

Instituto Tecnológico de Costa Rica



Escuela de Ingeniería en Computación

Lenguaje Betes Ra!

Compiladores e Intérpretes

Analizador:Parte 2

Grupo 20

Profesora:  
Aurelio Sanabria

Estudiantes:  
Alberto Zumbado Abarca  
Jonathan Quesada Salas  
Pablo Alberto Muñoz Hidalgo  
Jose Pablo Quirós Hidalgo  
Luis Andrés Rojas Murillo

Alajuela, 2022

## **Tabla de Contenidos**

<b>Lecciones Aprendidas</b>	<b>3</b>
Jonathan Quesada Salas	3
Alberto Zumbado Abarca	3
Pablo Alberto Muñoz Hidalgo	4
Luis Andrés Rojas Murillo	4
Jose Pablo Quirós Hidalgo	5
<b>Salud mental/ Amor propio/ Diversión/ Crecimiento Personal</b>	<b>6</b>
Jonathan Quesada Salas	6
Alberto Zumbado Abarca	8
Pablo Alberto Muñoz Hidalgo	10
Luis Andrés Rojas Murrillo	13
Jose Pablo Quirós Hidalgo	15

## **Lecciones Aprendidas**

### **Jonathan Quesada Salas**

Con lo que respecta a mis lecciones aprendidas puedo rescatar es que a la hora de hacer el ENUM del árbol de sintaxis abstracta se tiene que establecer una mayor de variables auto() ya que el analizador es una parte más avanzada que el explorador, por lo cual se necesita un mayor nivel de detalle con lo que respecta a la detección de estructuras sobre bloques de instrucciones, formato que deban de tener las operaciones, parámetros, declaración de variables, declaraciones y además tener en cuenta posibles errores, el cual se encargará la función “\_\_generar\_error”.

Adicionalmente algo relevante que aprendí sobre este proyecto fue el comportamiento que tiene el recorrido del ASA funciona de cierta manera distinta, ya que el la fase 1 del analizador se usó diccionarios los cuales contendrán reglas del programa (\_\_analizar\_función, \_\_analizar\_condicional, \_\_analizar\_repetición, \_\_analizar\_invocación, \_\_analizar\_operación) y reglas para literal (\_\_verificar\_texto, \_\_verificar\_entero, \_\_verificar\_flotante, \_\_verificar\_nulo) ya que estos mismos tienen en una misma “caja” diferentes funcionalidad que llamaban a las respectivas funciones a analizar, o bien comprobar la estructura que pueda tener un código en concreto.

Por último algo adicional que aprendí es que si bien es cierto que en la implementación del ASA respectivamente en el ENUM se tuvieron que asignar nuevas variables (auto()), estas mismas deben de ser llamadas en su totalidad en la implementación del analizador con el árbol de sintaxis abstracta.

### **Alberto Zumbado Abarca**

Para esta etapa como tal lo más importante fue analizar y comprender los diferentes componentes que puede tener el árbol y realizar la lógica respectiva de abajo hacia arriba de una forma más mental para comprender completamente cómo es que es que el árbol se rellena con los distintos nodos. Me pareció muy interesante el trabajar con la recursividad y ver cómo es posible la implementación de estos algoritmos por descenso.

Otro punto a destacar fue ver cómo realizar la estructura de datos del árbol mediante orientación a objetos, ya que como solo se había realizado en C, entonces cambiar de paradigma para crear el árbol fue algo interesante pero provechoso.

### **Pablo Alberto Muñoz Hidalgo**

En mi caso las lecciones aprendidas se resumen en tres puntos principales:

- Usar estructuras no es tan fácil cuando se deben usar de forma dinámica.
- Tener una estructura mental de cómo analizar el código no es nada fácil.
- Lograr plasmar la estructura y profundidad del código en un Árbol de Sintaxis Abstracta.

Comenzando por el primer punto me refiero a que en el curso de estructuras de datos utilizar árboles no era nada complicado, sin embargo en su momento tenía cierta complejidad y claro, ahora cuando ya hay que usar este árbol y no de manera alambrada como se hacía en antaño, ahí es donde viene lo complejo. En nuestro caso fue difícil crear nodos de la manera correcta, con sus respectivos padres. Además fue complicado el manejo de errores ya que, y esto va relacionado al segundo punto, tenemos que tener una estructura muy clara de cómo funcionaba el código, que errores y variables podría tener, como podía comenzar, como podía terminar y todo lo que no fuera eso pasarlo por error, sin embargo tampoco es la idea dejar por fuera código válido y pasarlo como si fuera un error, lo cual hizo más complejo el manejo del árbol y el analizador. Por último plasmar toda la estructura en código y verla reflejada en el árbol fué lo más complicado ya que como dije en el primer punto es muy difícil en comparación a cuando trabajábamos en C, hacer el árbol y recorrerlo de manera dinámica, osea que no sea alambrado fue complicado en sí, ahora tener que hacer este proceso cada vez que se crea un nodo y colocarlo donde se debe colocar es lo más difícil y en mi opinión donde radica la dificultad del proyecto. Debo mencionar lo obvio también, aprendí cómo funciona un ASA aplicado a un analizador y aprendí lo útil que es esta estructura para este tipo de proyectos.

### **Luis Andrés Rojas Murillo**

En cuestión a las lecciones aprendidas que podría destacar es importante decir que no hubieron muchas que considere nuevas ya que el árbol de sintaxis abstracta es

un árbol (estructura muy bien conocida) sin embargo fue importante lo relacionado a conocer que se debe de asignar al árbol y que se debe de concatenar, ya que durante la implementación del árbol tuvimos varios problemas relacionados a este dilema.

Por otro lado fue importante considerar la abstracción necesaria para poder saber los datos necesarios a incluir en el árbol de sintaxis abstracta y cuales funciones eran las que retornan los datos necesarios en el árbol y cuáles no, también se podría considerar la habilidad de determinar a nivel de diseño el donde captar los datos en incluirlos en el árbol.

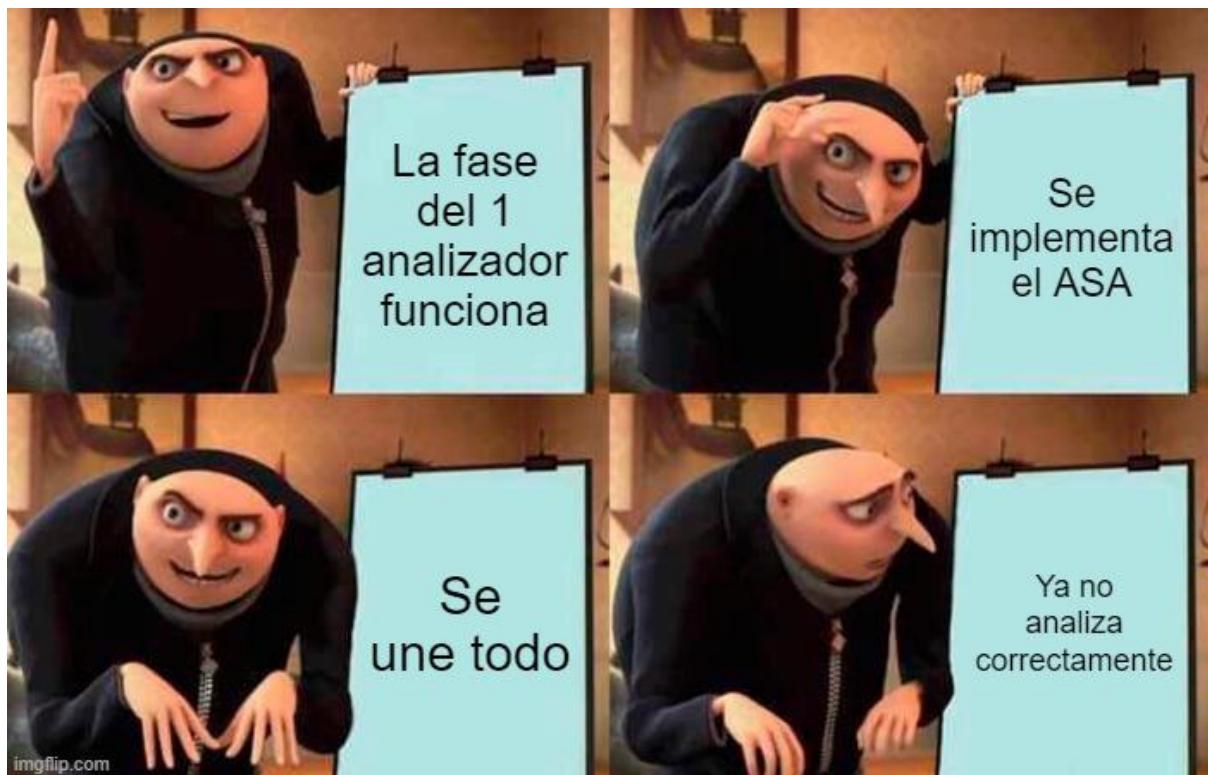
### **Jose Pablo Quirós Hidalgo**

Un árbol a pesar de ser una estructura conocida creo que nos exigió un poco más de nivel que el que teníamos de dicha estructura. El analizar el código punto por punto, y a la profundidad que había que analizarlo ya era una tarea no muy sencilla, y el implementar esto al árbol de sintaxis abstracta fue una tarea que tomó tiempo para pensar en cómo implementarlo, porque antes de empezar el análisis del problema y la implementación pensamos que no era tan complicado recorrer el código, pero no fue tan fácil como esperábamos, aprendí muchas cosas sobre la marcha, en cuanto a los mecanismos y estructuras para recorrer el código y en cuanto a las funciones utilizadas en el árbol.

Salud mental/ Amor propio/ Diversión/ Crecimiento Personal

Jonathan Quesada Salas





La fase 2 debe de funcionar  
bien si la fase 1 lo hacía

Si esta el ASA y los  
nuevos nodos bien implantados



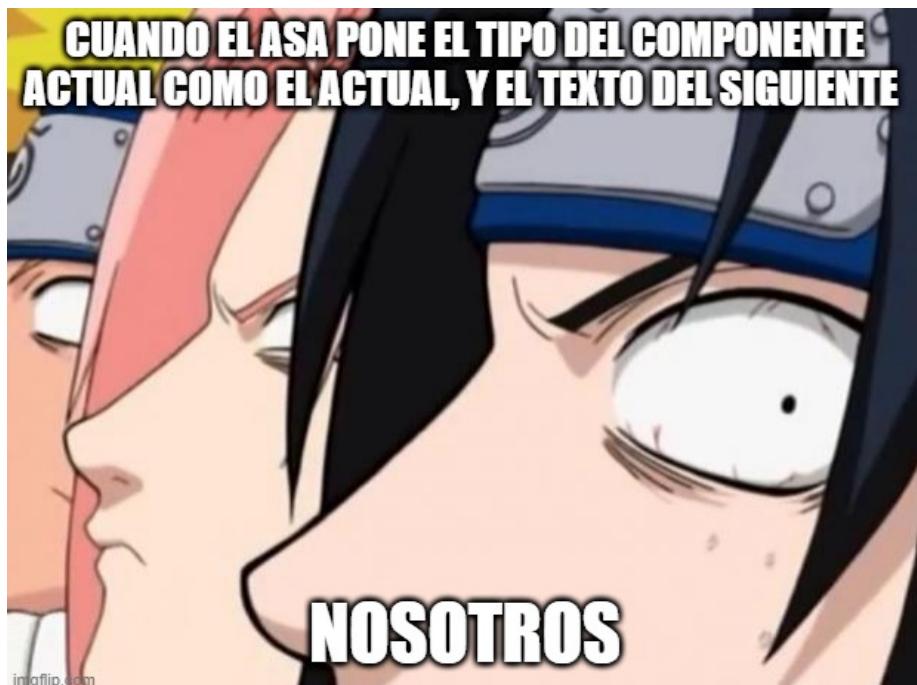
Alberto Zumbado Abarca



condicionales



map de  
componente y función



**NOSTROS: PORQUE NO CORRE?**

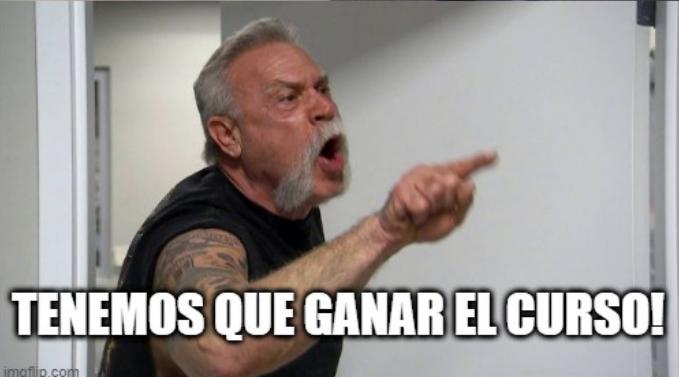
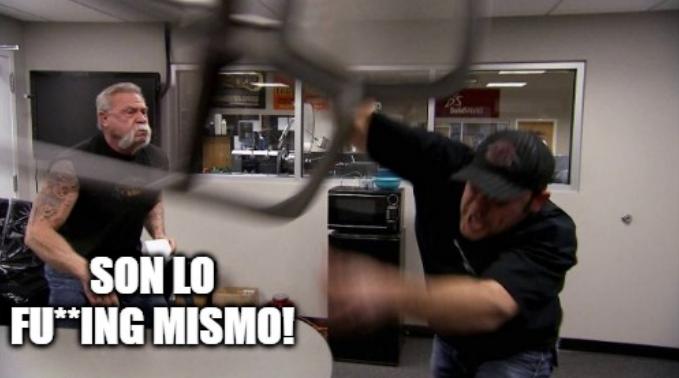
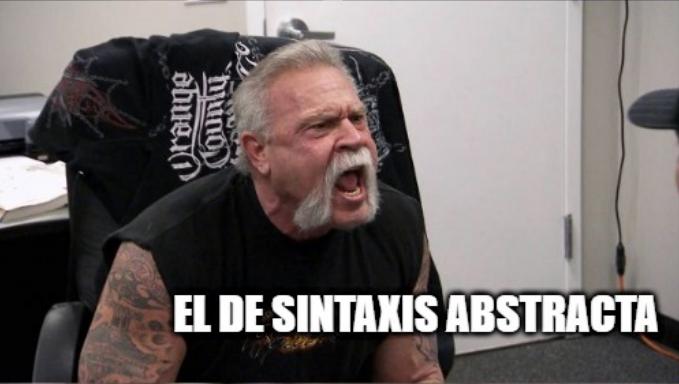
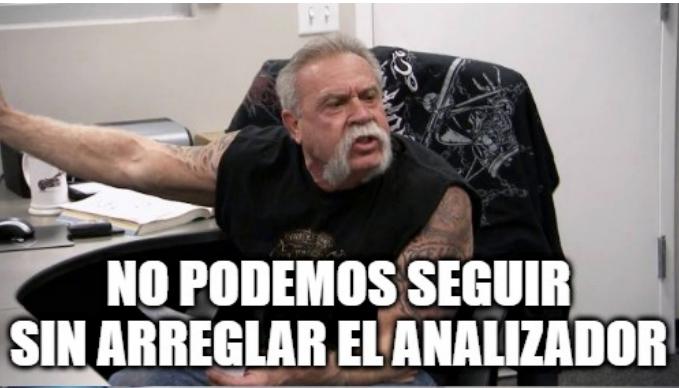


**PRIMERA VEZ CORRIENDO  
EL ANALIZADOR CON EL ASA**

imgflip.com

Pablo Alberto Muñoz Hidalgo







Luis Andrés Rojas Murrillo



Un árbol  
de sintaxis  
abstracta  
incompleto



El árbol  
con todos  
los valores  
que necesita





**Jose Pablo Quirós Hidalgo**



JAKE-CLARK.TUMBLR



yo diciendo que es  
cualquier vara lo que  
hay que hacer en esta fase



Yo viendo todo  
lo que había que hacer

[imgflip.com](#)