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

Etapa IV
Interpretador con verificaciones dinámicas

Finalmente, en esta última entrega usted debe *interpretar* programas escrito con el lenguaje *RangeX*. Para ello se debe apoyar de las estructuras arrojadas por el analizador sintáctico: la tabla de símbolos y el árbol sintáctico abstracto.

Debe recorrer el árbol e interpretarlo de manera que se ejecuten las instrucciones y evalúen las expresiones presentes en el programa. Usted debe implementar para las clases que representan a las instrucciones un procedimiento `ejecutar` y para las clases que representan expresiones una función `evaluar`.

Considere en tiempo de ejecución las siguientes verificaciones dinámicas:

- División por cero
  `Error: Intento de división por cero.`
- *Overflow*, el resultado de alguna operación da como resultado un número mayor al que puede representarse en 32 bits.
  `Error: Resultado no puede representarse en 32 bits.`

Si durante la verificación estática o dinámica hay errores, se debe reportar los errores y abortar la ejecución del intérprete.

Su intérprete deberá llamarse `rangex` el cual recibe el nombre de un archivo cuyo contenido es un código escrito en *RangeX*. La salida del mismo dependerá de las instrucciones `write`, `writeln` y `read` presentes en el programa o los errores que puedan presentarse.

Lenguajes y Herramientas a usar Sólo se permitirá el uso de las siguientes herramientas:

- *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 lexicográficos *Alex* 2.3.3.
  - Generador de analizadores sintácticos *Happy* 1.18.4.
- *Ruby:*
  - Interpretador *ruby* 1.8.7.
  - Expresiones regulares que provee el lenguaje.
  - Generador de analizadores sintácticos *Racc* 1.4.5.

Recuerde: Debe continuar trabajando con el lenguaje de programación escogido desde la primera entrega.

Entrega de la Implementación

- Un correo electrónico, a todos los preparadores, con el código fuente de su intérprete. 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 intérprete 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: Domingo 07 de Julio (Semana 11), hasta las 11:59:59 pm.
Valor: 10%.

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