Assignment No.3

AIM:

Generate the grammar for ‘C’ language.

THEORY:

Grammar is defined as
<V , ∑ , S , P>
Where V – set of nonterminals
∑ - set of terminals
S – Start symbol
P – Set of productions

For C-language

```
S  →  <Program>
∑ → {a, b, c,……..A, B, C,…..0, 1,……&,….}
V → {<expression>, <statement>, <var>, <expression>, <include-statement>}
P  → {<program>
       <include_statement>
       main ()
      {
             <dec_statement>,
             <statement>
      }
      }
      <include_statement>  →  #include ‘<’ <directive> ‘>’
                               <include_statement> |^
      <directive>  →  stdio.h |conio.h| ….
      <dec_statement>  →  <dd> <dec_statement> |^
      <dd> → <datatype> <variable> <next-var>;
      <datatype> → int | char | float
      <next-var>  →  <variable>  <next-var>  |^
      <var> → <letter> <next>
      <next> → <letter> <next>| <digit> <next>| <next>|^
      <letter> → A| B| C| …………..a|b|c|………|z
```

```
<digit> → 0|1|2|3|……….|9
<variable> → <var>=<number>|<var>=’<letter>’|<var>
<statement>  → <printf-statement>
               | <scanf-statement>
               | <if-statement>
               | <for-statement>
               | <while-statement>
<if-statement>   → if (<c-expression>)
                   | if (<c-expression>){<statement>} else {<statement>}
<expression>  → <expression>+<expression>|<var>|<number>
<term>  → <term>*<term>| <term>| <term><factor>
<factor> →(<expression>)|<expression>|<var>|<number>
<number> →<digit><num>
<num>→ <digit><num>| <digit><digit><var>|^
<no> → <digit> <no>|^
<while-statement> → while (<c-expression>){<statement>}
<for-statement> → for <expression>;<expression>;<expression>)
                   {<statement>}
<all-statement>  → <expression>
<scanf-statement>  → scanf (“%<sp><next-specifications> “ , <var>
                      <scanf-var>)
<scanf-var> → &<scanf-var>|^
<next-specifications> →  %<sp><next-specifications>|^
<sp> → d|f|…
<printf-statement>  → printf (“<string>” ,  <printf-var>)
<string> → <any>
<printf-statement>  →<var> < printf-var>|^
<any> → 0<any>| <any> |………|a<any>|……….|=<any>
```

CONCLUSION:

Thus the grammar for C-language is generated

REFERENCES:

- http://marvin.cs.uidaho.edu/Teaching/CS445/c-Grammar.pdf