ANSI C grammar, Lex specification

In 1985, Jeff Lee published this Lex specification together with a Yacc grammar for the April 30, 1985 ANSI C draft. Tom Stockfisch reposted both to net.sources in 1987; that original, as mentioned in the answer to question 17.25 of the comp.lang.c FAQ, can be ftp'ed from ftp.uu.net, file usenet/net.sources/ansi.c.grammar.Z.

I intend to keep this version as close to the current C Standard grammar as possible; please let me know if you discover discrepancies.

Jutta Degener, 1995

```
D                       [0-9]
L                       [a-zA-Z_]
H                       [a-fA-F0-9]
E                       [Ee][+-]?{D}+
FS                      (f|F|l|L)
IS                      (u|U|l|L)*

%{
#include <stdio.h>
#include "y.tab.h"

void count();
%}

%%
"/*"                    { comment(); }

"auto"                  { count(); return(AUTO); }
"break"                 { count(); return(BREAK); }
"case"                  { count(); return(CASE); }
"char"                  { count(); return(CHAR); }
"const"                 { count(); return(CONST); }
"continue"              { count(); return(CONTINUE); }
"default"               { count(); return(DEFAULT); }
"do"                    { count(); return(DO); }
"double"                { count(); return(DOUBLE); }
"else"                  { count(); return(ELSE); }
"enum"                  { count(); return(ENUM); }
"extern"                { count(); return(EXTERN); }
"float"                 { count(); return(FLOAT); }
"for"                   { count(); return(FOR); }
"goto"                  { count(); return(GOTO); }
"if"                    { count(); return(IF); }
"int"                   { count(); return(INT); }
"long"                  { count(); return(LONG); }
"register"              { count(); return(REGISTER); }
"return"                { count(); return(RETURN); }
"short"                 { count(); return(SHORT); }
"signed"                { count(); return(SIGNED); }
"sizeof"                { count(); return(SIZEOF); }
"static"                { count(); return(STATIC); }
"struct"                { count(); return(STRUCT); }
"switch"                { count(); return(SWITCH); }
"typedef"               { count(); return(TYPEDEF); }
"union"                 { count(); return(UNION); }
"unsigned"              { count(); return(UNSIGNED); }
"void"                  { count(); return(VOID); }
"volatile"              { count(); return(VOLATILE); }
```

```
"while"                 { count(); return(WHILE); }

{L}({L}|{D})*           { count(); return(check_type()); }

0[xX]{H}+{IS}?          { count(); return(CONSTANT); }
0{D}+{IS}?              { count(); return(CONSTANT); }
{D}+{IS}?               { count(); return(CONSTANT); }
L?'(\\.|[^\\'])+'       { count(); return(CONSTANT); }

{D}+{E}{FS}?            { count(); return(CONSTANT); }
{D}*"."{D}+({E})?{FS}?  { count(); return(CONSTANT); }
{D}+"."{D}*({E})?{FS}?  { count(); return(CONSTANT); }

L?\"(\\.|[^\\"])*\"     { count(); return(STRING_LITERAL); }

"..."                   { count(); return(ELLIPSIS); }
">>="                   { count(); return(RIGHT_ASSIGN); }
"<<="                   { count(); return(LEFT_ASSIGN); }
"+="                    { count(); return(ADD_ASSIGN); }
"-="                    { count(); return(SUB_ASSIGN); }
"*="                    { count(); return(MUL_ASSIGN); }
"/="                    { count(); return(DIV_ASSIGN); }
"%="                    { count(); return(MOD_ASSIGN); }
"&="                    { count(); return(AND_ASSIGN); }
"^="                    { count(); return(XOR_ASSIGN); }
"|="                    { count(); return(OR_ASSIGN); }
">>"                    { count(); return(RIGHT_OP); }
"<<"                    { count(); return(LEFT_OP); }
"++"                    { count(); return(INC_OP); }
"--"                    { count(); return(DEC_OP); }
"->"                    { count(); return(PTR_OP); }
"&&"                    { count(); return(AND_OP); }
"||"                    { count(); return(OR_OP); }
"<="                    { count(); return(LE_OP); }
">="                    { count(); return(GE_OP); }
"=="                    { count(); return(EQ_OP); }
"!="                    { count(); return(NE_OP); }
";"                     { count(); return(';'); }
("{"|"<%")              { count(); return('{'); }
("}"|"%>")              { count(); return('}'); }
","                     { count(); return(','); }
":"                     { count(); return(':'); }
"="                     { count(); return('='); }
"("                     { count(); return('('); }
")"                     { count(); return(')'); }
("["|"<:")              { count(); return('['); }
("]"|":>")              { count(); return(']'); }
"."                     { count(); return('.'); }
"&"                     { count(); return('&'); }
"!"                     { count(); return('!'); }
"~"                     { count(); return('~'); }
"-"                     { count(); return('-'); }
"+"                     { count(); return('+'); }
"*"                     { count(); return('*'); }
"/"                     { count(); return('/'); }
"%"                     { count(); return('%'); }
"<"                     { count(); return('<'); }
">"                     { count(); return('>'); }
"^"                     { count(); return('^'); }
"|"                     { count(); return('|'); }
"?"                     { count(); return('?'); }

[ \t\v\n\f]             { count(); }
.                       { /* ignore bad characters */ }
```

```
%%

yywrap()
{
        return(1);
}


comment()
{
        char c, c1;

loop:
        while ((c = input()) != '*' && c != 0)
                putchar(c);

        if ((c1 = input()) != '/' && c != 0)
        {
                unput(c1);
                goto loop;
        }

        if (c != 0)
                putchar(c1);
}


int column = 0;

void count()
{
        int i;

        for (i = 0; yytext[i] != '\0'; i++)
                if (yytext[i] == '\n')
                        column = 0;
                else if (yytext[i] == '\t')
                        column += 8 - (column % 8);
                else
                        column++;

        ECHO;
}


int check_type()
{
/*
* pseudo code --- this is what it should check
*
*       if (yytext == type_name)
*               return(TYPE_NAME);
*
*       return(IDENTIFIER);
*/

/*
*       it actually will only return IDENTIFIER
*/

        return(IDENTIFIER);
}
```