COMPILADORES
REPORTE #4
HOC4 -- Calculadora Cientifica

1 de Noviembre del 2018

PROFESOR: TECLA PARRA ROBERTO

ALUMNO: SALDAÑA AGUILAR ANDRÉS

GRUPO: 3CM8

En HOC4 se tiene la misma funcionalidad que en HOC3, ya que es una calculadora científica con variables, con la diferencia que este no hace uso de una tabla de símbolos para el almacenamiento de variables y predeterminados, sino de una máquina virtual.

Una máquina virtual de proceso, a veces llamada **"máquina virtual de aplicación"**, se ejecuta como un proceso normal dentro de un sistema operativo sirviendo de enlace entre un lenguaje de programación y el sistema operativo, realizando una interpretación u otra técnica de enlace entre fuente y código máquina.

Una característica esencial de las máquinas virtuales es que los procesos que ejecutan están limitados por los recursos y abstracciones proporcionados por ellas. Estos procesos no pueden escaparse de esta "computadora virtual".

En la gramática, las reglas de producción no sufrieron muchas modificaciones, sino sus acciones gramaticales, donde ocupamos ahora la máquina para la creacion y ejecucion de codigo,ahora explicare un poco el funcionamiento de la máquina virtual:

La máquina virtual tiene como recursos una pila del programa, la cual se ocupa en la ejecución del programa, para realizar operaciones aritméticas.
También tiene una RAM, tiene la función de guardar en él, las instrucciones a ejecutar, esta pila guarda solamente cadenas, con la cual después podemos posteriormente usar para invocar los métodos existentes y conseguir los parámetros necesarios
Para poder llevar un control en la pila, tenemos un contador del programa (PC) que indica la dirección de la siguiente instrucción a ejecutar y progp que nos dice la siguiente dirección donde podremos generar código.

Conociendo esto, podemos ahora entender las acciones gramaticales, por ejemplo en la impresión de un número complejo, ingresamos primeramente la instrucción "imprimeComplejo", que es un método encargado de imprimir el número que se ingrese, seguido de “STOP”, que es vital en cada fin de operación, ya que esta hace que el programa termine su ejecución, todo esto haciendo uso de “code”, que es el encargado de guardar en la RAM y actualizar el contador del programa, finalmente ejecutamos las instrucciones en la RAM con “execute”.

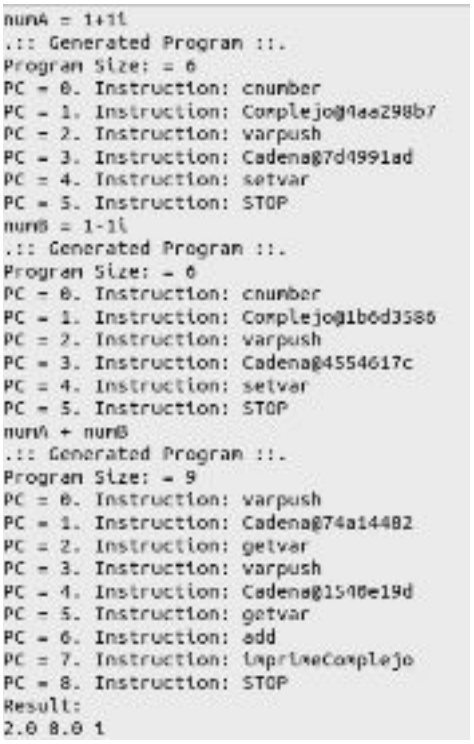
```
%right '='
%left '-' '+'
%left '*' '/'
%left '^'
%%
list: /* nada */
     | list '\n'
    | list asgn '\n'{ maq.code("STOP");
                      maq.execute(flag);
                      flag = true;
```

```
                }
    | list exp '\n' { maq.code("imprimeComplejo");
                     maq.code("STOP");
                     maq.execute(flag);
                     flag = true;
                    }
    ;
asgn: VAR '=' exp   { Cadena c = (Cadena)$1.obj;
                      maq.code("varpush");
                      maq.code(c);
                      maq.code("setvar");
                     }
    ;
exp: CNUMBER  { Complejo c = (Complejo)$1.obj;
              maq.code("cnumber"); //guardamos cadena en RAM
              maq.code(c); //guardamos variable en RAM
            }
  | VAR      { Cadena c = (Cadena)$1.obj;
              maq.code("varpush");
              maq.code(c);
              maq.code("getvar"); //conseguimos el valor de la variable para imprimirlo
            }
  | asgn
  | BLTIN '(' exp ')'{ Cadena c = (Cadena)$1.obj;
                       maq.code("stringpush");
                       maq.code(c);
                       maq.code("bltin");
                      }
  | exp '+' exp {maq.code("add");}
  | exp '-' exp {maq.code("sub");}
  | exp '*' exp {maq.code("mul");}
  | exp '/' exp {maq.code("div");}
  | exp '^' exp {maq.code("pow");}
  | '(' exp ')'  { $$= $2;}
  ;
```

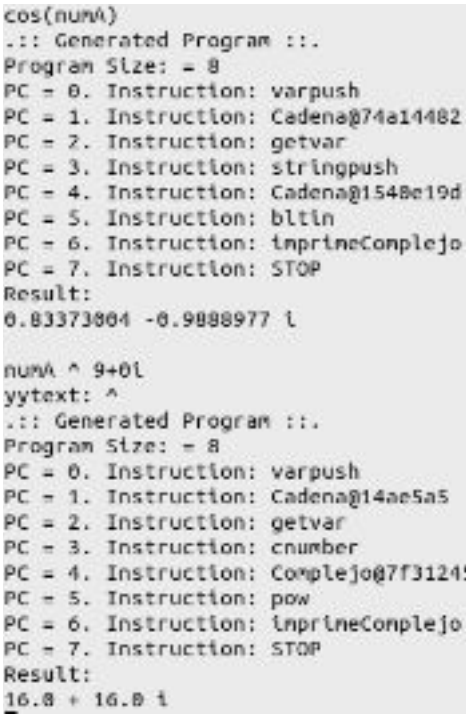
%%

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, adicionalmente imprimi el funcionamiento de la maquina para ver como ejecuta las instrucciones:

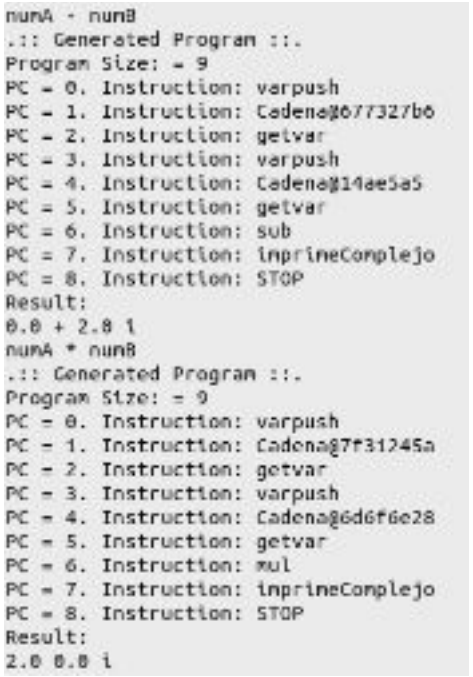


```
numA = 1+1i
.:: Generated Program ::.
Program Size: = 6
PC = 0. Instruction: cnumber
PC = 1. Instruction: Complejo@14ae298b7
PC = 2. Instruction: varpush
PC = 3. Instruction: Cadena@7d4991ad
PC = 4. Instruction: setvar
PC = 5. Instruction: STOP
numB = 1-1i
.:: Generated Program ::.
Program Size: = 6
PC = 0. Instruction: cnumber
PC = 1. Instruction: Complejo@1b6d3586
PC = 2. Instruction: varpush
PC = 3. Instruction: Cadena@4554617c
PC = 4. Instruction: setvar
PC = 5. Instruction: STOP
numA + numB
.:: Generated Program ::.
Program Size: = 9
PC = 0. Instruction: varpush
PC = 1. Instruction: Cadena@74a14482
PC = 2. Instruction: getvar
PC = 3. Instruction: varpush
PC = 4. Instruction: Cadena@1540e19d
PC = 5. Instruction: getvar
PC = 6. Instruction: add
PC = 7. Instruction: imprimeComplejo
PC = 8. Instruction: STOP
Result:
2.0 0.0 i
```

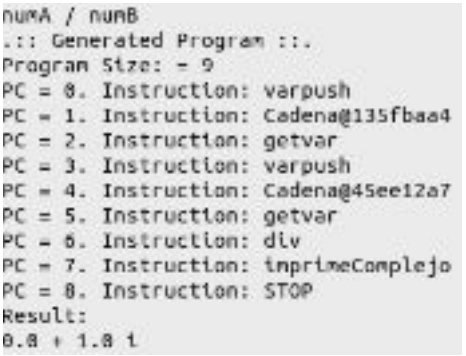


```
cos(numA)
.:: Generated Program ::.
Program Size: = 8
PC = 0. Instruction: varpush
PC = 1. Instruction: Cadena@74a14482
PC = 2. Instruction: getvar
PC = 3. Instruction: stringpush
PC = 4. Instruction: Cadena@1540e19d
PC = 5. Instruction: bltin
PC = 6. Instruction: imprimeComplejo
PC = 7. Instruction: STOP
Result:
0.83373004 -0.9888977 i

numA ^ 9+0i
yytext: ^
.:: Generated Program ::.
Program Size: = 8
PC = 0. Instruction: varpush
PC = 1. Instruction: Cadena@14ae5a5
PC = 2. Instruction: getvar
PC = 3. Instruction: cnumber
PC = 4. Instruction: Complejo@7f31245
PC = 5. Instruction: pow
PC = 6. Instruction: imprimeComplejo
PC = 7. Instruction: STOP
Result:
16.0 + 16.0 i
```

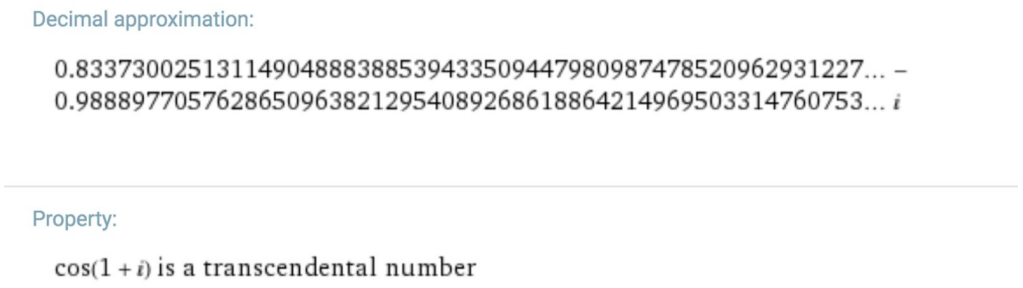


```
numA - numB
.:: Generated Program ::.
Program Size: = 9
PC = 0. Instruction: varpush
PC = 1. Instruction: Cadena@677327b6
PC = 2. Instruction: getvar
PC = 3. Instruction: varpush
PC = 4. Instruction: Cadena@14ae5a5
PC = 5. Instruction: getvar
PC = 6. Instruction: sub
PC = 7. Instruction: imprimeComplejo
PC = 8. Instruction: STOP
Result:
0.0 + 2.0 i
numA * numB
.:: Generated Program ::.
Program Size: = 9
PC = 0. Instruction: varpush
PC = 1. Instruction: Cadena@7f31245a
PC = 2. Instruction: getvar
PC = 3. Instruction: varpush
PC = 4. Instruction: Cadena@6d6f6e28
PC = 5. Instruction: getvar
PC = 6. Instruction: mul
PC = 7. Instruction: imprimeComplejo
PC = 8. Instruction: STOP
Result:
2.0 0.0 i
```

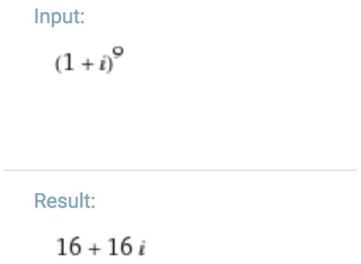


```
numA / numB
.:: Generated Program ::.
Program Size: = 9
PC = 0. Instruction: varpush
PC = 1. Instruction: Cadena@135fbaa4
PC = 2. Instruction: getvar
PC = 3. Instruction: varpush
PC = 4. Instruction: Cadena@45ee12a7
PC = 5. Instruction: getvar
PC = 6. Instruction: div
PC = 7. Instruction: imprimeComplejo
PC = 8. Instruction: STOP
Result:
0.0 + 1.0 i
```

Valor del coseno probado en wolfram:



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



Conclusión

El desarrollo de esta práctica tuvo como reto la creación de la máquina virtual y los métodos pertinentes para poder ejecutar instrucciones, la invocación de métodos y en que momento y orden se debe aumentar el contador del programa y sacar de la pila del programa, una vez solucionado eso lo demás permanece igual que en HOC3.