**Imagen que contiene Texto

Descripción generada automáticamente**

**Práctica 2: Gramáticas y generadores**

**automáticos**

**591000 – Compiladores**

**Adrian Valcarcel Alvarez 52003666E**

**Joseph Pajuelo Condori 12795971J**

Contenido

[Primera parte: árbol sintáctico para un lenguaje limitado (la búsqueda del tesoro). 3](#_Toc182495647)

[Nivel básico: 3](#_Toc182495648)

[Nivel Medio: 4](#_Toc182495649)

[Segunda parte: jugando a la búsqueda del tesoro 6](#_Toc182495650)

[Nivel Avanzado: 6](#_Toc182495651)

[Tercera parte: árbol sintáctico para un lenguaje simple (MiniB) 8](#_Toc182495652)

[Nivel Básico: 8](#_Toc182495653)

[Nivel Medio: 13](#_Toc182495654)

# Primera parte: árbol sintáctico para un lenguaje limitado (la búsqueda del tesoro).

## Nivel básico:

El programa se ejecuta con el comando

**Antlr4 MapaLexer.g4**

**Antlr4 MapaParser.g4**

**Javac \*.java**

**grun Mapa mapa -gui**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamentey luego se introduce el texto que queremos analizar, para el ejemplo propuesto por la practica el árbol AST es el siguiente:

Texto

Descripción generada automáticamente

Para un ejemplo que he propuesto yo:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Para otro ejemplo:

Texto

Descripción generada automáticamente

Para esta parte de la práctica se han desarrollado un lexer que contiene todos los posible tokens incluyendo los terminales, y los literales. Tiene la suficiente flexibilidad a la hora de analizar cada token.

## Nivel Medio:

El programa se ejecuta con el comando **java** **Analizador text.txt** (text.txt o el mapa que se quiera pasar para analizar), para este programa se han usado listeners en ANTLR, gracias a esto podemos introducir nuestro mapa para analizar y crear los arboles en formato de texto plano que salen con el nombre de Árbol.html.

**-**Mejora 1: Agregar múltiples coordenadas opcionales por barco

Cada barco puede estar en múltiples ubicaciones opcionales, separadas por comas. Esto permite que un barco esté "escondido" en varios lugares del mapa y se encuentre en más de una posición posible.

Para implementar esto en la gramática, modificamos la regla de coordenada en la regla ubicación para que acepte múltiples coordenadas separadas por comas.

**-**Mejora 2: Definir áreas de búsqueda o "zonas especiales"

Se incluye una nueva regla de zona para definir zonas del mapa con nombres específicos y límites de coordenadas. Estas zonas pueden ayudar a los jugadores a encontrar barcos en áreas específicas del mapa o darles indicaciones adicionales.

Ejemplo:

Captura de pantalla de computadora

Descripción generada automáticamente

# Segunda parte: jugando a la búsqueda del tesoro

## Nivel Avanzado:

El programa se ejecuta con el comando **java** **Juego text.txt** (text.txt o el mapa que se quiera pasar para analizar), este programa se ha ampliado creando el archivo Juego.java, este es el programa que se usa para iniciar el juego, dentro de encontramos las clases tesoro, mapa, jugador, y mapaLoader el cual extiende de MapaParserBaseListener para poder cargar el mapa para el juego.

El mapa tiene el tamaño que tenga el mapa de entrada.

El jugador selecciona casillas en las que bucear.

En caso de encontrar algún tesoro, se indica, junto a la puntuación acumulada.

El tesoro desaparece.

En caso de no encontrarlo, también se mostrará un mensaje informativo al jugador.

El juego tiene condición de fin de partida (e.g., se han encontrado todos los tesoros o se han terminado las casillas/intentos).

-Complicación 1:

Al usar una matriz para salvar las coordenadas de los tesoros estas empiezan por la coordenada 0,0 (debido a que los arrays siempre empiezan en la posición cero) y el juego no debe aceptar ceros en ninguna coordenada, o ninguna coordenada por encima del limite del mapa, esto se ha solucionado haciendo una matriz de arrays de capacidad n+1 y no permitiendo al jugador introducir el valor 0 o un numero mayor de n en las coordenadas. En caso de introducir cualquiera de esas dos opciones erróneas, se solicitarán otras coordenadas y no contará como intento.

-Complicación 2:

Para conseguir que el mapa se ajuste a la entrada, el mapa tiene que inicializarse con el tamaño máximo, por tanto, el árbol tiene que recorrer hasta el ultimo nodo para obtener las coordenadas mas altas, esto hace que el mapa no se cree hasta el final y los tesoros que están en nodos intermedios no puedan guardarse en el mapa puesto que todavía no está inicializado, para solucionar esto, se recorre el árbol dos veces, la primera para crear el mapa, y la segunda para esconder los tesoros.

**-**Mejora: Tablero de tamaño variable según la entrada

El juego se ajusta siempre al tamaño del mapa de entrada, siendo los límites máximos las coordenadas más altas tanto de X e Y de las zonas del mapa (las zonas son áreas delimitadas por coordenadas que se incluyeron como mejora en el nivel medio).

El formato del mapa es el siguiente:

*"Mapa de la Isla Misteriosa"*

*Zona "Bahia Escondida" limites 1,1 , 5,5 , 2,2*

*Zona "Isla Perdida" limites 6,6 , 9,9 , 14,15*

*"El Perla Negra" te da 100 puntos*

*"El Perla Negra" esta enterrado en 1,3 , 4,5 , 2,2*

*"La Galera Roja" te da 50 puntos*

*"La Galera Roja" esta enterrado en 2,1*

*"El Esmeralda" te da 60 puntos*

*"El Esmeralda" esta enterrado en 5,4 , 3,1*

*"El Santa Maria" te da 80 puntos*

*"El Santa Maria" esta enterrado en 2,3*

*"Fin del mapa"*

# Tercera parte: árbol sintáctico para un lenguaje simple (MiniB)

## Nivel Básico:

El programa se ejecuta con el comando

**Antlr4 miniBLexer.g4**

**Antlr4 miniBParser.g4**

**Javac \*.java**

**grun miniB programa -gui** y luego se introduce el texto que queremos analizar, para los ejemplos propuesto por la practica el árbol AST es el siguiente:

Imagen que contiene Word

Descripción generada automáticamenteDiagrama

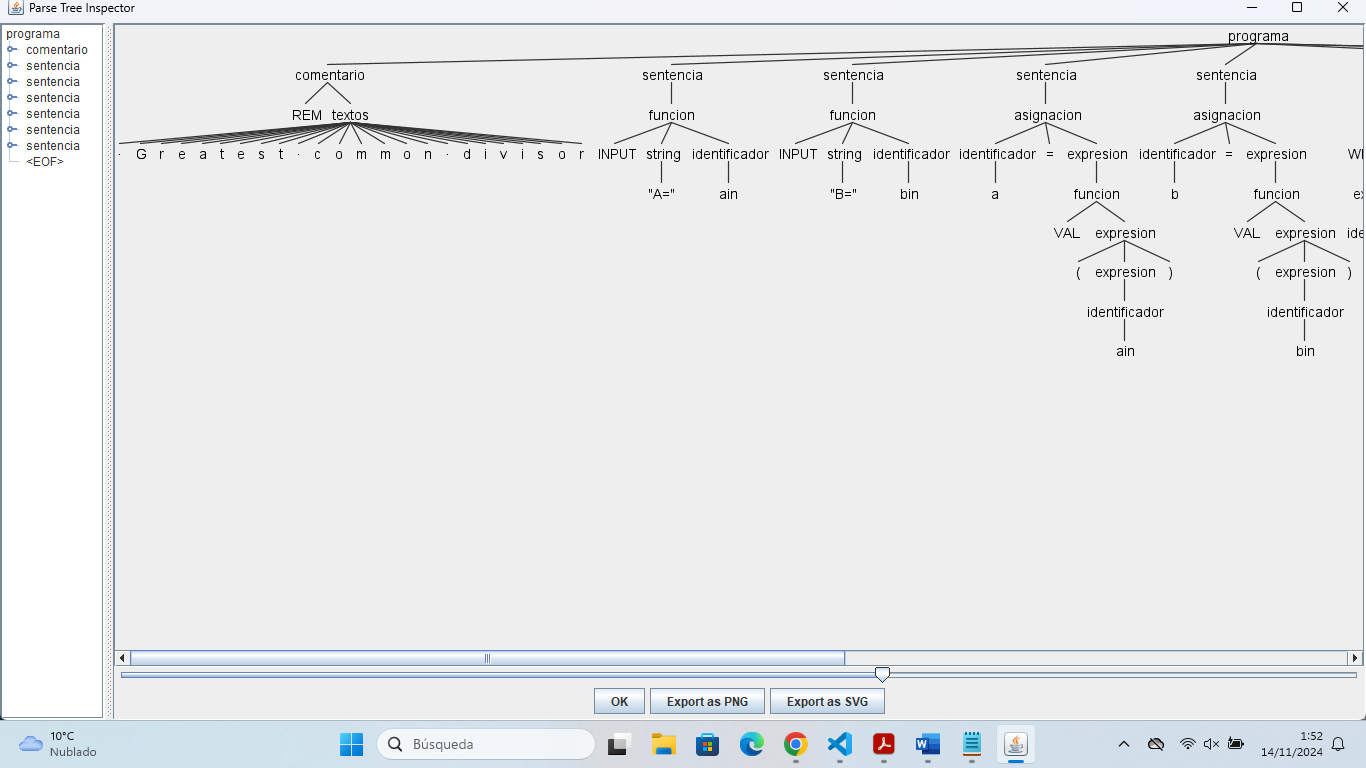
Descripción generada automáticamente con confianza media

Imagen que contiene Word

Descripción generada automáticamenteImagen que contiene Texto

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente con confianza media



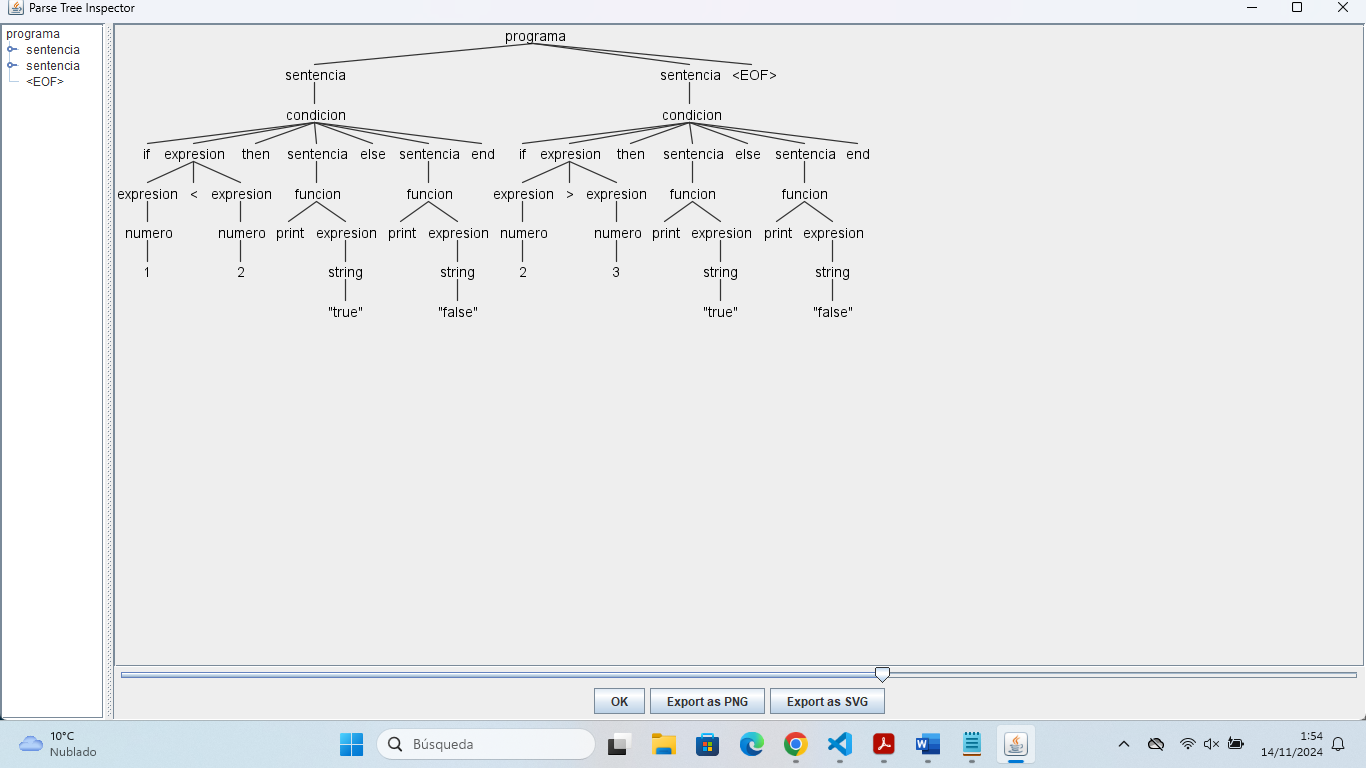
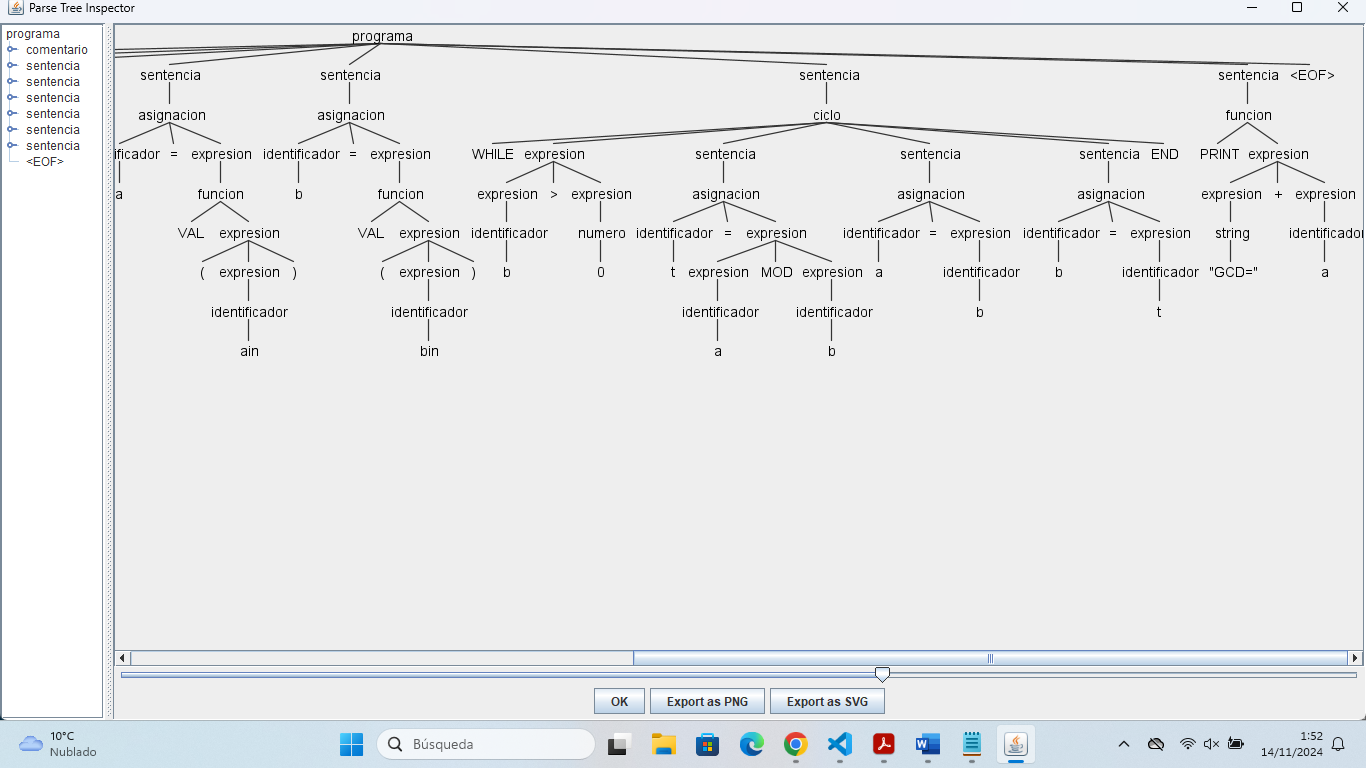


Imagen que contiene Diagrama

Descripción generada automáticamente

Imagen que contiene Gráfico

Descripción generada automáticamenteImagen que contiene Diagrama

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Imagen que contiene Diagrama

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

## Nivel Medio:

El programa se ejecuta con el comando **java** **Analizador text.txt** (text.txt o el archivo que se quiera pasar para analizar), el árbol de salida en texto plano se crea como Árbol.html.

**-**Mejora 1: Agregar un bloque SWITCH simple

**Diagrama

Descripción generada automáticamente**

-Mejora 2: Añadimos la regla función del usuario

**Imagen que contiene Diagrama

Descripción generada automáticamente**