COMPILADORES
REPORTE #3
HOC3 -- Calculadora científica

1 de Noviembre del 2018

PROFESOR: TECLA PARRA ROBERTO

ALUMNO: SALDAÑA AGUILAR ANDRÉS

GRUPO: 3CM8

Desarrollo

HOC 3 es una calculadora científica con variables, que puede realizar acciones de funciones como senos, cosenos y potencias, así como también puede contener algunas constantes predeterminadas, para poder realizar algunas de estas acciones necesitamos poder guardar de manera temporal estas variables en una estructura de datos accesible para el programa, que en este caso será una tabla de símbolos.

Esta práctica está hecha en lenguaje Java, por lo tanto la tabla de símbolos, es un **ArrayList** de Simbolos, los simbolos son un objeto de la clase Symbol que tiene como atributos nombre (‘para guardar el nombre de una variable’), tipo de dato y su valor.

En la gramática se agregaron las reglas de producciones necesarias para la creación de variables y predeterminados, así como las funciones:

Install: instala en la tabla de símbolos, predeterminados o variables)
LookupTable: busca y devuelve de la tabla de símbolos una variable o predeterminado.

A continuación, la gramática para la calculadora de complejos,

```
%right '='
%left '-' '+'
%left '*' '/'

%%
list: /* nada */
     | list '\n'
    | list asgn '\n' {}
     | list exp '\n' {Complejo res = (Complejo) $2.obj;imprimeComplejo(res);}
     ;
asgn: VAR '=' exp   { Cadena c = (Cadena) $1.obj;
                      Symbol s = lookUpTable(c.getCadena());
                      if (s == null)
                        install(c.getCadena(), (Complejo) $3.obj, (short) 1);
                      else
                        update(s, (Complejo) $3.obj);
                    }
    ;
```

```
exp:  CNUMBER                  {Complejo c = (Complejo) $1.obj;}
   | VAR                     { Cadena c      = (Cadena) $1.obj;
                             Symbol s      = lookUpTable(c.getCadena());
                             if (s != null) {
                               Complejo data = (Complejo) s.getData();
                               $$ = new ParserVal(data);
                             } else {
                               yyerror("Variable no declarada");
                               System.exit(0);
                             }
                            }
   | asgn
   | BLTIN '(' exp ',' DIG ')'  { Cadena c1       = (Cadena) $1.obj;
                             Symbol s        = lookUpTable(c1.getCadena());
                             String fName    = s.getName();
                             Complejo res    = new Complejo(0, 0);
                             Cadena powN     = (Cadena) $5.obj;

                             if (fName.compareTo("pow") == 0) {
                               res = f.pow((Complejo) $3.obj, powN);
                             }
                             $$ = new ParserVal(res);

                            }
   | BLTIN '(' exp ')'         { Cadena c        = (Cadena) $1.obj;
                             Symbol s        = lookUpTable(c.getCadena());
                             String fName    = s.getName();
                             Complejo res    = new Complejo(0, 0);

                             if (fName.compareTo("exp") == 0) {
                               res = f.exp((Complejo) $3.obj);
                             } else if (fName.compareTo("sin") == 0) {
                               res = f.sinus((Complejo) $3.obj);
                             } else if (fName.compareTo("cos") == 0) {
                               res = f.cosine((Complejo) $3.obj);
```

```
                        }
                        $$ = new ParserVal(res);
                      }
   | exp '+' exp          { Complejo c1   = (Complejo) $1.obj; Complejo c2 = (Complejo)
$3.obj;
                        Complejo res  = sumaComplejos(c1, c2);
                        $$          = new ParserVal(res);
                      }
   | exp '-' exp          { Complejo c1   = (Complejo) $1.obj; Complejo c2 = (Complejo)
$3.obj;
                        Complejo res  = restaComplejos(c1, c2);
                        $$          = new ParserVal(res);
                      }
   | exp '*' exp          { Complejo c1   = (Complejo) $1.obj; Complejo c2 = (Complejo)
$3.obj;
                        Complejo res  = multiplicaComplejos(c1, c2);
                        $$          = new ParserVal(res);
                      }
   | exp '/' exp          { Complejo c1   = (Complejo) $1.obj; Complejo c2 = (Complejo)
$3.obj;
                        Complejo res  = divideComplejos(c1, c2);
                        $$          = new ParserVal(res);
                      }
   | '(' exp ')'        {Complejo c = (Complejo) $2.obj; $$ = new ParserVal(c); }
   ;
%%
```

En la léxica agregamos la parte de una variable, que en este caso la escribí con la posibilidad de que el nombre de una variable sea una cadena seguida de un dígito.

```
nl          = \n | \r | \r\n
op          = [-+*()/=]
ws          = [ \t]+
digits      = [0-9]
```

```
number      = (0|[1-9]+{digits}*)\.?{digits}*
im          = [i]
complexnum  = {ws}*[-]*{ws}*{number}{ws}*[+|-]{ws}*{number}{ws}*{im}{ws}*
var         = [a-zA-Z][a-zA-Z]* | [a-zA-Z][a-zA-Z]*{digits}
built       = sin | exp | cos | pow
comma       = ,
%%

{built} {
 yyparser.yylval = new ParserVal(new Cadena(yytext()));
  return Parser.BLTIN;
}
{var} {
 yyparser.yylval = new ParserVal(new Cadena(yytext()));
 return Parser.VAR;
 }
{complexnum} { Auxiliar a = new Auxiliar();
              int [] cmp = a.obtenerComponentes(yytext());
              yyparser.yylval = new ParserVal( new Complejo(cmp[0], cmp[1]) );
              return Parser.CNUMBER;
            }
{op} | {comma} {return (int) yycharat(0); }
{digits} { yyparser.yylval = new ParserVal(new Cadena(yytext())); return Parser.DIG;}
{nl}   { return (int) yycharat(0); }
{ws}   {}
.      {}
/* error fallback */
[^]    { System.err.println("Error: unexpected character '"+yytext()+"'"); return -1; }
```

Resultados

Para probar el funcionamiento del programa, procedi a crear dos variables, num1 = 1+i y num2 = 1-i , con el propósito de realizar diferentes operaciones con ellos y conocer si el valor resultante es el correcto:

```
andres@andres-VirtualBox:/media/sf_Compiladores/Complejos_Java_ConTabladeSimbolo
s_Builtins/Complejos_Java$ java Parser
Calculadora Números Complejos Científica
Ingresa una expresion:
num1 = 1+1i
num2 = 1-1i
num1-num2
0.0 + 2.0 i
num1 + num2
2.0 0.0 i
num1 * num2
2.0 0.0 i
num1 / num2
0.0 + 1.0 i
cos(num1)
0.83373004 -0.9888977 i
pow(num1,9)
16.0 + 16.0 i
```

Valor del coseno probado en wolfram:

Decimal approximation:

$$0.833730025131149048883885394335094479809874785209629312 27\ldots - 0.988897705762865096382129540892686188642149695033147607 53\ldots\, i$$

Property:

$$\cos(1+i) \text{ is a transcendental number}$$

Valor de 1+i elevado a la novena potencia en wolfram:

Input:

$$(1+i)^9$$

Result:

$$16+16\,i$$

Conclusión

El desarrollo de esta práctica tuvo como reto la creación de la tabla de símbolos y los métodos pertinentes para poder instalar en él nuevos símbolos o buscar y conseguir la existencia de estos, una vez solucionado eso lo demás permanece igual, ya que el profesor nos apoyó con la creación de la gramática.