## Tarea 6 (Se puede hacer en equipo)

La gramática en notación BFN (Backus-Naur) que vimos en clase

```
<Programa> ::= "programa {"<BloqueSentencias>"}"
<BloqueSentencias> ::= <Sentencia>
<BloqueSentencias> ::= <Sentencia><BloqueSentencias>
<Sentencia> ::= <Asignacion>
<Asignacion> ::= <Id>""<Val>";"
<Val> ::= <Numero> | <id> | <Val>"+"<Val>
<Numero> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
<id> ::= "a" | "b" | "c"

permite derivar programas como el siguiente:

programa{
    a=1;
    c=2;
    b=a+b+1;
}
```

## **Ejercicios:**

- a) Deberán extenderla de forma que permita:
- valores númericos positivos enteros de más de un dígito, por ejemplo: 10, 1003, 29, etc.
- Valores booleanos "cierto" y "falso"
- Programas con estructuras de control tipo **if** e **if-else** con la siguiente sintaxis

```
si(identificador){
            bloque de sencencias
}

y

si(identificador){
            bloque de sencencias
}sino{
            bloque de sencencias
}
```

**Nota:** En la gramática lo que importa es que el programa esté bien formado, no importa si semánticamente tiene sentido, es decir en este punto no importa si el tipo de **identificador** no es un valor booleano

• Instrucciones de entrada y salida (lee y escribe) con la siguiente sintaxis:

lee **identificador**; (la funcionalidad que se pretende modelar es poder leer del teclado y almacenar el valor leido en **identificador**)

escribe **identificador**; (la funcionalidad que se pretende modelar es escribir en pantalla el valor de **identificador**)

Con los cambios introducidos, la gramática deberá permitir derivar las siguiente cadenas (programas), en azul se indican los nuevos elementos.

```
programa{
  a=cierto;
  d=falso;
  b=100;
  c=23;
  si(a){
    b=b+1;
    c=b;
  }
 si(d){
    b=0;
    c=c+10;
 }sino{
   b=b+1;
   c=c+2;
}
}
programa{
  a=cierto;
  d=falso;
  b=0;
  si(a){
    lee b;
  escribe b;
  si(d){
    b=b+10;
  }
 }
```

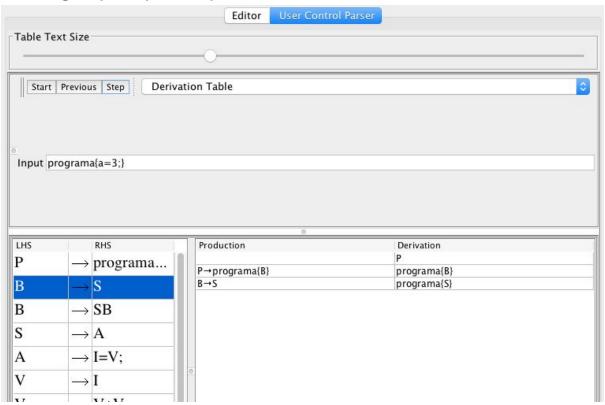
los saltos de linea no están contemplados en la gramática, son solo para facilitar la lectura, por lo que relmente las cadenas a derivar serían

 $programa\{a=cierto; d=falso; b=100; c=23; si(a)\{b=b+1; c=b;\} si(d)\{b=0; c=c+10;\} sino\{b=b+1; c=c+2;\}\}$ 

У

programa{a=cierto;d=falso;b=0;si(a){lee b;}escribe b;si(d){b=b+10;}}

- b) Verificar en JFLAP que la gramática, con las modificaciones introducidas, permite generar los programas antes mencionados. Se anexa el archivo .jff con la gramática base. Para probar la gramática:
  - i) ir a input->user control parse
  - ii) seleccionar derivation Table
  - iii) introducir la cadena a derivar en el campo input
  - iv) oprimir el botón start para comenzar la derivación
  - v) seleccionar del lado izquierdo la regla a aplicar sobre la cadena de entrada
  - vi) luego step para aplicarla o previous para deshacer el cambio
  - vii) Repetir desde el paso *v* hasta terminar la derivación o no haya más reglas que se puedan aplicar



c) Dar un ejemplo de un programa con dos errores de sintaxis y explicar cuáles son los errores

## NOTA: Para esta tarea los entregables (se suben por separado) son:

- 1. El archivo .jff con la gramática modificada
- 2. Un archivo PDF con:
  - a. la gramática modificada en notación BNF (inciso a)
  - b. las capturas de pantalla de la tabla de derivación (final) de los dos programas
  - c. el programa con errores