Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 – Traductores e Interpretadores
Abril – Julio 2013

# Etapa II
# Análisis Sintáctico con Árbol Sintáctico Abstracto (AST)

En esta entrega usted deberá diseñar una gramática libre de contexto para el lenguaje *RangeX* y utilizarla para implantar un reconocedor que lo reconozca. Su reconocedor debe además construir el árbol sintáctico abstracto del programa reconocido e imprimirlo de forma legible por pantalla.

Su gramática debe ser lo suficientemente completa tal que pueda reconocer cada instrucción y expresión de *RangeX*, y combinarlos para hacer programas. Note que su gramática solo debe chequear la forma de la entrada, pero no su significado; es decir, no es necesario que detecte problemas de tipos o variables no declaradas.

Para construir el árbol sintáctico debe implementar las clases necesarias para representar las instrucciones y expresiones de *RangeX*. La representación debe usar tipos recursivos de datos y aprovechar la herencia de clases.

Su programa principal deberá llamarse `rangex` y recibirá como único argumento el nombre del archivo a analizar. La ejecución del mismo mostrará en pantalla la representación del árbol de sintaxis (descrita más adelante). Si el analizador lexicográfico consigue errores, deben reportarse todos y abortar la ejecución de la misma manera que se realizó en la primera entrega.

Basta que su reconocedor aborte ante el primer error de sintaxis. Si bien las herramientas en uso le permitirían implantar una recuperación de errores sintácticos completa, no es el objetivo de este curso, así que no invierta tiempo en ello. Sin embargo, al encontrar el primer error sintáctico, el reconocedor debe abortar la ejecución indicando la línea y columna del error, así como el token que causó el problema. Por ejemplo:

```
program
  begin
    a = ;
    b = ;
  end
```

Al pasar este archivo por su programa debe obtenerse un resultado similar a:

```
$ ./rangex con_errores_sintacticos.rgx
Error de sintaxis en línea 3, columna 9:  token ';' inesperado.
```

Para la impresión del árbol sintáctico considere el siguiente ejemplo de código en *RangeX*:

```
program
begin
  declare
     a, b as int;
     x, y as range

  // iterar sobre el primer rango
  read a;
  read b;
  x = a..b;
  for i in x do
    writeln "Variable \"i\" es igual a: ", i;

  // iterar sobre el segundo rango
  read y;
  for j in y do
    write j, ", ";

  a = 3 + b;
  b = -4;

  case b of
   x    -> writeln b
   y    -> writeln a
   a..b -> begin
             declare z as range
             writeln a, b;
             z = x <> y;
             writeln bottom(z), "..", top(z)
           end
  end;

  while i < 10 do
     begin
      read i;
      write "Still here!"
     end
end
```

La impresión de árbol podrá verla en el archivo `ejemplo_enunciado.output`.

**Lenguajes y Herramientas a usar** Para la implementación de este proyecto se permitirá el uso de las siguientes herramientas. Estos son:

- *Python:*
  - Interpretador *python* 2.6.6.
  - Generador de analizadores lexicográficos y sintácticos *PLY* 3.3.3.
- *Haskell:*
  - Compilador *ghc* 6.12.1.
  - Generador de analizadores sintácticos *Happy* 1.18.4.
- *Ruby:*
  - Interpretador *ruby* 1.8.7.
  - Generador de analizadores sintácticos *Racc* 1.4.5.

**Entrega de la Implementación**

- Un archivo llamado `gramatica.txt` que contenga la gramática propuesta por usted para reconocer *RangeX*. Esta gramática debe coincidir con la implementada por su reconocedor y debe seguir el siguiente formato:

  Símbolo inicial: `S`

  ```
  S  ->  A B C
     |   A B

  A  ->  C
  ...
  ```

- Un correo electrónico, a su preparador asignado, con el código fuente de su entrega y la definición de la gramática. Todo el código debe estar debidamente documentado.

  El código fuente debe estar en un archivo comprimido `tar.gz` de la siguiente manera: `EXGY.tar.gz` donde `X` es el número (sin el número cero) de la entrega e `Y` es el número (con el número cero cuando aplique) del grupo asignado. El correo debe titularse `CI3725 - Entrega X Grupo Y`.

- Incluya un Makefile si va a utilizar Haskell. Si su entrega no compila no será corregido.
- Un *breve* informe (`README.txt`) explicando la formulación/implementación/problemas de su analizador sintáctico y justificando todo aquello que usted considere necesario.
- Respete las reglas de juego expuestas en la página oficial del curso.

*Nota: Es importante que su código pueda ejecutarse en las máquinas del LDC, pues es ahí y únicamente ahí donde se realizará su corrección.*

**Referencia Bibliográfica**

[WM95] R. Wilhelm & D. Maurer. *Compiler Design.* Addison-Wesley, 1995.

[S97] T. Sudkamp. *Languages and Machines.* Second Edition. Addison-Wesley, 1997.

**Fecha de Entrega:** Viernes 07 de Junio (Semana 7), hasta las 11:59:59 pm.
**Valor:** 8%.

---

E.H. Novich, R. Monascal, H. González, J. Goncalves, J. Sanchez y M. Woo / Abril 2013