



Tecnológico Nacional de México en Celaya

Lenguajes y Autómatas II

ACTIVIDAD 1

DOCENTE: Ricardo González González

Alumnos:

Hernandez Martinez Alina Adriana

Cid Soto Pablo

Cárdenas Ávila Ricardo

Rico Gómez Eduardo Daniel

Celaya , Gto a 14 de Febrero del 2020



"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

DEPARTAMENTO DE SISTEMAS COMPUTACIONALES E INFORMÁTICA

ASUNTO: SOLICITUD DE ACTIVIDADES

Celaya, Guanajuato, **31/Enero/2020**

LENGUAJES Y AUTÓMATAS II
DOCENTE DESIGNADO: ISC. RICARDO GONZÁLEZ GONZÁLEZ
SEMESTRE ENERO-JUNIO 2020

ACTIVIDAD 1 (VALOR 70 PUNTOS)

LEA CUIDADOSAMENTE, Y REALICE LAS SIGUIENTES 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 DE UNA TAREA SENCILLA, PUES DEMANDA TIEMPO PARA INVESTIGAR, LEER, ANALIZAR, REDACTAR, ILUSTRAR Y PROponER DE MANERA PROFESIONAL LOS TEMAS PROPUESTOS EN LA ESTRUCTURA TEMÁTICA DE ESTA MATERIA.

1. INTRODUCCIÓN.

COMO EQUIPO, O DE MANERA INDIVIDUAL, INVESTIGAR Y CON UNA REDACCIÓN PROPIA, DEFINIR LOS SIGUIENTES CONCEPTOS. DESPUÉS ELABORAR LAS ILUSTRACIONES PERTINENTES Y DE SU AUTORÍA QUE APoyEN DICHOS CONCEPTOS.

- A. QUÉ ES UN LENGUAJE (NATURAL / FORMAL), Y CUÁLES SON SUS CARACTERÍSTICAS MÁS IMPORTANTES.
- B. DEFINA QUÉ ES LA GRAMÁTICA, LA SINTAXIS Y LA SEMÁNTICA. DÉ UN PAR DE EJEMPLOS DE CADA UNA DE ÉSTOS CONCEPTOS. ADEMÁS EXPLIQUE SU CORRELACIÓN.
- C. ELABORE UNA CRONOLOGÍA DE LAS APORTACIONES EN LA VIDA PROFESIONAL DE NOAM CHOMSKY.
- D. DESARROLLE EL TEMA, "CATEGORÍAS DE GRAMÁTICAS, SEGÚN NOAM CHOMSKY". EXPLICANDO MUY BIEN CADA UNA DE ÉSTAS, Y ACOMPAÑANDO CON EJEMPLOS MUY CLAROS DE CADA UNO DE LOS TIPOS DE GRAMÁTICAS.





SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO
en Celaya

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

- E. DEFINA LO QUÉ ES UN AUTÓMATA, CUALES SON SUS CARACTERÍSTICAS, Y EXPLIQUE CÓMO SE CONSTRUYEN ÉSTOS.
- F. QUÉ ES UNA EXPRESIÓN Y EN QUÉ CATEGORÍAS SE DIVIDEN.
- G. QUÉ ES UN COMPILADOR, Y QUÉ ES UN INTÉPRETE. DEFINA SUS CARACTERÍSTICAS, SIMILITUDES Y DIFERENCIAS. POR ÚLTIMO DIGA CON QUÉ CRITERIOS SE ELIGE DESARROLLAR O USAR, UNO U OTRO.

2. INVESTIGACIÓN.

COMO EQUIPO, O DE MANERA INDIVIDUAL, INVESTIGAR Y CON UNA REDACCIÓN PROPIA, DEFINIR LOS SIGUIENTES CONCEPTOS. DESPUÉS, ELABORAR LAS ILUSTRACIONES PERTINENTES Y DE SU AUTORÍA QUE APOYEN A DICHOS CONCEPTOS.

- A. EL PROCESO DE COMPILACIÓN QUE SE EXPLICÓ LA PRIMER CLASE, ES DECIR TODAS LAS ETAPAS INVOLUCRADAS EN EL PROCESO DE "REDUCIR" UN LENGUAJE DE ALTO NIVEL A UNO DE BAJO NIVEL.
- B. DESCRIBIR A PROFUNDIDAD LAS ACCIONES CONCRETAS QUE IMPLICA EL REALIZAR CADA UNO DE LOS ANÁLISIS EN UN PROCESO DE COMPILACIÓN (LÉXICO, SINTÁCTICO Y SEMÁNTICO).
- C. QUÉ ES UNA TABLA DE SÍMBOLOS, PARA QUÉ SIRVE, EN QUÉ ETAPAS O ANÁLISIS DEL PROCESO DE COMPILACIÓN SE USA Y POR ÚLTIMO, QUÉ ESTRUCTURA POR LO GENERAL TIENE DICHA TABLA.
- D. COMO EQUIPO, O DE MANERA INDIVIDUAL, HAGA UN RESÚMEN MUY COMPLETO DE LO APRENDIDO EN LA MATERIA PRECEDENTE A ÉSTA (LA I), Y QUE USTED YA CURSÓ. ADEMÁS EXPLIQUE EL PROYECTO FINAL QUE ELABORÓ EN DICHA MATERIA.

MUY IMPORTANTE: ESTE EJERCICIO DEBE SER IMPLEMENTADO A MODO DE REPASO Y DEBE INCLUIR LAS EXPERIENCIAS DE LOS INTEGRANTES DEL EQUIPO.

- E. COMO EQUIPO, O DE MANERA INDIVIDUAL, PROPONER LOS REQUISITOS FORMALES Y NECESARIOS DE UN MICRO LENGUAJE PROTOTIPO, BASADO EN AQUELLOS DE TIPO C-STYLE.
- F. COMO EQUIPO, O DE MANERA INDIVIDUAL, EXPLICAR QUÉ ES Y CÓMO TRABAJA UNA PILA SEMÁNTICA. INCLUIR UN EJEMPLO DE CÓMO SE IMPLEMENTARÁ CON CÓDIGO FUENTE DICHA PILA, TOMANDO COMO BASE EL ANALIZADOR SEMÁNTICO DEL LENGUAJE PROTOTIPO DEFINIDO EN EL PUNTO ANTERIOR.



Antonio García Cubas Pte, No. 600 esq. Av. Tecnológico, Col. Alfredo V. Bonfil, C.P. 38010
Celaya, Gto. Ap 57, Comutador (461) 6117575 e-mail: lince@itcelaya.edu.mx
www.itcelaya.edu.mx



SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO
en Puebla

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

- G. COMO EQUIPO, HACER UNA MONOGRAFÍA DE INVESTIGACIÓN ACERCA DEL TEMA, ESQUEMAS DE TRADUCCIÓN, PROponiendo EJEMPLOS QUE SE APEGUEN A LA DEFINICIÓN DEL LENGUAJE PROTOTIPO QUE DESARROLLARÁ COMO PROYECTO FINAL.

NOTA : SI REQUIERE SABER O REAFIRMAR LAS CARACTERÍSTICAS DEL TIPO DE DOCUMENTO QUE SE LE ESTÁ SOLICITANDO (MONOGRAFÍA DE INVESTIGACIÓN), O ALGÚN OTRO QUE SE LE SOLICITARÁ, PUEDE CONSULTAR LA SIGUIENTE PÁGINA, DONDE DE FORMA BREVE SE EXPlica SU SIGNIFICADO Y CÓMO SE DEBE ELABORAR.

FUNDAMENTOS DE INVESTIGACIÓN AQUÍ.

- H. COMO EQUIPO, O DE MANERA INDIVIDUAL, DESARROLLAR EL TEMA QUE ABORDA LA GENERACIÓN DE TABLAS DE SÍMBOLOS Y DIRECCIONES, EXPLICANDO CON CLARIDAD LO QUE SE DEBE ENTENDER POR SÍMBOLOS Y DIRECCIONES, ASÍ COMO LA IMPORTANCIA QUE TIENEN EN EL PROCESO DE ANÁLISIS EN LA COMPILACIÓN E INTERPRETACIÓN.

- I. COMO EQUIPO, O DE MANERA INDIVIDUAL, EXPLICAR QUE CONSIDERACIONES SE DEBEN TOMAR PARA EL MANEJO DE LOS ERRORES SEMÁNTICOS.

EXPLIQUE AL MENOS 3 EJEMPLOS DE ERRORES SEMÁNTICOS Y DE CÓMO DAR TRATAMIENTO A ÉSTOS PARA LA DEFINICIÓN EN SU LENGUAJE PROTOTIPO PROPUESTO EN ESTAS MISMAS ACTIVIDADES.

ADICIONALMENTE EXPLICAR EN QUÉ CONSISTEN LOS ERRORES LÉXICOS Y SINTÁCTICOS.

EXPLICAR EJEMPLOS MUY CONCRETOS DE ÉSTOS, PARA QUE AYUDEN A COMPRENDER CÓMO CADA ANALIZADOR REQUIERE DE UN TRATAMIENTO PARTICULAR DE LOS ERRORES QUE SE PUEDEN PRESENTAR.



Antonio García Cubas Pte. No. 600 esq. Av. Tecnológico, Col. Alfredo V. Bonfil, C.P. 38010
Celaya, Gto. Ap 57, Comutador (461) 6117575 e-mail: lince@itcelaya.edu.mx
www.itcelaya.edu.mx



SEP
SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO
CAMPUS CELAYA

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

NOTA A CONSIDERAR.

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.

NOTA GENERAL DE LA ACTIVIDAD:

SE DEBE CONSIDERAR Y TOMAR EN CUENTA PARA EL CORRECTO CUMPLIMIENTO DE ESTA ACTIVIDAD, LO SOLICITADO EN LA GUÍA TUTORIAL, CONCRETAMENTE EN EL PUNTO 3 INCISO I (*Trabajo en equipo*).



Antonio García Cubas Pte. No. 600 esq. Av. Tecnológico, Col. Alfredo V. Bonfil, C.P. 38010
Celaya, Gto. Ap 57, Comutador (461) 6117575 e-mail: lince@itcelaya.edu.mx
www.itcelaya.edu.mx



SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO
en Celaya

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

LA NOMENCLATURA SOLICITADA PARA ENVIAR SU TRABAJO ES LA SIGUIENTE:

AAAA-MM-DD_MATERIA_DOCUMENTO_EQUIPO_NOCTROL_APELLIDOS_NOMBRE_SEM.PDF

(NOTA: *** TODO EN MAYÚSCULA ***)

DONDE:

AAAA : AÑO
MM : MES
DD : DÍA
MATERIA : LAII, PG, PE, LI
DOCUMENTO : A1-ACTIVIDAD 1, P1-PRACTICA 1, R1-REPORTE 1, T1-TAREA 1, PGI-PROGRAMA,
ETC. (CAMBIANDO EL NÚMERO CONSECUТИVO POR EL QUE CORRESPONDA)
EQUIPO : NÚMERO DEL EQUIPO QUE CORRESPONDA SEGÚN INDICACIÓN DEL PROFESOR.
NOCTROL : SU NÚMERO DE CONTROL
APELLIDOS : SUS APELLIDOS
NOMBRE : SU NOMBRE
SEM : EL PERIODO SEMESTRAL EN CURSO: ENE-JUN / AGO-DIC

EJEMPLO :

SI EL TRABAJO SE HACE EN EQUIPO.

2020-01-31_LAII_A1_EQUIPO_99_9999999_PEREZ_PEREZ_JUAN_ENE-JUN20.PDF

SI EL TRABAJO SE HACE INDIVIDUALMENTE.

2020-01-31_LAII_A1_9999999_PEREZ_PEREZ_JUAN_ENE-JUN20.PDF



Antonio García Cubas Pte. No. 600 esq. Av. Tecnológico, Col. Alfredo V. Bonfil, C.P. 38010
Celaya, Gto. Ap 57, Comutador (461) 6117575 e-mail: lince@itcelaya.edu.mx
www.itcelaya.edu.mx



SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO
en Celaya

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

OBSERVACIONES:

- ✓ LA REVISIÓN SERÁ EN DIVERSAS VERTIENTES. PUEDE SER AL MOMENTO DE SOLICITAR LA CARPETA DE EVIDENCIAS, O BIEN PUEDE SER AL SOLICITAR LA EXPOSICIÓN DE LA TAREA EN LA CLASE. TAMBIÉN PUEDE SER POR SOLICITUD EXPRESA DEL INTERESADO A PARTICIPAR EN CLASE EXPONIENDO BREVEMENTE SU ACTIVIDAD.
- ✓ AQUELLAS ACTIVIDADES EN FORMATO DIGITAL SE DEBERÁN TENER SIEMPRE, Y EN TODO MOMENTO A LA MANO EN UNA MEMORIA USB.
- ✓ ÉSTAS ACTIVIDADES PODRÁN SER SOLICITADAS EN LA CLASE, O BIEN PARA SU ENVÍO A UNA CUENTA DE CORREO.
- ✓ ESTAS ACTIVIDADES DEBEN ESTAR LISTAS E INTEGRADAS A LA CARPETA DE EVIDENCIAS (FÍSICAMENTE) A LA FECHA DE ENTREGA INDICADA AL FINAL DE ÉSTE DOCUMENTO.
- ✓ CADA HOJA QUE ENTREGUE DE SU ACTIVIDAD, DEBERÁ ESTAR FIRMADA AL MARGEN DERECHO, INCLUIDA LA PROPIA SOLICITUD DE LA ACTIVIDAD.
- ✓ 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.
- ✓ CON ESTA ACTIVIDAD, USTED DEBERÁ IR INTEGRANDO SUS CARPETAS FÍSICA Y ELECTRÓNICA DE EVIDENCIAS, Y AL FINAL DEL SEMESTRE EN UN DISCO COMPACTO HARÁ ENTREGA DE SU CARPETA ELECTRÓNICA DE EVIDENCIAS.
- ✓ PARA TENER DERECHO A LA REVISIÓN Y EVALUACIÓN DE SUS ACTIVIDADES, DEBE REGISTRAR SU ASISTENCIA A CLASE, EL DÍA SEÑALADO PARA LA ENTREGA DE LA MISMA.
- ✓ FALTAR A CLASE EL DÍA DE LA ENTREGA, ANULA LA REVISIÓN DE SUS EVIDENCIAS.
- ✓ POR ÚLTIMO, POR FAVOR GESTIONE APROPIADAMENTE SU TIEMPO, Y SEA PUNTUAL EN SU ENTREGA Y ASÍ EVITAR PROBLEMAS DE NULIDAD POR EXTEMPORANEIDAD.
- ✓ AÚN PARA TRABAJOS EN EQUIPO APlican TODAS LAS MISMAS OBSERVACIONES ANTERIORES.



Antonio García Cubas Pte. No. 600 esq. Av. Tecnológico, Col. Alfredo V. Bonfil, C.P. 38010
Celaya, Gto. Ap 57, Comutador (461) 6117575 e-mail: lince@itcelaya.edu.mx
www.itcelaya.edu.mx



SEP

SECRETARIA DE
EDUCACIÓN PÚBLICA



• TECNOLÓGICO NACIONAL DE MÉXICO
en Coahuila

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

FECHA DE ENTREGA:

VÍA CORREO ELECTRÓNICO, EL VIERNES 14 DE FEBRERO DEL 2020, CON HORA LÍMITE DE ENTREGA HASTA LAS 14:00 HORAS (2 DE LA TARDE).

EN CASO DE QUE EL TRABAJO SE HAYA ELABORADO EN EQUIPO, EL JEFE DEL MISMO SERÁ EL ÚNICO RESPONSABLE DE ENVÍAR LA ACTIVIDAD A LA SIGUIENTE CUENTA DE CORREO:

ricardo.gonzalez@itcelaya.edu.mx

MUY IMPORTANTE:

DESPUÉS DE LAS 14:00 HRS. EN PUNTO, LA ACTIVIDAD YA SERÁ CONSIDERADA COMO EXTEMPORÁNEA Y NO CONTARÁ COMO EVIDENCIA PARA SU EVALUACIÓN.

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

RECUERDE ANEXAR A SU ARCHIVO .PDF DE EVIDENCIAS, ESTA SOLICITUD DE ACTIVIDADES CON TODAS LAS HOJAS FIRMADAS EN EL MARGEN DERECHO.

POR ÚLTIMO, CONSIDERE EL SIGUIENTE CALENDARIO OFICIAL, PARA GESTIONAR ADECUADAMENTE EL TIEMPO QUE SE LE OTORGÁ PARA DESARROLLAR ESTA ACTIVIDAD.

CALENDARIO 2019-2020 - EVALUACIÓN 1

中		上		下		左		右		外	
四	五	二	三	一	六	九	八	七	十	十一	十二
5	6	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39	40	41

EVALUACIÓNFORMATIVA DE LA ESTADÍSTICA PARCIAL

A) **Qué es un lenguaje (natural/formal), y cuáles son sus características más importantes.**

Un lenguaje es un sistema mediante el cual se puede llevar a cabo la comunicación; este sistema puede llevarse mediante distintas maneras, las cuales pueden ser sonoras (utilizando la voz o sonidos), corporales (mediante el cuerpo o señas), con gráficos (mediante la escritura o dibujos), entre otras maneras.

Todo lenguaje en esencia es un código de signos en el cual cada signo tiene algún significado. Ya sea en la comunicación hombre-máquina o entre seres humanos, el lenguaje nos permite que, a partir de un mensaje dado por un emisor, se interprete por el receptor y obtengamos una salida o respuesta como resultado.

Para fines de la materia y junto a nuestro conocimiento previo, vamos a definir al lenguaje como un conjunto de palabras dadas por un conjunto de símbolos finito (llamado alfabeto y representado por el símbolo Σ), el lenguaje serán todas las combinaciones resultantes del alfabeto y sera representado por Σ^* .

Ejemplo: El lenguaje está dado por el alfabeto del sistema binario

$$\Sigma = \{0, 1\}$$

$$\Sigma^* = \{0, 1, 00, 01, 11, \dots, 10101110, \dots\}$$

El lenguaje puede ser tan complejo o tan simple dependiendo la manera en la que fueron definidos, el lenguaje conformado por los números decimales es sencillo para nosotros los humanos comprenderlo ya que nos podemos apoyar de los dedos de nuestra mano para representarlo, a diferencia del lenguaje

(Firma)
llevado a cabo por el sistema hexadecimal en el cual la manera en que lo comprendemos puede ser sencilla pero la manera de representarlo a demás a entender en ese lenguaje ya se dificulta.

• Lenguaje natural

El lenguaje natural es formado por la manera en que nos comunicamos día a día con otros seres humanos, el proceso mediante el cual lo hacemos es casi inconsciente ya que lo utilizamos desde nuestros primeros años de vida, se fue desarrollando y llevando a cabo por la experiencia humana y como lo dice su nombre se va dando de manera "natural".

Aplicado en la informática, los desarrolladores y científicos han intentado llevar un lenguaje natural de los humanos a los computadores, de modo que no hiciera falta llevar a cabo una traducción para generar la comunicación, pero esto problemático fue lo que llevó a la generación de diferentes lenguajes formales en la informática, ya que los humanos contamos con diferentes idiomas, tonos de voz, maneras de hablar, formas de expresarnos y organizar las oraciones, estas diferentes características llevó a estandarizar los lenguajes formales para la comunicación hombre-máquina.

De igual manera los lenguajes naturales tienen una característica que dice que son polisémicos, es decir una sola palabra puede tener diversos significados, esto dificulta más la implementación de un lenguaje natural en los ordenadores.

• Lenguaje formal (lenguaje que tienen su lógica clara)

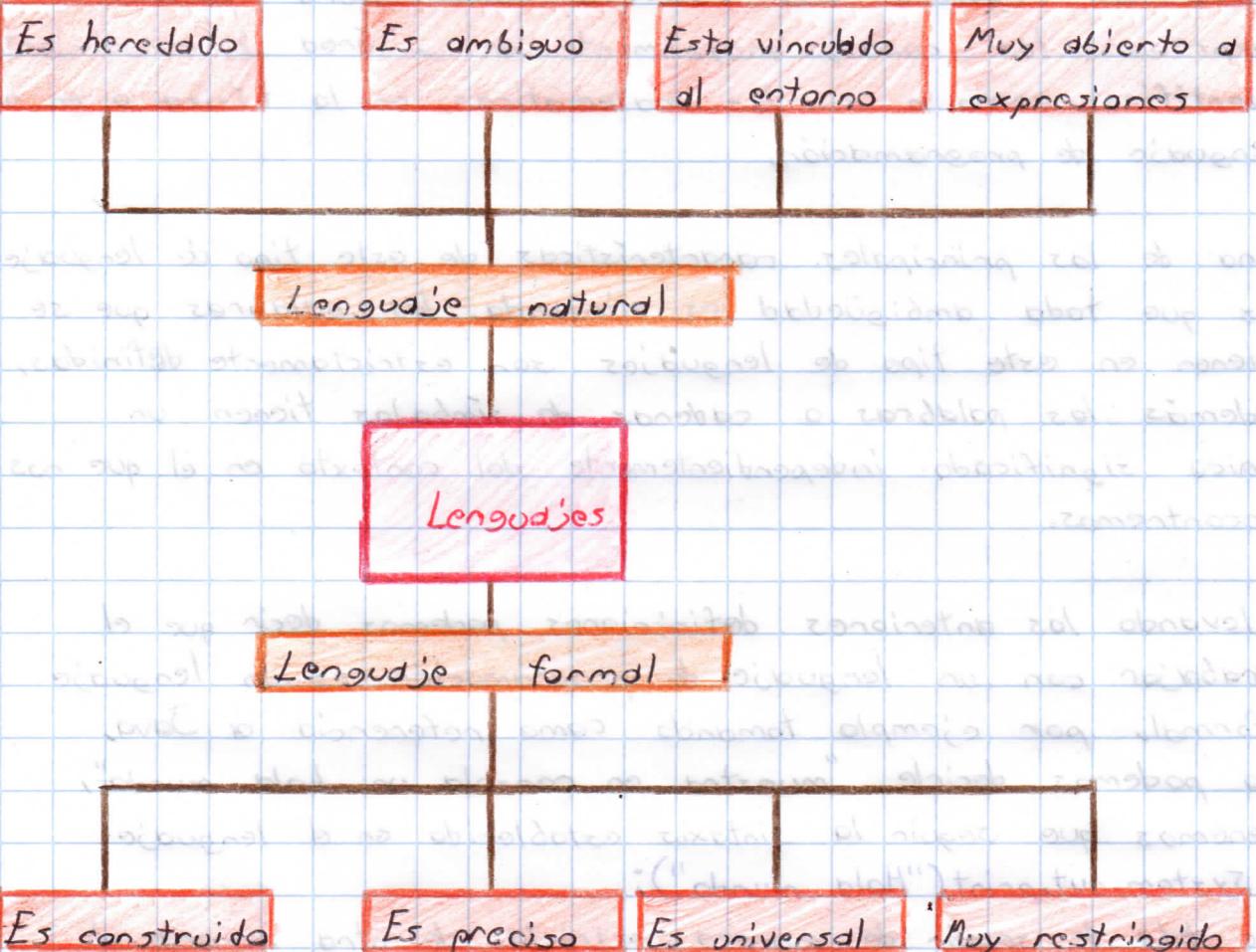
El lenguaje formal tiene como propósito expresar las diferentes situaciones que se dan a cabo en algún área donde se requiera una estricta nomenclatura para expresar las cosas, mayormente en un área de conocimiento científico como lo son las matemáticas, en la física o en un lenguaje de programación.

Una de las principales características de este tipo de lenguaje es que toda ambigüedad es eliminada, las palabras que se tienen en este tipo de lenguajes son estrictamente definidas, además las palabras o cadenas de símbolos tienen un único significado independientemente del contexto en el que nos encontramos.

Llevando las anteriores definiciones podemos decir que el trabajar con un lenguaje de programación es un lenguaje formal, por ejemplo tomando como referencia a Java, no podemos decirle "muestra en consola un hola mundo", tenemos que seguir la sintaxis establecida en el lenguaje:
`System.out.print("Hola mundo");`

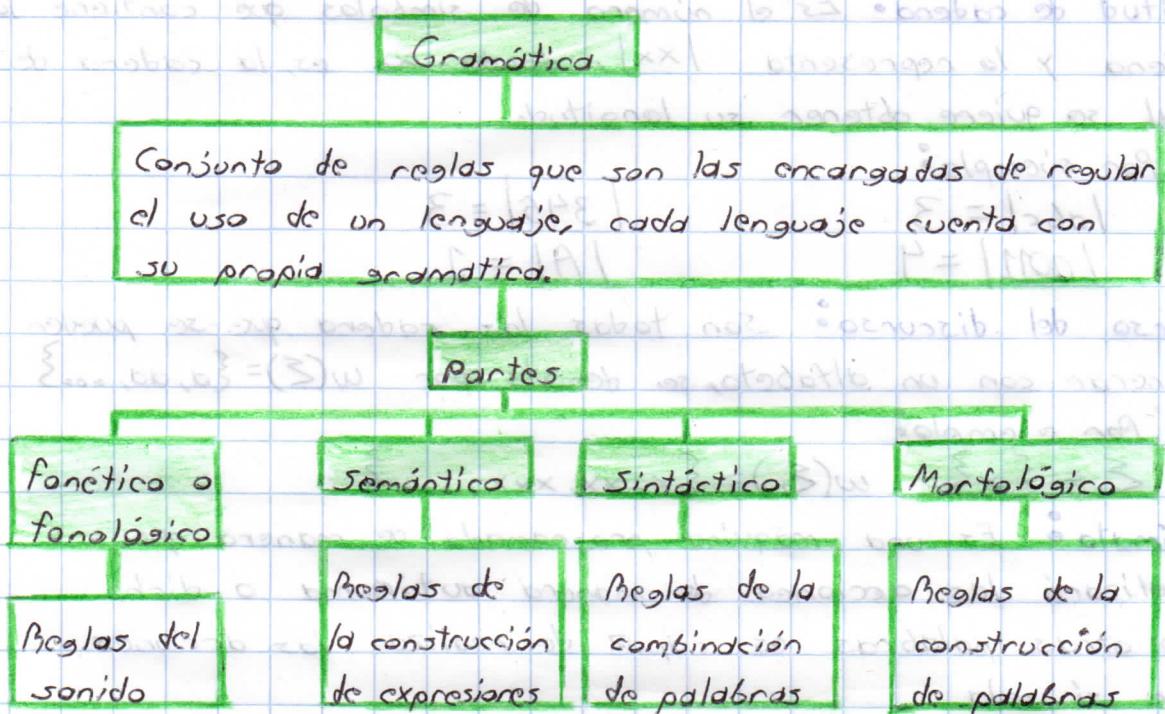
se debe de seguir de manera estricta, de otra manera el lenguaje no podrá entenderlo y no podrá traducirlo al lenguaje máquina para que el computador ejecute las acciones dadas.

Lenguaje natural vs Lenguaje formal



B) Defina qué es la gramática, la sintaxis y la semántica. Dé un par de ejemplos de cada uno de estos conceptos. Además, explique su correlación.

Partamos del siguiente mapa conceptual para ver lo que es la gramática dentro de la materia en cuestión.



Como podemos ver, la esencia de la gramática son las reglas que sigue un lenguaje o una cadena de símbolos para pertenecer al lenguaje, exactamente eso mismo es en el área de los lenguajes y automatas.

Para poder apoyar el concepto es necesario definir algunos otros cómo lo son los siguientes:

- **Símbolo**: Son la manera en la que se representa el lenguaje, comúnmente con letras, números u otros caracteres
 - **Alfabeto**: Es el conjunto finito de símbolos.
- Por ejemplo:

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

- **Cadena:** Es un conjunto de símbolos finitos utilizado dentro de algún alfabeto.

Ejemplo: Del alfabeto $\Sigma = \{a, b, c\}$, algunas cadenas válidas serían "a", "abc", "adc", "daabcc", etc.

- **Cadena vacía:** Es aquella cadena que no tiene elementos y se puede denotar de distintas maneras por ejemplo: ϵ .

- **Longitud de cadena:** Es el número de símbolos que contiene la cadena y la representa $|xx|$, donde xx es la cadena de la cual se quiere obtener su longitud.

Por ejemplo:

$$|abc| = 3$$

$$|0011| = 4$$

$$|345| = 3$$

$$|A| = 1$$

- **Universo del discurso:** Son todas las cadenas que se pueden generar con un alfabeto, se denota por $w(\Sigma) = \{a, aa, \dots\}$

Por ejemplo:

$$\Sigma = \{x\} \quad w(\Sigma) = \{x, xx, xxx, \dots\}$$

- **Autómata:** Es una máquina programada de manera que realizará las acciones de manera autónoma o dicho en otras palabras es capaz de realizar las acciones por sí sola.

Como podemos ver, las anteriores conceptos retoman cosas ya vistas anteriormente con lo que ahora podemos proceder a definir la gramática de una manera formal.

El autor Chomsky del cual retomaremos sus definiciones más adelante además de sus aportaciones define a la gramática de la siguiente manera: "Descripción formalizada de un lenguaje. Una gramática genera o describe un lenguaje".

Una gramática es una cuadrigua definida por:

$$G = \langle V_t, V_n, S, P \rangle$$

Dónde:

$$G = \text{Gramática}$$

V_t = Conjunto finito de símbolos terminales.

V_n = Conjunto no finito de símbolos terminales.

S = Símbolo inicial que pertenece a V_n .

P = Conjunto de reglas de derivación.

Operaciones con la gramática:

Consiste en aplicar las reglas de remplazo iniciando con S y termina cuando la cadena obtenida contiene únicamente símbolos de V_t .

" \rightarrow " = Símbolo de derivación

Ejemplo 1:

$$G = \langle V_t, V_n, S, P \rangle$$

$$V_t = \{d\}$$

$$V_n = \{S\}$$

$$S = S$$

$$P = \{ \textcircled{1} S \rightarrow dS \quad \textcircled{2} S \rightarrow d \}$$

$$\bullet S \xrightarrow{1} dS \xrightarrow{1} ddS \xrightarrow{2} ddad$$

$$\bullet S \xrightarrow{2} d$$

$$\bullet S \xrightarrow{1} dS \xrightarrow{2} dd$$

Las cadenas resultantes de la gramática son

$$\{d, dd, ddad, \dots\}$$

Sintaxis

En la cuadriga de la gramática, el componente P designa las reglas de las secuencias que debe seguir la cadena para ser considerada válida dentro de un lenguaje, este componente P son las sintaxis que sigue la cadena.

Visto lo anterior podemos ver una cadena válida y ver cadenas no válidas en el lenguaje de algún gramática.

Ejemplo:

$$G_3 = \langle V_t, V_n, S, P \rangle$$

$$V_t = \{x\}$$

$$V_n = \{S\}$$

$$S = S$$

$$P = \{ \begin{array}{l} ① S \rightarrow x \\ ② S \rightarrow xS \end{array} \}$$

Cadenas válidas:

$$\bullet x \quad \bullet xxx \quad \bullet xx$$

Cadenas no válidas:

$$\bullet xa \quad \bullet 28x \quad \bullet xx3$$

Podemos observar que las cadenas no válidas no pueden ser generadas de ninguna manera ya que, como primer punto no contamos con más símbolos finales aparte "x", por lo que cualquier cadena que contenga algo diferente a ello será considerada como no válida.

Semántica

La semántica es la parte que tiene como propósito verificar el significado de los开端 que componen un lenguaje.

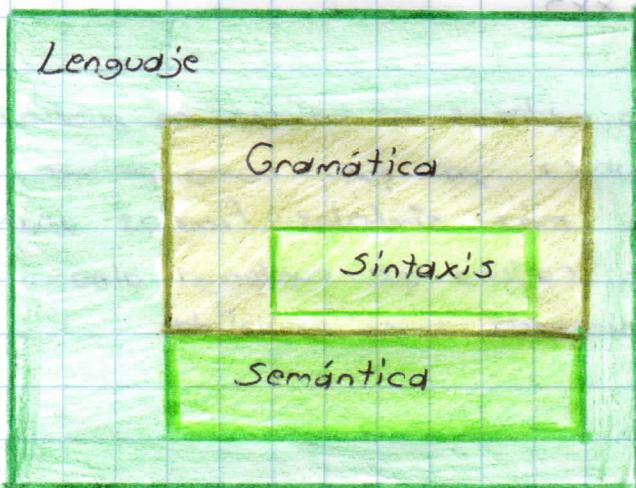
De manera sencilla podemos mencionar el siguiente ejemplo:

En una caja tengo 5 monedas, 3 de ellas de \$10 y 2 de ellas de \$5, las de \$10 son para Juan y las de \$5 son para Miguel, de esta manera ya se le dio un significado a cada tipo de moneda.

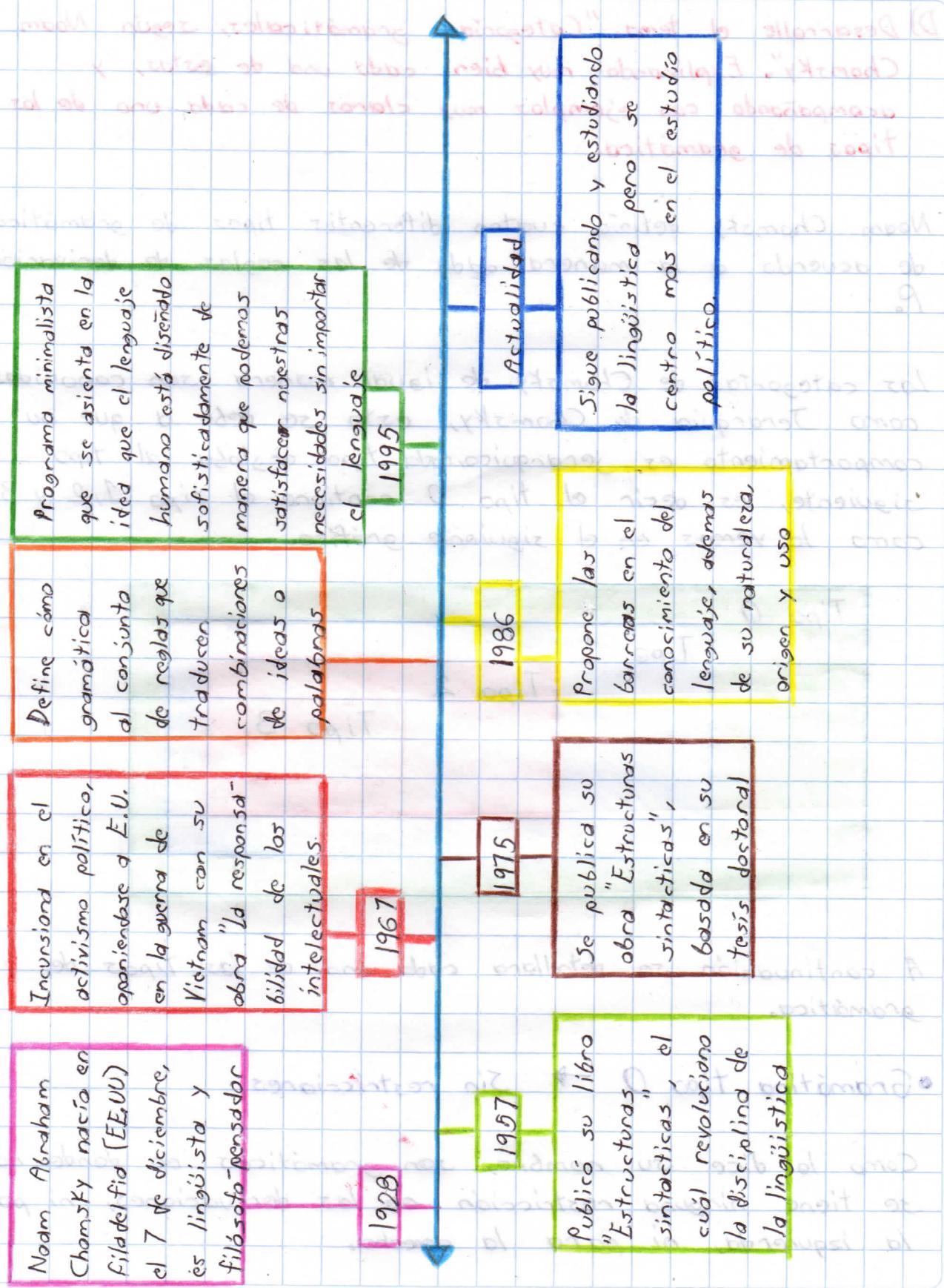
Utilizando las bases anteriormente mencionadas de la gramática se dice que el análisis semántico es la parte posterior al análisis sintáctico donde se registra la información adicional necesaria para ser procesada e interpretada por el lenguaje.

Correlación

La gramática, la sintaxis y la semántica son partes del lenguaje las cuales definen como es, que lo compone y que significan los开端 dentro de él.



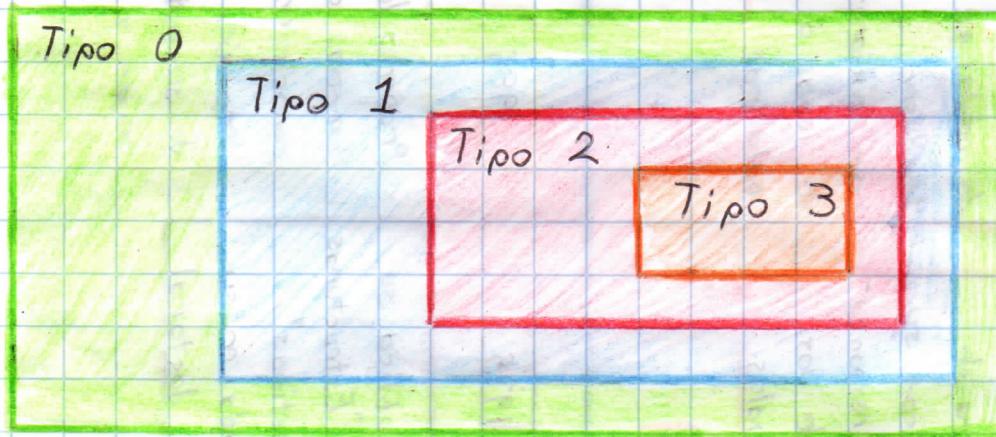
C) Elabore una cronología de las aportaciones en la vida profesional de Noam Chomsky.



D) Desarrolle el tema, "Categorías gramáticas, según Noam Chomsky". Explicando muy bien cada uno de éstas, y acompañando con ejemplos muy claros de cada uno de los tipos de gramática.

Noam Chomsky definió cuatro diferentes tipos de gramática de acuerdo a la manera dada de las reglas de derivación P.

Las categorías de Chomsky de igual manera son conocidas como Jerarquía de Chomsky, esto se debe a que su comportamiento es jerárquico, cada tipo engloba al tipo siguiente, es decir el tipo 0 contiene el tipo 1, 2 y 3, como lo vemos en el siguiente gráfico.



A continuación se detallará cada uno de los tipos de gramática.

• Gramática tipo 0 → Sin restricciones

Como lo dice su nombre, son gramáticas en donde no se tiene ninguna restricción en las derivaciones, ni para la izquierda, ni para la derecha.

No existe algoritmo finito que diga si una cadena o no sigue las reglas de una gramática.

Estas gramáticas generan a todos los lenguajes que son reconocidos por una máquina de Turing.

Ejemplo: $G = \langle V_n, V_t, S, P \rangle$

$$V_n = \{S, A, B\} \quad V_t = \{a, b\}$$

$$P = \{ \begin{array}{l} ① S \rightarrow \epsilon \\ ② S \rightarrow AaBS \\ ③ BS \rightarrow BB \\ ④ Bb \rightarrow bb \\ ⑤ A \rightarrow ab \end{array} \}$$

$$S = S$$

Como vemos, la gramática genera cadenas que se extienden tanto a izquierda como a derecha.

Gramática tipo 1 → Dependiente del contexto

Este tipo de gramática acota las derivaciones del lado izquierdo, únicamente permitiendo las del lado derecho.

Los lenguajes generados a partir de esta gramática son los lenguajes que de igual manera son reconocidos por la máquina de Turing, pero que se encuentran acotada por la izquierda.

Ejemplo: $G = \langle V_n, V_t, S, P \rangle$

$$V_n = \{X, Y, Z\} \quad V_t = \{x, y, z\} \quad S = S$$

$$P = \{ \begin{array}{ll} ① S \rightarrow X & ⑤ yY \rightarrow yy \\ ② X \rightarrow xXYZ & ⑥ yZ \rightarrow yz \\ ③ X \rightarrow xyZ & ⑦ zZ \rightarrow zz \\ ④ ZY \rightarrow YZ & \end{array} \}$$

Como vemos, ya no se deriva a la izquierda, pero para derecha se sustituyen varios símbolos.

• Gramática tipo 2 → Libre de contexto

La principal característica que la gramática libre de contexto es que únicamente permiten las derivaciones cuando en la parte izquierda solamente se encuentran símbolos no terminales, es decir se encuentran con ~~sólo~~ el no terminal y puede derivar a un conjunto de símbolos terminales y no terminales, matemáticamente se representa como:

$$A \rightarrow a \quad \text{dónde: } A \in V_n \text{ y } a \in (V_n \cup V_t)^+$$

Ejemplo: $G = \langle V_n, V_t, S, P \rangle$

$$V_n = \{A, B\} \quad V_t = \{a, b\} \quad S = A$$

$$P = \{ \begin{array}{ll} ① A \rightarrow aB & ④ B \rightarrow Ab \\ ② A \rightarrow bA & ⑤ B \rightarrow bb \\ ③ A \rightarrow a & ⑥ B \rightarrow aBB \end{array} \}$$

• Gramática tipo 3 → Regulares

También conocidas como gramáticas lineales o la derecha, todas sus reglas de producción se rigen por alguna de las siguientes dos formas:

- Comienza por un no terminal y deriva a uno terminal seguido de uno terminal

$$A \rightarrow aB$$

- Comienza por un no terminal y deriva a uno terminal

$$A \rightarrow a$$

Este tipo de gramática es la que rige a los autómatas finitos.

10/07/2023

Ejemplo: $G = \langle V_n, V_t, P, S \rangle$

$$V_n = \langle S, X, Y \rangle \quad V_t = \{a, b\} \quad S = S$$

$$P = \{ \begin{array}{l} ① S \rightarrow XS \\ ② S \rightarrow YS \\ ③ S \rightarrow ZS \\ ④ S \rightarrow X \\ ⑤ S \rightarrow Y \\ ⑥ S \rightarrow Z \end{array} \}$$

$$② S \rightarrow YS$$

$$③ S \rightarrow ZS$$

$$④ S \rightarrow X$$

$$⑤ S \rightarrow Y$$

$$⑥ S \rightarrow Z \}$$

Como podemos ver a través de los ejemplos dados, conforme se adentra en la jerarquía, se van adquiriendo restricciones, de manera de resumir y ver las diferencias obtenemos el siguiente cuadro:

Tipo	Lenguaje que genera	Autómata que reconoce	Normas de producción	Ejemplo de una derivación
0	Recursivamente enumerable (LRE)	Máquina de Turing	$\alpha A \beta \rightarrow \gamma$	$aBcD \rightarrow aDBb$ $aBD \rightarrow BaC$
1	Dependiente del contexto (LSC)	Autómata ligeramente acotado	$\alpha A \beta \rightarrow \alpha \gamma \beta$	$aBC \rightarrow aBCaD$
2	Independiente del contexto (LLC)	Autómata con pila	$A \rightarrow \gamma$	$B \rightarrow aBC$
3	Regular (LR)	Autómata finito	$A \rightarrow \gamma$ $A \rightarrow a\theta$	$A \rightarrow a$ $A \rightarrow aB$

Simbología:

a = Terminal

A, B, C, D = No terminal

α, β = Cadena posiblemente vacía

γ = Cadena no vacía

e) Autómatas

La teoría de autómatas, el estudio de dispositivos de cálculo abstractos. En los años treinta, Alan Turing estudió una máquina abstracta su objetivo era describir muy puntualmente los límites entre lo que una máquina de cálculo podía y no podía hacer, de hecho aún se siguen aplicando estas conclusiones a las máquinas que utilizamos hoy en día. Posteriormente en las décadas de los cuarenta y cincuenta se investigaron más a fondo las máquinas más simples, las que denominamos "Autómatas finitos" estos se plantearon para modelar el funcionamiento del cerebro que se terminaron usando para otros propósitos.

Los Autómatas finitos es un modelo de gran utilidad tanto para Hardware y Software.

Los autómatas trabajan junto con otros términos:

Alfabeto: conjunto finito de símbolos y no vacío.

Cadena de caracteres: Es una secuencia de símbolos finito.

Cadena vacía: es la cadena que no contiene símbolos.

El autómata finito, tiene estados, cambia de estado a otro en respuesta a una entrada.

Los autómatas finitos se dividen en deterministas y no deterministas

Autómata determinista

El autómata determinista es aquel que solo puede estar en un estado a la vez.

Consta de lo siguiente:

- Conjunto finito de estados (Q).
- Conjunto finito de símbolos de entrada (Σ)
- función de transición
- Un estado inicial
- Conjunto de estados finales de aceptación.

Autómata no determinista

Tiene la capacidad de encontrarse en varios estados a la vez. Comúnmente se utiliza para buscar palabras clave dentro de un texto. Este también tiene la capacidad de convertirse en autómatas deterministas.

Comparte las mismas características que un autómata determinista.

Representación Gráfica.

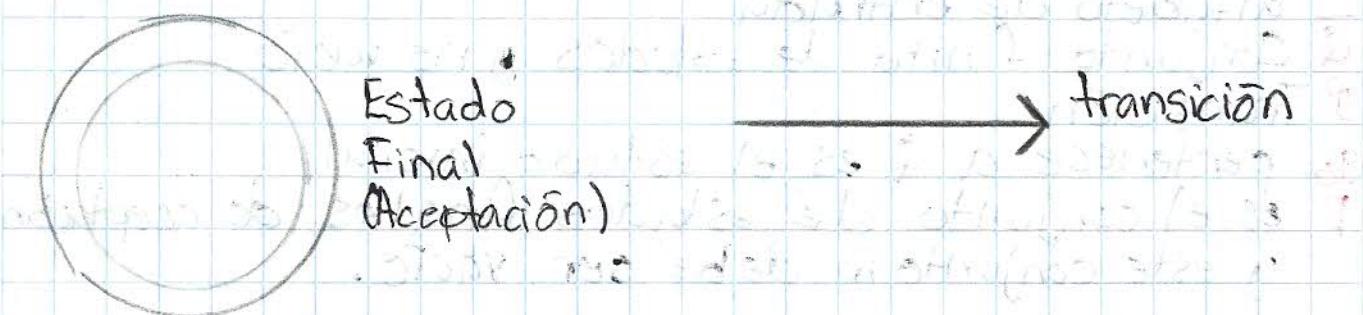
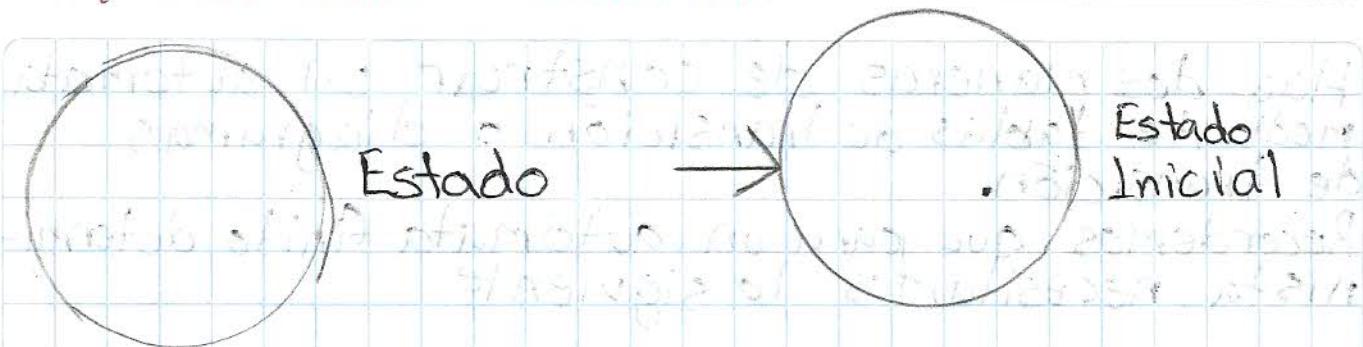


Diagram illustrating the alphabet (Σ):

Σ • **Alfabeto**

O, E → Etiquetas

Σ^* * Conjunto de todas las palabras posibles

W Palabra

$|W|$ Longitud de palabra

L Lenguaje

↓ Cerradura de Kleene (*)

Construcción de Automátas

Hay dos maneras de construir un autómata mediante tablas de transición o diagramas de transición.

Recordemos que para un autómata finito determinista necesitamos lo siguiente

Σ alfabeto de entrada

Q Conjunto finito de estados y no vacío

δ Transición

q_0 pertenece a Q es el estado inicial

F es el conjunto de estados finales de aceptación y este conjunto no debe ser vacío.

Para la construcción de un autómata necesitamos un lenguaje como en el siguiente ejemplo

$$L = \{a^n b a^m \mid n > 0, m > 0\}$$

$Q = \{q_0, q_1, q_2, q_3\}$ este depende de cuantos estados necesitamos

$\Sigma = \{a, b\}$ nuestro alfabeto

$F = \{q_3\}$ depende de donde lleguen nuestros estados de aceptación, tanto en conjunto Q , como F dependen de la lógica del diseñador de como va a construir su autómata

ahora empezamos a crear nuestras transiciones

$$\delta(q_0, a) = q_1$$

$$\delta(q_1, a) = q_1$$

$$\delta(q_1, b) = q_2$$

$$\delta(q_2, a) = q_3$$

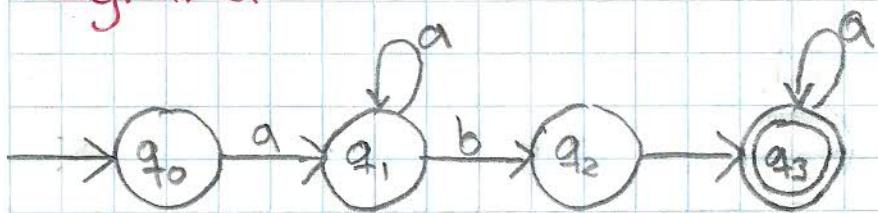
$$\delta(q_3, a) = q_3$$

Con nuestras transiciones ya podemos realizar la tabla y el diagrama.

Tabla

		Alfabeto		
		a	b	
E	$\rightarrow q_0$	q_1		
S		q_1	q_2	\rightarrow Estado inicial
+				*
q				Estado final
d				
0				
s	$\downarrow * q_3$	q_3	q_3	

Diagrama



\rightarrow Estado inicial

\circ Estado final

\overrightarrow{a} transición

f) Expresión

Expresiones Regulares

Son una secuencia de caracteres que forman un patrón de búsqueda. Estas las utilizan para encontrar patrones de cadenas de caracteres. Estas expresiones regulares denotan Lenguajes

Las expresiones regulares cuentan con operadores.

Unión U

La unión de dos o más lenguajes ejemplo

$$A = \{a, A\}$$

$$B = \{B\}$$

$$C = \{c, C\}$$

$$A \cup B \cup C = \{a, A, B, c, C\}$$

Concatenación .

La concatenación de dos o más lenguajes se denotan con un punto en medio de los lenguajes y este se tiene que respetar el orden, ejemplo:

$$A = \{A, a\}$$

$$B = \{B\}$$

$$A \cdot B = \{AB, aB\}$$

Clausura o Cerradura de Kleene L*

Se designa como un superíndice de un asterisco en el lenguaje, este permite crear cadenas utilizando las cadenas del lenguaje con repeticiones infinitas ejemplo:

$$A = \{a, ab, b\}$$

$$A^* = \{a, aa, aaa, ab, abab, aabab, b, bb, bbb, abb, \dots\}$$

g) Compilador

Es un software que traduce un programa escrito en un lenguaje de alto nivel como java, python, c++, etc., en un lenguaje máquina que pueda interpretar el procesador.

El compilador tiene distintas etapas.

• Analizador Léxico

Primera etapa, se encarga de dividir el programa en tokens dependiendo de la tabla de símbolos del lenguaje.

• Analizador sintáctico

Es la segunda fase del proceso de compilación, generan un árbol sintáctico que permite representar de una forma más simple el programa fuente.

Algunos compiladores actuales utilizan estructuras de objetos para representar a un programa, de esta forma existe una clase específica para representar cada posible token de nuestra tabla de símbolos.

• Analizador semántico

Valida compatibilidad de tipos de datos, que las variables se encuentren programadas y se encuentren dentro del contexto. El objetivo es que tenga un significado exacto y que no pueda fallar en tiempo de ejecución.

2023
FEB
Nº 1

• Generación código intermedio

Después de los análisis se genera un código intermedio del código fuente, este código se encuentra entre un lenguaje de alto nivel y el lenguaje de máquina, este código intermedio es más fácil de transformar a lenguaje máquina.

• Optimización de código

Esta etapa se realiza sobre el código intermedio, es esta elimina el código innecesario, y organiza la secuencia de declaraciones para acelerar la ejecución del programa.

• Generación de código máquina

Ya con el código intermedio optimizado se realiza la generación y mapeo al lenguaje máquina

Interprete

Es un software que recibe un programa en lenguaje de alto nivel, lo analiza y lo ejecuta.

Estos no generan un archivo ejecutable, por lo que cada vez que se ejecuta se vuelve a analizar el programa. Por lo tanto los intérpretes son más lentos que los compiladores.

Similitudes

- Los dos analizan el código fuente en busca de errores, ya sean léxicos, sintácticos y semánticos

Diferencias

- El compilador genera un archivo ejecutable
- El intérprete puede ejecutarse incluso con errores, cuando llega al error este se detiene
- Es más fácil distribuir un compilado que interpretarlo en diferentes máquinas

Compilador vs Interpretante

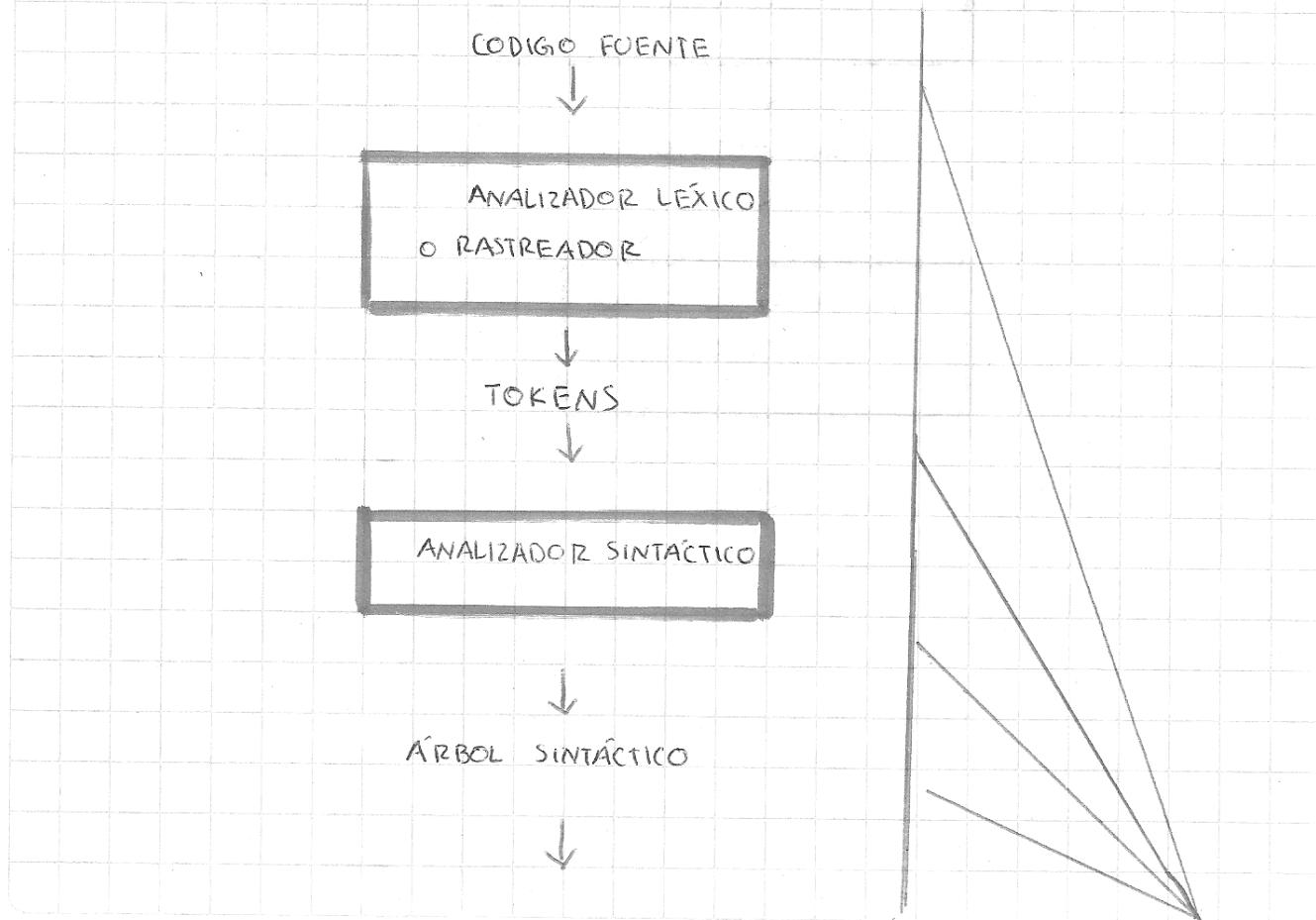
El compilador es mejor utilizarlo en proyectos pequeños, ya que no tarda mucho en compilarse, en cambio imaginemos que estamos desarrollando un sistema operativo donde este tiene cientos de miles o millones de líneas de código, si necesitamos probar un cambio lo que tardaría en compilarse sería demasiado a comparación de si usamos un intérprete y probar los cambios.

(A) El proceso de compilación que se explicó en la primera clase, es decir todas las etapas involucradas en el proceso de reducción lenguaje de alto nivel a uno de bajo nivel.

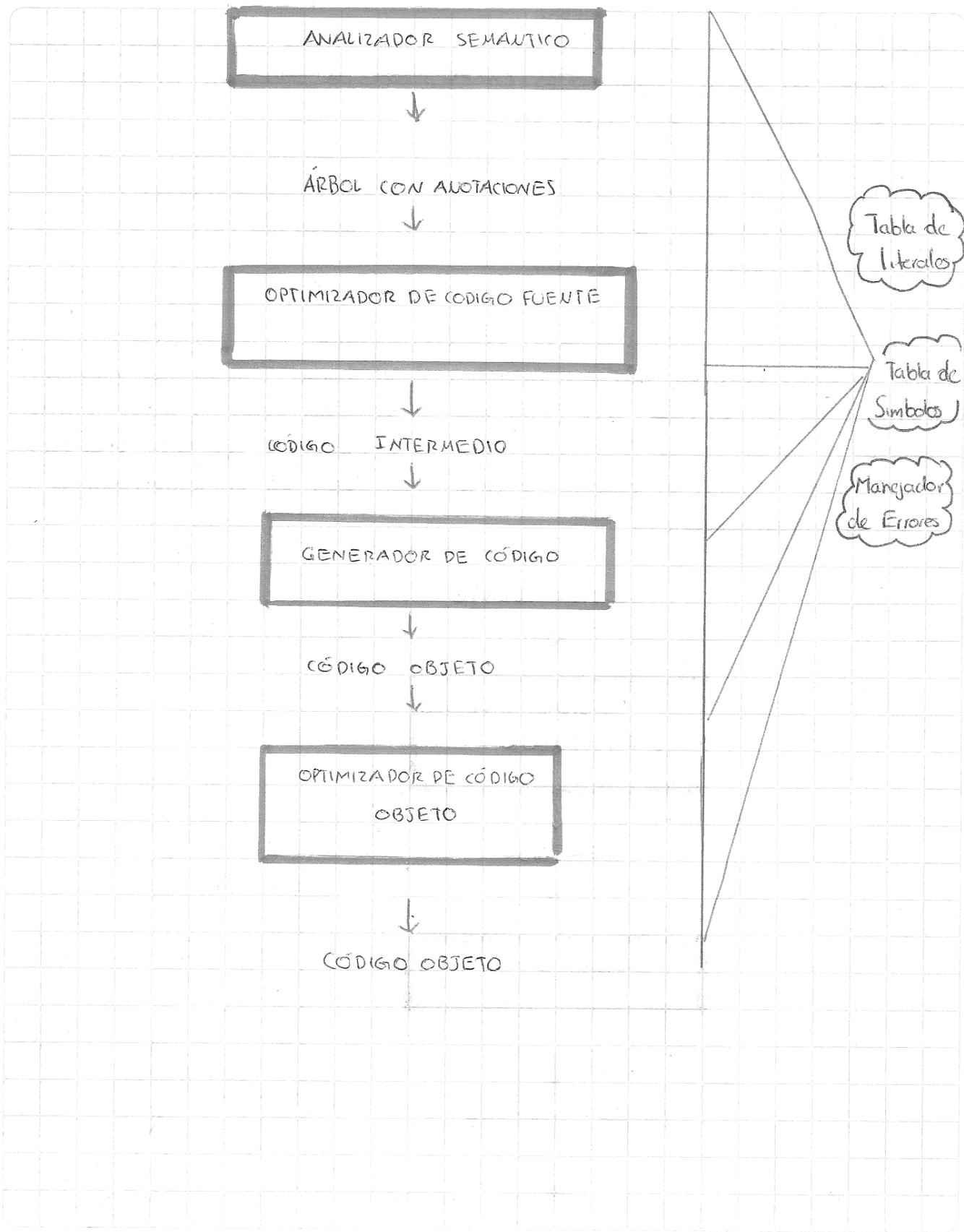
(B) Describir a profundidad las acciones concretas que implica el realizar cada uno de los análisis en un proceso de compilación (Léxico, sintáctico, Semántico).

PROCESO DE TRADUCCIÓN

Un compilador se compone internamente de varias etapas que realizan operaciones lógicas. Se suele pensar que tal vez cada etapa es independiente como si fueran piezas separadas pero la realidad es que cada una de ellas se integran de manera conjunta. Las fases de un compilador se ilustran a continuación.



*Foto
NJP*



ANALIZADOR LÉXICO O RASTREADOR

En esta fase el compilador efectúa la lectura del programa fuente generalmente conformado por un flujo de caracteres. El rastreador realiza lo que se conoce como análisis léxico: RECOLECTA SECUENCIAS DE CARACTERES EN UNIDADES SIGNIFICATIVAS LLAMADAS TOKENS, las cuales son como palabras de un lenguaje natural.

Por ejemplo: considere la siguiente línea de código, que podría ser parte de un programa en C

a [index] = 4 + 2

Este código contiene 12 caracteres diferentes de un espacio en blanco pero sólo 8 tokens:

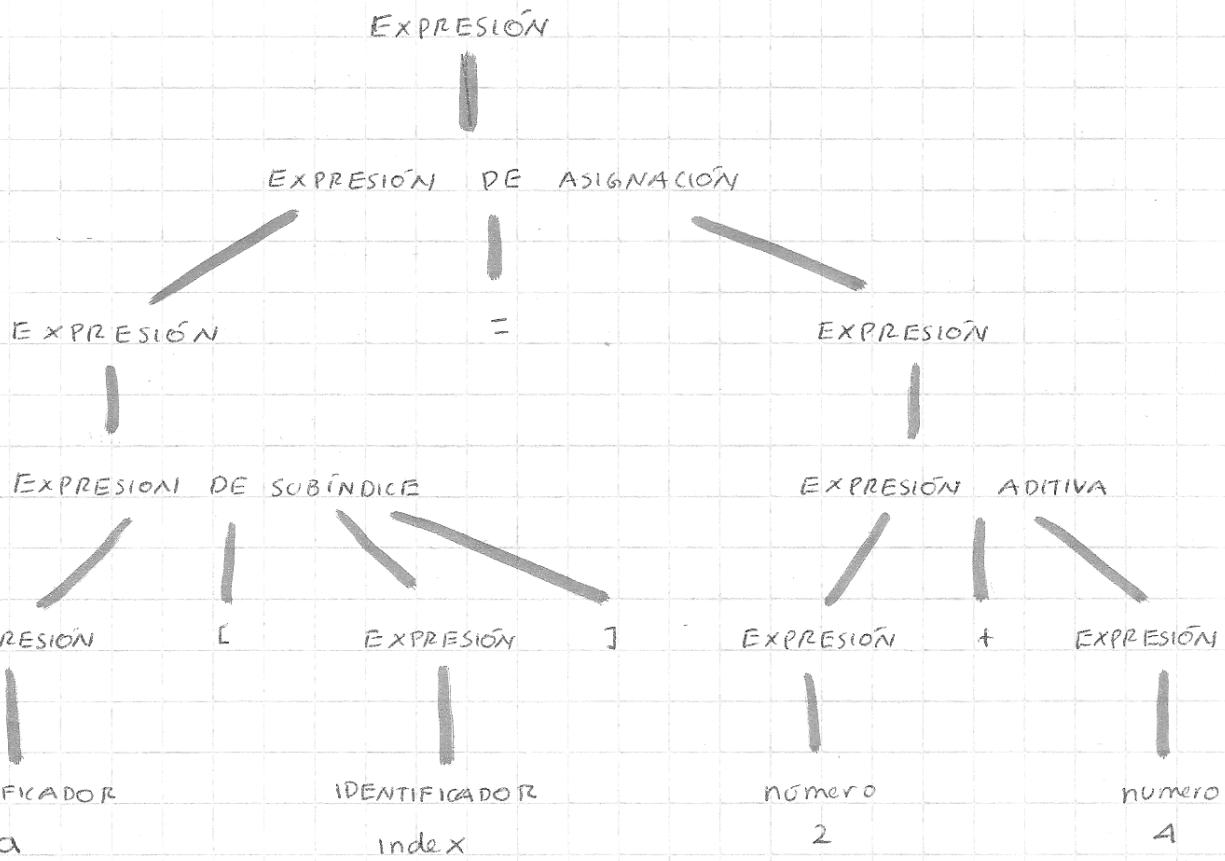
a	identificador
[carriléte izquierdo
index	identificador
]	carriléte derecho
=	asignación
4	número
+	signo más
2	número

ANALIZADOR SINTÁCTICO (PARSER)

El analizador sintáctico recibe el código fuente en forma de tokens proveniente del analizador léxico y realiza el análisis sintáctico, que determina LA ESTRUCTURA DEL PROGRAMA. ESTO ES SEMEJANTE A REALIZAR EL ANÁLISIS GRAMATICAL SOBRE UNA FRASE EN UN LENGUAJE NATURAL.

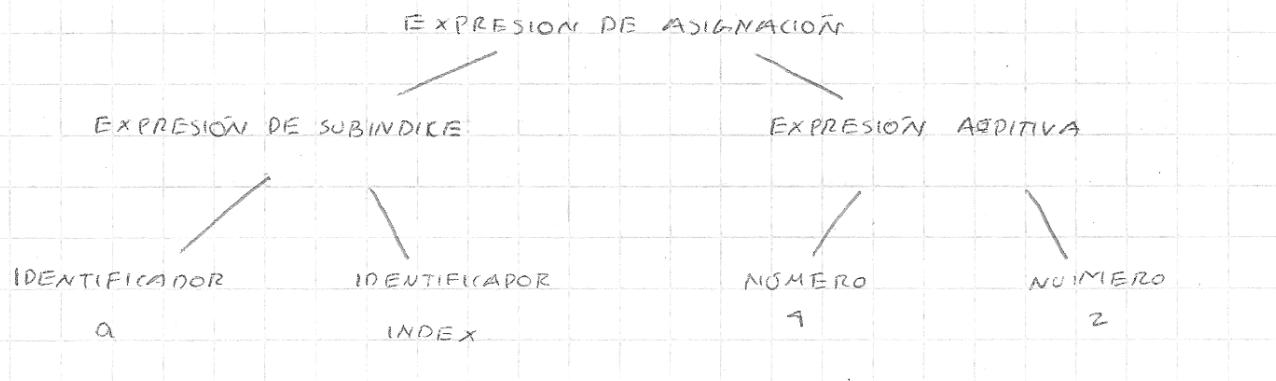
Los resultados de este análisis por lo regular se presentan como un árbol de análisis gramatical o un árbol sintáctico.

Por ejemplo: consideremos nuevamente la línea de código que ya habíamos dado. Esta estructura se puede representar como un árbol de análisis gramatical de la forma siguiente.



Los nodos internos del árbol de análisis gramatical están etiquetados con los nombres de las estructuras que representan y que las hojas del árbol representan la secuencia de tokens de la entrada.

los analizadores sintácticos tienden a generar un árbol sintáctico en su lugar el cual es una condensación de la información contenida en el árbol de análisis gramatical. un árbol sintáctico abstracto para la expresión de asignación es:



02.7
lunes
MAY

ANALIZADOR SEMÁNTICO

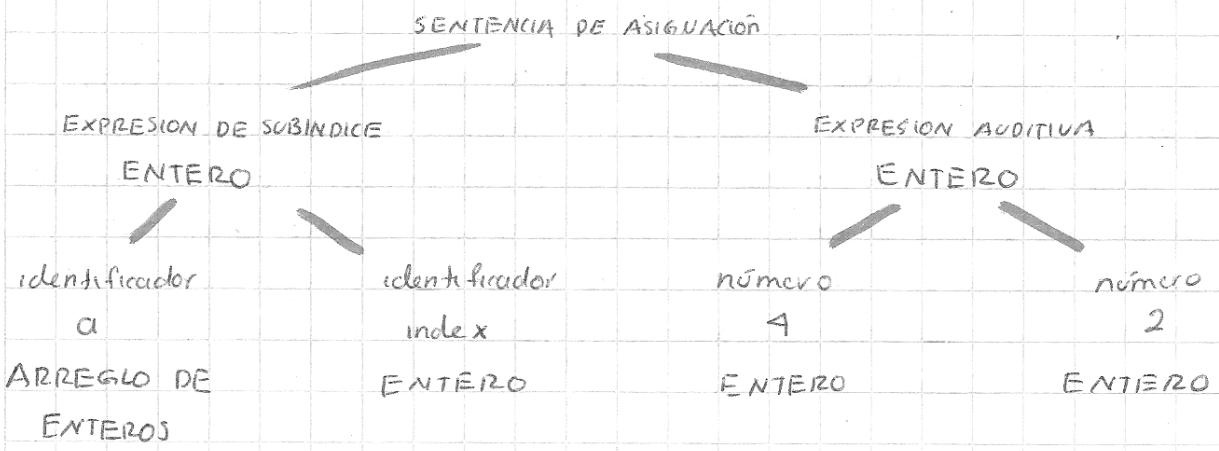
La semántica de un programa es su significado, en oposición a su sintaxis o estructura. La semántica de un programa determina su comportamiento durante el tiempo de ejecución.

Se hace referencia a tales características como semántica estática, y el análisis de tal semántica es tarea del analizador semántico. Las características típicas de la semántica estática en los lenguajes de programación comunes incluyen las declaraciones y la verificación de tipos. Las partes extra de la información como los tipos de datos que se calculan mediante el analizador semántico se llaman atributos y con frecuencia se agregan al árbol como anotaciones o "decoraciones".

Por ejemplo:

$a[\text{index}] = 4 + 2$; la información que se tendría que obtener antes del análisis de esta línea sería que a sea un arreglo de valores enteros y que index sea una variable entera.

Entonces el analizador semántico registraría el árbol sintáctico con los tipos de todas las subexpresiones y posteriormente verificaría que la asignación tuviera sentido para estos tipos, y en caso de que no fuese así devolvería un error de correspondencia de tipo. Para este ejemplo todos los tipos tienen sentido y el resultado del análisis semántico en el árbol sintáctico podría representarse por el siguiente árbol con anotaciones:



ESTRUCTURAS DE DATOS PRINCIPALES EN UN COMPILEADOR

TOKENS Cuando un analizador léxico recorre los caracteres en un token, generalmente representa el token de manera simbólica, es decir, como un valor de un tipo de datos enumerado que representa el conjunto de tokens del lenguaje fuente.

Clases de tokens:

- ▷ Palabras reservadas (if, then, ...)
- ▷ Símbolos especiales (operadores aritméticos, logarítmicos)
- ▷ Cadenas no específicas (idem, número real, ...)

La cadena de caracteres que se ha reconocido como un token determinado se denomina lexema.

Los tokens se especifican mediante expresiones regulares.

EXPRESIÓN REGULAR	TOKEN
$[a-z][a-z0-9]^*$	id
if	if
$[0-9]^+$	entero
>	mayor
\geq	mayoring
/	div

ÁRBOL SINTÁCTICO Si el analizador sintáctico genera un árbol sintáctico, por lo general se construye como una estructura estandar basada en un apuntador que se asigna de manera dinámica a medida que se efectúa el análisis sintáctico. El árbol entero puede entonces conservarse como una variable simple que apunta al nodo raíz. Cada nodo con la estructura de registros con la información recolectada por el analizador semántico y sintáctico.

TABLA DE SÍMBOLOS:

22/9
2019
Nº 9

Esta estructura de datos mantiene la información asociada con los identificadores: funciones, variables, constantes y tipos de datos. LA TABLA DE SÍMBOLOS INTERACTUA CON CASI TODAS LAS FASES DEL COMPILADOR.

ANÁLISIS LÉXICO

ANALIZADOR SINTÁCTICO

ANALIZADOR SEMÁNTICO

→ Agregara tipos de datos
y otra información

}

Pueden introducir identificadores dentro de la tabla

Utilizaran { OPTIMIZACIÓN DE
la información CÓDIGO Y GENERACIÓN

Por la tabla para
efectuar secciones
apropiadas de
código objeto .

Puesto que la tabla de símbolos tendrá solicitudes de acceso con tanta frecuencia, las operaciones de inserción, eliminación y acceso necesitan ser eficientes.

① QUE ES UNA TABLA DE SÍMBOLOS , PARA QUE SIRVE , EN QUE ETAPAS O PROCESOS DE COMPILACIÓN SE USA Y POR ÚLTIMO QUE ESTRUCTURA
POR LO GENERAL TIENE DICHA TABLA

Anteriormente hice mención a grandes rasgos en cuanto a la tabla de símbolos , en este apartado trataré de profundizar un poco mas.

La tabla de símbolos es la estructura utilizada por el compilador para almacenar los atributos asociados a los símbolos que se utilizan en el lenguaje de programación .

los atributos que esta estructura almacenará para cada símbolo pueden ser:

(ID)

TIPO (DATO)	VALOR (LEXEMA)	DIRECCIÓN DE MEMORIA	LINEA	ÁMBITO
----------------	-------------------	----------------------	-------	--------

La misión de la tabla de símbolos es colaborar en las comprobaciones semánticas y facilitar la generación de código.

ESTRUCTURA

ID Identifica de manera única cada elemento que ha sido reconocido en el análisis léxico mediante un número consecutivo o una clave

LEXEMA Hace referencia a la palabra que ha sido reconocida por el analizador léxico

TOKEN Palabras reservadas

Identificadores, Tipos de operadores, delimitadores, cadenas y constants, los cuales han sido categorizados

TIPO DE DATO

VALOR

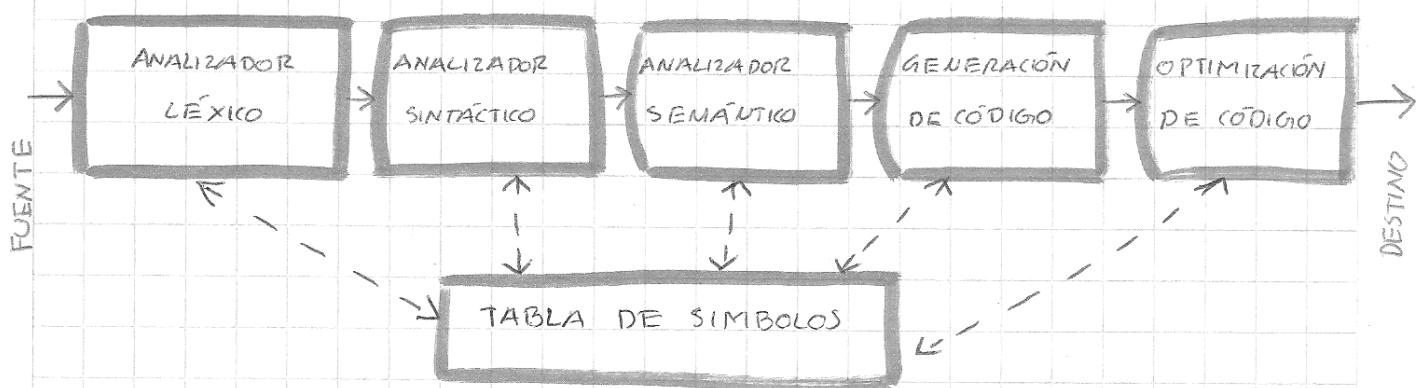
ID	LEXEMA	TOKEN	TIPO DE DATO	VALOR	LINEA
01	CADENA	1	-	-	4
02	=	2	-	-	32

20/10/2023

Las operaciones que puede realizar son

INSERCIÓN, BUSQUEDA ó CONSULTA, ACTUALIZACIÓN Y ELIMINACIÓN
la eficiencia de estos operaciones, depende de la estructura de datos
utilizada y puede hacer que el compilador consuma mucho tiempo en los
accesos a la misma.

La tabla de símbolos es de suma importancia ya que interviene en todos
los procesos del compilador la utilizan el ANALIZADOR LÉXICO,
SINTÁCTICO Y SEMÁNTICO para introducir información y el GENERADOR
DE CÓDIGO INTERMEDIO, la fase de Optimización y la de generación de
código para generar el código necesario



La tabla de símbolos contiene información útil para poder compilar, por tanto existe en tiempo de compilación y no de ejecución.

Sin embargo, en un intérprete, dado que la compilación y ejecución se producen a la vez, la tabla de símbolos permanece todo el tiempo

D) Resumen de LA I

En la materia de lenguajes y automatas 1 se abordan diferentes temas empezando con teoría de conjuntos pues ésta se usa para representar los conceptos que se tratan de entender a lo largo de la materia.

Se empieza a conocer lo que es un lenguaje formal que se refiere al conjunto de cadenas finitas de símbolos primitivos, estas cadenas son formadas por un alfabeto que es un conjunto no vacío de símbolos y por una gramática que es un conjunto de reglas para formar cadenas con símbolos de dicho alfabeto.

También se aborda el tema de gramáticas en el que existen varios tipos de éstas:

Sin restricciones

Tipo 0

Contextuales

Tipo 1

Gramáticas

Tipo 2

libre de contexto

Tipo 3

Regulares

333 *para*
10/10 *feliz*

Se centra en los lenguajes regulares que se describen con expresiones regulares que expresan las cadenas que se pueden aceptar o no por autómatas finitos que tienen estados y un control que cambia por agentes externos, en los autómatas finitos se encuentran dos tipos, el determinista que consiste en estar en solo un estado y, no determinista que consiste en estar en varios estados a la vez.

Uno de los temas más importantes es la máquina de Turing que es un autómata finito que tiene una cinta infinita donde se pueden escribir y leer datos.

Se abordan los compiladores que son traductores de código fuente a un lenguaje que pueda entender la máquina.

Se hace enfasis en 3 partes del compilador que son:

Analizador léxico:

Se encarga de que cada carácter escrito en el código pertenezca al alfabeto del lenguaje

Analizador sintáctico:

Se encarga de analizar las cadenas con base a la gramática creada para el lenguaje

Analizador semántico:

Se encarga de que las cadenas tengan sentido en el contexto del lenguaje.

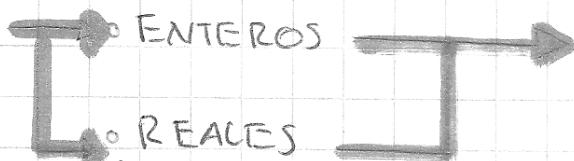
En el proyecto de la materia se desarrollo un compilador con su análisis léxico, sintáctico. Se propuso una gramática con elemento que nosotros mismos elegimos. Se tomaron en cuenta tokens, y mostrada algunos errores que se presentaban cuando no seguían la gramática.

Final
Avr
Bueno
Otro

⑤ Proponer los requisitos formales y necesarios de un micro lenguaje basado en aquellos de tipo C style

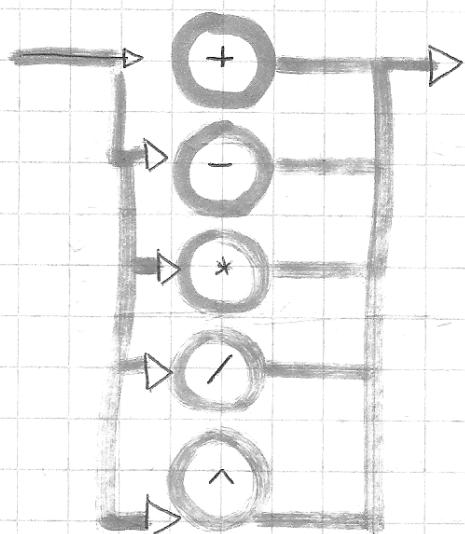
TIPOS DE DATOS Propiedad que determina su dominio, que operaciones puede realizar y como se almacena internamente por el computador, todos los variables que aparecen en un programa tienen un tipo.

Para el lenguaje prototipo los tipos de datos seleccionados son:

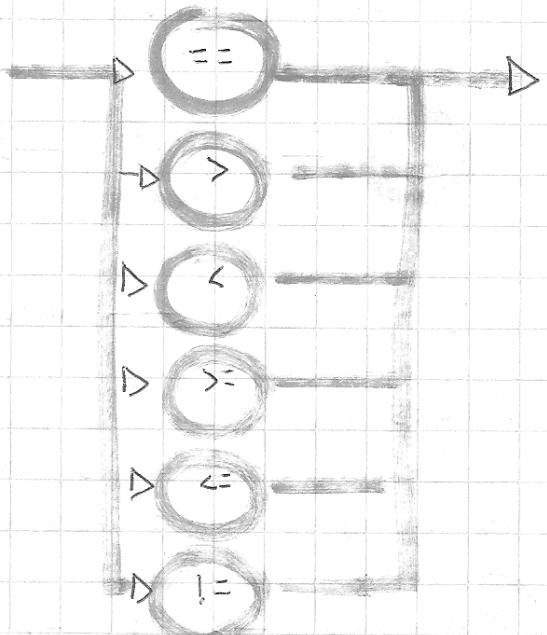


OPERADORES Los símbolos que indican como se deben manipular las variables de un lenguaje. Estos símbolos junto con las variables forman una expresión que se ocupa para realizar el cálculo de un valor.

ARITMÉTICOS



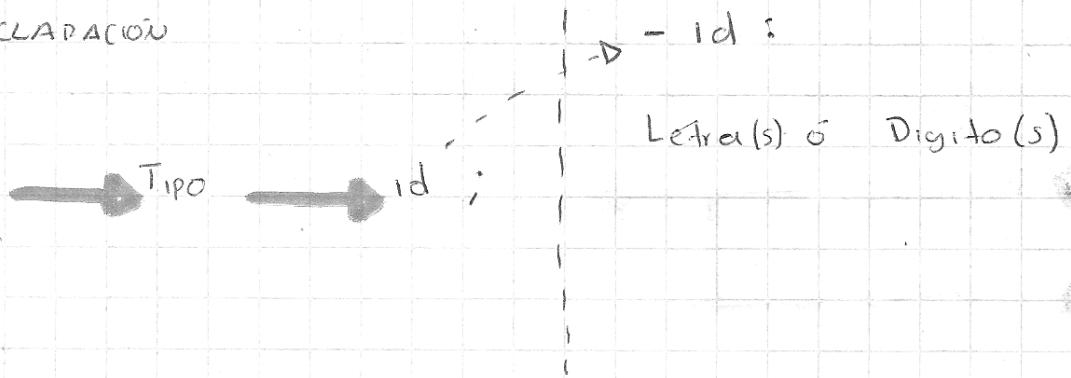
RELACIONALES



*sur
m.
7/07/2023*

IDENTIFICADORES conjunto de caracteres alfanuméricos, sirven para identificar las entidades de un programa (clases, objetos, variables, atributos, parámetros, funciones, etc...). Estos pueden ser combinaciones de letras y números.

- DECLARACIÓN



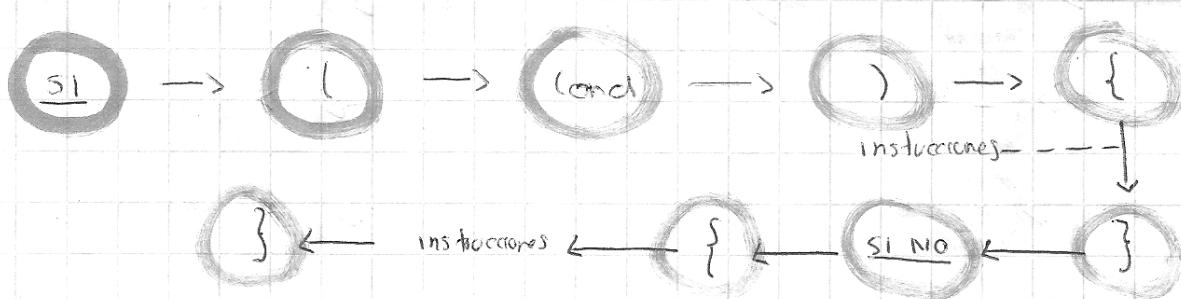
CONDICIONES Para que las secciones de código se ejecuten primero debe de cumplirse una condición.

Para nuestro lenguaje prototipo se seleccionan las siguientes:

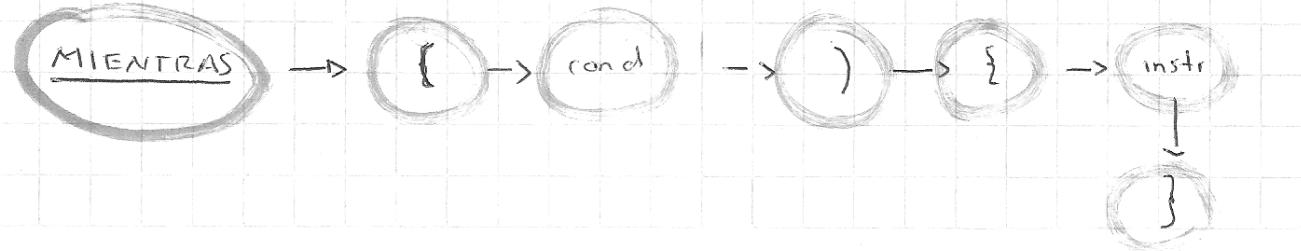
- SI

- SINO

y su estructura será la siguiente:



BUCLES Es una sentencia que se repite varias veces mientras se cumpla una condición determinada, si esta condición no se cumple, no se ejecutarán o dejará de ejecutar un conjunto o bloque



PALABRAS RESERVADAS Y ESPECIFICACIONES

close	CLASE
Principal	PRINCIPAL
Entero	ENTERO
Real	REAL
{	LLAVE IDEM
}	LLAVE D
(PARI
)	PARO
SI	SI
Sino	SINO
Mientras	MIENTRAS
+ - * / ⁿ	OPARIT
=	ASIG
,	COMA
Lid > :: = L (D1L)*	ID
Metodo	METODO
:	PUNCO
! = < > L = > = = =	OPREL

⑥ PILA SEMANTICA

Es una estructura de datos que junto con la tabla de análisis se determinan los estados, se utiliza para almacenar estados en un análisis sintáctico descendente para obtener un árbol semántico ascendente para obtener un árbol semántico.

PILAS

Son estructuras de datos que se utilizan generalmente para simplificar ciertas operaciones de programación. Estas operaciones se pueden implementar mediante un arreglo (estático) o listas enlazadas (dinámicas).

*St. 2017
Fernando
Perez*

Las pilas tienen dos operaciones básicas:

PUSH Insertar un elemento en la pila

POP Sacar un elemento de la pila

ANALISIS SEMANTICO

Detecta la validez semántica de las sentencias aceptadas por el análisis sintáctico. El analizador semántico puede trabajar de forma simultánea con el analizador sintáctico.

La semántica son el conjunto de reglas que especifican el significado de cualquier sentencia sintácticamente correcta y escrita en un determinado lenguaje.

ANALIZADOR SINTACTICO ASCENDENTE

Este tipo de análisis intenta probar todas las posibles operaciones mediante un método de fuerza bruta, hasta llegar al árbol sintáctico o bien agotar todas las opciones en cuyo caso la cadena se rechaza.

Algoritmo de pila que compiebla el orden en el que el analizador léxico le va entregando los tokens válidos y proporciona el árbol sintáctico que lo reconoce.

G) Esquemas de traducción

"Un esquema de traducción es una gramática independiente de contexto en la que se asocian atributos con los símbolos gramaticales"

En tu sistema de producciones coloca acciones semánticas entre llaves del lado derecho.

Al crear tu esquema de traducción tienes que tener en cuenta que tus atributos deben estar disponibles cuando una acción lo necesite. Debes asegurarte que tu atributo ya haya sido calculado cuando una acción lo llame.

Ejemplo

$L \rightarrow E_n$	{ print(E.val); }
$E \rightarrow E_1 + T$	{ E.val = E_1.val + T.val; }
$E \rightarrow T$	{ E.val = T.val; }
$T \rightarrow T_1 * F$	{ T.val = T_1.val * F.val; }
$T \rightarrow F$	{ T.val = F.val; }
$F \rightarrow (E)$	{ F.val = E.val; }
$F \rightarrow \text{digit}$	{ F.val = digit.lexval; }

- Atributos heredados

Son los atributos que se calculan por el principio de composicionalidad generalizado. Tienen la capacidad de compartir entre clases. Son los valores de entrada

- Atributos sintetizados

Se calculan por el principio de composicionalidad y poseen una gramática s-atribuida. Son los valores de salida

H) Generación de la tabla de símbolos y direcciones

La tabla de símbolos es una estructura de datos que almacena toda la información de un identificador en el lenguaje fuente.

Sus dos principales objetivos son revisar la semántica del código y nos ayuda en la generación de código. Estas funciones se realizan insertando y recuperando datos de la tabla, estos datos son los atributos de los identificadores. Los atributos pueden variar respecto al lenguaje de programación pero de forma general son

- Nombre de identificador
- Tipo: Las variables se ingresa el tipo de dato y para las funciones se ingresa el tipo que devuelve esta
- Dirección en tiempo de ejecución: Se almacena el dato e indica donde encontrarlo.
- Número de dimensiones: Te muestra la dimensión de un array o da cantidad de parámetros en una función
- Tamaño máximo de las dimensiones
- Tipo y forma de acceso
- Valor del descriptor de ficheros
- Número de linea donde se declaró
- Número de linea en donde se hace uso
- Campo puntero

- Nombre del identificador

Se deben asociar en la tabla de símbolos para que el analizador semántico y el generador de código puedan encontrarlo. La manera en que se almacena depende del lenguaje de programación, pero una solución general es usar un descriptor de cadenas. El descriptor contiene la posición y la longitud del string, este método puede ser lento a la hora de acceder pero ahorras almacenamiento.

Nombre de la variable													
Posición	Longitud	0	1	2	3	4	5	6	7	8	9	10	11
0	9	companylavt											
9	2												

- Dirección en memoria (offset)

Este atributo ayuda a la generación del código objeto reemplazando los nombres de identificadores por direcciones en las que se encuentran, esto sucede porque el código máquina solo trabaja con direcciones.

Las direcciones son relativas y el linker las transforma en direcciones absolutas.

- Tipo

Se usa cuando los lenguajes tienen diferentes tipos de dato. El analizador semántico se ayuda del tipo de dato para comprobar sentencias. También nos indica la cantidad de memoria que va ocupar en tiempo de ejecución.

- Número de dimensiones, de miembros o de parámetros
Son usados para la comprobación semántica. La dimensión de un array es usada en el cálculo de la ubicación de un elemento dentro del array.
El número de parámetros en la llamada y declaración de funciones debe ser igual.
- Valor máximo de las dimensiones de arrays
Se usa para la generación de código pues se usa para calcular la ubicación de un elemento en el arreglo. Si este atributo no es usado se tendrá que usar una lista dinámica con los rangos de cada dimensión del array
- Tipo y forma de acceso de los miembros de estructuras, registros, uniones y clases
Se usa para tener en cuenta las formas en las que se puede acceder a algún elemento. Las clases pueden ser public, private, protected
- Tipo de los parámetros de las funciones, procedimientos o métodos de las clases
Se usan para análisis semántico. Los lenguajes no tienen un límite de parámetros se implementa una lista dinámica para almacenarlos.
- Descriptor de ficheros
Se usa para generar código por medio del descriptor de fichero de bajo nivel al que se le asocia un identificador del fichero

- Otros atributos

Las listas de referencias cruzadas ayudan en la depuración del código. Contiene la línea de texto donde el identificador fue declarado y donde se hace referencia por primera vez, también contiene las líneas donde se usa.

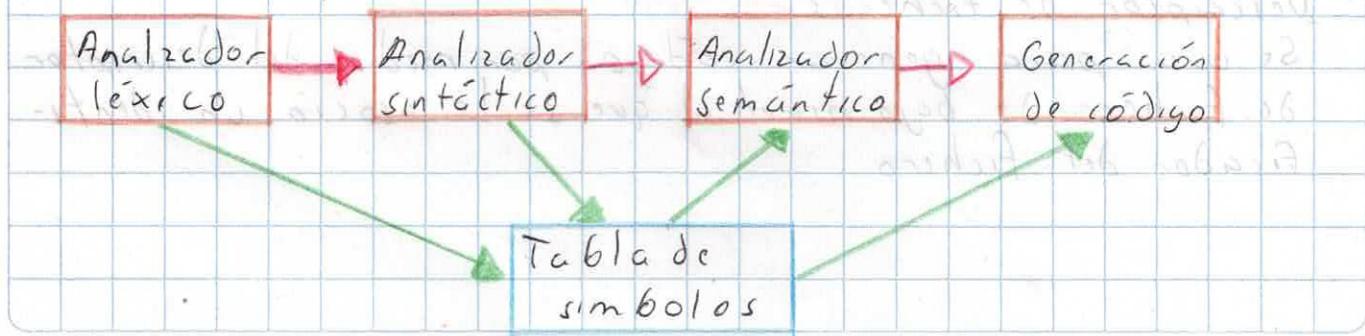
Identificador	Tipo	Dimensión	Declarada en	Referencias
cont	int	0	5	11, 22, 25
num	real	0	5	10, 11, 13, 24

Los símbolos son los identificadores del código fuente que son almacenados en la tabla de símbolos junto con sus atributos.

Las direcciones son la representación de los identificadores en código máquina pues ahí solo existen direcciones.

La tabla de símbolos está presente en la compilación e interpretación desde el análisis léxico hasta la generación de código.

El analizador léxico y sintáctico inserta los identificadores en la tabla de símbolos, el analizador semántico se apoya de la tabla para comprobar el código al contexto y el generador de código intermedio usa las direcciones de memoria de cada identificador.



1) Manojo de errores

Los errores se presentan cuando una instrucción no se ha construido correctamente y esto repercuten en el programa con su mal funcionamiento.

Las etapas de análisis léxico, análisis sintáctico y análisis semántico son las encargadas en la detección de errores.

Es una de las importantes tareas del compilador pero de las más difíciles de realizar. Un error puede provocar una reacción en cadena evitando el compilamiento completo del código.

El manojo de errores es muy importante a la hora de compilar, este nos ayuda a que al momento de encontrar un error no se pare y siga buscando todos los posibles errores que tenga el código.

Los errores semánticos son más difíciles de detectar pues la sintaxis del código es correcta pero la semántica no es la correcta. El compilador solo revisa la gramática del lenguaje, que tenga la estructura correcta pero no revisa que tenga sentido.

Los errores semánticos algunas veces provocan la finalización de un programa pero no muestra error alguno y en ocasiones el programa continua su ejecución pero como consecuencia se tiene un resultado erróneo.

Para tratar los errores semánticos se tiene la comprobación de tipos. Es capaz de encontrar todos los errores de tipo en la compilación del código y con esto ejecutar el programa sin ningún error de tipo.

La comprobación de tipos tiene que informar el error que se tiene, en donde se dio el error y seguir con el análisis del programa.

En la comprobación de tipos se tienen que realizar algunas acciones para continuar con la detección de errores.

- Conversión de tipos

Esto es necesario para usar funciones y consiste en combinar el tipo de dato para reutilizar las funciones necesarias para continuar.

- Coerción

Es una conversión de tipos que el compilador realiza para su correcta ejecución. También existe la conversión explícita que sucede cuando el programador realiza la conversión.

- Sobrecarga de operadores

Se determina el tipo de dato en todas las expresiones involucradas en la sobrecarga.

- Funciones polimórficas.

Son funciones que pueden cumplir a distintos tipos.

Un comprobador de tipos tiene que realizar o cumplir dos objetivos

1- Asignación de tipos

Declarar cada una de las expresiones

2- Evaluación y comprobación de tipos

Realizar las acciones de conversión, coerción, sobre carga y funciones polimórficas en las expresiones, funciones y sentencias.

Existen diferentes tipos de errores semánticos, se pueden presentar los siguientes:

- Conversiones de tipos no permitidas

in

```
int x  
x = 4.5
```

El tipo de dato que se tiene que ingresar en x es un `int` pero tiene un error con un tipo de dato ingresando `double`

- Variables usadas y no definidas

```
int x, r  
r = x + b
```

La variable `b` no está definida por lo tanto tiene un error la sentencia

- Operando de tipos no compatibles

`IF ('abc' * 5) a = 3`

El operando no puede ser usado y no tiene sentido la expresión que se da.

Se tiene que considerar este tipo de errores a la hora de realizar nuestro compilador.

- Errores léxicos

Se detectan cuando un carácter no pertenece al alfabeto del lenguaje en un identificador, número y errores en palabras reservadas

Ejemplos

`int 5ab@` Error de nombre

`5+7,8` Número incorrecto

`while` Error en palabra reservada

- Errores sintácticos

Se detectan cuando una expresión aritmética o paréntesis no estén equilibrados

Ejemplo

`IF (x == 5) ^`

`z = 3;`

`h = 4;`

`t`

Error al no colocar llave de apertura.