

Instituto Tecnológico de Costa Rica



Escuela de Ingeniería en Computación

Lenguaje Betes Ra!

Compiladores e Intérpretes

Analizador: Análisis por Descenso Recursivo

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

Lecciones Aprendidas	3
Jonathan Quesada Salas	3
Pablo Alberto Muñoz Hidalgo	5
Luis Andrés Rojas Murrillo	6
Jose Pablo Quirós Hidalgo	6
Salud mental/ Amor propio/ Diversión/ Crecimiento Personal	8
Jonathan Quesada Salas	8
Alberto Zumbado Abarca	10
Pablo Alberto Muñoz Hidalgo	12
Luis Andrés Rojas Murrillo	14
Jose Pablo Quirós Hidalgo	16
Revisión de la gramática	18

Lecciones Aprendidas

Jonathan Quesada Salas

Con lo que respecta a las lecciones aprendidas por mi parte se focalizan en un principio en cuanto a la secuencia y el orden que tiene el analizador para poder analizar seguidamente cada componente léxico la cual, hubo un problema con el mismo porque el analizador no pasaba por el siguiente componente, se quedaba estancado en el primer componente, por lo cual como grupo se ideó la idea de plantear una función que se encargue de pasar por el siguiente componente y dicha función a continuación:

```
def __pasar_siguiente_componente(self):
    self.posición_componente_actual += 1
    if self.posición_componente_actual >= self.cantidad_componentes:
        return
    self.componente_actual = self.componentes_léxicos[self.posición_componente_actual]
```

En cuanto respecta a el código anterior se llaman en funciones de un nivel determinando de abstracción como “__analizar_valor”, “__analizar_bloque_instrucciones”, “__verificar” y “__verificar_tipo_componente”. Para que de esta manera dichas funciones sean el puente para controlar qué componentes léxicos pueden llegar a pertenecer a una estructura determinada, en el caso de los Betes Ra! serían la asignación, repetición, invocación, función y condicionales.

Adicionalmente, también puedo resaltar que a la hora de analizar un bloque de instrucciones es importante contemplar la totalidad de las palabras reservadas por medio de condicionales en una función y no tratarlos a parte en otras funciones, para que en otras funciones puedan cumplir funciones específicas y puntuales como puede ser funciones como “__analizar_valor” y “__analizar_número”.

```
if self.componente_actual.texto in ["GLOB", "LOC"]:
    self.__pasar_siguiente_componente()
    if self.componente_actual.texto in ["Flotante", "Entero", "Texto"]:
        self.__pasar_siguiente_componente()
```

self.__analizar_asignación()

Alberto Zumbado Abarca

Con lo que respecta a las lecciones aprendidas, me parece importante rescatar el hecho de pensar en distintas formas de poder realizar el análisis por descenso recursivo en nuestro lenguaje y poder utilizarlo a favor de que analice correctamente la estructura sintáctica de nuestro lenguaje. Cabe resaltar que en esta etapa quedaron más claros como es que se realizan esas verificaciones a la sintaxis, ya que en la etapa anterior pensaba en consideradas veces intentar resolver errores de sintaxis cuando no correspondía en esa etapa, por lo que aprendí de mejor forma a cómo encontrar errores sintácticos cuando se tiene una gramática definida.

De igual forma, rescato el hecho de haber entendido mejor como funcionaba la clase "ComponenteLéxico", ya que antes simplemente se generaba, pero ya poder utilizarla y entenderla más a fondo para poder realizar las validaciones respectivas fue muy interesante.

Para resaltar la forma en que se dio el aprendizaje de estos puntos respectivos, para el como poder implementar el análisis, aprendí como usar un diccionario en python para poder mapear los métodos al tipo o al texto del componente, y así no tener que preguntar por cada elemento, si no que nada más si estaba en la estructura, obtener el valor que en este caso correspondía a una función. Me parece pertinente destacar el aprendizaje de esa funcionalidad de python de poder usar una función como valor de un diccionario.

El aprendizaje para el manejo sintáctico se vio revisando constantemente la gramática, y así intentar replicar la gramática a la función que analizaba ese bloque en específico; eso juntandolo con el aprendizaje del funcionamiento de la clase mencionada anteriormente, el cual se dió haciendo las comparaciones respectivas con las condiciones entre el tipo y el texto del componente con lo que se supone que va en la gramática, por lo que se dio el aprendizaje de ambos puntos.

Y para finalizar, se dio el aprendizaje el ver la forma de programar otro tipo de programas, que a pesar de que son muy lineales, suelen complicarse si no se tiene cuidado.

Pablo Alberto Muñoz Hidalgo

En este proyecto aprendí que cada lo complicado que es analizar paso por paso un lenguaje de programación y su funcionamiento, por ejemplo, cuando se analiza una asignación puede ser muy parecido a cuando se analiza un identificador o llamado a variable normal, todo esto en nuestro caso por lo que otra lección que aprendí es que todo se debe ordenar de manera lógica y seguir un proceso determinado para poder analizar con éxito una parte de código.

Donde más aprendí y con diferencia fue en el apartado de función ya que en nuestro caso se nos complicó un poco como grupo ya que nuestra gramática permite realizar acciones en funciones como ciclos, otras funciones e incluso operaciones matemáticas lo que aumenta considerablemente la complejidad de análisis de estas, aprendí que una función es un mini-programa dentro del programa grande por lo que es como si todo se analiza desde cero.

También aprendí mucho con la categorización de variables en el código, a esto me refiero a que hay que contemplar si una variable es entero, flotante o texto por ejemplo, todo esto lleva un análisis por lo que se vuelve un sistema de alta complejidad, en nuestro caso aprendimos también que la modularización y el orden son fundamentales, sin esto no lo hubiéramos logrado ya que todo debe estar ordenado en caso de errores saber en dónde está el error.

Adicionalmente aprendí a que un analizador es un proceso es como un camino que va por pasos, en este caso con muchos condicionales que verifican que una situación no sea otra, hay que tener muy claro el proceso o flujo que sigue un analizador antes de modificarlo ya que, como fue en nuestro caso, tuvimos problemas con la función ya que no habíamos considerado hasta el final las operaciones matemáticas, las cuales pueden ir incluso en una asignación por lo que tener bien claro el flujo de información o de análisis es fundamental para no tener problemas en un futuro.

Por último y para no hacerlo demasiado largo, aprendí que no se pueden contemplar todas las posibilidades y que siempre se debe estar modificando la gramática y los demás componentes, incluido el analizador ya que siempre surgen necesidades nuevas, problemas nuevos que deben ser solucionados.

Luis Andrés Rojas Murrillo

Creo que en este proyecto es importante destacar que parte de las lecciones más importantes que adquirí es el conocimiento de estos protocolos vitales para el funcionamiento de todas las herramientas que se utilizan en la actualidad y que comúnmente no se conocen o no se les presta la atención que merecen y ni se le dan los méritos a sus creadores.

También se me presentó la oportunidad de interiorizar el porqué de muchos hechos que damos por sentados en la programación como lo es el caso de los tipos que, sabemos que debemos de respetar pero no conocíamos el trasfondo del asunto, toda su categorización y procesamiento.

Una de las problemáticas en las que caímos es que al igual que en la entrega anterior existían elementos de la gramática que no estaban planteados desde ese enfoque funcional que necesitamos para que el analizador funciones de manera adecuada y tuvimos que aplicar varias correcciones que nos ayudaron este aspecto. Y por último saliendo un poco de la temática habitual del curso, en mi caso creo que al igual que en cualquier proyecto de carácter evolutivo o en donde se va agregando mas código y va creciendo el proyecto, es importante mantener prácticas que mejoren la legibilidad y mantenibilidad del código para futuras modificaciones además del trabajo conjunto por parte del equipo y la toma de decisiones sobre los puntos estratégicos que hay que mejorar.

Jose Pablo Quirós Hidalgo

De este proyecto quiero rescatar varios puntos que me parecieron interesantes y varios de los que tuve un aprendizaje importante.

Lo primero creo que fue notar errores cometidos en la gramática, y no errores que no tenían sentido, si no cosas que parecían bien planteadas hasta el explorador,

pero a la hora de ir realizando el análisis vimos que habían varios cambios que ir realizando en la gramática para que lo realizado en el analizador tuviera más sentido. También fue muy interesante saber que lenguajes que utilizamos pueden funcionar de manera similar o al menos pasaron por esta etapa también. Como funciona analizar cada palabra o carácter en cada línea, el cómo va comparando uno por uno con lo que según nuestra gramática es el carácter o palabra esperad@. Y me gustaría recalcar el excelente trabajo en equipo que hubo durante este proyecto, donde conversando entre todos fuimos encontrando errores en el explorador y fuimos haciendo en equipo los cambios necesarios, de la misma manera que construimos el analizador, donde nos dividimos ciertas tareas para empezar y fuimos conversando como equipo cada cambio que hacíamos y entre todos compartimos ideas para hacer el analizador de manera que todos estuviéramos de acuerdo.

Salud mental/ Amor propio/ Diversión/ Crecimiento Personal

Jonathan Quesada Salas







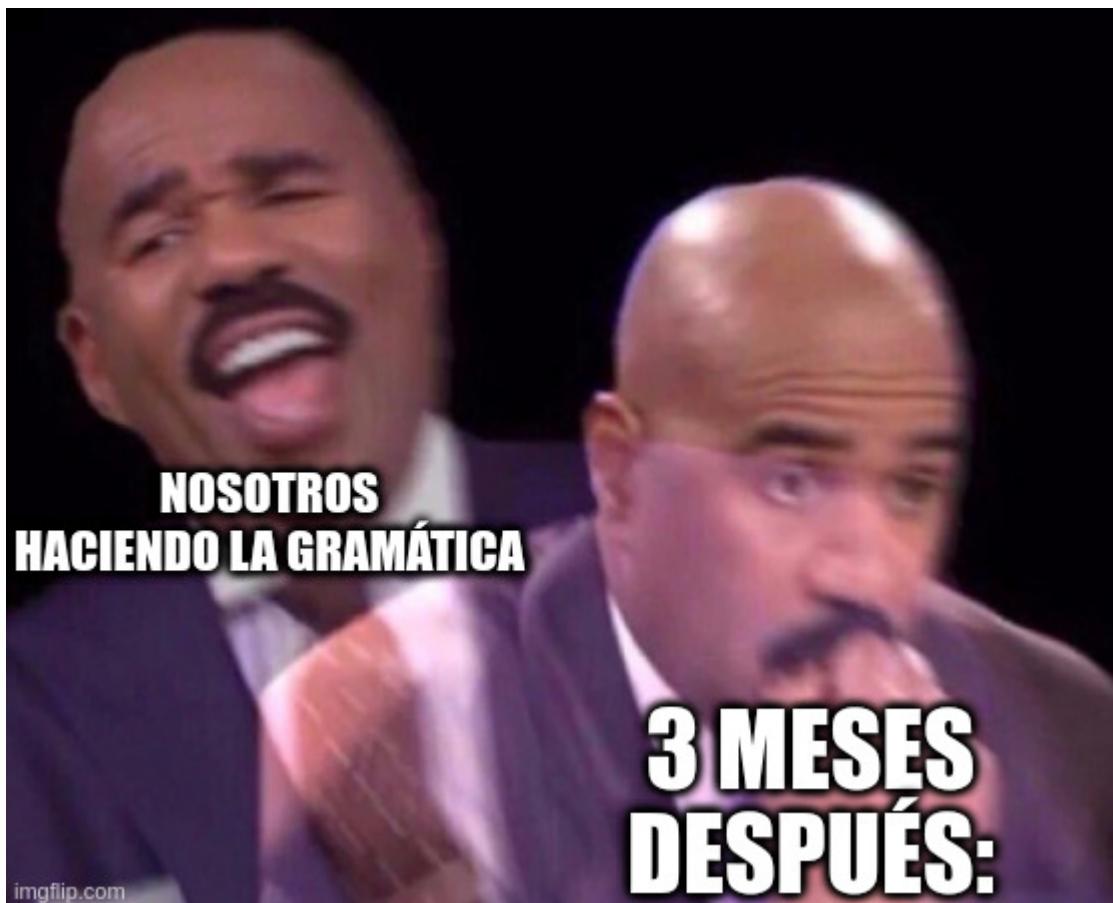
Alberto Zumbado Abarca



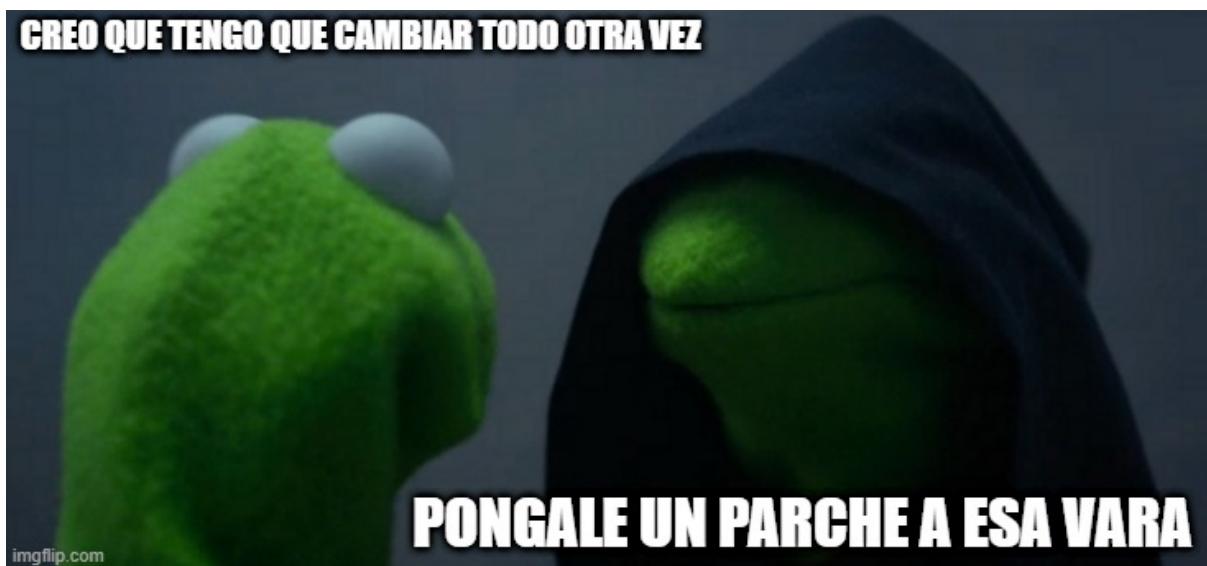
**NOSTROS HACIENDO LA GRAMÁTICA
CON ERRORES A FUTURO PARA EL EXPLORADOR**

AURELIO:

imgflip.com



Pablo Alberto Muñoz Hidalgo



Logramos analizar una función

Logramos analizar ***solo*** una función



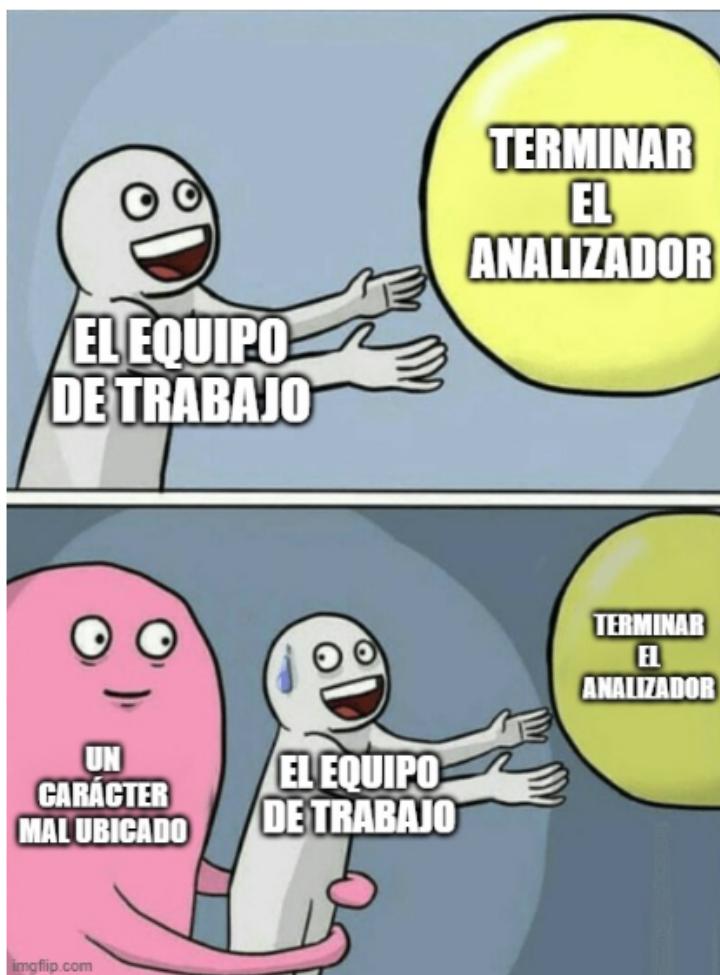
Como se veía la gramática al principio

Como se ve ahora
haciendo el analizador





Luis Andrés Rojas Murrillo



Me dijiste que generar
el analizador era sencillo

Si, la gramática
estaba bien estructurada



Jose Pablo Quirós Hidalgo





Revisión de la gramática

- El grupo consideró que era más factible modificar el formato de la estructura de las funciones, repeticiones y condicionales, de forma que empiecen por su palabra reservada establecida y seguidamente por los parámetros dentro de llaves para así representar aún mejor el enfoque hacia los conjuntos. Y para representar el contenido de las mismas se establece con las siguientes flechas, al inicio -> y para el final <- , el cual anteriormente existía la presencia de un símbolo de dólar en la gramática.

Por ejemplo:

Func nombreFunción {parametro1} ->

 #C Bloque de instrucciones

 <-

Rep {ExpCondicional} ->

 #C Bloque de instrucciones

 <-

Si {Comparación} ->

 #C Bloque de instrucciones

 <-

Se consideró de esta forma ya que al estar programando la lista de componentes léxicos y viendo el analizador vimos como ya el verificar los componentes léxicos es más sencillo de lo que se esperaba, por lo que se eliminaron esos componentes que consideramos extra.

- También como grupo se estableció que el signo de # antes de las palabras reservadas como LOC, GLOB, CONS, Texto, Flotante, Entero, era innecesario por lo cual se consideró eliminar eso de la gramática.
- De igual forma, otro de los cambios realizados fue en la regla Operación. La gramática antes del cambio estaba representada de la siguiente manera:

Asignación ::= TipoVariable TipoValor Identificador (<=>) (Operación | Valor);

Operación ::= Valor (Operadores Valor)+

Valor ::= (Identificador | Literal)

El problema consistía en que la asignación luego del componente que funciona como "igual", puede recibir un valor o una operación. Valor puede ser un identificador o literal, mientras que operación de igual forma comienza con un identificador o literal, por lo que para no tener que buscar el componente venidero para así hacer esa diferenciación se decidió añadir al inicio de una operación la palabra reservada Operar, para poder diferenciar entre ambas reglas. Ahora, cada vez que se quiera hacer una Operación se deberá comenzar con un Operar.

Asignación ::= TipoVariable TipoValor Identificador (<=>) (Operación | Valor);

Operación ::= Operar Valor (Operadores Valor)+

Valor ::= (Identificador | Literal)

Los mismos cambios se realizaron para regla de invocación de una función, ya que esta comenzaba con un identificador y podría generar problemas con la regla de operar variable.

A nivel de implementación se realizaron los respectivos cambios en el explorador para comprender los nuevos componentes léxicos de la gramática, y de igual forma se hicieron otros cambios para poder generar unos mejores errores en la etapa del analizador.