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

Etapa III
Tabla de Símbolos y Verificaciones estáticas

Para esta entrega usted debe modificar la gramática propuesta para *RangeX* con las sugerencias dadas por su preparador. Además, proveerá su implementación del tipo de datos para la tabla de símbolos y debe llenarla haciendo uso de sus operaciones.

La tabla de símbolos es un tipo de datos asociado a un *bloque*. Usted debe implementar un módulo correspondiente a la tabla de símbolos **SymTable** el cual debe incluir como mínimo las siguientes funciones:

- `SymTable.new()`: o algún método de nombre análogo que permita crear una tabla de símbolos vacía.
- `SymTable.insert(...)`: inserta un elemento en la tabla de símbolos.
- `SymTable.delete(...)`: borra un elemento de la tabla de símbolos.
- `SymTable.update(...)`: actualiza la información de un elemento en la tabla de símbolos.
- `SymTable.isMember(...)`: preguntar la existencia de un elemento en la tabla de símbolos.
- `SymTable.find(...)`: retorna la información de un elemento en la tabla de símbolos, asumiendo que el mismo existe.

Nota: Los parámetros de cada método se dejan a su preferencia y conveniencia.

Su programa principal deberá llamarse `rangex` y recibirá como argumento el nombre del archivo a analizar. En adición a la impresión del árbol sintáctico abstracto (Entrega II), se debe imprimir en pantalla la tabla de símbolos (el formato de impresión se describe más adelante). En caso de error, sólo imprimir los errores conseguidos durante la verificación estática.

Los errores (y su formato) que se deben reportar para la verificación estática son:

- Todos los errores lexicográficos.
  `Error en línea 5, columna 4:  caracter inesperado "ñ".`
- El primer error sintáctico.
  `Error de sintaxis en línea 5, columna 4:  token 'begin' inesperado.`
- Utilización de variables no declaradas.
  `Error en línea 5, columna 4: no puede usar la variable "x" pues no ha sido declarada.`
- Redeclaración de variables en el mismo bloque.
  `Error en línea 5, columna 4: la variable "y" ya ha sido declarada.`
- Errores de tipo. Por ejemplo, sumar un booleano y un entero.
  `Error en línea 5, columna 4: intento de "sumar" la variable "p" del tipo "bool" y la variable "x" del tipo "entero".`

- Intento de uso de la variable de una iteración determinada. `Error en línea 5, columna 4: se intenta modificar la variable "i" la cual pertenece a una iteración.`

Importante: Recuerde aplicar las reglas de alcance de las variables definidas en el lenguaje *RangeX*.

Nota: El módulo **SymTable** debe estar en un archivo aparte llamado *SymTable.py*, *SymTable.hs* o *SymTable.rb*, según sea el caso del lenguaje escogido.

Imprima la tabla de símbolo (y el árbol de sintaxis abstracto) con el siguiente formato:

```
BLOQUE
  TABLA DE SIMBOLOS
  variable: x | tipo: int
  variable: y | tipo: int
  variable: z | tipo: range
  variable: p | tipo: int

  ASIGNACION
    var: x
    val: CONSTANTE_ENT
           valor: 1
  SEPARADOR
  ASIGNACION
    var: y
    val: CONSTANTE_ENT
           valor: 1
  SEPARADOR
  ASIGNACION
    var: z
    val: EXPRESION_BIN
           operador: instanciacion
           operando izq: VARIABLE
                           nombre: x
           operando der: VARIABLE
                           nombre: y
  SEPARADOR
  CONDICIONAL
    condicion: EXPRESION_BIN
                 operador: igual que
                 operando izq: VARIABLE
                                 nombre: x
                 operando der: VARIABLE
                                 nombre: y
    verdadero: ASIGNACION
                 var: x
                 val: EXPRESION_BIN
                        operador: suma
                        operando izq: FUNCION_EMB
                                        nombre: rtoi
                                        argumento: VARIABLE
                                                     nombre: z
                        operando der: CONSTANTE_ENT
```

```
                                        valor: 1
    falso: BLOQUE
             TABLA DE SIMBOLOS
             variable: z | tipo: int
             variable: q | tipo: range

             ASIGNACION
               var: z
               val: FUNCION_EMB
                      nombre: rtoi
                      argumento: EXPRESION_BIN
                                   operador: instanciacion
                                   operando izq: VARIABLE
                                                   nombre: x
                                   operando der: VARIABLE
                                                   nombre: y
             SEPARADOR
             WRITELN
               expresion: VARIABLE
                            nombre: z
```

La impresión anterior pertenece a un código escrito en *RangeX* que luce así:

```
program
begin
  declare
     x, y as int;
     z as range;
     p as int

  x = 1;
  y = 1;
  z = x..y;
  if x == y then
    x = rtoi(z) + 1
  else
    begin
      declare
         z as int;
         q as range
      z = rtoi(x..y);
      writeln z
    end
end
```

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.

Entrega de la Implementación

- Un correo electrónico, a todos los preparadores, con el código fuente de su entrega. 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 implementación 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 23 de Junio (Semana 9), hasta las 11:59:59 pm.
Valor: 7%.

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