Expression Evaluator Grammar

©1999-2017 Garth Santor

# Variables in Expressions

<expression> ::= <variable> ‘=’ <expression> ;

<expression> ::= <real expression> | <integer expression> | <Boolean expression> | <variable> ;

# Real Expressions

<real expression> ::= <real expression> <binary real operator> <real expression> ;

<real expression> ::= <unary real operator> <real expression> ;

<real expression> ::= <real function-1 keyword> ‘(‘ <real expression> ‘)’ ;

<real expression> ::= <real function-2 keyword> ‘(‘ <real expression> ‘,’ <real expression> ‘)’ ;

<real expression> ::= <real literal> ;

<real expression> ::= <integer expression> ;

# Integer Expressions

<integer expression> ::= <integer expression> <binary integer operator> <integer expression> ;

<integer expression> ::= <unary integer operator> <integer expression> ;

<integer expression> ::= <integer function-1 keyword> ‘(‘ <integer expression> ‘)’ ;

<integer expression> ::= <integer function-2 keyword> ‘(‘ <integer expression> ‘,’ <integer expression> ‘)’ ;

<integer expression> ::= <integer expression> ‘!’ ;

<integer expression> ::= <integer literal> | <binary literal>;

# Boolean Expressions

<Boolean expression> ::= <Boolean expression> <binary Boolean named operator> <Boolean expression> ;

<Boolean expression> ::= <unary Boolean named operator> <Boolean expression> ;

<Boolean expression> ::= <Boolean literal> ;

# Keywords, Operators & Variables

<variable> ::= <identifier> ∉ <keyword> ;

<identifier> ::= <alpha> { <alphanum> } ;

<keyword> ::= <Boolean literal> | <real constant> | <function keyword> | <named operator> ;

<function keyword> ::= <real function keyword> | < integer function keyword> ;

<real function keyword > ::= <real function-2 keyword> | <real function-1 keyword> ;

<real function-2 keyword > ::= ‘arctan2’ | ‘max’ | ‘min’ ;

<real function-1 keyword > ::= ‘abs’ | ‘arccos’ | ‘arcsin’ | ‘arctan’ | ceil’ | ‘cos’ | ‘exp’ | ‘floor’ | ‘lb’ | ‘ln’ | ‘log’ | ‘sin’ | ‘sqrt’ | ‘tan’|’fib’|’pel’|’syl’ ;

<integer function keyword> ::= <integer function-2 keyword> | <integer function-1 keyword> ;

<integer function-2 keyword > ::= ‘max’ | ‘min’ ;

<integer function-1 keyword > ::= ‘abs’ | ‘result’ | ‘binary’ ;

<named operator> ::= <binary named operator> | <unary named operator> ;

<binary named operator> ::= <binary integer named operator> | <binary Boolean named operator> ;

<unary named operator> ::= <unary Boolean named operator> ;

<unary real operator> ::= ‘+’ | ‘-‘ ;

<binary real operator> ::= ‘+’ | ‘-‘ | ‘\*’ | ‘/’ | ‘\*\*’;

<unary integer operator> ::= ‘+’ | ‘-‘ ;

<binary integer operator> ::= <binary integer named operator> | ‘+’ | ‘-‘ | ‘\*’ | ‘/’ | ‘%’ | ‘&’ | ‘^’ | ‘|’ | ‘<<’ | ‘>>’ | ‘\*\*’;

<binary integer named operator> ::= ‘mod’ ;

<binary Boolean named operator> ::= ‘and’ | ‘or’ | ‘xor’ | ‘nand’ | ‘nor’ | ‘xnor’ ;

<unary Boolean named operator> ::= ‘not’ ;

<real constant> ::= ‘PI’ | ‘E’ ;

# Literals

<binary literal> ::= ‘0b’ <binary digit> + ;

<Boolean literal> ::= ‘true’ | ‘false’ ;

<integer literal> ::= <digit> + ;

<real literal> ::= <digit>+ <decimal point> <digit>+ ;

<decimal point> ::= ‘.’ ;

<alphanum> ::= <alpha> | <digit> ;

<binary digit> ::= ‘0’ | ‘1’ ;

<digit> ::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’ ;

<alpha> ::= <lower case alpha> | <upper case alpha> ;

<lower case alpha> ::= ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’ | ‘g’ | ‘h’ | ‘i’ | ‘j’ | ‘k’ | ‘l’ | ‘m’ | ‘n’ | ‘o’ | ‘p’ | ‘q’ | ‘r’ | ‘s’ | ‘t’ | ‘u’ | ‘v’ | ‘w’ | ‘x’ | ‘y’ | ‘z’ ;

<upper case alpha> ::= ‘A’ | ‘B’ | ‘C’ | ‘D’ | ‘E’ | ‘F’ | ‘G’ | ‘H’ | ‘I’ | ‘J’ | ‘K’ | ‘L’ | ‘M’ | ‘N’ | ‘O’ | ‘P’ | ‘Q’ | ‘R’ | ‘S’ | ‘T’ | ‘U’ | ‘V’ | ‘W’ | ‘X’ | ‘Y’ | ‘Z’ ;

# Meaning

|  |  |
| --- | --- |
| \*\* | Power operation |
| ! | Factorial operation – only valid for integers |
| result(*i*) | The value of previous results. result(1) is the first result, result(2) is the second result, etc. |
| log | log10 |
| ln | loge |
| lb | log2 |