



DEPARTAMENTO DE SISTEMAS COMPUTACIONALES E INFORMÁTICA

ASUNTO: SOLICITUD DE ACTIVIDADES

Celaya, Guanajuato, 12 / octubre / 2022

LENGUAJES Y AUTÓMATAS II
DOCENTE DESIGNADO: ISC. RICARDO GONZÁLEZ GONZÁLEZ
SEMESTRE AGOSTO-DICIEMBRE 2022

ACTIVIDAD 6 (VALOR 44 PUNTOS)

LEA CUIDADOSAMENTE, Y REALICE LAS SIGUIENTE ACTIVIDADES, CONSIDERANDO LOS CRITERIOS DE CALIDAD PROPUESTOS EN LOS DOCUMENTOS DE LA [GUÍA TUTORIAL](#), Y LA [RÚBRICA DE EVALUACIÓN](#),

EL LECTOR DEBE TOMAR MUY EN CUENTA QUE ESTA ACTIVIDAD ES UN EXAMEN, Y NO UNA SIMPLE TAREA, PUES DEMANDA DEDICACIÓN PARA INVESTIGAR, LEER, ANALIZAR, REDACTAR, ILUSTRAR Y PROponER DE MANERA PROFESIONAL LOS TEMAS PROPUESTOS EN LA ESTRUCTURA TEMÁTICA DE ESTA ASIGNATURA.

4. ANÁLISIS SEMÁNTICO.

INVESTIGUE, LEA, COMPREnda Y ELABORE UNA **MONOGRAFÍA TÉCNICA** COMPLETAMENTE APEGADA A LO SOLICITADO EN LA [GUÍA TUTORIAL](#) (PUNTO 3, INCISO a) ACERCA DE LOS SIGUIENTES TEMAS :

- TEMA 4.1 ÁRBOLES DE EXPRESIONES.
- TEMA 4.2 ACCIONES SEMÁNTICAS DE UN ANALIZADOR SINTÁCTICO.
- TEMA 4.3 COMPROBACIONES DE TIPOS EN EXPRESIONES.
- TEMA 4.4 PILA SEMÁNTICA EN UN ANALIZADOR SINTÁCTICO.
- TEMA 4.5 ESQUEMA DE TRADUCCIÓN.
- TEMA 4.6 GENERACIÓN DE LA TABLA DE SÍMBOLOS Y DE DIRECCIONES.
- TEMA 4.7 MANEJO DE ERRORES SEMÁNTICOS.

CONSIDERACIÓN :

DEBE USTED ENTENDER EL VALOR QUE TIENE ESTA ACTIVIDAD Y QUE LOS TEMAS ANTES REFERIDOS, PARA NADA DEBEN SER ABORDADOS COMO SIMPLES CONCEPTOS REDACTADOS CON LA LIGEREZA, PUES ESTA ACTIVIDAD ESTÁ CONSIDERADA COMO UN EXAMEN.





EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO[®]

Instituto Tecnológico de Celaya
Departamento de Sistemas y Computación

ANALICE CADA TEMA, SUS CARACTERÍSTICAS, SU IMPORTANCIA, SUS CONCEPTOS, SUS EJEMPLOS, SUS ILUSTRACIONES, Y LOS TIPOS DE EVIDENCIAS QUE USARÁ PARA DEMOSTRAR QUE USTED HA ADQUIRIDO UN VERDADERO CONOCIMIENTO ACERCA DE ÉSTOS.

A MODO DE PRÁCTICAS REALICE ESTE PUNTO Y ELABORE EJERCICIOS NECESARIOS CON LOS CUÁLES USTED DEMUESTRE

- ELABORE DOS VIDEOS (NO MÁS DE 25 MINUTOS) DISTRIBUIDOS DE LA SIGUIENTE FORMA Y EN LOS QUE EXPONGA SUS CONOCIMIENTOS ADQUIRIDOS. DESPUÉS COLOQUE SUS MATERIALES EN YOUTUBE E INCLUYA LAS LIGAS EN SU EXAMEN.

VIDEO 1 : TEMAS 4.1, 4.2, 4.3, 4.4

VIDEO 2: TEMAS 4.5, 4.6, 4.7

MUY IMPORTANTE: SI ESTA ACTIVIDAD ES ENTREGADA EN EQUIPO, CADA UNO DE LOS INTEGRANTES DE ÉSTE DEBEN PARTICIPAR EN CADA VIDEO, EXponiendo JUNTO A SUS COMPAÑEROS CADA TEMA SOLICITADO.

POR FAVOR NO USE APUNTADORES O MATERIALES DE APOYO TAN SOLO LEER LOS CONCEPTOS. LA IMPORTANCIA Y EL VALOR DE LOS VIDEOS RADICA EN EXPRESAR Y EVALUAR CORRECTAMENTE SU CONOCIMIENTO EN ESTOS TEMAS.

IMPORTANTE: SI LO REQUIERE PUEDE CONSULTAR EL [SIGUIENTE DOCUMENTO](#) PARA ORIENTAR SU TRABAJO EN CONOCER QUÉ ES Y CÓMO HACER UNA MONOGRAFÍA CON EL RIGOR ACADÉMICO REQUERIDO.

POR ÚLTIMO, RECUERDE LEER LA [GUÍA TUTORIAL](#) PARA EL CORRECTO TRATAMIENTO DE ESTE INCISO.

¿ QUÉ SE CALIFICARÁ ?

LA RÚBRICA PARA EVALUAR ESTA ACTIVIDAD ESTARÁ INTEGRADA POR LOS SIGUIENTES CRITERIOS.

- LA OPORTUNIDAD. SI EL TRABAJO FUE ENTREGADO OPORTUNAMENTE.
- LA COMPRENSIÓN. SE VALORARÁ EL GRADO DE COMPRENSIÓN DEL TEMAS ANALIZADOS.
- LA CALIDAD. SI LAS EVIDENCIAS ENVIADAS CORRESPONDEN A LA CALIDAD ESPERADA PARA ESTE NIVEL PROFESIONAL QUE SE CURSA.
- LA CAPACIDAD DE SÍNTESIS. SI LAS EVIDENCIAS ENTREGADAS TIENEN EL NIVEL DE DETALLE Y PROFUNDIDAD REQUERIDA, O EN BIEN SI SE OMITIERON CONCEPTOS CON EL AFÁN DE SIMPLIFICAR Y ENTREGAR UN MATERIAL ACADÉMICA Y TÉCNICAMENTE POBRE.



Av. Antonio García Cubas #600 esq. Av. Tecnológico,
Colonia Alfredo V. Bonfil, C.P. 38010 Celaya, Gto.
Tel. 01 (461) 611 75 75
e-mail: lince@celaya.tecnm.mx
tecnm.mx | celaya.tecnm.mx





EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO[®]

Instituto Tecnológico de Celaya
Departamento de Sistemas y Computación

- e. LA CREATIVIDAD. LA MANERA EN QUE SE EXPRESAN LOS CONCEPTOS Y EL TRATAMIENTO QUE SE DA A LA INFORMACIÓN ANALIZADA PARA QUE ÉSTA SEA COMPRESIBLE EN SU ESENCIA.

IMPORTANTE : CUENTA CON EL TIEMPO SUFICIENTE PARA REALIZAR ESTA ACTIVIDAD Y SUMAR PUNTOS IMPORTANTES A SU CALIFICACIÓN DE ESTA EVALUACIÓN.

IMPORTANTE: TODO EL MATERIAL ESCRITO DEBERÁ SER HECHO A MANO.



Av. Antonio García Cubas #600 esq. Av. Tecnológico,
Colonia Alfredo V. Bonfil, C.P. 38010 Celaya, Gto.
Tel. 01 (461) 611 75 75
e-mail: lince@celaya.tecnm.mx
tecnm.mx | celaya.tecnm.mx





CONSIDERACIONES.

CADA UNO DE LOS PUNTOS ANTERIORES DEBE SER DESARROLLADO CON LA PROFUNDIDAD ACORDE A UN NIVEL PROFESIONAL, Y APEGÁNDOSE COMPLETAMENTE A LAS DIRECTRICES DE LA GUÍA TUTORIAL.

NO CONCIBA ESTE TRABAJO, COMO UN SIMPLE RESUMEN O EJERCICIO DE TRANSCRIPCIÓN, PUES EL VALOR INDICADO AL INICIO DE ESTA ACTIVIDAD LE DARÁ A USTED UNA BUENA IDEA DE LO QUE SE ESPERA DE ELLA, EN CUANTO A CALIDAD Y EL APRENDIZAJE OBTENIDO, MISMO QUE SERÁ PUESTO A PRUEBA MEDIANTE UN EXAMEN ESCRITO O BIEN ORAL EN CLASE.

SI DECIDIÓ ELABORAR ESTA ACTIVIDAD EN EQUIPO, CADA INTEGRANTE DE ÉSTE DEBERÁ POSEER EL MISMO NIVEL DE CONOCIMIENTO, PUES TAN SOLO REPARTIR TEMAS ENTRE LOS INTEGRANTES DEL EQUIPO, SUPONDRIÁ UN GRAVE ERROR DE INTERPRETACIÓN A LA INTENCIÓN DIDÁCTICA REAL DE ESTA ACTIVIDAD.

POR ÚLTIMO, ESTA ACTIVIDAD SOLO SE PODRÁ DESARROLLAR EN EQUIPO, SI SE REGISTRÓ EN UNO PREVIAMENTE, UTILIZANDO EL FORMATO ENTREGADO EN LA ACTIVIDAD INICIAL. DE LO CONTRARIO DEBERÁ ELABORAR Y ENTREGAR LA ACTIVIDAD DE FORMA INDIVIDUAL.

LA ENTREGA DE DICHO REGISTRO SE HARÁ VÍA CORREO ELECTRÓNICO ENVIANDO ÉSTE AL PROFESOR DESIGNADO, Y POSTERIORMENTE EN CLASE ENTREGANDO LA HOJA EN FÍSICO.

OBSERVACIONES:

- CADA HOJA QUE ENTREGUE DE SU ACTIVIDAD, DEBERÁ ESTAR FIRMADA AL MARGEN DERECHO, INCLUIDA LA PROPIA SOLICITUD DE LA ACTIVIDAD.
- **INTEGRE TODO SU TRABAJO EN UN SOLO ARCHIVO DE TIPO .PDF, Y ASIGNE EL NOMBRE QUE A CONTINUACIÓN SE INDICA.**

NO OLVIDE ANEXAR LAS HOJAS DE ESTA ACTIVIDAD Y DE SU TRABAJO DESPUÉS DE SU PORTADA.

- UNA VEZ ELABORADA SU ACTIVIDAD, RECUERDE DIGITALIZARLA Y NOMBRARLA EN BASE A LA NOMENCLATURA QUE SE INDICA MÁS ADELANTE EN ESTE DOCUMENTO.
- SI SUS EVIDENCIAS ENVIADAS POR CORREO, NO CUMPLEN CON LA NOMENCLATURA SOLICITADA, NO SERÁN CONSIDERADAS COMO EVIDENCIAS PARA SU EVALUACIÓN.
- **POR ÚLTIMO, POR FAVOR GESTIONE APROPIADAMENTE SU TIEMPO, Y SEA PUNTUAL EN SU ENTREGA Y ASÍ EVITAR PROBLEMAS DE NULIDAD POR EXTEMPORANEIDAD.**





EDUCACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO[®]

Instituto Tecnológico de Celaya
Departamento de Sistemas y Computación

LA NOMENCLATURA SOLICITADA PARA ENVIAR SU TRABAJO ES LA SIGUIENTE :

AAAA-MM-DD_TNM_CELAYA_MATERIA_DOCUMENTO_[EQUIPO]_NOCTROL_APELLIDOS_NOMBRE_SEM.PDF

(NOTA : *** TODO DEBE SER ESCRITO USANDO LETRAS MAYÚSCULAS ***)

DONDE :

TNM_CELAYA	: INSTITUCIÓN ACADÉMICA
AAAA	: AÑO
MM	: MES
DD	: DÍA
MATERIA	: LAI , LI MÁS EL GRUPO (-A , -B, -C)
DOCUMENTO	: A1-ACTIVIDAD 1, P1-PRACTICA 1, R1-REPORTE 1, TI-TAREA 1, PG1-PROGRAMA, ETC. (CAMBIANDO EL NÚMERO CONSECUATIVO POR EL QUE CORRESPONDA)
[EQUIPO]	: NÚMERO DEL EQUIPO QUE CORRESPONDA SEGÚN INDICACIÓN DEL PROFESOR. [OPCIONAL]
NOCTROL	: SU NÚMERO DE CONTROL
APELLIDOS	: SUS APELLIDOS
NOMBRE	: SU NOMBRE
SEM	: EL PERIODO SEMESTRAL EN CURSO: AGO-DIC

EJEMPLO :

SI EL TRABAJO SE SOLICITÓ EN EQUIPO.

2022-10-12_TNM_CELAYA_LAI-A_A6_EQUIPO_99_9999999_PEREZ_PEREZ_JUAN_AGO-DIC22.PDF

DONDE EL NOMBRE DEBERÁ CORRESPONDER AL JEFE DE EQUIPO QUE HACE LA ENTREGA DEL TRABAJO.

SI EL TRABAJO SE SOLICITÓ INDIVIDUALMENTE.

2022-10-12_TNM_CELAYA_LAI-A_A6_9999999_PEREZ_PEREZ_JUAN_AGO-DIC22.PDF



Av. Antonio García Cubas #600 esq. Av. Tecnológico,
Colonia Alfredo V. Bonfil, C.P. 38010 Celaya, Gto.
Tel. 01 (461) 611 75 75
e-mail: lince@celaya.tecnm.mx
tecnm.mx | celaya.tecnm.mx





FECHA Y HORA DE ENTREGA:

LA INDICADA EN LA PLATAFORMA VIRTUAL.

EN CASO DE QUE EL TRABAJO SE HAYA SOLICITADO EN EQUIPO, EL JEFE DEL MISMO SERÁ EL ÚNICO RESPONSABLE DE ENVIAR LA ACTIVIDAD EN LA PLATAFORMA VIRTUAL.

MUY IMPORTANTE:

1. DESPUÉS DE LA HORA INDICADA EN LA PLATAFORMA VIRTUAL (AÚN CUANDO SOLO SEA UN MINUTO O VARIOS), LA ACTIVIDAD SERÁ CONSIDERADA COMO EXTEMPORÁNEA Y NO CONTARÁ COMO EVIDENCIA PARA SU EVALUACIÓN.

SE LE SUGIERE ENVIAR CON ANTICIACIÓN SU ACTIVIDAD A FIN DE EVITAR CONFLICTOS POR NO ENTREGAR ÉSTA A TIEMPO.

BAJO NINGÚN PRETEXTO O JUSTIFICACIÓN SE ACEPTARÁN LOS TRABAJOS EXTEMPORÁNEOS, EVITE LA PENA DE RECORDAR A USTED QUE EL VALOR DE LA PUNTUALIDAD ES PARTE IMPORTANTE DE SUS EVIDENCIAS Y ES EL PRIMER PUNTO QUE SE HA DE EVALUAR.

2. NO OLVIDE ANEXAR A SU ARCHIVO .PDF DE EVIDENCIAS UNA PORTADA PROFESIONAL, Y ESTA SOLICITUD DE ACTIVIDADES CON TODAS LAS HOJAS FIRMADAS EN EL MARGEN DERECHO.
3. POR ÚLTIMO, TODA EVIDENCIA GENERADA QUE CONTENGA AL MENOS UNA TRANSCRIPCIÓN DE CUALQUIER FUENTE Y DE CUALQUIER TIPO, ES DECIR CON MATERIAL PLAGIADO SERÁ ANULADA DE FORMA INCONTROVERTIBLE.



~~scribo~~

TECNOLOGICO NACIONAL DE MEXICO

~~scribo~~

EN CELAYA

LENGUAJES Y AUTOMATAS I

I.S.C Ricardo González González Grupo: A

Equipo 3:

- García Ramírez Luis David (19030263)
- Pérez Cabrera José Eduardo (19030985)
- Ramírez García Marco Isaias (19030260)

Presentan ACTIVIDAD 7.
ANALIZADOR SEMÁNTICO.

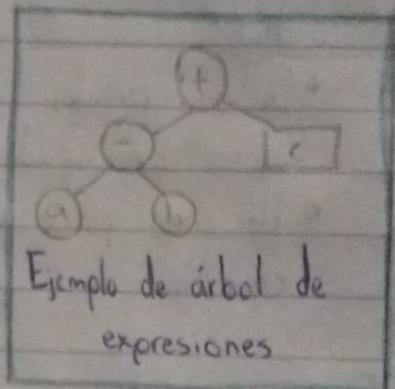
A 20 de Octubre de 2022.

Introducción

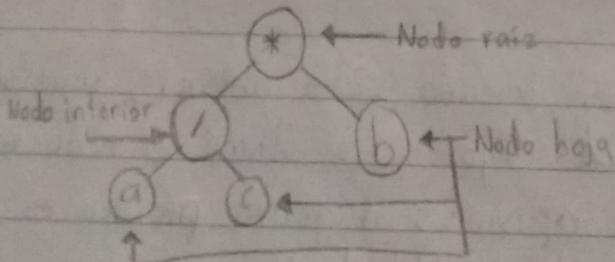
En el presente trabajo se abordarán temas de importancia para comprender mejor el funcionamiento del análisis semántico. Veímos como es que trabaja el analizador semántico con ayuda de los árboles de expresiones, que como su nombre lo dice nos ayudará a representar las expresiones en un árbol así como comprobar que tipo de expresiones estamos hablando. Abordaremos las estructuras de datos que son usadas para auxiliar al analizador semántico como lo son la pila semántica y la tabla de símbolos y direcciones, que estas últimas son de mucha ayuda para realizar las comprobaciones semánticas. Los árboles y gramáticas se complementan entre si, ya que una gramática puede ser representada por un árbol de allí nacen los esquemas de traducción, que como hemos visto es una gramática que se presenta en forma de árbol. Todos estos temas antes mencionados son importantes de abordar para conocer de mejor manera como es que trabajan los analizadores semánticos.

4.1. Árboles de expresiones

En la fase del análisis semántico existe una estructura llamada "árbol de expresiones", estos árboles representan el código en forma de datos. La forma de esta estructura, como su nombre lo dice, es en forma de un árbol invertido, está conformado por nodos y cada nodo representa una expresión del código, por ejemplo, los símbolos aritméticos o los símbolos de igualdad, entre otros.



El árbol de expresiones cuenta con 3 tipos de nodos que se utilizan para su construcción, el nodo raíz, el cuál es el nodo principal donde empieza la construcción del árbol, los nodos interiores, los cuales son los nodos que están al interior de nodo y conectan con otros nodos pero nunca son los finales, y por último, los nodos hojas, los cuales son aquellos nodos que son los últimos de cada rama. Ejemplo de los tipos de nodos:



El árbol cuenta con 2 características importantes que se deben de tener siempre en cuenta a la hora de su construcción:

- Todas las hojas están etiquetadas con un solo operando.
- Todas las nodos interiores están etiquetados por un operador.

El árbol también cuenta con otras características que deben ser tomadas en cuenta, estas reglas pueden parecer muy fáciles de implementar pero

~~Matemática~~
hoy que tener cuidado en implementarlas correctamente porque pueden hacer que nuestro árbol este incorrecto desde un principio. Las características son:

- La raíz siempre debe ser un operador.
- Las hojas siempre son operandos.
- Si un operador tiene mayor prioridad que la raíz, entonces se coloca como hijo.
- Si un operador tiene igual o menor prioridad que un nodo, entonces se coloca como parente.
- Un nodo puede contener como hijo otro subárbol que contiene una pequeña expresión.

Existen recorridos para los árboles de expresiones, los cuales se diferencian en la manera en que se obtiene la expresión y como es que estas expresiones se leen. Son 3 recorridos diferentes: preorden, postorden y inorden, el recorrido a elegir es a elección del usuario y depende del objetivo que este el usuario buscando para la utilización del recorrido.

Recorrido postorden.

El recorrido postorden o también llamado "orden posterior", consiste en recorrer primero cada uno de los nodos hijos y por último el nodo raíz. El orden como se recorre el árbol de expresiones de forma postorden es de la siguiente manera: subárbol izquierdo, subárbol derecho y nodo raíz.

Recorrido inorden.

El recorrido inorden consiste en recorrer desde el nodo que se encuentra más a la izquierda de todos, después sigue con el nodo raíz, y por último, termina con los nodos del lado derecho. El orden como se recorre el árbol de expresiones de forma inorden es: subárbol izquierdo, nodo raíz y

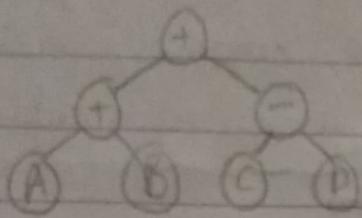
Manzana

subárbol derecho.

Recorrido preorden.

El recorrido preorden consiste en primero recorrer el nodo raíz, luego el subárbol izquierdo y por último el árbol derecho. El orden de como se recorre es: nodo raíz, subárbol izquierdo y subárbol derecho.

Ejemplo de recorridos.



Inorden: (A+B)+(C+D)

Preorden: + + A B - C D

Postorden: A B + C D + -

En pocas palabras, los árboles de expresiones sirven para ver la construcción de las diferentes expresiones que se pueden presentar en el código, se utilizan para comprobar que las expresiones estén construidas correctamente.

4.2. Acciones semánticas de un analizador sintáctico.

Las acciones semánticas que realiza un analizador sintáctico en el momento que realiza su fase de análisis son diversas, el principal objetivo de algunas de estas acciones es encontrar errores en la gramática dada para el código.

La primera acción semántica que realiza un analizador sintáctico es checar el código de entrada en base a la gramática establecida, en caso de que el analizador determine que el código de entrada es válido, entonces suministra un árbol sintáctico que lo reconoce, el cual es capaz de ser reconocido por los tokens del analizador léxico.

El analizador sintáctico también accede a la tabla de símbolos, esto para poder realizar el trabajo de la fase de análisis sintáctico, el cual tiene que ir a la tabla de símbolos para ir llevando un registro de los símbolos que van apareciendo en el código de entrada, y así guardar información para su misma fase de análisis o para agudar a las siguientes fases de análisis.

El analizador sintáctico también es responsable de checar los tipos de los datos, ya que debe verificar que por ejemplo, una cadena no quiera ser introducida a una variable tipo entera, o también del que el tipo que quiera ser declarada la variable sea un tipo de dato aceptado por el compilador.

También debe de tener la opción de generar código intermedio, esto para que ayude al usuario a crear un código más rápido y sin errores, y así ahorrar más trabajo al momento del análisis, ya que aseguramos que parte del código es correcta.

Tabla de símbolos:

Es una estructura que usa un compilador para manipular los identificadores que aparecen en un código fuente.

Manejo de errores

Una acción semántica muy importante con la cuál el analizador sintáctico debe de contar es que debe de ser capaz de gestionar los errores cuando estos sean detectados, para que el usuario sea capaz de visualizar que su código de entrada tiene errores y obviamente debe corregirlos, los mensajes de error que se le manden al usuario deben contener la suficiente información para que el usuario pueda detectar el error y solucionarlo lo más rápido que se pueda.

En pocas palabras, el analizador sintáctico realiza casi todas las operaciones de compilación, ayudándose paralelamente con el analizador léxico, la tabla de símbolos y el manejador de errores para asegurar que el código de entrada pase al siguiente analizador sin ningún tipo de error.

Analizador sintáctico:
Es un programa que analiza una cadena de entrada según las reglas de la gramática dada.

Manejo de errores sintácticos

El manejo de errores sintácticos es la parte más elaborada de un compilador, porque además de los procedimientos para detectar errores, también nos debe de informar de todos los errores que se hayan recuperado y no solo parar cuando se encuentre uno. Los objetivos de un manejador de errores sintácticos son los siguientes:

- Debe indicar de forma clara y precisa el error y su localización.
- Recuperarse del error para seguir examinando el código de entrada.
- Por ningún motivo debe ralentizar significativamente la compilación.

El manejo de errores es una parte en la cuál debemos cuidar hasta el más mínimo detalle, ya que es la parte la cuál nos garantizara un código

Tipo de gramática para un analizador sintáctico.

La gramática de la cual se debe basar un analizador sintáctico es la gramática libre de contexto, ya que es mucho más fácil de comprender y nos facilita la corrección de errores, además de evitar la ambigüedad para así localizar los errores con mayor precisión.

La gramática libre de contexto es la siguiente:

(Gramática: $G(N, T, P, S)$).

N = No terminales.

T = Terminales.

P = Reglas de producción.

S = Axioma inicial.

Gramática libre de contexto: Es la gramática en donde cada regla de producción es de la forma: $V \rightarrow N$.

Comprobación de tipos de expresiones

Para este tema es de suma importancia conocer a profundidad los tres procesos de análisis que hace un compilador para generar código fuente. Se hace uso extensivo sobre los temas singulares de tipos de datos y de expresiones, el compilador al tratar de ejecutar un código debe ser capaz de identificar el

tipo de operación que se hace en el proceso y validar que los tipos de datos que se usan son compatibles entre ellos, para realizar correctamente

las evaluaciones se deben evaluar las expresiones de tipos, las tablas de simbolos son de mucha ayuda ya que si el lenguaje evaluado es fuertemente tipificado, será fácil acceder a la información con los propiedades de los operandos involucrados y evaluar las expresiones con sus respectivas gramáticas.

String x = "Hola";
int y = 25;
int z = x + y;

Análisis Tipos

- | | |
|---|---------------|
| ✓ | A. Léxico |
| ✓ | A. Sintáctico |
| ✗ | A. Semántico |

~~11/04/2022~~
Las comprobaciones de tipos de expresiones forman parte de los últimos dos procesos de análisis los cuales son el sintáctico y semántico.

String[] array = {"Hola", "Mundo"}
✓ ✓ ✓ ✓ ✓

String dato = array["Hola"];
✓ ✓ ✓

Tipo

Detectar este tipo de caso a veces puede resultar algo complejo si se tiene en cuenta que Segun la construcción de la gramática para validar la declaración es valida pero el tipado no corresponde a lo establecido por la construcción de como se accede a un arreglo.

Dependiendo el lenguaje de programación se pueden complicar más o menos estos análisis ya que hay que recordar que para interpretar las instrucciones de un código fuente para un lenguaje específico, se pueden tener diferentes características y funcionalidades.

~~Maestro~~
También influye el tipo de compilación ya que se pueden compilar o interpretar los códigos fuente ya sea por bloques o por instrucciones.

A este tipo de verificaciones se le llaman estática y dinámica ya que se depende de si se está haciendo el proceso en tiempo de compilación por bloques o interpretación por instrucciones y así segmentar la carga.

Se deberá validar la compatibilidad ya que muchas veces, la cantidad de memoria o los tipos de caracteres que pueden tener los tipos de datos no son compatibles y generan muchos problemas a futuro por la simbología o la cantidad de bytes ocupados.

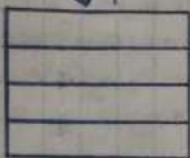
Dentro de las tablas de símbolos se cuentan con punteros hacia otra información, con la finalidad de alinearse los parámetros para evaluar los tipos.

Según la gramática del lenguaje se pueden contar con diferentes tipos de expresiones como pueden ser las expresiones regulares ER.



Pila Semántica en un analizador

Una pila es una estructura de datos con funcionalidades específicas que sirven para almacenar y extraer datos siguiendo las reglas del acrónimo LIFO



LIFO

Last In
First Out

Para entender su funcionamiento de una manera simple se puede hacer una analogía como la es tener un montón de monedas "apiladas" para ingresar uno nuevo se hace colocándola al final de la pila y para recoger la moneda de igual manera se toma la última moneda de la pila.

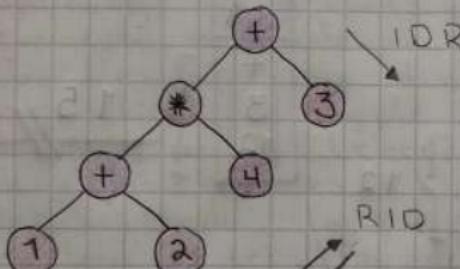
Para cualquier analizador semántico de pila es fundamental conocer los métodos para las operaciones los cuales son PUSH y POP.

El análisis semántico utiliza como base de entrada el desarrollo de un árbol sintáctico para hacer la comprobación estructurada de los aquellos errores que puede presentar.

~~Matemática~~

La finalidad de la pila es permitir la evaluación organizada de los informes con el fin de validar que se hayan construido bien las expresiones teniendo en cuenta la importancia de clasificar los operandos y los operadores para comparar los tipos de datos usados.

Árbol Sintáctico



$((1+2)*4)+3)$



IDR

IDR
+
3
3
*
4
4
+
2
2
1

RID

RID
3
4
2
1
+
*
+
*
+

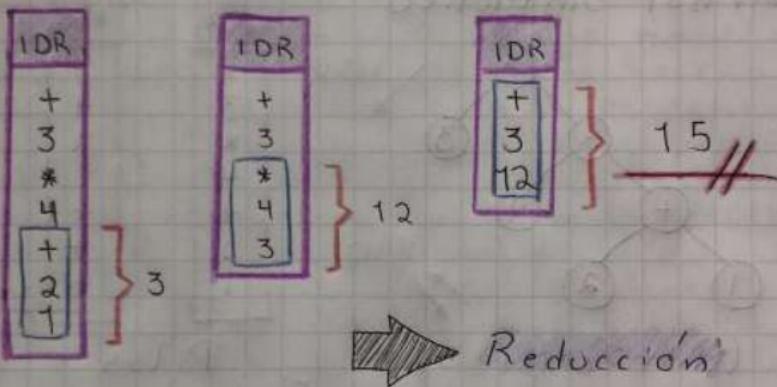
Pilas

Evaluador por
Medio de recorridos

Cuando se tiene una expresión que sigue con una expresión negativa se debe hacer el desarrollo de las producciones de la gramática para ser desarrollada y mientras completa con las producciones y alfabeto no será capaz de detectar errores lógicos.

Para hacer las evaluaciones de las pilas sera necesario de ir aplicando los métodos Push y Pop para la manipulación de los datos, el compilador hará el proceso de evaluaciones con la pila buscando llegar a un solo registro.

Evaluación de la pila tipo IDR



El funcionamiento de la pila consiste en pasar los datos de un árbol sintáctico a una pila semántica, por los métodos de apilación y desapilación se aplicarán las evaluaciones a los operadores y operandos, realizando las operaciones correspondientes y reduciendo la pila con el fin de llegar a un solo registro el cual tendrá como dato el resultado final del conjunto de operaciones.

4.5 ESQUEMA DE TRADUCCIÓN

Un Esquema de traducción se considera como una gramática libre del contexto. en la misma se insertan fragmentos de código en la parte derecha de las reglas de producción. A estas mismas se le llaman como acciones semánticas, a estas se encargan de calcular y modificar los atributos asociados a los nodos de los árboles sintácticos.

El orden con el que evalúan los fragmentos es un recorrido del primero en profundo dentro de los árboles sintácticos.

Primero se tiene que construir el árbol sintáctico para poder aplicar el esquema de traducción, después se aplican las acciones empotradas, siguiendo las reglas en el orden de recorrido primero profundo. Es importante que la gramática no sea ambigua, ya que, esto puede generar que una frase tenga dos o más significados, lo que lleva que halla dos árboles de sintaxis lo cual podría llevar a diferentes resultados.

Para evitar lo último mencionado es necesario que el árbol sintáctico concreto se esté hablando.

Ejemplo: Supongamos que tenemos la siguiente regla
 $A \rightarrow^2 B$.

Después insertamos un fragmento de código
 $A \rightarrow^2 \text{acción3} B$

La acción se ejecutará después de que se hallan terminado el recorrido del árbol A , pero antes de que se ejecute el subárbol B .

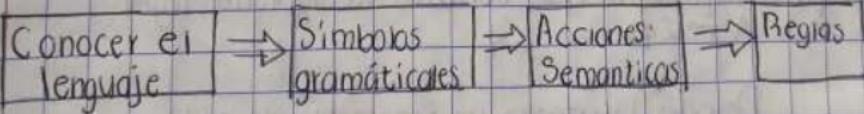
Volvamos el siguiente esquema de traducción recibe como entrada una expresión en infijo y como salida una expresión aritmética en postfijo.

$$\begin{array}{l} \text{Expr} \rightarrow \text{expr1 - NUM} \quad ? \$\text{expr1}\text{TRAZ} = \$\text{expr1}\text{VALZ} \\ \qquad \qquad \qquad " \cdot \text{NUM} \text{VALZ} \cdot " \cdot " \cdot " \\ \text{expr} \rightarrow \text{NUM} \quad ? \$\text{expr}\text{TRAZ} = \$\text{NUM}\text{VALZ} \end{array}$$

Podemos ver que aparecen variables sintácticas en una regla de producción y esta se indexan, con el fin de identificar de qué nodo del análisis se está hablando. La indexación de tipo hash es la que se usa cuando estamos hablando del atributo de un nodo.

En el ejemplo anterior tenemos que VALZ es un atributo de los nodos de tipo NUM, el cual representa el valor numérico, para acceder a este basta con poner \$NUMVALZ. Paralelamente, \$exprTRAZ denota el atributo de traducción de los nodos de tipo expr.

Requisitos para hacer un esquema de traducción.



Es necesario conocer el lenguaje con el que se está trabajando, ya que esto nos va a llevar a no cometer errores como pueden ser las ambigüedades antes mencionadas. A su vez, los símbolos gramáticos son necesarios ya que con estos empezaremos a construir las reglas.

Por último las acciones semánticas nos ayudarán a identificar acerca de lo que estamos hablando dentro del contexto del lenguaje.

Proceso para generar o usar un esquema de traducción

Un esquema de traducción dirigido por Sintaxis se puede implementar con base a un árbol de análisis sintáctico para posteriormente hacer un recorrido en pre-orden del árbol para al mismo tiempo ir ejecutando las acciones semánticas.

- Construir el árbol del análisis sintáctico sin tomar en cuenta acciones semánticas.
- Añadir nuevos hijos a cada símbolo de la parte derecha como nuevos hijos que contienen las acciones semánticas que se deben realizar.
- Realizar el recorrido en preorden del árbol del análisis sintáctico.

Ejemplo de un esquema de traducción.

Gramática

$$L \rightarrow E_n$$

$$E \rightarrow E_1 + T$$

$$E \rightarrow T$$

$$T \rightarrow T_1 \cdot F$$

$$T \rightarrow F$$

$$F \rightarrow (E)$$

$$F \rightarrow \text{digit}$$

$$\{ \text{print}(E.\text{val}), \}$$

$$\{ E.\text{val} = E.\text{val}_1 + T.\text{val}; \}$$

$$\{ E.\text{val} = T.\text{val}; \}$$

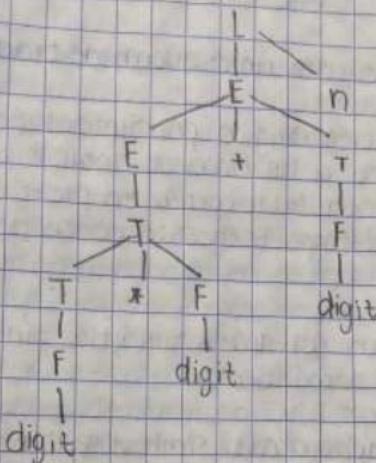
$$\{ T.\text{val} = T_1.\text{val} \times F.\text{val}; \}$$

$$\{ T.\text{val} = F.\text{val}; \}$$

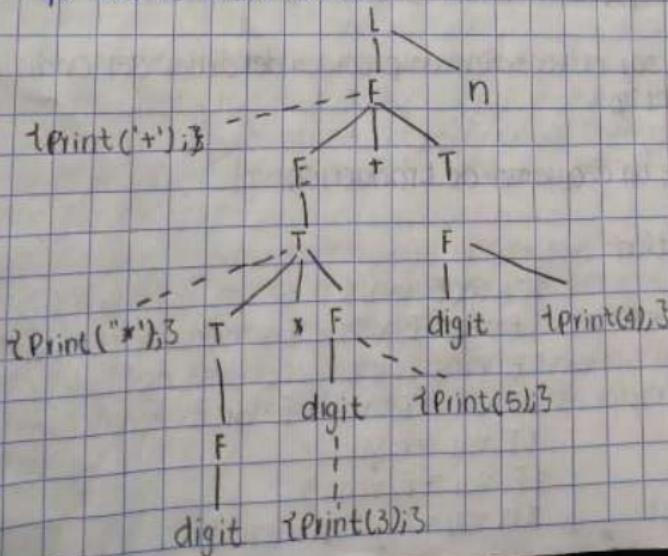
$$\{ F.\text{val} = E.\text{val}; \}$$

$$\{ F.\text{val} = \text{digit}.1\text{exVal}; \}$$

Árbol de Sintaxis del ejemplo anterior para la expresión $3 * 5 + 4$



Esquema de traducción.



TEMA 4.6 GENERACIÓN DE LA TABLA DE SÍMBOLOS Y DIRECCIONES

Una tabla de símbolos es una estructura de datos la cual almacena información semántica. Como sabemos la semántica trata sobre el sentido o el significado que tiene cada elemento en una oración, en el caso de los computadores nos referimos a semántica como el significado asociado a las estructuras formales de lenguaje.

Su objetivo principal es delimitar el contexto de las oraciones es decir, que el lenguaje con el que estamos trabajando se pueda describir recursivamente en términos de otros lenguajes o de los símbolos terminales.

El contenido de la misma es de dos tipos, la primera se enfoca en el propio símbolo y de los atributos necesarios para poder definir el símbolo hablando en términos de sintaxis o por otro lado ya en la generación de código, los atributos dependen a nivel general si:

- Gestión de memoria.
- Lenguaje en bloques o no.
- Generación del código.

Construcción de la tabla de símbolos.

1. Análisis léxico:

- Inserta los símbolos encontrados en la tabla.
- Crea un tabla parcialmente
- Indica la línea del código fuente de donde fue encontrada.

2. Análisis Semántico

- Añade los tipos, en caso de que procedan a los símbolos dentro de la tabla.

Operaciones Sobre la tabla de símbolos.

1. Insertar.
2. Consultar.
3. Modificar.

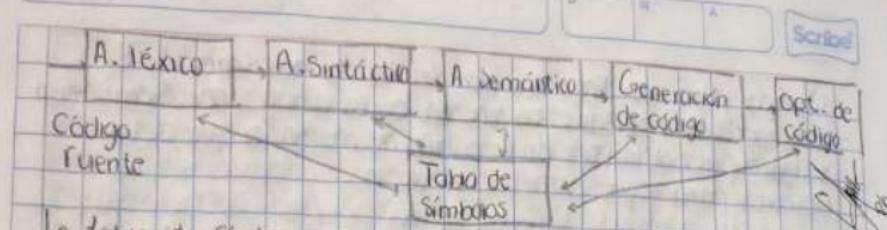
Consideraciones Sobre la tabla de símbolos.

La tabla de símbolos puede o no iniciar con información previa, en caso de iniciar con información pueden ser:

- Constantes : PI, E, etc.
- Funciones de librerías : EXP, LOG, etc.
- Palabras Reservadas: Esto facilita mucho el trabajo, después de reconocer un identificador lo busca en la tabla de símbolos y en caso de ser una palabra reservada retorna el token asociado.

Durante la ejecución van apareciendo nuevas declaraciones de identificadores, el analizador léxico o semántico va insertando nuevos símbolos a la tabla, evitando que haya entradas repetidas.

El encargado de efectuar las Comprobaciones Sensibles al contexto es el analizador semántico con apoyo de la tabla de símbolos. El generador de código intermedio usa las direcciones de memoria que se le fue asignada a cada identificador dentro de la tabla.



La Tabla de Símbolos contiene la información necesaria para Compilar, esto significa que el tiempo que demora es tiempo de Compilación y no de ejecución.

En un intérprete, dado que la compilación y ejecución son hechas a la par la tabla de símbolos permanece todo el tiempo.

En la implementación de la tabla de símbolos, el cómo se distribuye la información dependerá mucha de las características del lenguaje con el que se esté trabajando, también dependerá de las restricciones de los símbolos. Está conformado por Campos.

El Campo para el Símbolo se divide en un formato fijo y variable respecto al primero. Como su nombre nos lo indican es el encargado de establecer un límite de caracteres que conforman los símbolos. En el Segundo se tiene que la tabla de símbolos es un auxiliar en donde se introducen los símbolos de forma consecutiva.

Campo de dirección: Se divide en dos, lenguajes sin y con estructura de bloques, se necesita del número y dirección.

Campo tipo: Aquí se añaden cuando encontramos la declaración explícita o implícita de una variable. Es usada para determinar la memoria de nuestro programa.

Campo núm. de dimensiones /núm. de parámetro. El ~~Analizador Semántico~~ encargado de realizar esta acción es el analizador Semántico y ayuda para delimitar el tamaño de memoria que se necesita para representar un símbolo.

Campo lista cruzada de referencia y puntero de orden. Son muy útiles para los programadores del traductor de la tabla de símbolos representada como un objeto.

Organización de la tabla de símbolos.

1. Lenguaje SIN estructura de bloque: Es por medio de una lista la cual está dividida orden o desordenada. Se representa por árboles binarios y tabla de hash respectivamente.
2. Lenguaje CON estructura de bloque: Una tabla para cada bloque con estructura de pila.

Ejemplo de una tabla de símbolos.

IDENTIFICADOR	DIRECCIÓN	TIPO	DIMENSIÓN
a	Static+0	int	10
x	Static+10	float	0
i	Static+12	Double	0
s	Static+13	String	3

4.7 MANEJO DE ERRORES SEMÁNTICOS

Los errores semánticos pasan cuando la sintaxis es correcta pero carece de significado. Como se ha visto en los anteriores trabajos los analizadores léxico y semántico obedecen a las reglas del lenguaje, pues el analizador semántico no es la excepción. Como tal un compilador o intérprete no pueden detectar por su propia cuenta este tipo de errores, ya que estos sólo se ocupan de verificar que esté bien escrito en términos del lenguaje.

Un ejemplo de lo antes mencionado lo tenemos en conversiones de tipos que no son permitidas, en Java anteriormente realizar la conversión de un float a un int era no permitido, actualmente se puede realizar, pero antes lanzaba un error como el siguiente.

```
int x;  
x = 4.51;
```

Error: Ej1.java [6:1] possible loss of precision

Otros ejemplos comunes de este tipo de errores es:

- Variables usadas pero no definidas
- Operando incompatibles

```
int b,a,
```

```
c = b*a;
```

Error: Ej1. C is not defined

```
int i=0
```

```
if(x==5) x=0;
```

Error: Ej1. [7:1] Operator == cannot be applied to
int | int

Como mencionamos anteriormente, el compilador no ~~puede~~ de detectar por si solo errores Semánticos, por lo cual este recurre a la tabla de Símbolos, ya que esta le da información al compilador sobre el contexto que se tenga sobre los identificadores y de esta forma permitir al identificador ser un operando de cualquier operador del lenguaje.

Cuando hacemos esto podemos evitar la producción de un mensaje de error cada vez que la variable no sea definida o también si el operando no coincide con el operador que se le está dando.

Este tipo de errores hasta cierto modo son fáciles de depurar, puesto que ni el compilador ni el sistema proporcionan información sobre qué está fallando. Lo único que se tiene como información es que el programa no está funcionando como se debería.

Cuando se crea un lenguaje se debe prestar atención a los posibles errores que el mismo pueda producir. Ya sean errores léxicos, sintácticos o semánticos se debe tener en cuenta lo siguiente.

1. Errores de entrada
2. Recuperar los errores sin perder mucha información.
3. Se le debe proporcionar al programador información sobre el error con el fin de corregirlo.

~~Manejamiento de errores~~

Cuando un programa produce errores estos se almacenan en una llamada pila de errores. Si recordamos lo que es una pila, es una estructura de datos de tipo Fila (First In Last Out) lo que significa que el primero en entrar es el último en salir. Correspondiendo a la pila de errores esta almacenada los errores de forma que el primer error detectado será el último en salir. En la pila se pueden almacenar cualquier tipo de error. Ejemplo:

1. ent i = 0; → Produce un error Léxico.
2. int o = x; → Produce un error Sintáctico
3. S = B; → produce un error Semántico

Podemos observar un pequeño código en el cuál produce tres tipos de errores Léxico, Sintáctico y Semántico, entonces en la pila de errores regresaría lo siguiente:

- Error en la línea 3. S no definida
- Error en la línea 2. Sintaxis incorrecta.
- Error en la línea 1. ent no es tipo de dato.

la pila de errores debe mostrar al programador información detallada sobre los errores que la misma lance, ya que un error puede ocasionar más errores posteriormente, para ello es necesario saber dónde se originó todo.

Como se mencionó anteriormente los errores Semánticos se auxilian de la tabla de simbolos para detectarlos, sin embargo, actualmente gracias al avance de las tecnologías hoy IDEA que facilitan la detección de errores ante de que se compilar el código.

~~Matemática~~

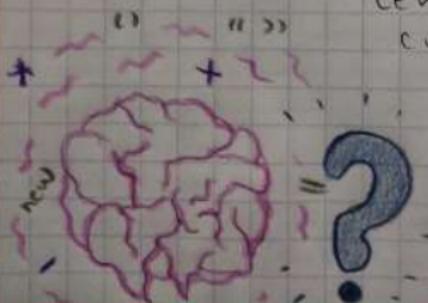
Estas IDEA previenen posibles errores de todo tipo léxico, sintáctico y semántico. Por lo que son de mucha ayuda al programador a la hora de estar realizando el código.

Incluso muchos lanzan advertencias antes de compilar el programa en caso de que un error halla sido detectado preguntandole al programador si está seguro de querer ejecutar ciún cuando se encontraron errores.

Conclusion

Los temas vistos anteriormente hacen referencia a una parte fundamental del proceso que lleva a cabo un compilador para funcionar correctamente a partir de sus tres tipos de análisis.

En este punto de análisis semántico ya se da por hecho el análisis léxico aplicado al alfabeto y el sintáctico que está enfocado en las gramáticas. Posiblemente sea el más complejo de identificar a simple vista ya que a diferencia del léxico y el sintáctico la estructura está completamente desarrollada, lo que se tendrá que evaluar es el tipo de parámetros que soliciten las instancias llamadas por de esta manera manejar el código fuente. Muchas de las veces se depende de la situación específica que se tenga teniendo en cuenta factores como el



lenguaje de programación, el grado de tipado o los tipos de datos que sea capaz de recibir el lenguaje de programación y el compilador.

*Chico
Máscara
C*

Links de los videos

Primer video:

<https://youtu.be/kmOF8xMiEnw>

Segundo video:

https://www.youtube.com/watch?v=mMN7I1xz78o&ab_channel=RAM%C3%8DREZGARC%C3%8DAMARCOISA%C3%8DAS

Bibliografía.

- ITPN (NA). Unidad I: Análisis Semántico. Recuperado el 18 de octubre de 2022, de:
<http://itpn.mx/recursosisc/7semestre/leenguajesyautomatas2/Unidad%201.pdf>
- lenguajes y automatas II (NA). 11: Árboles de expresiones. Recuperado el 18 de octubre de 2022, de:
<https://5e344735705b1.site123.me/unidad-i-%C3%81nalisissem%C3%A1ntico/11-%C3%81rboles-de-expresiones>
- Autnor. (2021). Que es el recorrido preorden Inorden y postorden. Recuperado el 18 de octubre de 2022, de
<https://cortocualquierconsejo.com.mx/quees-el-recorrido-preorden-inorden-y-postorden/>
#Que es un recorrido Postorden
- Aquilino Adolfo Juan Fuente. (2006). Tabla de símbolos de procesadores de lenguaje. Recuperado el 19 de octubre de 2022, de