Gramática libre de contexto.
Mini java

Integrantes: Cordero Hernández Marco Ricardo

1. Define formalmente la gramática.

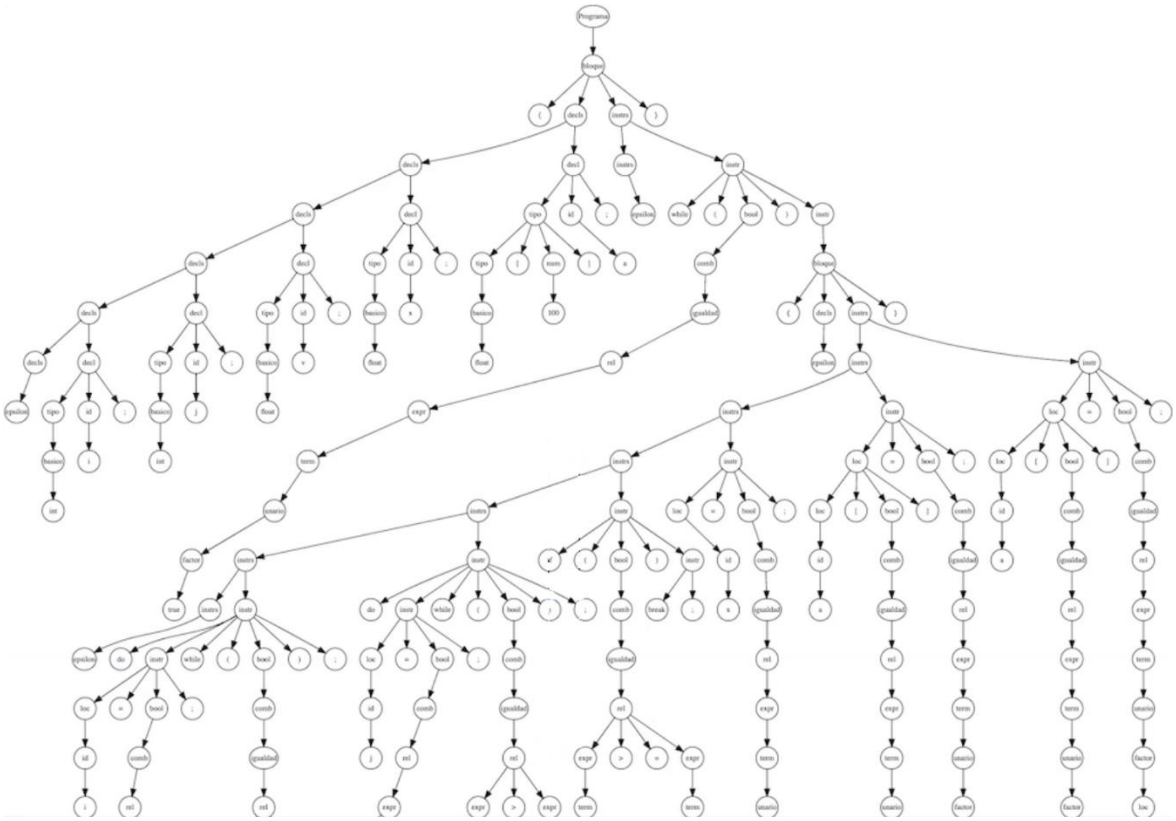
$$
\begin{array}{rcl}
\textit{programa} & \rightarrow & \textit{bloque} \\
\textit{bloque} & \rightarrow & \{\ \textit{decls instrs}\ \} \\
\textit{decls} & \rightarrow & \textit{decls decl} \mid \epsilon \\
\textit{decl} & \rightarrow & \textit{tipo}\ \mathbf{id}\ ; \\
\textit{tipo} & \rightarrow & \textit{tipo}\ [\ \mathbf{num}\ ] \mid \mathbf{basico} \\
\textit{instrs} & \rightarrow & \textit{instrs}\ \textit{instr} \mid \epsilon \\
\textit{instr} & \rightarrow & \textit{loc} = \textit{bool}\ ; \\
 & \mid & \mathbf{if}\ (\ \textit{bool}\ )\ \textit{instr} \\
 & \mid & \mathbf{if}\ (\ \textit{bool}\ )\ \textit{instr}\ \mathtt{else}\ \textit{instr} \\
 & \mid & \mathbf{while}\ (\ \textit{bool}\ )\ \textit{instr} \\
 & \mid & \mathbf{do}\ \textit{instr}\ \mathbf{while}\ (\ \textit{bool}\ )\ ; \\
 & \mid & \mathbf{break}\ ; \\
 & \mid & \textit{bloque} \\
\textit{loc} & \rightarrow & \textit{loc}\ [\ \textit{bool}\ ] \mid \mathbf{id} \\
\textit{bool} & \rightarrow & \textit{bool}\ ||\ \textit{comb} \mid \textit{comb} \\
\textit{comb} & \rightarrow & \textit{comb}\ \&\&\ \textit{igualdad} \mid \textit{igualdad} \\
\textit{igualdad} & \rightarrow & \textit{igualdad} == \textit{rel} \mid \textit{igualdad}\ != \textit{rel} \mid \textit{rel} \\
\textit{rel} & \rightarrow & \textit{expr} < \textit{expr} \mid \textit{expr} <= \textit{expr} \mid \textit{expr} >= \textit{expr} \mid \\
 & & \textit{expr} > \textit{expr} \mid \textit{expr} \\
\textit{expr} & \rightarrow & \textit{expr} + \textit{term} \mid \textit{expr} - \textit{term} \mid \textit{term} \\
\textit{term} & \rightarrow & \textit{term} * \textit{unario} \mid \textit{term} / \textit{unario} \mid \textit{unario} \\
\textit{unario} & \rightarrow & !\ \textit{unario} \mid -\ \textit{unario} \mid \textit{factor} \\
\textit{factor} & \rightarrow & (\ \textit{bool}\ ) \mid \textit{loc} \mid \mathbf{num} \mid \mathbf{real} \mid \mathbf{true} \mid \mathbf{false}
\end{array}
$$

2. Describe los patrones para cada uno de los terminales.

terminal	descripción / patrón	ejemplos
num	(0+1+…+9)*	100, 45, 430876
(	(	(
)	)	)
id	$[a\text{-}z]^+ ([0\text{-}9]^+ \mid [a\text{-}z]^+)^*$	var1, id20,
If	if	If
while	while	while
do	do	do
break	break	break
;	;	;
real	-?([0-9])+(.[0-9]+)?((E|e)[0-9]+(.[0-9]+)?)?	
true	true	true
false	false	false

3. Genera el árbol de derivación para la siguiente entrada, utiliza graphviz para generarlo. (Ignora los comentarios)

```
{                    // Archivo prueba
   int i; int j; float v; float x; float[100] a;
   while( true ) {
      do i = i+1; while( a[i] < v);
      do j = j-1; while( a[j] > v);
      if( i >= j ) break;
      x = a[i]; a[i] = a[j]; a[j] = x;
   }
}
```



4. Analiza si la gramática es ambigua. Si es ambigua encuentra un ejemplo con sus respectivos arboles de derivación.
   Utilizando el ejemplo { int [10] a ; float b ; if(a[2] == b) if(b<3+2) a[2]=b*2; else b=3; } se puede decir que la gramática si es ambigua, puesto que se pueden generar dos árboles a partir de la instrucción *b < 3 + 2*.