Instituto Tecnológico de Chilpancingo

Ingeniería en sistemas computacionales

Lenguajes y Autómatas II

Catedrático:

Alfredo de Jesús Canto Cetina

Trabajo:

"Analizador Recursivo en Ruby"

Integrantes del Equipo:

- Cynthia Daniela García González
- Abigail Mosso Martínez
- Guillermo Peña Figueroa

Índice

Contenido:

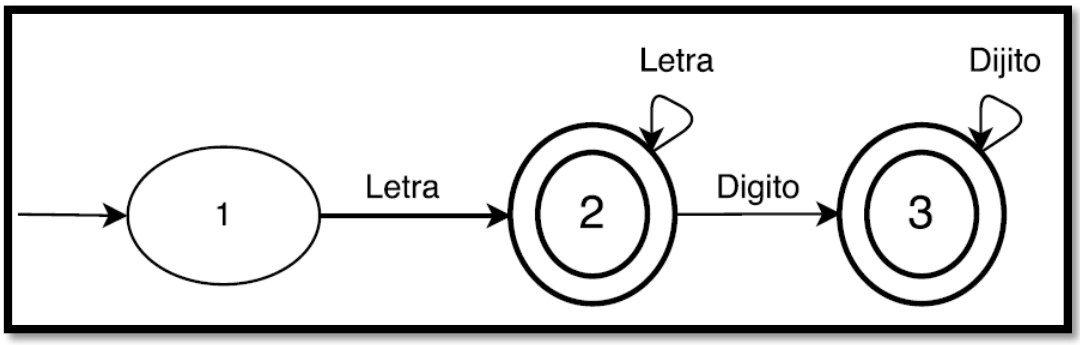
Introducción:

A lo largo del curso de verano, aprendimos a manejar Ruby. Ruby es un lenguaje de programación orientado a objetos: todos los tipos de datos son un objeto, incluidas las clases y tipos que otros lenguajes definen como primitivas, (como enteros, booleanos, y "nil"). Toda función es un método. Las variables siempre son referencias a objetos, no los objetos mismos. Ruby soporta herencia con enlace dinámico y métodos (pertenecientes y definidos por una sola instancia más que definidos por la clase). A pesar de que Ruby no soporta herencia múltiple, la clases pueden importar módulos.

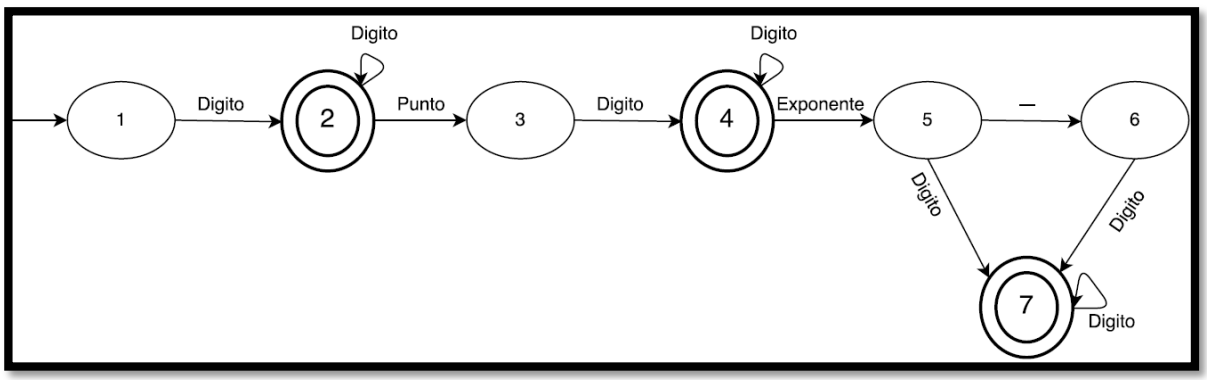
En este último programa programado en ruby, realizamos un analizador léxico recursivo, con la ayuda de los autómatas ya conocidos como: identificador y número real con exponente, así mismo de los diagramas de flujo de: Factor, termino, expresión y expresión simple. En el cual corroboramos su sintaxis correcta.

Autómatas

Autómata Identificador

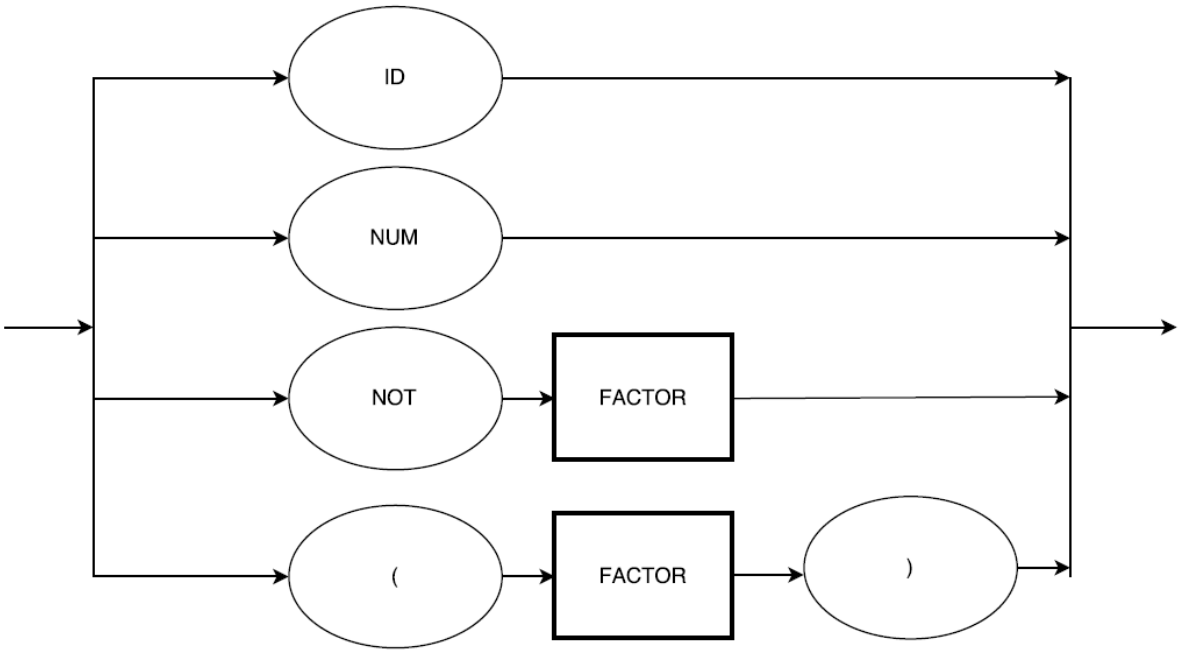


Autómata numero general (Numero, real y numero real con exponente)

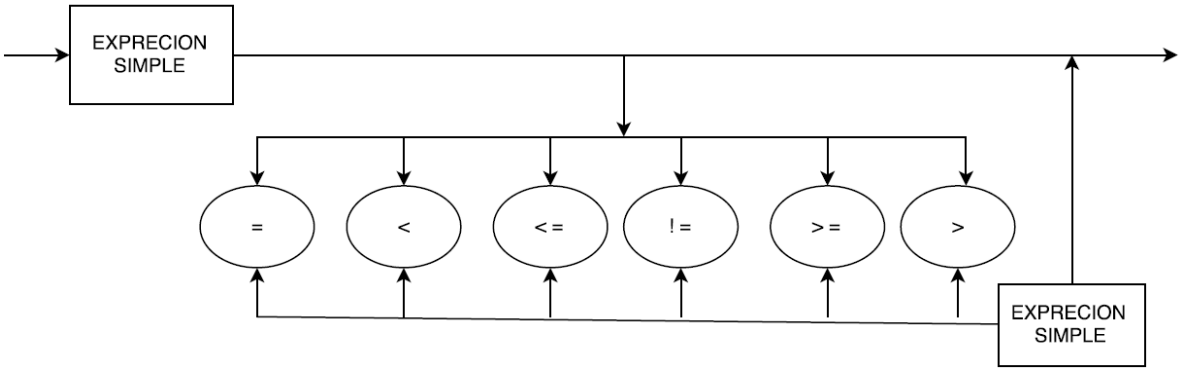


Diagramas de flujo

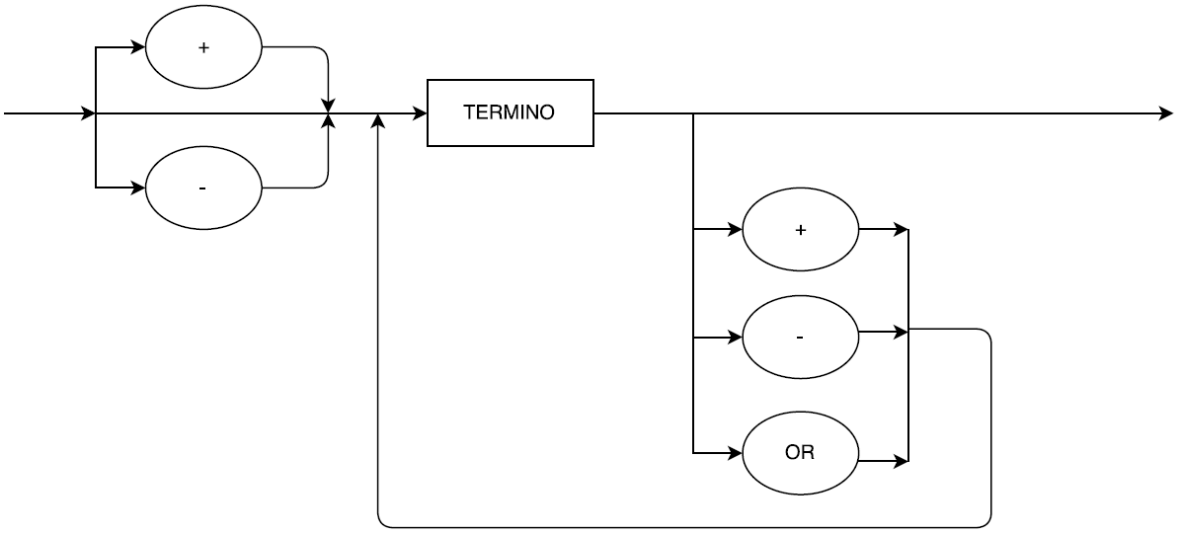
Factor



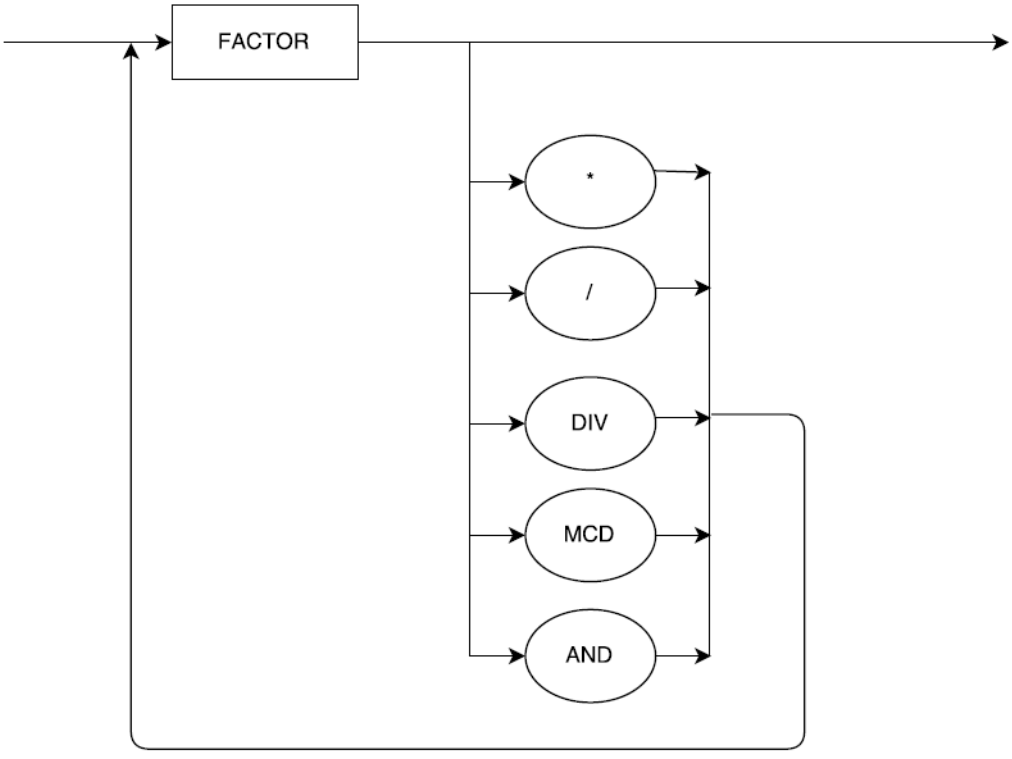
Expresión



Expresión simple



Término



Codificación en ruby:

Interfaz Gráfica:

```
#****************************************
#Instituto Tecnologico de Chilpancingo   *
#Ingeniería en sistemas computacionales  *
#Lenguajes y automatas II                *
#Elaborado por:                          *
#-Cynthia Daniela García González        *
#-Abigail Mosso Martinez                 *
#-Guillermo Peña Figueroa                *
#****************************************

#*****IMPORTACION DE CLASE *******
require_relative 'Analizador_Recursivo.rb'

#*******IMPORTACION DE LIBRERIAS ******
require'fox16'
include Fox

#********CLASE INTERFAZ QUE HEREDA LOS COMPONENTES DE FXMAINWINDOWS *******
class Interfaz < FXMainWindow
  #***CONSTRUCTOR****
  def initialize(app)
    super(app, "Interfaz Grafica", :width=>800, :height=>600)
#********COMPONENTES QUE SE UTILIZAN EN LA INTERFAZ *******
    @textArea = FXText.new(self, :opts=>LAYOUT_EXPLICIT, :x=>50, :y=>80,
                           :width=>700, :height=>400)
    @textArea.font = FXFont.new(app,"Modern, 220, BOLD,0")
    @btnAnalizar = FXButton.new(self, "Analizar", :opts=> LAYOUT_EXPLICIT,
                                :x=>350,:y=>490,:width=>150, :height=>40)
    @btnAnalizar.font = FXFont.new(app, "Mordern, 220, BOLD,0")

    @lblRespuesta = FXLabel.new(self, "Analizador Recursivo", :opts=>LAYOUT_EXPLICIT,
                                :x=>280, :y=>20, :width=>250, :height=>40)
    @lblRespuesta.font = FXFont.new(app, "Modern, 220, BOLD,0")
    @lblRespuesta.backColor = 'Brown'
```

```
#***************ANALIZADOR RECURSIVO ******************
    @btnAnalizar.connect(SEL_COMMAND)do
      inicio = AnalizadorRecursivo.new(@textArea.text.to_s,app)
      inicio.secuencia()

    end
    #*****FONDO DEL ANALIZADOR ********
    lblFondo = FXLabel.new(self,"", :opts=>LAYOUT_EXPLICIT, :x=>0, :y=>0,
                           :width=>800, :height=>600)
    lblFondo.icon = FXJPGIcon.new(app, File.open("madera.jpg","rb").read)
    lblFondo.layoutHints = LAYOUT_CENTER_X|LAYOUT_CENTER_Y

  end


  def create
    super
    show(PLACEMENT_SCREEN)
  end
end
#***CREAMOS LA INTERFAZ GRAFICA ****
app=FXApp.new
Interfaz.new(app)
app.create
app.run
```

Clase Analizador léxico

```
#*********************************************
#Tecnologico Nacional de Mexico              *
#Instituto Tecnologico de Chilpancingo       *
#Ingeniería en sistemas computacionales      *
#Lenguajes y Automatas 2                     *
#Autores:                                    *
#-Cynthia Daniela García González            *
#-Abigail Mosso Martinez                     *
#-Guillermo Peña Figueroa                    *
#*********************************************

require'fox16'
include Fox

class AnalizadorRecursivo
#******** CONSTRUCTOR DE LA CLASE ***********
  def initialize(cadena,aplicacion)

    @cadena=cadena
    @arreglo=Array.new
    @arreglo=@cadena.scan(/[\-\+\*\/\'\:\,\_\.\?\[\]\¿\[\{\}\^\|
                           \"\#\$\%\&\@\~\<\>\=\(\)\;\!]|
                           [0-9]+\.[0-9]+|[A-Za-z0-9]*/)
    @arreglo.push("FDC")
    @contador=0
    @mensaje="¡CADENA ACEPTADA! :D"
    @app=aplicacion
    @aux=Array.new

    for k in  0... @arreglo.length
      if @arreglo[k].to_s.length>0
        @aux.push(@arreglo[k])
      end
    end
    @arreglo=@aux
  end
```

```ruby
#***** Metodo para identificar un Operador Logico *****
def esOperadorLogico(cadena)
  if  /[\<\>\!]/.match(cadena)
    return true
  end
  return false
end

#***** Metodo para identificar un Numero General *****
def esNumero(entero)

  if  /^[0-9]+(\\.[0-9]+)?([eE\\^][0-9]+)?$/.match(entero)
    return true
  end
  return false
end

#********  Metodo para identificador *******
def esIdentificador(palabra)
  if /^[A-Za-z]+[0-9]*[A-Za-z]*$/.match(palabra)
    return true
  end
  return false
end

#***** Metodo para identificar una negación *****
def esNot(simbolo)
  if /\!/.match(simbolo)
    return true
  end
  return false
end
```

```ruby
#**** Metodo para identificar un operador matematico ****
def esOperadorMatematico(operador)
  if/[\-\/\+\*\%]/.match(operador)
    return true
  end
  return false
end

#*****Metodo para identificar un signo ****
def esSigno(signo)
  if /[\-\+]/.match(signo)
    return true
  end
  return false
end

#****Metodo para obtener un token ****
def getToken
  puts  @token=@arreglo[@contador]
  if @arreglo[@contador].to_s=="FDC"
  else
    @contador+=1
  end
  return  @token
end

#***** Metodo para identificar un numero real ******
def esReal (caracter)
  if caracter =~ /[0-9]+\.[0-9]+/
    #^[-]?\d*[.]?\d*$
    return true
  else
    return false
  end
end
```

```ruby
#*****METODO FACTOR*****
  def factor()
    if  esIdentificador(@token)
      getToken
    elsif esNumero(@token)
      getToken
    elsif esReal(@token)
     getToken
    elsif esNot(@token)
      getToken
      factor()

    elsif @token.to_s=="("
      @checar=true
      getToken
      expresion()
      if @token.to_s==")"
        getToken
      else
        puts"error1"
        puts  @mensaje="Error falta parentesis"

      end
    else
      puts "error2 #{@token}"
      @mensaje="Error de expresion"
    end
  end
```

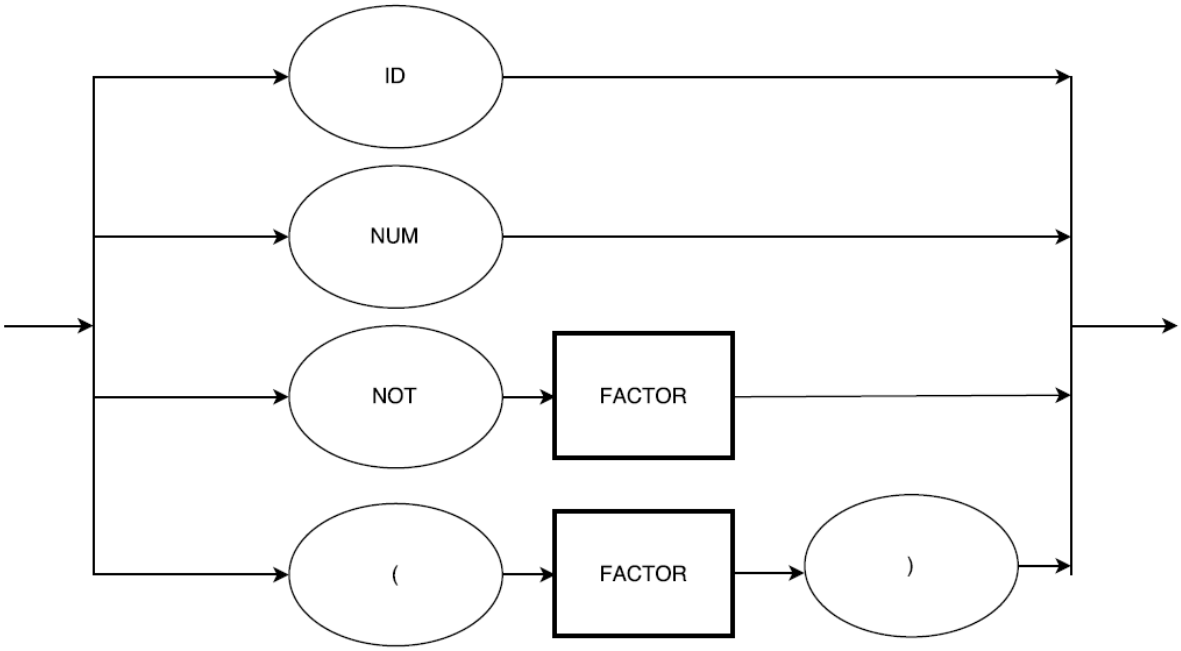
```ruby
#***** MÉTODO TERMINO *****
def termino()
  factor()
  while(esOperadorMatematico(@token))
    puts"OM #{@token}"
    getToken
    puts" #{@token}"
    factor()
  end
end

#****** MÉTODO EXPRESION SIMPLE *****
def expresionSimple()
  if esSigno(@token)
    getToken
  end
  termino()
  while(@token.to_s=="-" ||@token.to_s=="+" ||@token.to_s=="|" ||@token.to_s == "^")
    getToken
    termino()
  end
end
```

```ruby
#***** MÉTODO EXPRESIÓN *****
def expresion()
  expresionSimple()
  if esOperadorLogico(@token)
    getToken
    if @token.to_s=="="
      getToken
    end
    expresionSimple()
  end
end
```

```
#****METODO SECUENCIA ****
def secuencia()
  getToken
  loop do
    expresion()

    while(@token.to_s!=";")

      if @mensaje=="Error de expresion"
        @mensaje="Error de expresion"
        break
      elsif @mensaje=="Error falta parentesis"
        @mensaje="Error falta parentesis )"
        break
         else
        @mensaje="Error de expresion"
      end

      if @token.to_s=="FDC"
        @mensaje="falta ;"
        break
      end
      getToken
    end
    getToken
    if @token.to_s=="FDC"
      puts"segun..."
      break
    end
  end


  if @mensaje.to_s=="¡CADENA ACEPTADA! :D"

    FXMessageBox.information(@app,MBOX_OK,"------INFORMACION------",@mensaje.to_s)

  else
    FXMessageBox.error(@app,MBOX_OK,"---INFORMACION---",@mensaje.to_s)
  end
 end
end
```

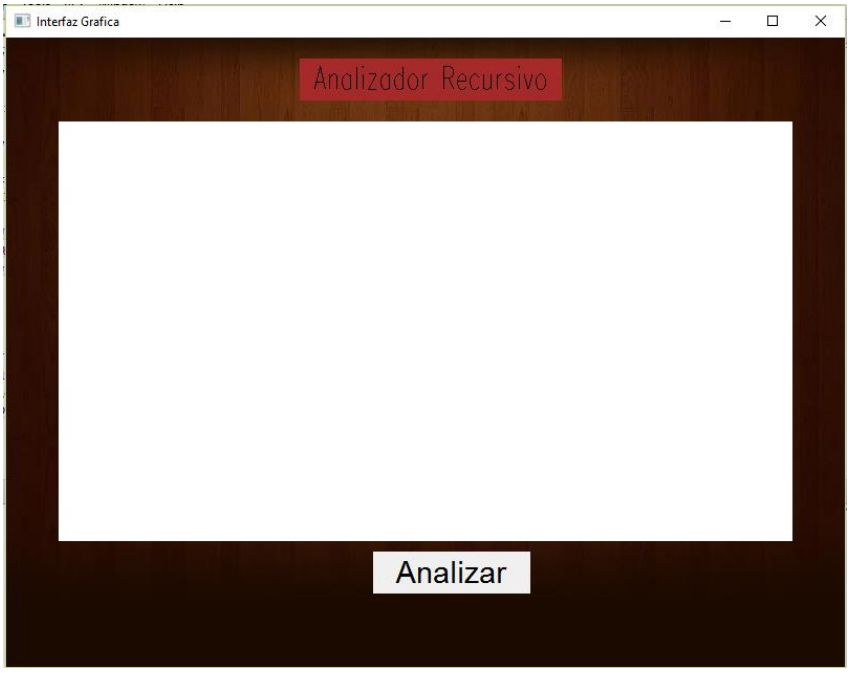
Pruebas del analizador Léxico Recursivo

1.- FACTOR



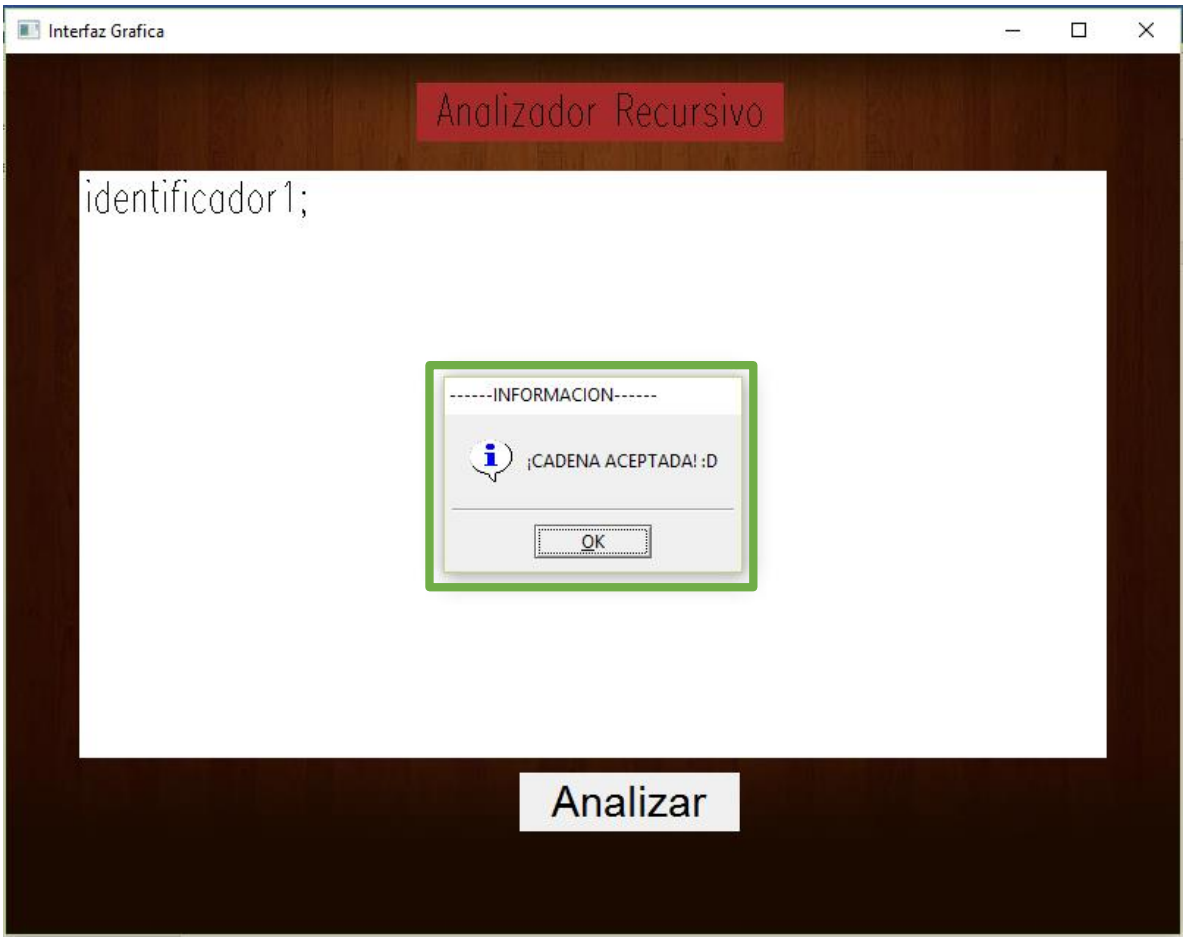
Ahora comprobaremos con el programa el diagrama de flujo del FACTOR.

La interfaz gráfica

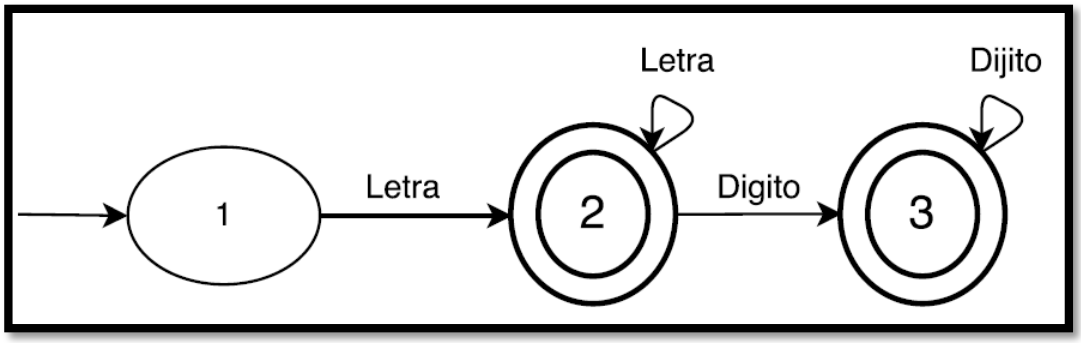


Si ingresamos un identificador:

Y damos clic en analizar:

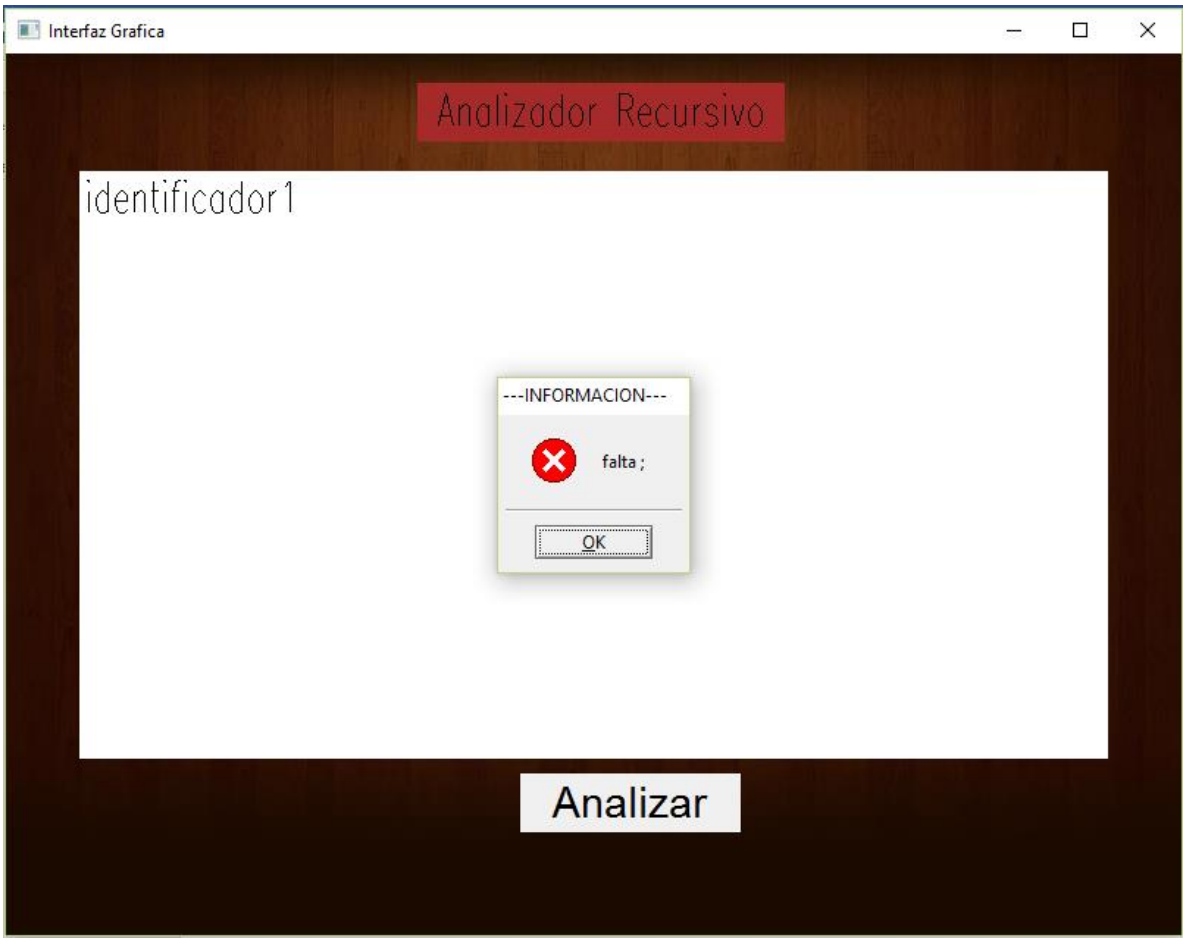


Como nos podemos dar cuenta en nuestro autómata identificador



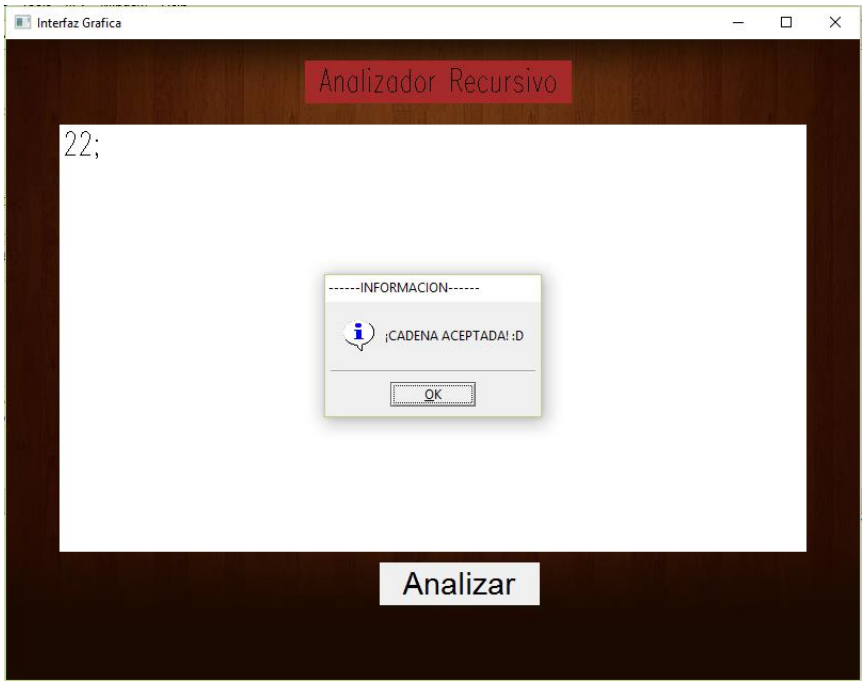
Nuestro estado inicial es 1, si es letra, nuestro estado cambiaria a estado: 2, si nuestro estado es digito cambiaria a estado: 3 y como estado 3 es final de cadena, la cadena es ACEPTADA.

Ahora para comprobar que nuestro delimitador ";" funciona lo quitaremo y analizaremos la cadena:

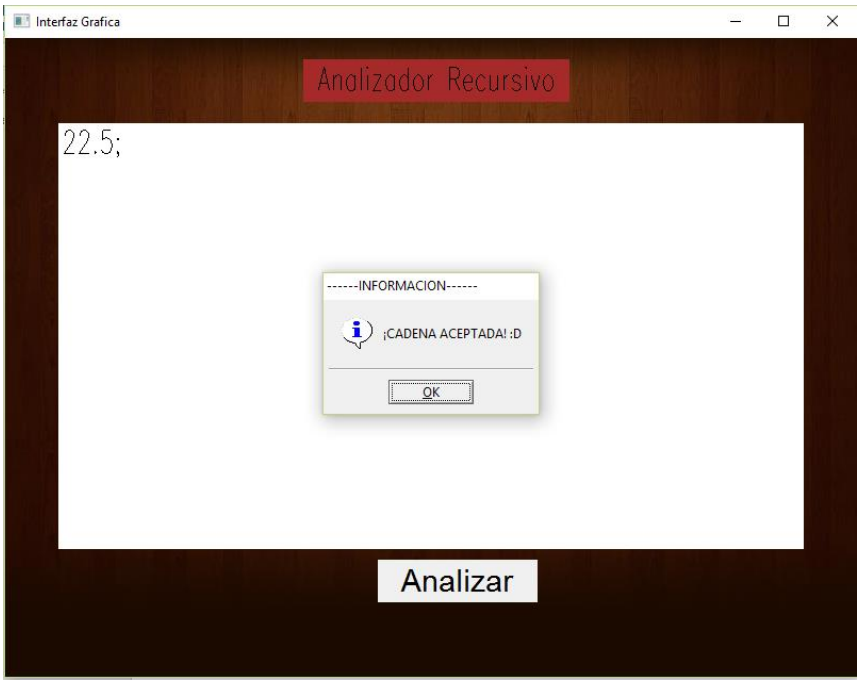


Como nos podremos dar cuenta el autómata identificador si esta correcto, pero nos indica que nos falta el delimitador ";"

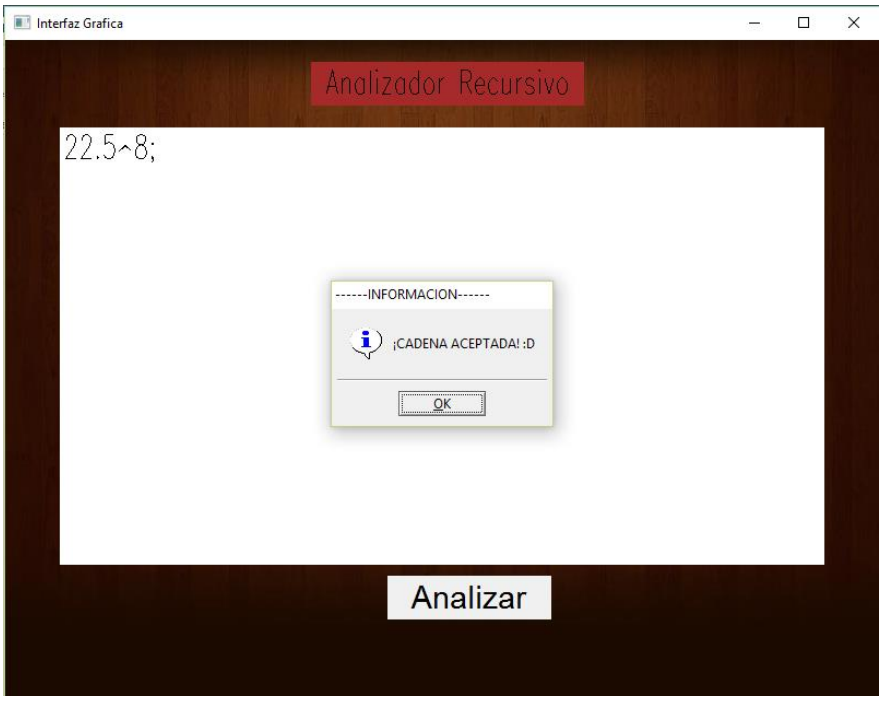
Ahora probaremos el numero



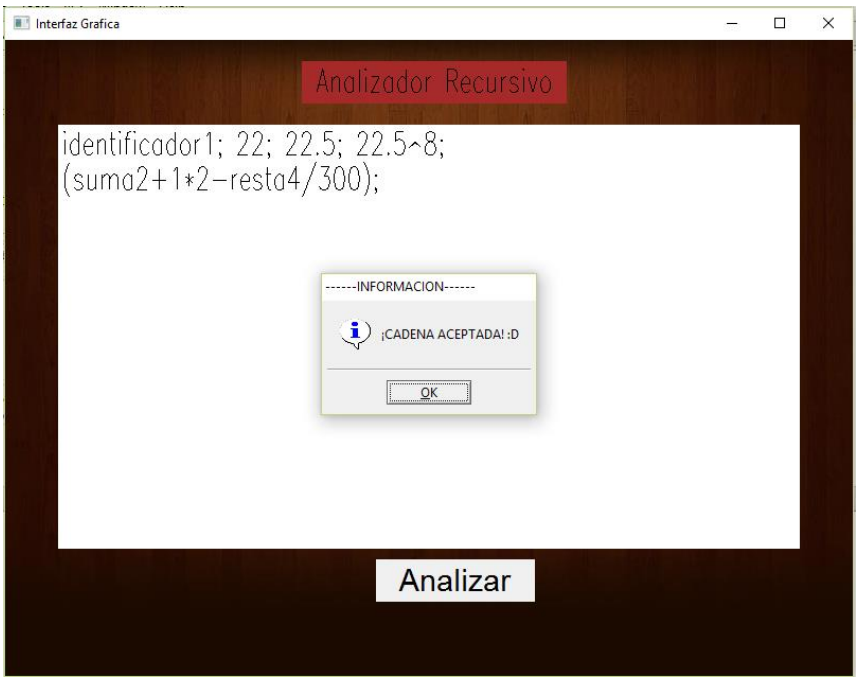
Y con número real



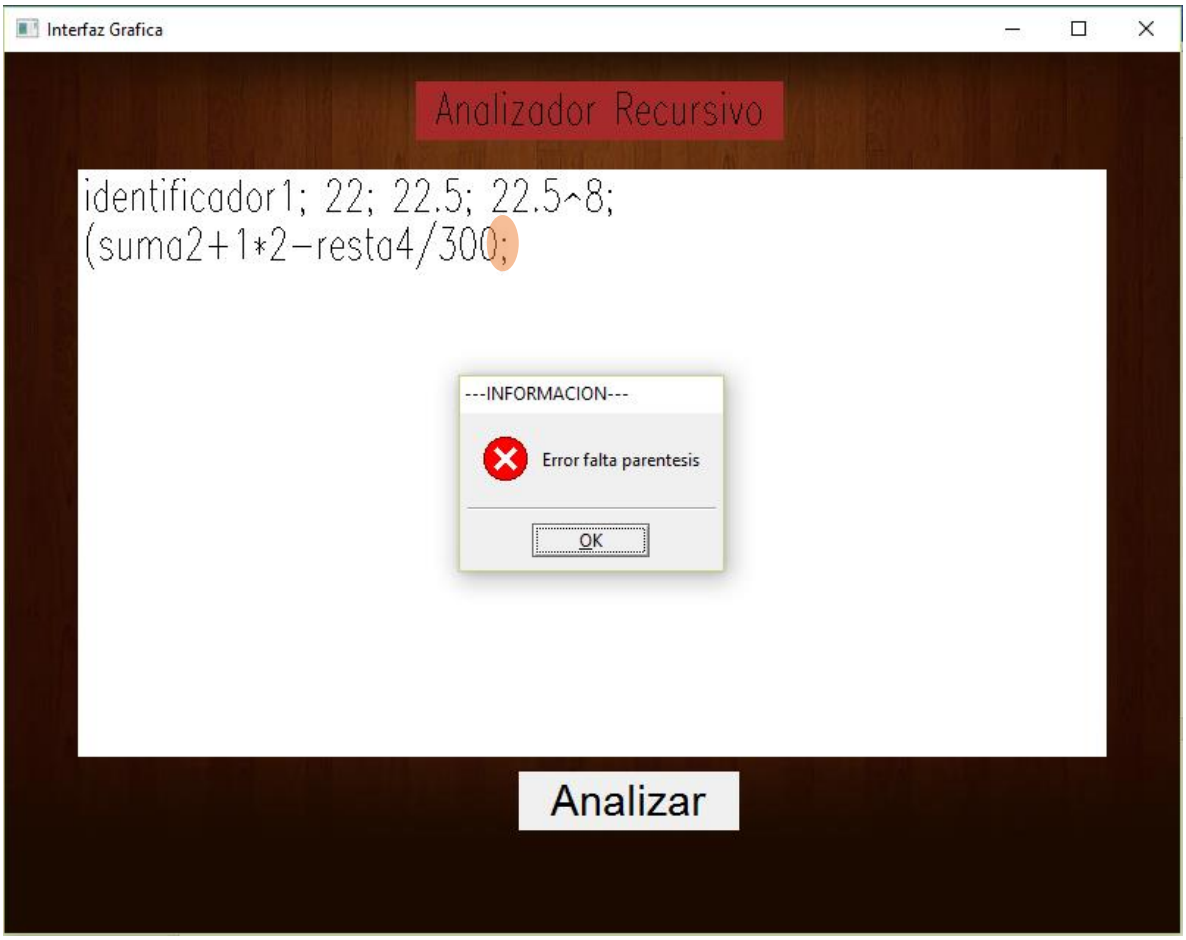
Número real con exponente



Expresion



Si le quitamos un paréntesis:



Ahora probamos todos los diagramas de flujo: