to take place.
For example, suppose that the token name DIGIT has been defined in the declarations section of the Yacc specifications file.

```
yylex () {
external int yylval;
int c;
...
c = getchar ();
... switch (c) {
... case '0':
case '1':
...
case '9':

yylval = c - '0';
return (DIGIT);
... }
...
```

The intern is to return a token number of DIGIT, and a value equal to the numerical value of the digit. Provided that the lexical analyzer code is placed in the programs section of the specification file.

As mentioned above, the token numbers may be chosen by Yacc or by the user. In the default situation, the numbers are chosen by Yacc. The default token number for a literal character is the numerical value of the character in the local character set.

Comparing Sentence types :-
Sentences give structure to language and in English, they come in four types : Simple, compound, complex and compound-complex.

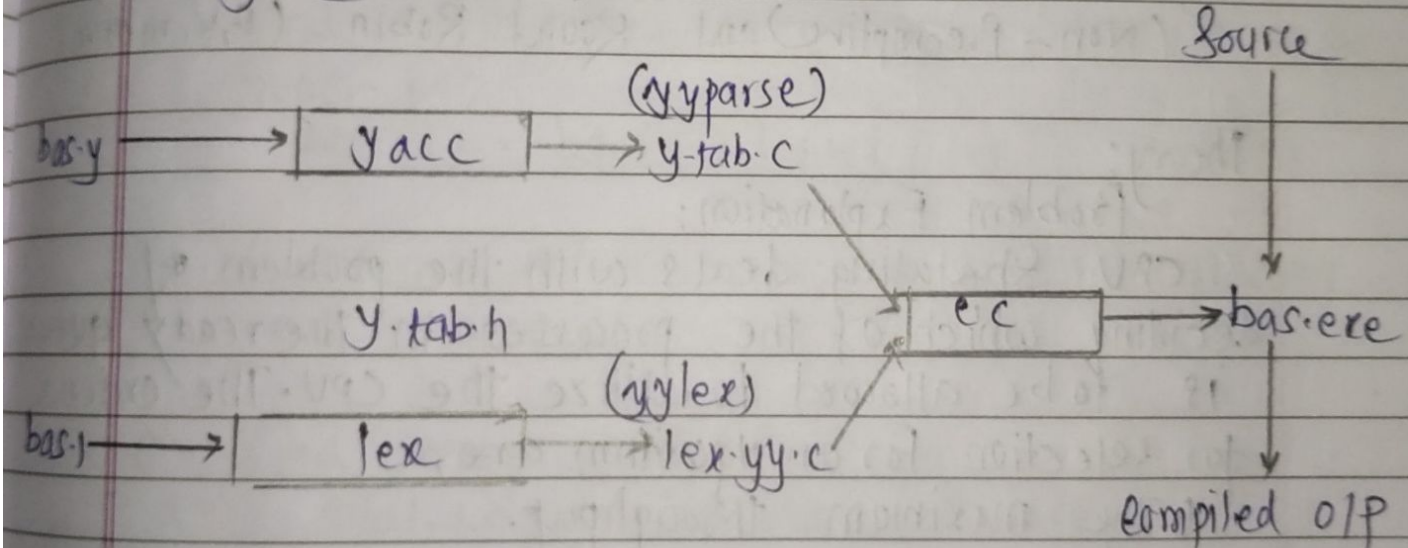
1. The simple sentence is an independent clause with one subject and one verb. For example: we are the indian.

2. The compound sentence is two or more independent clause, joined with comma, semicolon & conjunctions.

Application

YACC is used to generate parsers, which are an integral part of compiler.

Design (architecture):



Conclusion:

Thus, we have studied lexical analyzer, Syntax. anlysis and implemented Lex & Yacc application for Syntax analyzer to validate the given infix expression.

Name: Pritam Gotiram Mogal **DivRoll**: TE –B-17

Assignment No. 07 [LEX Program]

Problem Satement: Write a program using YACC specifications to implement syntax analysis phase of compiler to recognize simple and compound sentences given in input file

---

1. Code b5.l:

```
%{
   #include<stdio.h>
   int simple=0;
%}

%%
[ \t\n][aA][nN][dD][ \t\n] {simple=1; }
[ \t\n][bB][uU][tT][ \t\n] {simple=1; }
[ \t\n][oO][rR][ \t\n] {simple=1; }
. ;
%%

int yywrap(){

}

int main(){
   printf("Enter sentence:  \n");
yylex();     if(simple==1){
       printf("compound\n\n");
   }
else{
       printf("simple\n\n");
   }
return 0;
}
```

OUTPUT

```
Pritam-spos@Pritam-HP:~/SPOSL/LexProgram/B5$  lex b5.l

Pritam-spos@Pritam-HP:~/SPOSL/LexProgram/B5$  gcc lex.yy.c

Enter sentence:

Hi Friends

Simple

Pritam-spos@Pritam-HP:~/SPOSL/LexProgram/B5$  lex b5.l
```

```
Pritam-spos@Pritam-HP:~/SPOSL/LexProgram/B5$  gcc lex.yy.c

Pritam-spos@Pritam-HP:~/SPOSL/LexProgram/B5$   ./a.out

Enter sentence:

Hi friends or chai pilo

Compound
```