Tecnológico de Monterrey

Tarea 5
Diseño de Compiladores

Para esta tarea es necesario que tengan funcionando las siguientes herramientas: flex, bison y gcc. La tarea deben entregarla el viernes 30 de diciembre de 2018.

Problema 1

El objetivo del ejercicio es que construyan un reconocedor sintáctico y un intérprete, usando flex, bison y el lenguaje de programación C del lenguaje de programación descrito más abajo.

El lenguaje maneja dos tipos de datos, enteros y números de punto flotante y su sistema de tipos es fuerte, esto es, las operaciones no pueden llevarse a cabo si los tipos de los operandos no son iguales.

El intérprete debe recibir como entrada un archivo de texto que contenga un programa en el lenguaje de programación descrito más abajo. Una vez que el reconocedor sintáctico lo reconoce como un programa válido en el lenguaje, el intérprete debe interpretar (ejecutar) el código del programa. Para esto, el intérprete debe recorrer el árbol sintáctico reducido que construye el reconocedor sintáctico. El reconocedor debe hacer la revisión de tipos y enviar mensaje de error cuando se encuentre con errores de sintaxis, tipos y declaración de variables.

La semántica del lenguaje la discutiremos en clase. El nombre del archivo que contiene el programa a interpretar debe pasársele al intérprete como un parámetro y no redireccionando la entrada desde el teclado. Esto es, no se puede usar el operador "<" para que el intérprete lea el archivo.

Lo que aparece en ***negritas*** son los símbolos terminales, y obviamente se refiere a lo que debe reconocer el reconocedor léxico. El reconocedor léxico deben hacerlo usando *flex*.

Las llamadas a función usarán el paso de parámetros por valor. Una función termina cuando ejecuta una instrucción **return** o cuando ejecuta la última instrucción posible dentro del cuerpo de la función. Si hay un **return**, entonces la función debe devolver el valor de la expresión que acompaña al **return**. Si la última instrucción ejejcutada no es un **return**, entonces debe devolver 0.

Cada función debe tener una tabla de símbolos asociada, donde aparecen las variables locales y los parámetros formales.

$$
\begin{array}{lcl}
prog & \rightarrow & \textbf{program id \{ } opt\_decls\ opt\_fun\_decls \textbf{ \}} \ stmt \\
opt\_decls & \rightarrow & decls \mid \varepsilon \\
decls & \rightarrow & dec \ \textbf{;} \ decls \mid dec \\
dec & \rightarrow & \textbf{var id:} \ tipo \\
tipo & \rightarrow & \textbf{int} \mid \textbf{float} \\
opt\_fun\_decls & \rightarrow & fun\_decls \mid \varepsilon \\
fun\_decls & \rightarrow & fun\_declsfun\_dec \mid fun\_dec \\
fun\_dec & \rightarrow & \textbf{fun id (}oparams\textbf{) :} \ tipo \ \textbf{\{} \ opt\_decls \ \textbf{\}} \ stmt \\
oparams & \rightarrow & params \mid \varepsilon \\
params & \rightarrow & \textbf{var id :} \ tipo \\
stmt & \rightarrow & assign\_stmt \\
 & \mid & if\_stmt \\
 & \mid & iter\_stmt \\
 & \mid & cmp\_stmt \\
assig\_stmt & \rightarrow & \textbf{set id} \ expr\textbf{;} \\
 & \mid & \textbf{read} \ id\textbf{;} \\
 & \mid & \textbf{print} \ expr\textbf{;} \\
 & \mid & \textbf{return} \ expr\textbf{;} \\
if\_stmt & \rightarrow & \textbf{if (}expresion\textbf{)} \ stmt \\
 & \mid & \textbf{ifelse (}expresion\textbf{)} \ stmt \ stmt \\
iter\_stmt & \rightarrow & \textbf{while (}expresion\textbf{)} \ stmt \\
 & \mid & \textbf{for set id} \ expr \ \textbf{to} \ expr \ \textbf{step} \ expr \ \textbf{do} \ stmt \\
cmp\_stmt & \rightarrow & \textbf{\{ \}}|\textbf{\{} \ stmt\_lst \ \textbf{\}} \\
stmt\_lst & \rightarrow & stmt \mid stmt\_lst \ stmt
\end{array}
$$

$$
\begin{array}{lcl}
expr & \rightarrow & expr \ \mathbf{+} \ term \\
 & | & expr \ \mathbf{-} \ term \\
 & | & term \\
 & & \\
term & \rightarrow & term \ \mathbf{*} \ factor \\
 & | & term \ \mathbf{/} \ factor \\
 & | & factor \\
 & & \\
factor & \rightarrow & \mathbf{(} \ expr \ \mathbf{)} \\
 & | & \mathbf{id} \\
 & | & \mathbf{numi} \\
 & | & \mathbf{numf} \\
 & | & \mathbf{id} \ \mathbf{(} \ opt\_exprs \ \mathbf{)} \\
 & & \\
opt\_exprs & \rightarrow & expr\_lst \\
 & | & \varepsilon \\
 & & \\
expr\_lst & \rightarrow & expt\_lst\mathbf{,} \ expr \\
 & | & expr \\
 & & \\
expresion & \rightarrow & expr < expr \\
 & | & expr > expr \\
 & | & expr = expr \\
 & | & expr <= expr \\
 & | & expr >= expr
\end{array}
$$