Tarea 2 - Intérpretes

Profesor: Gonzalo Navarro
Auxiliar: Rodrigo Fuentes
Ayudantes: Javiera Alegría
Gabriel Chaperón
Matías Rojas

Fecha de entrega: 16 de Diciembre, 2018

Descripción

El objetivo de esta tarea es generar un intérprete para un lenguaje similar al de la P1 del control 2. Para esto, puede utilizar distintas herramientas que construyen parsers. Algunas son:

- PLY (Python Lex-Yacc): `https://www.dabeaz.com/ply/`
- ANTLR: `https://github.com/antlr/antlr4`
- BYACC/J: `http://byaccj.sourceforge.net/`
- Flex + Bison: `http://aquamentus.com/flex_bison.html`
- Lex + Yacc: `http://dinosaur.compilertools.net`
- JFlex: `http://jflex.de`

Si conoce otra herramienta similar también puede utilizarla.

El lenguaje:

Como recordará, el lenguaje descrito era un lenguaje para manejar enteros, que permitiera las siguientes características:

(a) Mantener variables escalares.

(b) Asignarles el resultado de expresiones aritméticas.

(c) Leer un entero de una entrada y guardarlo en una variable.

(d) Imprimir el resultado de una expresión aritmética.

(e) La sentencia condicional if-then-else según el resultado de una expresión aritmética (cero = falso, distinto a cero = verdadero)

(f) La sentencia while-do, de la misma forma.

Las expresiones aritméticas pueden tener constantes enteras, suma, resta, multiplicación, división, signo negativo, paréntesis y comparación (a través de ==, !=, <, >, >= y <=), como notará estas últimas fueron agregadas para la tarea. Las variables no necesitan declararse.

A continuación, una gramática libre de contexto que especifica el lenguaje en forma precisa.

$$
\begin{array}{lll}
< S > & ::= & < Asign >; \mid \\
 & & < Read >; \mid \\
 & & < Print >; \mid \\
 & & < If >; \mid \\
 & & < While >; \mid \\
 & & \{< S >\} \mid \\
 & & < S >< S > \\
 & & \\
< Asign > & ::= & < V >=< Exp > \\
 & & \\
< V > & :== & < Let > \mid < V >< Let > \mid < V >< Dig > \\
 & & \\
< Num > & ::= & < Dig >< Num > \mid < Dig > \\
 & & \\
< Let > & ::= & a \mid b \mid ... \mid z \\
 & & \\
< Dig > & ::= & 0 \mid 1 \mid 2 \mid ... \mid 9 \\
 & & \\
< Exp > & ::= & < Exp > + < Exp > \mid \\
 & & < Exp > - < Exp > \mid \\
 & & < Exp > * < Exp > \mid \\
 & & < Exp > / < Exp > \mid \\
 & & (< Exp >) \mid \\
 & & - < Exp > \mid \\
 & & < Exp >< C >< Exp > \mid \\
 & & < Num > \mid \\
 & & < V > \\
 & & \\
< C > & ::= & == \mid != \mid > \mid < \mid >= \mid <=
\end{array}
$$

$$
\begin{aligned}
&< Read > && ::= && < V >= read() \\
&< Print > && ::= && print(< Exp >) \\
&< If > && ::= && if(< Exp >)then < S > else < S > \quad | \\
& && && if(< Exp >)then < S > \\
&< While > && ::= && while(< Exp >)do < S >
\end{aligned}
$$

Observaciones

- Se deben permitir separadores en cualquier parte del programa de forma libre (espacios, tabs, new lines)
- Las expresiones aritméticas deben respetar la precedencia de la forma que conocemos. Usted puede modificar la gramática para forzar eso, o en general, para facilitar el parsing.

Entrega

- El objetivo final es que su programa reciba como input el nombre de un archivo que contenga el programa escrito en el lenguaje antes descrito y lo ejecute.
- El plazo de la entrega vence el día 16 de Diciembre. La entrega puede ser personal o en grupos de máximo tres personas. Debe implementar su programa en Python, C, C++ o Java.
- Además del código fuente debe entregar un breve informe conteniendo una descripción de su programa, instrucciones de compilación y ejecución, y ejemplos de uso.

Ejemplo

El siguiente es un programa válido que calcula fibonacci.

```
n = read();
f1 = 1;
if (n <= 1) print(n);
else
   { f2 = 1;
     while (n > 2)
       { f = f1+f2;
         f1 = f2;
         f2 = f;
         n = n-1;
       }
     print (f2);
   }
```