Add new double

\_\_\_\_\_\_

// MiniLan.lex

import java.lang.System;

import java\_cup.runtime.Symbol;

%%

%{

// This variable determines if the scanner prints

// messages on the screen (true) or if it only returns

// the token to the parser (false). You can change

// this value depending on whether you are interested

// in knowing what the scanner does or not.

boolean output = true;

%}

%char

%public

%cup

%full

%type java\_cup.runtime.Symbol

DIGIT=([0-9])

INTEGER=({DIGIT}+)

LETTER=([a-zA-Z])

BLANK=(" "|\t)

REAL=(({DIGIT}+"."{DIGIT}\*)|({DIGIT}\*"."{DIGIT}+))

COMMENT=("//"({LETTER}|{DIGIT}|{BLANK})\*)

IDENT=({LETTER}(({LETTER}|{DIGIT}|"\_")\*({LETTER}|{DIGIT}))?)

%%

begin {if (output) System.out.println("SCANNER:: BEGIN");

return new Symbol(sym.BEGIN);}

end {if (output) System.out.println("SCANNER:: END");

return new Symbol(sym.END);}

print {if (output) System.out.println("SCANNER:: PRINT");

return new Symbol(sym.PRINT);}

; {if (output) System.out.println("SCANNER:: EOS");

return new Symbol(sym.EOS);}

"(" {if (output) System.out.println("SCANNER:: LP");

return new Symbol(sym.LP);}

")" {if (output) System.out.println("SCANNER:: RP");

return new Symbol(sym.RP);}

"+" {if (output) System.out.println("SCANNER:: PLUS");

return new Symbol(sym.PLUS);}

- {if (output) System.out.println("SCANNER:: MINUS");

return new Symbol(sym.MINUS);}

\\* {if (output) System.out.println("SCANNER:: MULT");

return new Symbol(sym.MULT);}

"/" {if (output) System.out.println("SCANNER:: DIV");

return new Symbol(sym.DIV);}

{INTEGER} {if (output) System.out.println("SCANNER:: NUMBER <"+yytext()+">");

return new Symbol(sym.NUMBER, new Double(yytex());}

{REAL} {if (output) System.out.println("SCANNER:: NUMBER <"+yytext()+">");

return new Symbol(sym.NUMBER, new Double(yytex());}

{IDENT} {if (output) System.out.println("SCANNER:: IDENT <"+yytext()+">");

return new Symbol(sym.IDENT);}

{COMMENT} {if (output) System.out.println("SCANNER:: COMMENT <"+yytext()+">");}

(" "|\t|\n|\r)+ {}

. {if (output) System.out.println("SCANNER:: Unmatched input "+ yytext());}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1.2.1

// MiniLan.lex

import java.lang.System;

import java\_cup.runtime.Symbol;

%%

%{

// This variable determines if the scanner prints

// messages on the screen (true) or if it only returns

// the token to the parser (false). You can change

// this value depending on whether you are interested

// in knowing what the scanner does or not.

boolean output = true;

%}

%char

%public

%cup

%full

%type java\_cup.runtime.Symbol

DIGIT=([0-9])

INTEGER=({DIGIT}+)

LETTER=([a-zA-Z])

BLANK=(" "|\t)

REAL=(({DIGIT}+"."{DIGIT}\*)|({DIGIT}\*"."{DIGIT}+))

COMMENT=("//"({LETTER}|{DIGIT}|{BLANK})\*)

IDENT=({LETTER}(({LETTER}|{DIGIT}|"\_")\*({LETTER}|{DIGIT}))?)

%%

begin {if (output) System.out.println("SCANNER:: BEGIN");

return new Symbol(sym.BEGIN);}

end {if (output) System.out.println("SCANNER:: END");

return new Symbol(sym.END);}

print {if (output) System.out.println("SCANNER:: PRINT");

return new Symbol(sym.PRINT);}

; {if (output) System.out.println("SCANNER:: EOS");

return new Symbol(sym.EOS);}

"(" {if (output) System.out.println("SCANNER:: LP");

return new Symbol(sym.LP);}

")" {if (output) System.out.println("SCANNER:: RP");

return new Symbol(sym.RP);}

"+" {if (output) System.out.println("SCANNER:: PLUS");

return new Symbol(sym.PLUS);}

- {if (output) System.out.println("SCANNER:: MINUS");

return new Symbol(sym.MINUS);}

\\* {if (output) System.out.println("SCANNER:: MULT");

return new Symbol(sym.MULT);}

"/" {if (output) System.out.println("SCANNER:: DIV");

return new Symbol(sym.DIV);}

{INTEGER} {if (output) System.out.println("SCANNER:: NUMBER <"+yytext()+">");

return new Symbol(sym.NUMBER, new Double(yytext()));}

{REAL} {if (output) System.out.println("SCANNER:: NUMBER <"+yytext()+">");

return new Symbol(sym.NUMBER, new Double(yytext()));}

{IDENT} {if (output) System.out.println("SCANNER:: IDENT <"+yytext()+">");

return new Symbol(sym.IDENT);}

{COMMENT} {if (output) System.out.println("SCANNER:: COMMENT <"+yytext()+">");}

(" "|\t|\n|\r)+ {}

. {if (output) System.out.println("SCANNER:: Unmatched input "+ yytext());}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import java.io.FileReader;

import java\_cup.runtime.\*;

parser code{:

public static void main(String args[]) throws Exception {

new parser(new Yylex(new FileReader(args[0]))).parse();

} // End Main

:}

terminal BEGIN, END, PRINT;

terminal PLUS, MINUS, MULT, DIV;

terminal EOS, LP, RP;

terminal Double NUMBER;

terminal IDENT;

non terminal program, instructions, instruction, sentence, printSentence;

non terminal Double arithExpr, term, factor;

program ::= BEGIN instructions END {: System.out.println(" PARSER:: program <== BEGIN instructions END"); :}

;

instructions ::= instructions instruction {: System.out.println(" PARSER:: instructions <== instructions instruction");:}

| instruction {: System.out.println(" PARSER:: instructions <== instruction");:}

;

instruction ::= sentence EOS {: System.out.println(" PARSER:: instruction <== sentence EOS");:}

;

sentence ::= printSentence {: System.out.println(" PARSER:: sentence <== printSentence ");:}

;

printSentence ::= PRINT LP arithExpr:a RP

{:

RESULT=a;

System.out.println(" PARSER:: printSentence<"+RESULT+"> <== PRINT LP arithExpr<"+a+"> RP");

:}

;

arithExpr ::= arithExpr:a PLUS term:t

{:

RESULT=a+t;

System.out.println(" PARSER:: arithExpr<"+RESULT+"> <== arithExpr<"+a+"> PLUS term<"+t+">");

:}

| arithExpr:a MINUS term:t

{:

RESULT=a-t;

System.out.println(" PARSER:: arithExpr<"+RESULT+"> <== arithExpr<"+a+"> MINUS term<"+t+">");

:}

| term:t

{:

RESULT=t;

System.out.println(" PARSER:: arithExpr<"+RESULT+"> <== term<"+t+">");

:}

;

term ::= term:t MULT factor:f

{:

RESULT=t\*f;

System.out.println(" PARSER:: term<"+RESULT+"> <== term<"+t+"> MULT factor<"+f+">");

:}

| term:t DIV factor:f

{:

RESULT=t/f;

System.out.println(" PARSER:: term<"+RESULT+"> <== term<"+t+"> DIV factor<"+f+">");

:}

| factor:f

{:

RESULT=f;

System.out.println(" PARSER:: term<"+RESULT+"> <== factor<"+f+">");

:}

;

factor ::= NUMBER:v

{:

RESULT=v;

System.out.println(" PARSER:: factor<"+RESULT+"> <== NUMBER<"+v+">");

:}

| LP arithExpr:a RP

{:

RESULT=a;

System.out.println(" PARSER:: factor<"+RESULT+"> <== LP arithExpr<"+a+"> RP ");

:}

| PLUS factor:f

{:

RESULT=f;

System.out.println(" PARSER:: factor<"+RESULT+"> <== PLUS factor<"+f+">");

:}

| MINUS factor:f

{:

RESULT=-f;

System.out.println(" PARSER:: factor<"+RESULT+"> <== MINUS factor<"+f+"> ");

:}

;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

begin

print( -(6 - 3) / 2.0 );

print(2 + 4\*5 - 6/3\*2);

print(2 + 4\*(5 - 6.0)/3\*2);

end

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

C:\Users\UO294067\Desktop\AMD Paula\U.2\ParserMiniLan (3)\ParserMiniLan>generateParser.bat

C:\Users\UO294067\Desktop\AMD Paula\U.2\ParserMiniLan (3)\ParserMiniLan>java -jar jflex-full-1.7.0.jar MiniLan.lex

Reading "MiniLan.lex"

Constructing NFA : 113 states in NFA

Converting NFA to DFA :

........................................

42 states before minimization, 29 states in minimized DFA

Old file "Yylex.java" saved as "Yylex.java~"

Writing code to "Yylex.java"

C:\Users\UO294067\Desktop\AMD Paula\U.2\ParserMiniLan (3)\ParserMiniLan>java -jar java-cup-11b.jar MiniLan.cup

Warning : Terminal "IDENT" was declared but never used

------- CUP v0.11b 20160615 (GIT 4ac7450) Parser Generation Summary -------

0 errors and 1 warning

14 terminals, 8 non-terminals, and 17 productions declared,

producing 33 unique parse states.

1 terminal declared but not used.

0 non-terminal declared but not used.

0 productions never reduced.

0 conflicts detected (0 expected).

Code written to "parser.java", and "sym.java".

---------------------------------------------------- (CUP v0.11b 20160615 (GIT 4ac7450))

C:\Users\UO294067\Desktop\AMD Paula\U.2\ParserMiniLan (3)\ParserMiniLan>javac -cp "java-cup-11b-runtime;./\*" \*.java

Yylex.java:592: warning: [removal] Double(String) in Double has been deprecated and marked for removal

return new Symbol(sym.NUMBER, new Double(yytext()));

^

1 warning

C:\Users\UO294067\Desktop\AMD Paula\U.2\ParserMiniLan (3)\ParserMiniLan>runParser.bat

C:\Users\UO294067\Desktop\AMD Paula\U.2\ParserMiniLan (3)\ParserMiniLan>java -classpath ".;.\java-cup-11b-runtime.jar;.\java-cup-11b.jar;.\jflex-full-1.7.0.jar" parser test

SCANNER:: BEGIN

SCANNER:: PRINT

SCANNER:: LP

SCANNER:: MINUS

SCANNER:: LP

SCANNER:: NUMBER <6>

SCANNER:: MINUS

PARSER:: factor<6.0> <== NUMBER<6.0>

PARSER:: term<6.0> <== factor<6.0>

PARSER:: arithExpr<6.0> <== term<6.0>

SCANNER:: NUMBER <3>

SCANNER:: RP

PARSER:: factor<3.0> <== NUMBER<3.0>

PARSER:: term<3.0> <== factor<3.0>

PARSER:: arithExpr<3.0> <== arithExpr<6.0> MINUS term<3.0>

SCANNER:: DIV

PARSER:: factor<3.0> <== LP arithExpr<3.0> RP

PARSER:: factor<-3.0> <== MINUS factor<3.0>

PARSER:: term<-3.0> <== factor<-3.0>

SCANNER:: NUMBER <2.0>

SCANNER:: RP

PARSER:: factor<2.0> <== NUMBER<2.0>

PARSER:: term<-1.5> <== term<-3.0> DIV factor<2.0>

PARSER:: arithExpr<-1.5> <== term<-1.5>

SCANNER:: EOS

PARSER:: printSentence<-1.5> <== PRINT LP arithExpr<-1.5> RP

PARSER:: sentence <== printSentence

SCANNER:: PRINT

PARSER:: instruction <== sentence EOS

PARSER:: instructions <== instruction

SCANNER:: LP

SCANNER:: NUMBER <2>

SCANNER:: PLUS

PARSER:: factor<2.0> <== NUMBER<2.0>

PARSER:: term<2.0> <== factor<2.0>

PARSER:: arithExpr<2.0> <== term<2.0>

SCANNER:: NUMBER <4>

SCANNER:: MULT

PARSER:: factor<4.0> <== NUMBER<4.0>

PARSER:: term<4.0> <== factor<4.0>

SCANNER:: NUMBER <5>

SCANNER:: MINUS

PARSER:: factor<5.0> <== NUMBER<5.0>

PARSER:: term<20.0> <== term<4.0> MULT factor<5.0>

PARSER:: arithExpr<22.0> <== arithExpr<2.0> PLUS term<20.0>

SCANNER:: NUMBER <6>

SCANNER:: DIV

PARSER:: factor<6.0> <== NUMBER<6.0>

PARSER:: term<6.0> <== factor<6.0>

SCANNER:: NUMBER <3>

SCANNER:: MULT

PARSER:: factor<3.0> <== NUMBER<3.0>

PARSER:: term<2.0> <== term<6.0> DIV factor<3.0>

SCANNER:: NUMBER <2>

SCANNER:: RP

PARSER:: factor<2.0> <== NUMBER<2.0>

PARSER:: term<4.0> <== term<2.0> MULT factor<2.0>

PARSER:: arithExpr<18.0> <== arithExpr<22.0> MINUS term<4.0>

SCANNER:: EOS

PARSER:: printSentence<18.0> <== PRINT LP arithExpr<18.0> RP

PARSER:: sentence <== printSentence

SCANNER:: PRINT

PARSER:: instruction <== sentence EOS

PARSER:: instructions <== instructions instruction

SCANNER:: LP

SCANNER:: NUMBER <2>

SCANNER:: PLUS

PARSER:: factor<2.0> <== NUMBER<2.0>

PARSER:: term<2.0> <== factor<2.0>

PARSER:: arithExpr<2.0> <== term<2.0>

SCANNER:: NUMBER <4>

SCANNER:: MULT

PARSER:: factor<4.0> <== NUMBER<4.0>

PARSER:: term<4.0> <== factor<4.0>

SCANNER:: LP

SCANNER:: NUMBER <5>

SCANNER:: MINUS

PARSER:: factor<5.0> <== NUMBER<5.0>

PARSER:: term<5.0> <== factor<5.0>

PARSER:: arithExpr<5.0> <== term<5.0>

SCANNER:: NUMBER <6.0>

SCANNER:: RP

PARSER:: factor<6.0> <== NUMBER<6.0>

PARSER:: term<6.0> <== factor<6.0>

PARSER:: arithExpr<-1.0> <== arithExpr<5.0> MINUS term<6.0>

SCANNER:: DIV

PARSER:: factor<-1.0> <== LP arithExpr<-1.0> RP

PARSER:: term<-4.0> <== term<4.0> MULT factor<-1.0>

SCANNER:: NUMBER <3>

SCANNER:: MULT

PARSER:: factor<3.0> <== NUMBER<3.0>

PARSER:: term<-1.3333333333333333> <== term<-4.0> DIV factor<3.0>

SCANNER:: NUMBER <2>

SCANNER:: RP

PARSER:: factor<2.0> <== NUMBER<2.0>

PARSER:: term<-2.6666666666666665> <== term<-1.3333333333333333> MULT factor<2.0>

PARSER:: arithExpr<-0.6666666666666665> <== arithExpr<2.0> PLUS term<-2.6666666666666665>

SCANNER:: EOS

PARSER:: printSentence<-0.6666666666666665> <== PRINT LP arithExpr<-0.6666666666666665> RP

PARSER:: sentence <== printSentence

SCANNER:: END

PARSER:: instruction <== sentence EOS

PARSER:: instructions <== instructions instruction

PARSER:: program <== BEGIN instructions END

C:\Users\UO294067\Desktop\AMD Paula\U.2\ParserMiniLan (3)\ParserMiniLan>