PRÁCTICA 2: GRAMÁTICAS Y GENERADOS AUTOMÁTICOS

Ingeniería de computadores 2023/2024

Álvaro Merino

Pedro García

Karim Laouini

Índice

Parte 1

Gracias al parser y lexer que hemos desarrollado, hemos generado los siguientes árboles de sintaxis que reflejan cómo se estructuran las interacciones en nuestro sistema. A continuación, mostramos un ejemplo del árbol antes de realizar las modificaciones, y después, incluyendo las nuevas funcionalidades que hemos implementado.

En esta actualización, hemos añadido dos nuevas interacciones al sistema: **"peligro"** y **"criatura"**. Estas permiten representar situaciones más complejas y dinámicas en la narrativa, como encuentros con criaturas específicas o eventos peligrosos. Esto amplía las posibilidades de diseño y profundidad en el juego.

Ejecución

Dentro de la carpeta Parte1_2 y desde una terminal deberás ejecutar los siguientes comandos:

antlr MapaLexer.g4 : este comando compila el lexer de la parte 1.

antlr MapaParser.g4 : este comando compila el parser de la parte 1.

javac *.java : este comando compila todos los archivos .java de la parte1.

Si quieres utilizar el comando que crea el árbol y/o los tokens con ANTLR:

grun Mapa mapa -tokens -gui ejemplo.field

Si quieres ejecutar el programa para analizar un fichero generando su árbol en texto plano:

java Main

Después seleccionar la opción de Analizar pulsando 1:

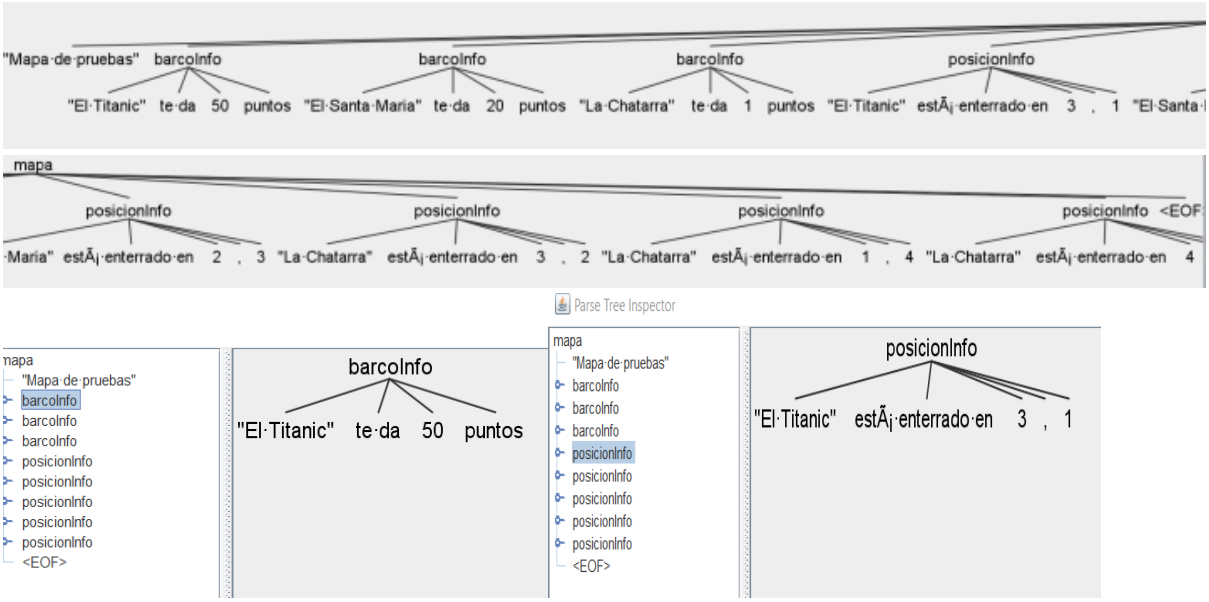
```
Elija una opcion:
1. Analizar un archivo
2. Iniciar el juego
3. Salir
1
Ingrese el archivo a analizar: ejemplo.field
```

Y al final te solicita que incluyas el archivo que quieres comprobar (debes añadir su extensión)

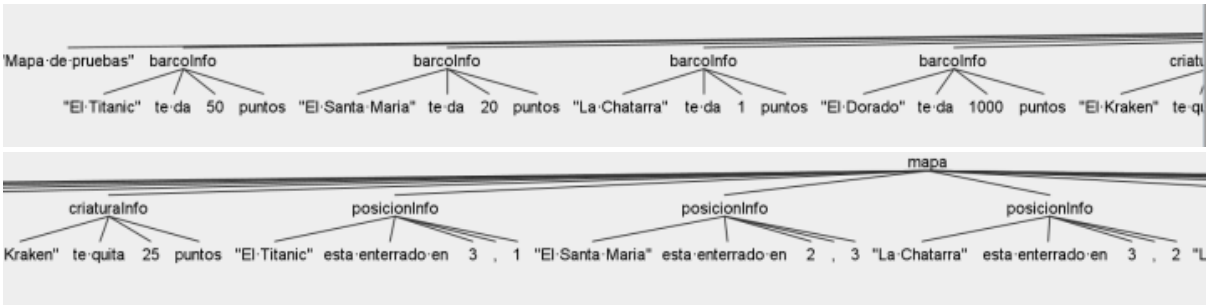
Árboles AST

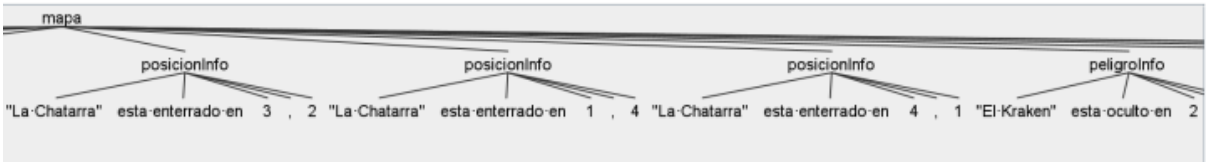
A continuación, se muestra la evolución de los árboles sintácticos:

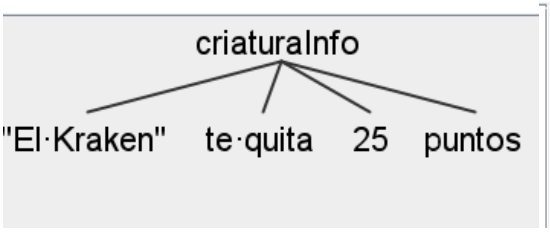
1. **Árbol antes de las modificaciones:**

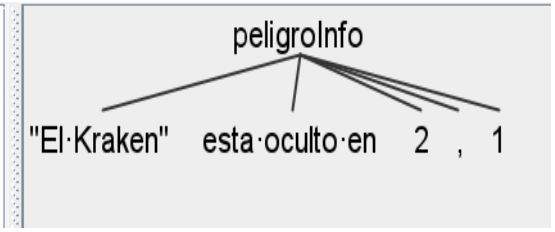


2. **Árbol después de las modificaciones:** [Ejemplo que incorpora "peligro" y "criatura"].









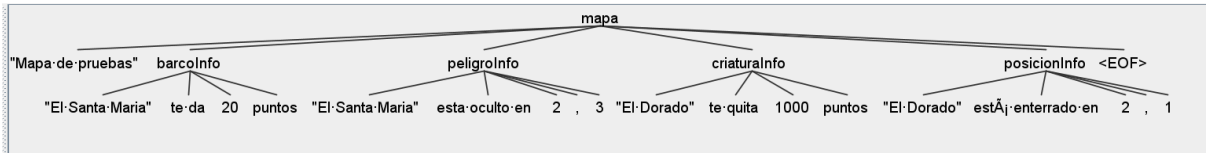
Que posteriormente hemos pasado a formato texto-plano y quedaría tal que asi:

```
Arbol sintactico:
mapa
  "Mapa de pruebas"
  barcoInfo
    "El Titanic"
    te da
    50
    puntos
  barcoInfo
    "El Santa Maria"
    te da
    20
    puntos
  barcoInfo
    "La Chatarra"
    te da
    1
    puntos
  posicionInfo
    "El Titanic"
    estÃ¡ enterrado en
    3
    ,
    1
  posicionInfo
    "El Santa Maria"
    estÃ¡ enterrado en
    2
    ,
    3
  posicionInfo
    "La Chatarra"
    estÃ¡ enterrado en
    3
    ,
    2
  posicionInfo
    "La Chatarra"
    estÃ¡ enterrado en
    1
```

```
posicionInfo
  "La Chatarra"
  estÃ¡ enterrado en
  1
  ,
  4
posicionInfo
  "La Chatarra"
  estÃ¡ enterrado en
  4
  ,
  1
<EOF>
```

3. Segundo Mapa de ejemplo con su árbol:

Usando el comando grun



Usando el análisis del Main:

```
Ingrese el archivo a analizar: ejemploExtra.field

Arbol sintactico:
mapa
  "Mapa de pruebas"
  barcoInfo
    "El Santa Maria"
    te da
    20
    puntos
  peligroInfo
    "El Santa Maria"
    esta oculto en
    2
    ,
    3
  criaturaInfo
    "El Dorado"
    te quita
    1000
    puntos
  posicionInfo
    "El Dorado"
    estÃ¡ enterrado en
    2
    ,
    1
  <EOF>
```

Parte 2

En esta parte funciona como ampliación de la parte 1, en concreto hemos añadido un archivo juego.java, el cuál nos servirá para crear un mapa de dimensiones 5x5, donde se colocarán barcos según el archivo/mapa introducido, estos barcos nos darán una puntuación si marcamos las coordenadas de donde se encuentran, para luego desaparecer.

Ejecución

Siguiendo los pasos de la ejecución en el apartado 1, hasta llegar al momento de selección de opción por consola, en el cual escogemos la opción 2 y nos pedirá que seleccionemos el archivo/mapa para jugar:

```
Elija una opcion:
1. Analizar un archivo
2. Iniciar el juego
3. Salir
2
Ingrese la ruta del archivo para configurar el juego: ejemplo.field
Ingrese coordenadas (x,y) o 'salir' para terminar:
```

A continuación explicamos las posibilidades del juego.

Resultado

En la siguiente captura podremos observar las tres posibilidades que tenemos en el juego, en este caso una coordenada (1,2) la cuál no tiene nada, la coordenada (3,2) ha encontrado un barco por lo que ha recibido 10 puntos y si ponemos una coordenada fuera de los límites como es la (6,6) nos lo indica, mientras que la puntuación se acumula.

```
Ingrese la ruta del archivo para configurar el juego: ejemplo.field
Ingrese coordenadas (x,y) o 'salir' para terminar: 1,2
No hay tesoro aqui. Sigue buscando.

Puntuacion acumulada: 0
Ingrese coordenadas (x,y) o 'salir' para terminar: 3,2
Â¡Encontraste un barco! Puntuacion: 10

Puntuacion acumulada: 10
Ingrese coordenadas (x,y) o 'salir' para terminar: 6,6
Â¡Coordenadas fuera del mapa!
Puntuacion acumulada: 10
Ingrese coordenadas (x,y) o 'salir' para terminar: salir
Fin del juego.
```

Parte 3

Siguiendo el mismo enfoque detallado en la primera parte, hemos diseñado un lexer y un parser capaces de procesar correctamente todos los ejemplos proporcionados por el profesor. Estos componentes generan árboles de sintaxis abstracta (AST) que reflejan fielmente la estructura lógica de cada ejemplo.

Una vez confirmada la funcionalidad con los ejemplos dados, hemos creado dos casos adicionales que permiten corroborar que nuestra gramática es robusta y que puede manejar escenarios más complejos o variados. Este proceso garantiza que el lexer y el parser no solo son adecuados para los ejemplos iniciales, sino que también tienen la flexibilidad necesaria para ser escalables y adaptarse a futuras expansiones.

Ejecución

Para compilar el Lexer: **antlr EjemploLexer.g4**
Para compilar el Parser: **antlr EjemploParser.g4**
Para compilar los programas .java: **javac *.java**
Si queremos ver el árbol mostrado en el terminal poner el comando:

java Main nombre (nombre del archivo)

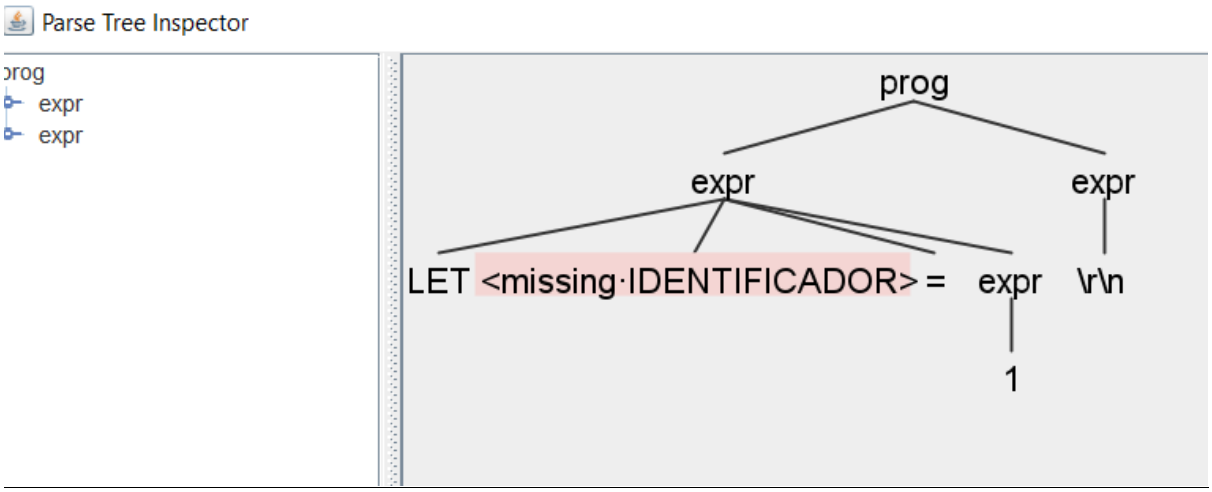
Si queremos en este caso ver los árboles AST y los tokens:

grun Ejemplo prog -gui -tokens nombre (nombre del ejemplo)

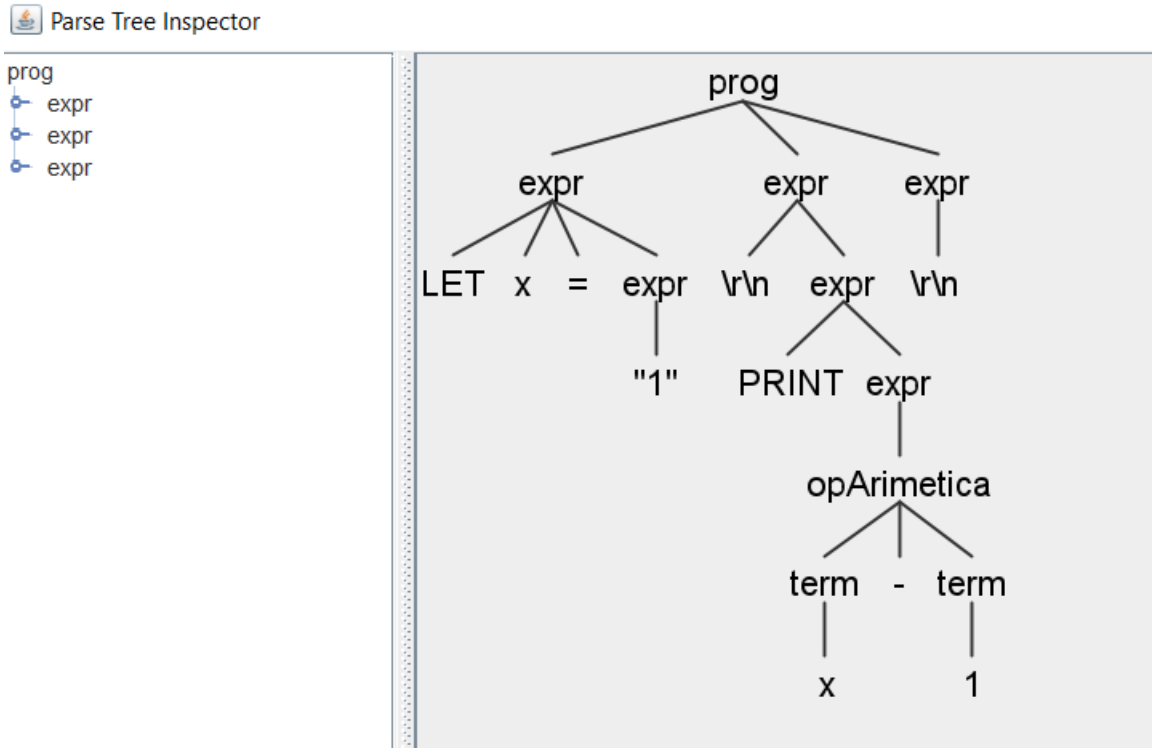
Árboles AST

A continuación, se muestran los árboles AST generados tanto para los ejemplos originales como para los casos adicionales creados.

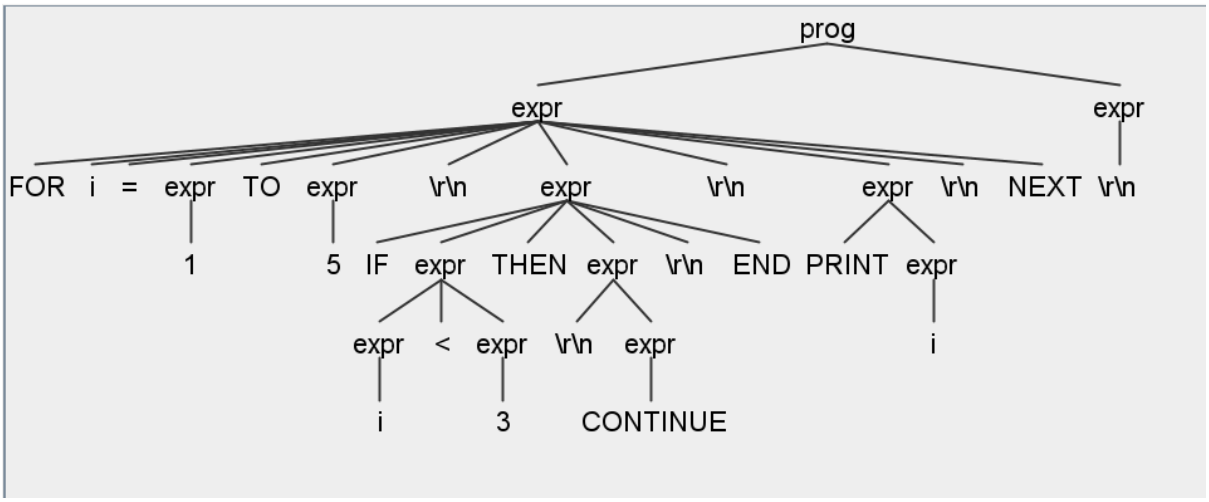
Error_sintactico.bas:



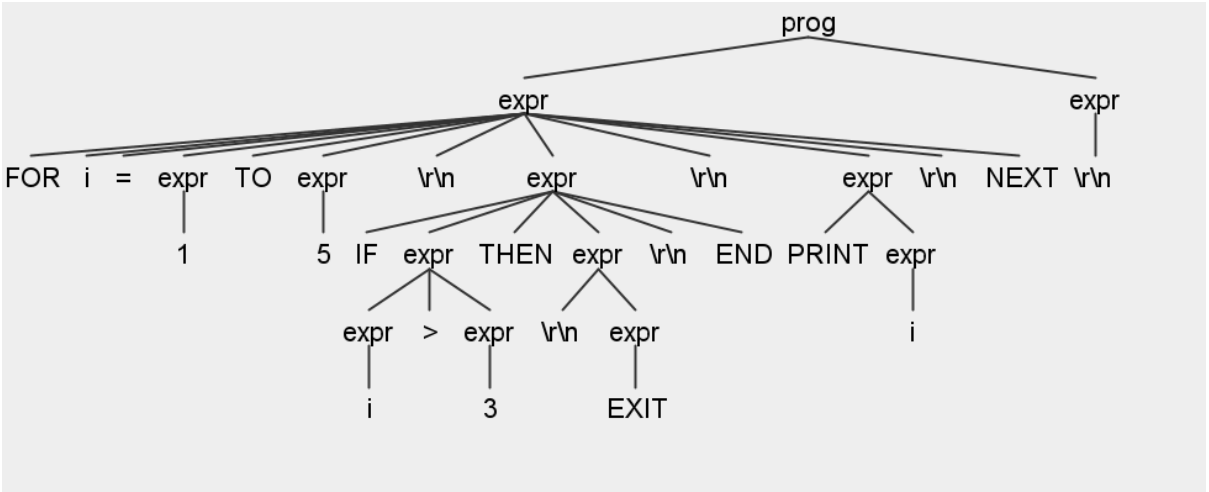
Error_tipo.bas:



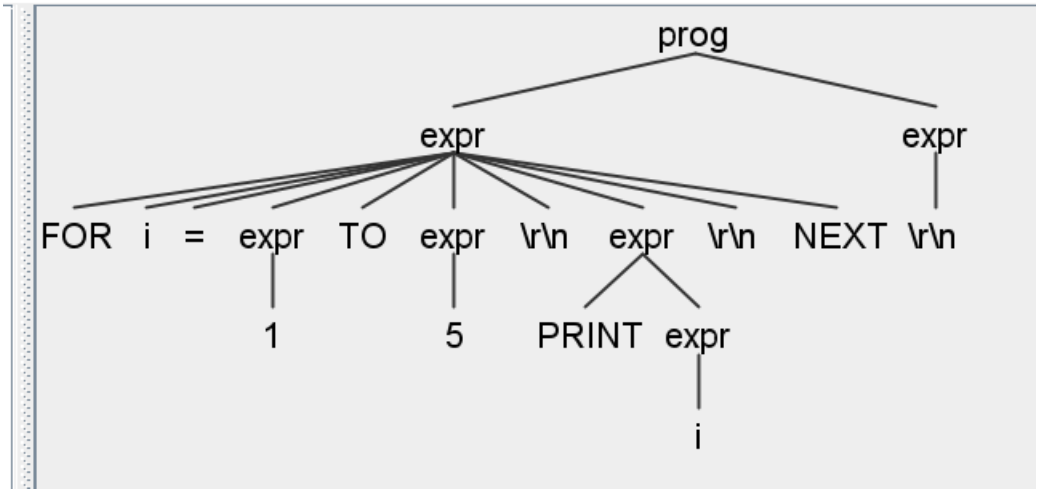
For_continue.bas:



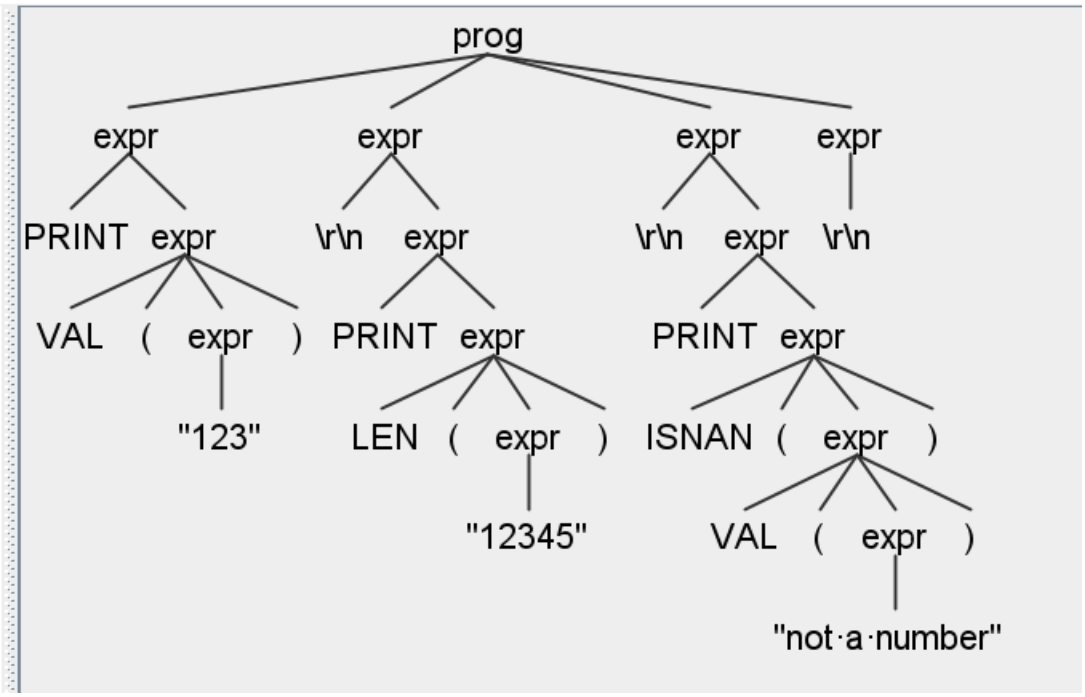
For_exit.bas:



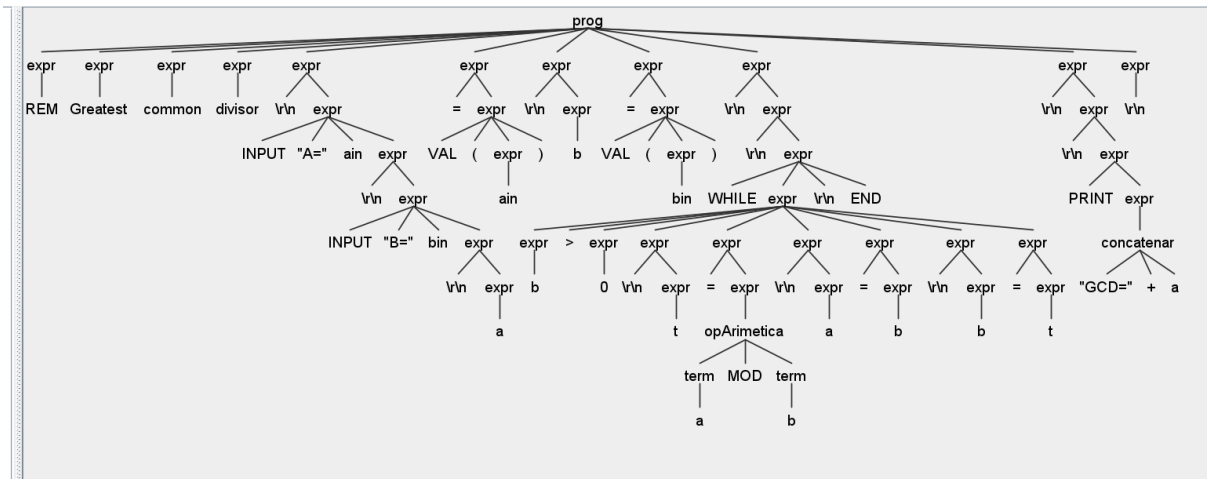
For_simple.bas:



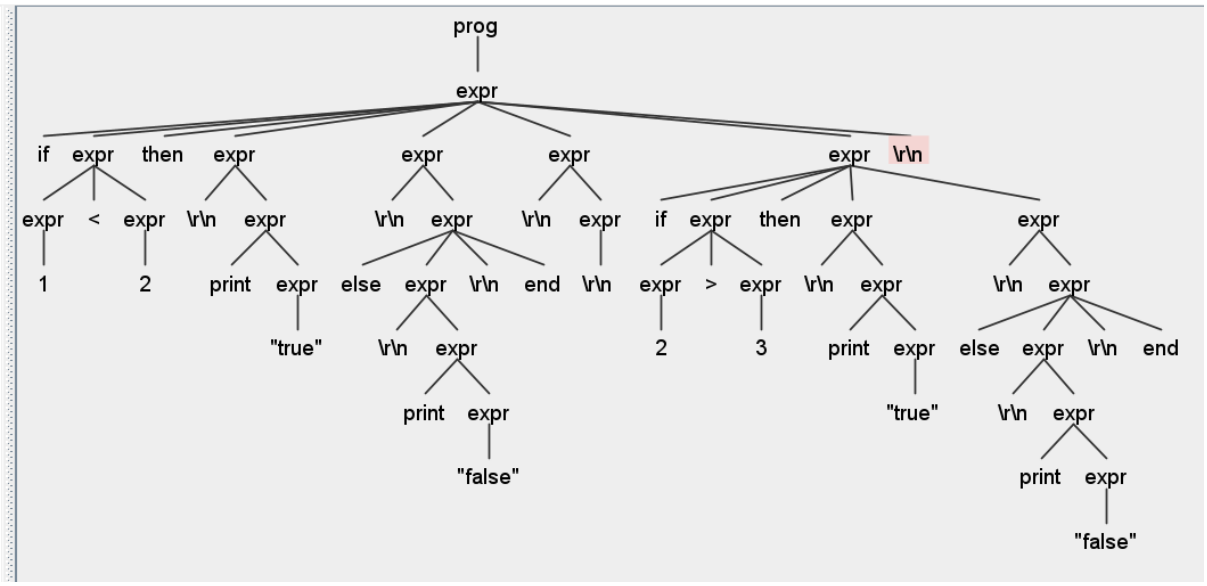
Functions.bas:



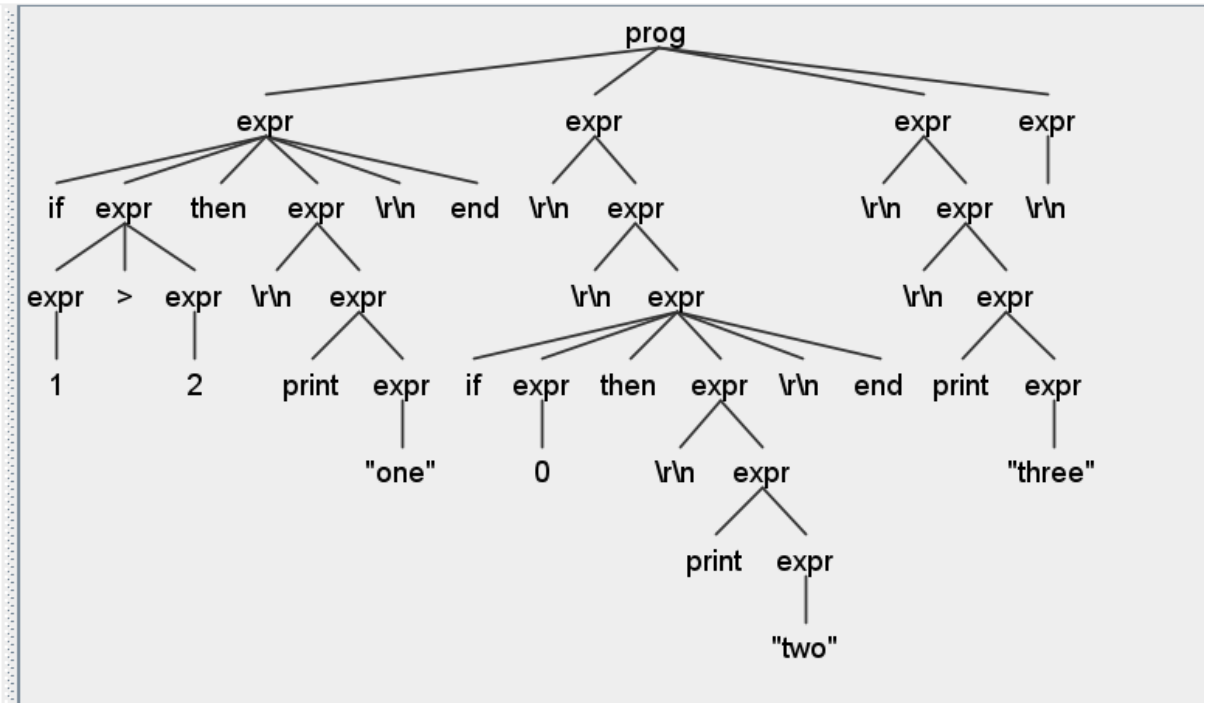
Gcd_euclid.bas:



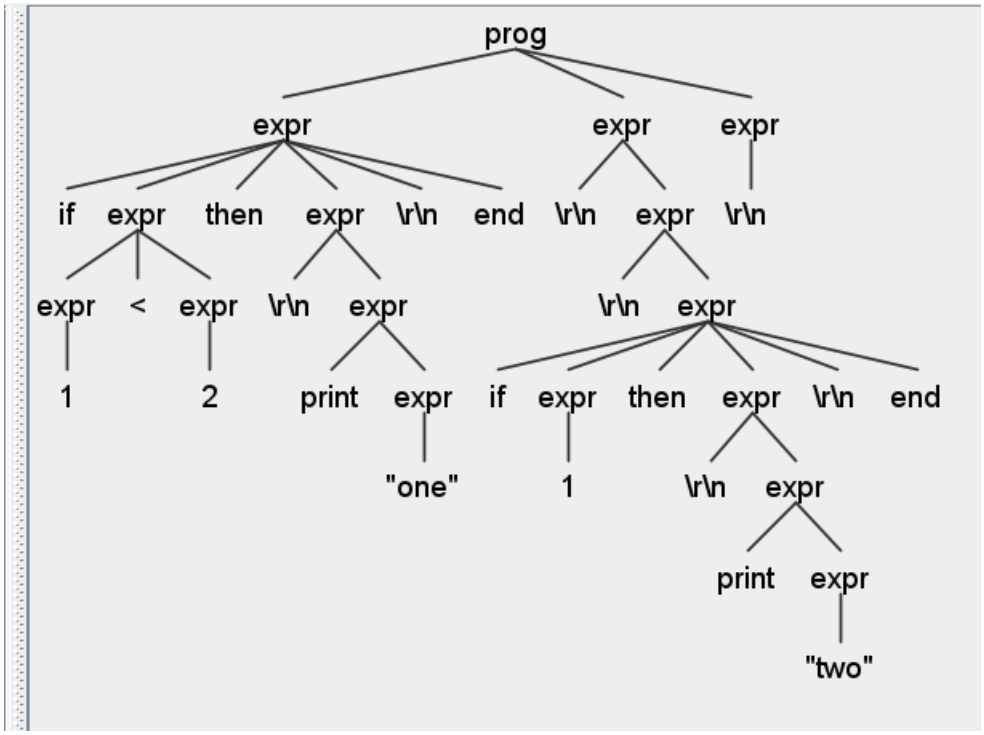
If_else.bas:



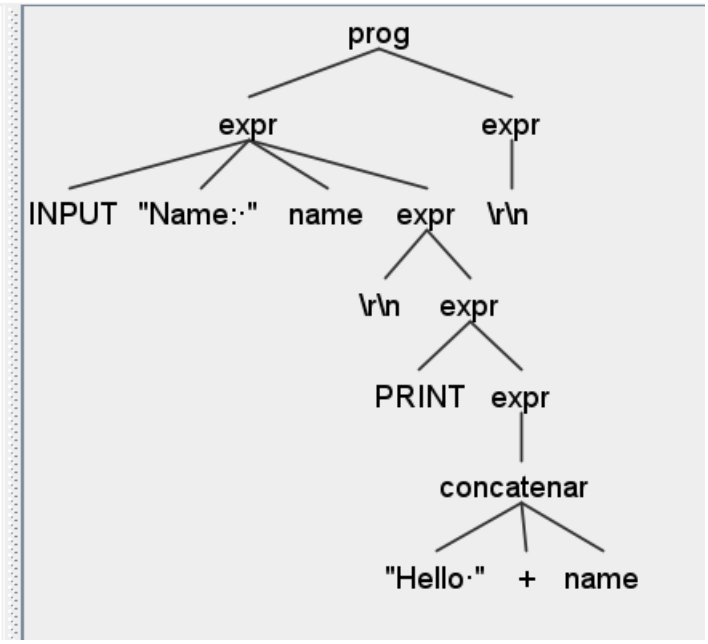
If_simple_false.bas:



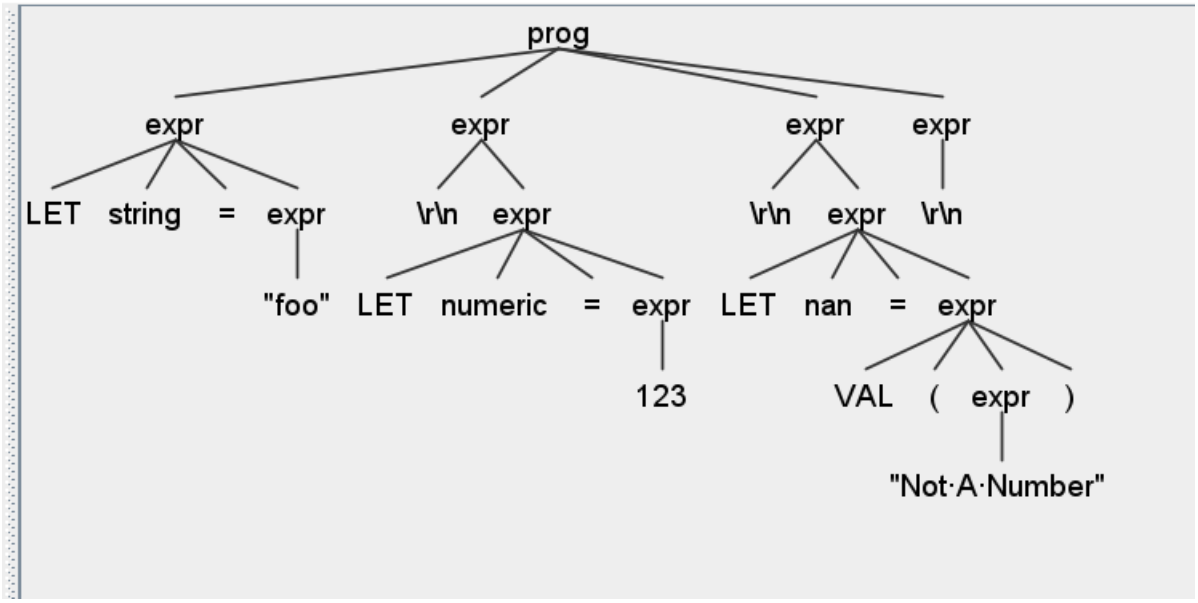
If_simple_true.bas:



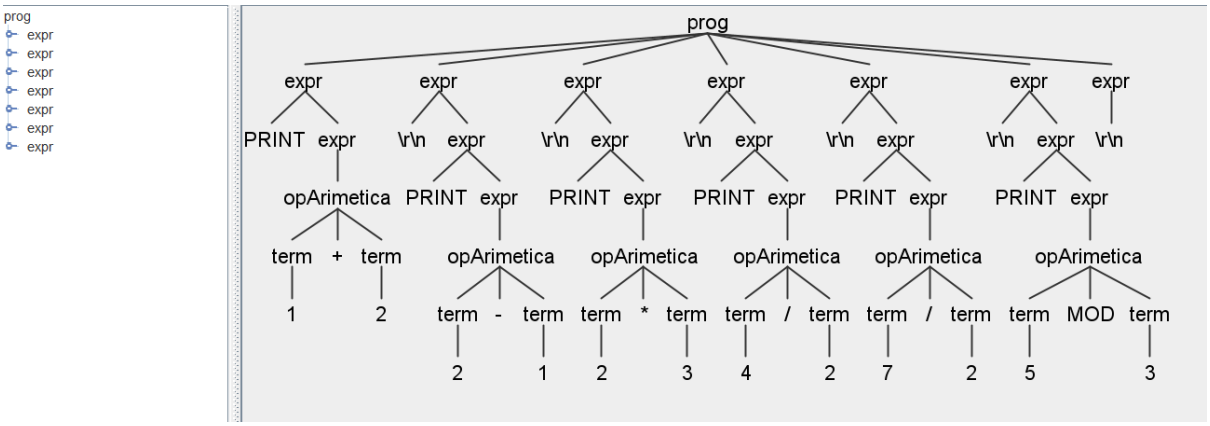
Input.bas:



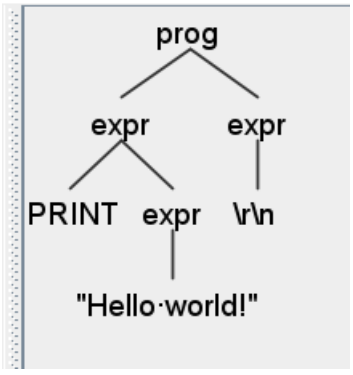
Let.bas:



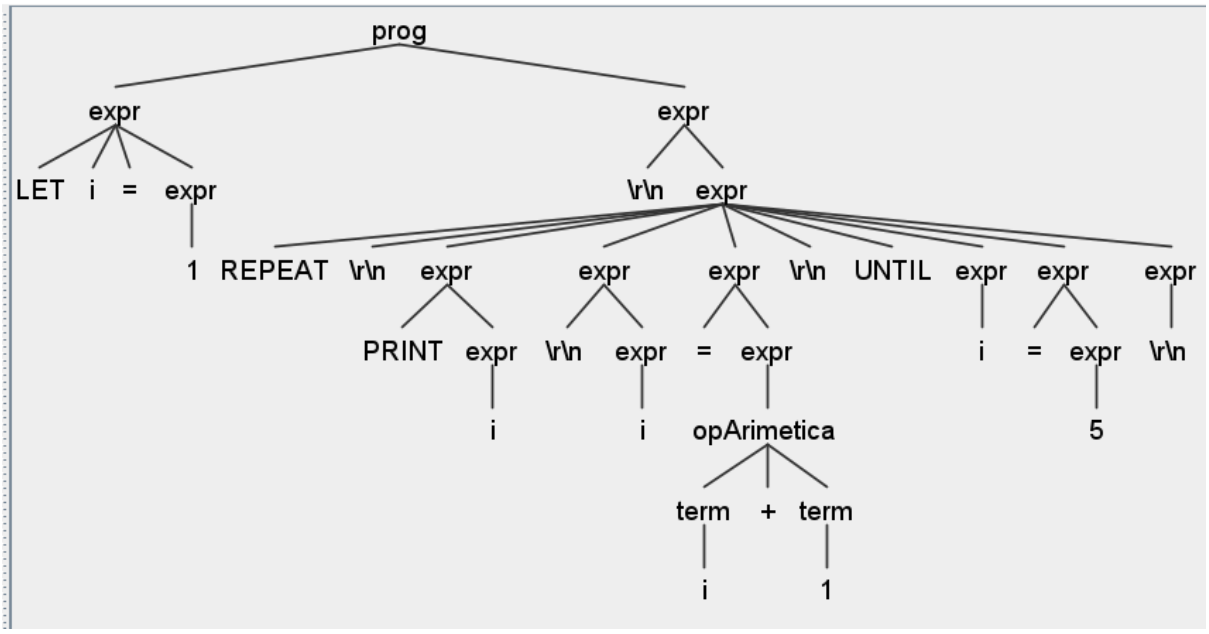
Operations.bas:



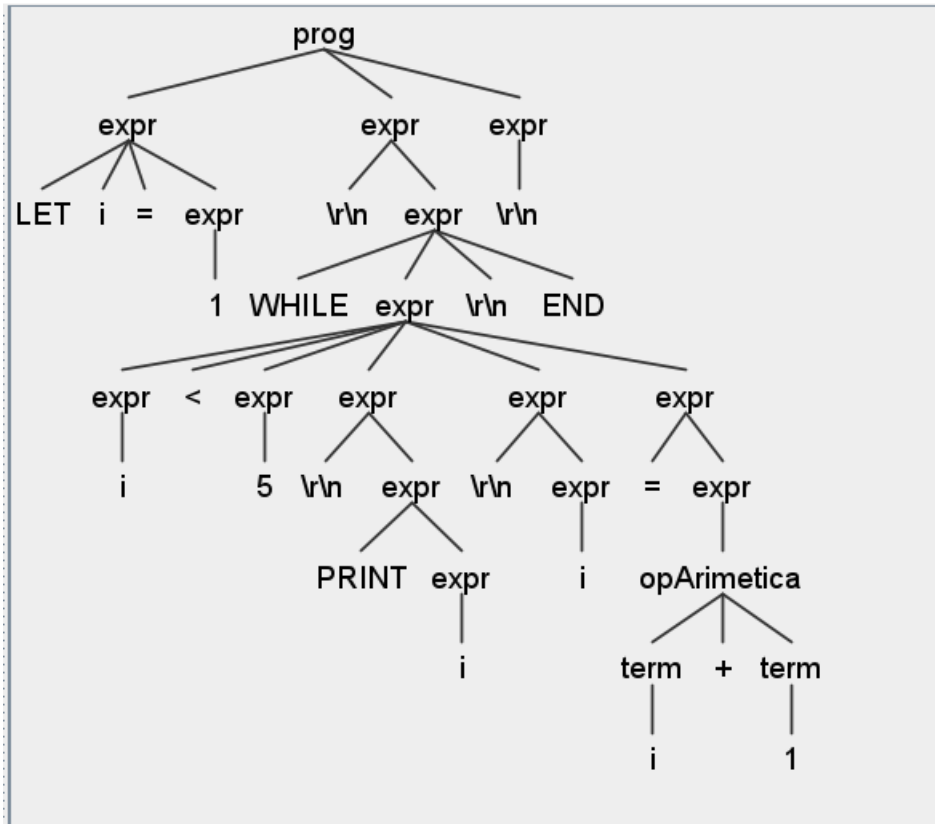
Print.bas:



Repeat.bas:

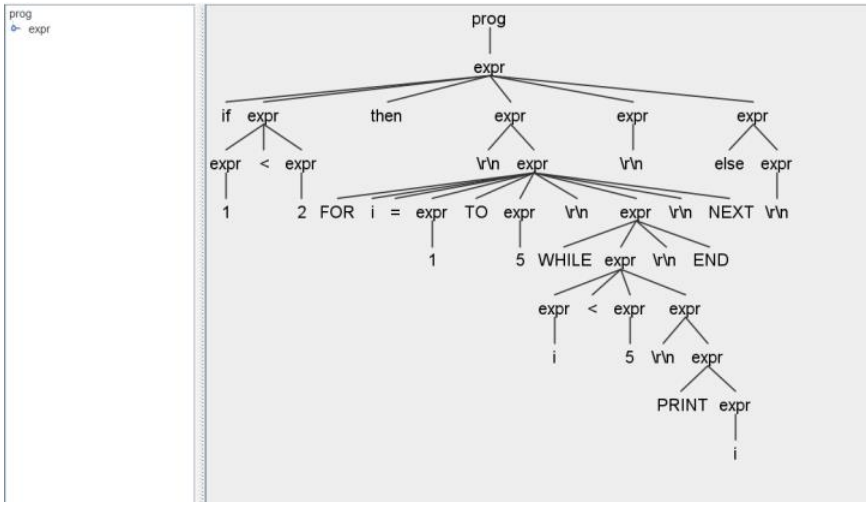


While.bas:

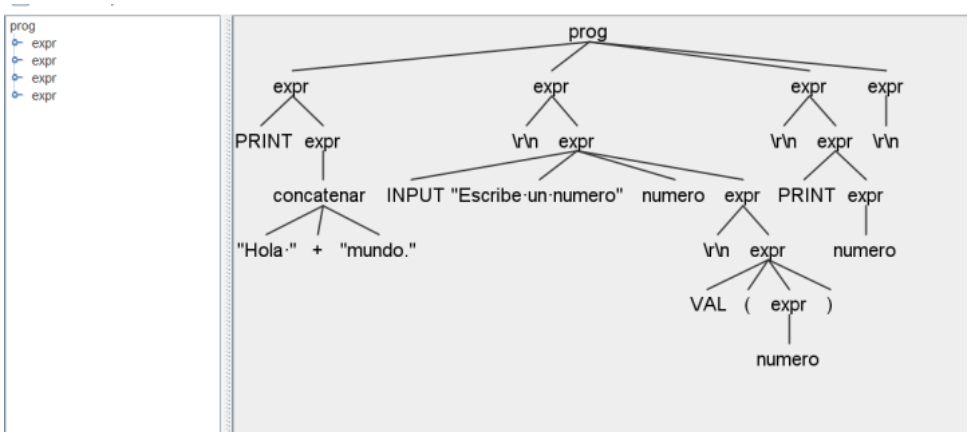


Y esto son los dos Extras (Extra1 y Extra2) que hemos incluido como ejemplos para corroborar que toda la gramática funciona correctamente:

Extra1.bas:



Extra2.bas:



Árbol AST en modo fichero de texto plano de uno de los ejemplos (operaciones.bas):

```
Arbol:
prog
  expr
    PRINT
    expr
      opArimetica
        term
          1
        +
        term
          2
  expr


    expr
      PRINT
      expr
        opArimetica
          term
            2
          -
          term
            1
  expr


    expr
      PRINT
      expr
        opArimetica
          term
            2
          *
          term
            3
  expr


    expr
      PRINT
```

```
  expr
    PRINT
    expr
      opArimetica
        term
          4
        /
        term
          2
expr


  expr
    PRINT
    expr
      opArimetica
        term
          7
        /
        term
          2
expr


  expr
    PRINT
    expr
      opArimetica
        term
          5
        MOD
        term
          3
expr
```

Dificultades

La parte 2 ha dado problemas en el momento de añadir nuestra mejora de la parte 1, en este caso era poner criaturas que restaban, en este caso o se colocaban criaturas o barcos en las casillas por lo que hemos decidido que solo haya barcos en el mapa 5x5, además todas tienen la misma puntuación.

En la parte 3 hemos encontrado dificultades al hacer la gramática de todos los ejemplos, sobre todo en los ejemplos relacionados con el IF donde teníamos errores después de cada END, aunque lo hemos solucionado añadiendo el elemento INTRO cuando correspondía. También encontramos problemas a la hora de reconocer que ha terminado los ejemplos, por lo que hemos optado en modificar todos los ejemplos añadiendo una línea en blanco (un INTRO) al final de cada archivo para que nuestra gramática detecte correctamente que termina en una línea en blanco y no nos den errores.