PROJECT: PARSER FOR C LANGUAGE USING YACC

INTRODUCING LEX AND YACC

**SUBMITTED TO** **–**  Dr. Asif Ekbal

**SUBMITTED BY –**  HIMANSHU GARG

**ROLL NO –** 1301CS20

**DATE OF SUBMISSION** **–**  31/03/2016

**GROUP** – 10

**IIT PATNA**

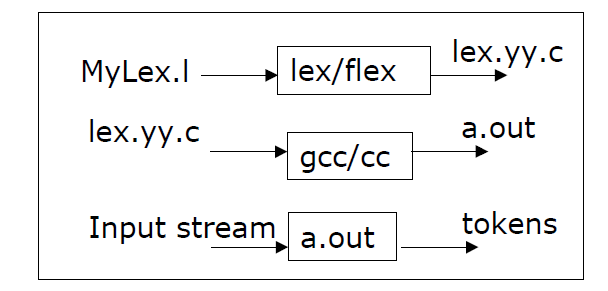
**Objective : Generate a Parser for C language using YACC**

**Tools Used:**

* Using Lex to build Scanner
* Using YACC for parser Generator

**Lex : A lexical Analyzer**

* **Using Lex to build Scanner**

****

* Lex file is used to identify tokens in the input file.
* Generated tokens are used to create symbol table which can be helpul in error recovery and display error message in console.
* Write lex specification

1. Save it in a file(MyLex.l).

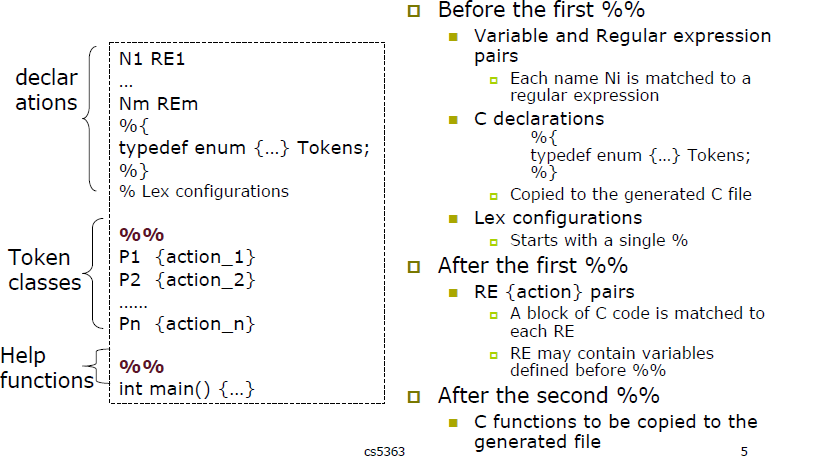
* Compile the lex specification file by invoking lex/flex

1. Lex Mylex.l
2. A lex.yy.c is generated by lex.
3. Rename lex.yy.c file if desired

* Compile the generate C file

1. gcc –c lex.yy.c (or gcc –c MyLex.c)

* **Structure of a Lex specification file**



* **Example of Lex Specification**

* **alpha [a-zA-Z]**
* **digit [0-9]**

Regular expressions for finding alphabets and digits

* **[\n]**

Regular expression for newline and count line number

* **{digit}+**

Regular expression to identify a number

* **{alpha}({alpha}|{digit})\***

Regular expression to identify an identifier

* **\/\/.\*;**
* **\/\*.(\n).\\/;**

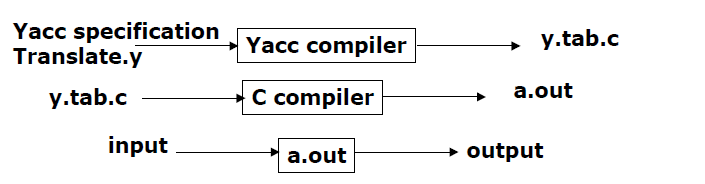
Regular expression to identify an identifier

* **H {a-fA-F0-9}**
* **E {Ee]{+-}{D}+**
* **FS (f|F|l|L)**
* **IS (u|U|l|L)\***
* **0[xX]{H}+{IS}?**
* **0{D}+{IS}?**
* **{D}+{IS}?**
* **L?’(\\.|[^\\’])+’**
* **{D}\*”.”{D}+({E})?{FS}?**
* **{D}+”.”{D}\*({E})?{FS}?**

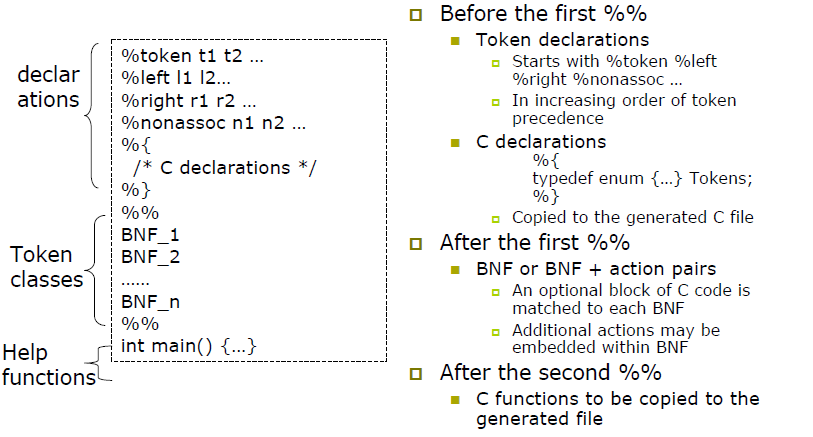
Regular expression to identify hexadecimalconstants

**YACC : Yet another compiler compiler**

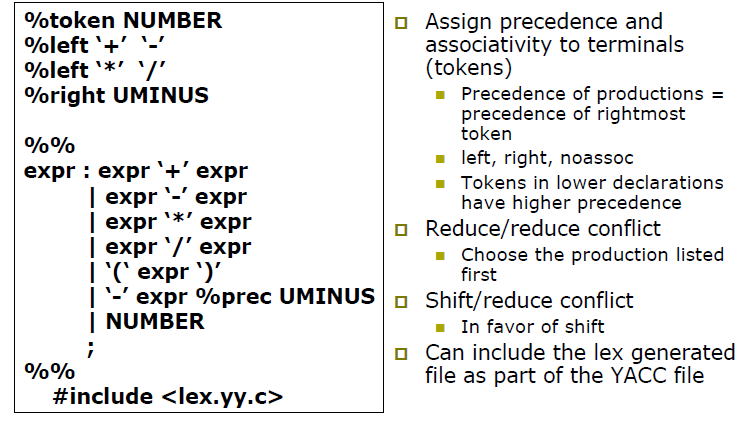
* **YACC : LALR parser generators**
* Automatically generate LALR parsers
* More powerful than LR(0) and weaker than LR(1)
* Created by S.C. Johnson in 1970’s



* Compile your YACC specifications file by invoking yacc
* Compile your generated C file: gcc –c y.tab.c (or gcc –c Translate.c)
* **The Structure of a YACC specification file**
* Write production rules according to the given grammar to parse the input.
* Identify tokens using lex file.
* Use different techniques to handle errors.

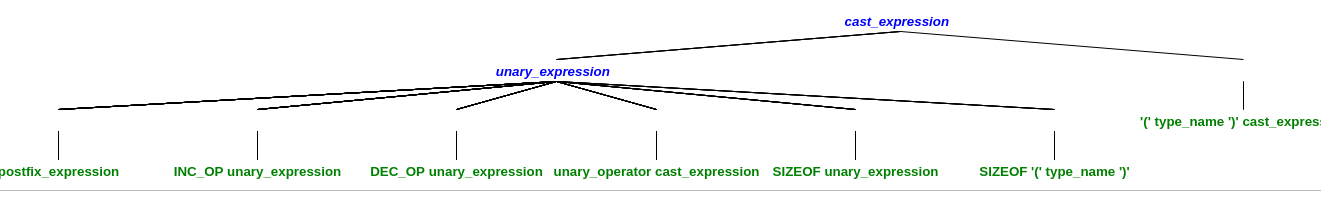


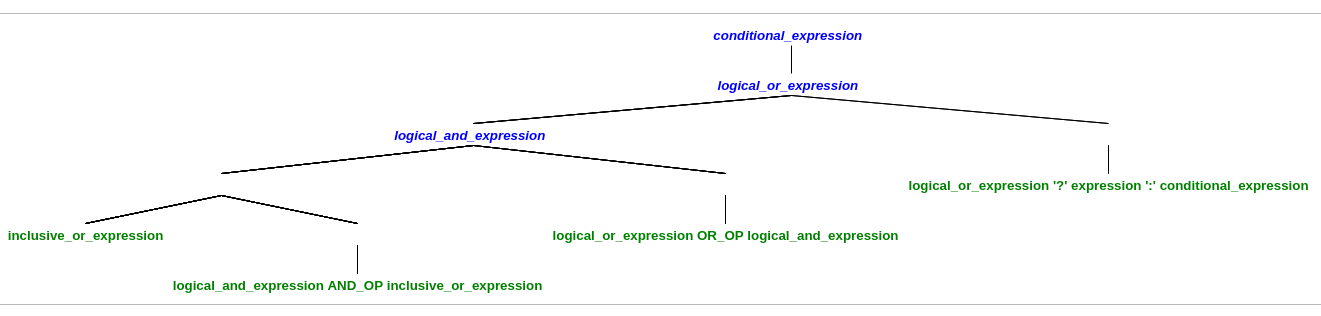
* **Example YACC Specificatios**

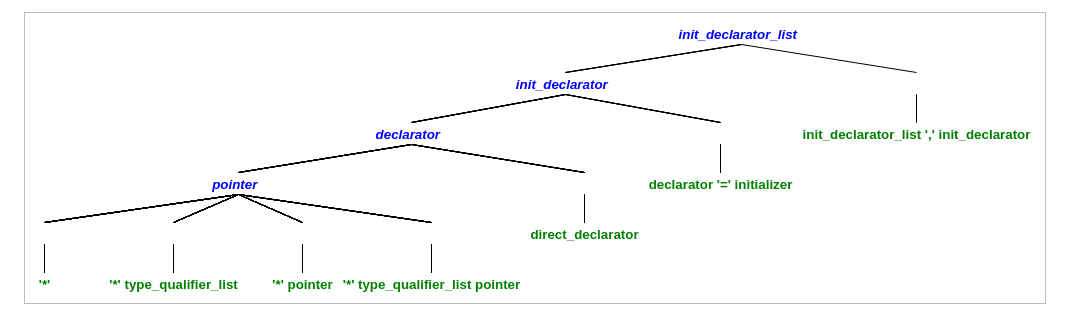


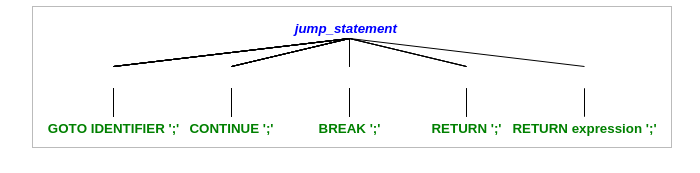
**Contribution Towards Project**

* Identify tokens from grammar and write suitable regex for them.
* Generate symbol table from tokens.
* Write production rules in yacc file according to given grammar and modify them according to the requirement of the feature.
* Write functions to verify identifiers whether they are declared or not in their scope. If not, print error message on console.
* Identify a way on how to use variables of yacc file into lex file and vice-versa.
* Generate parse trees for some of the production rules.

****

****

****

****