

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

Celaya, Guanajuato, 25 / marzo / 2025

ASUNTO: **SOLICITUD DE ACTIVIDADES**

LENGUAJES Y AUTÓMATAS II

DOCENTE DESIGNADO: ISC. RICARDO GONZÁLEZ GONZÁLEZ
SEMESTRE ENERO-JUNIO 2025

ACTIVIDAD 6 (VALOR 44 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 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.

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.

- a. **LA OPORTUNIDAD.** SI EL TRABAJO FUE ENTREGADO OPORTUNAMENTE.
- b. **LA COMPRENSIÓN.** SE VALORARÁ EL GRADO DE COMPRENSIÓN DEL TEMAS ANALIZADOS.
- c. **LA CALIDAD.** SI LAS EVIDENCIAS ENVIADAS CORRESPONDEN A LA CALIDAD ESPERADA PARA ESTE NIVEL PROFESIONAL QUE SE CURSA.
- d. **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.
- 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 SUFFICIENTE PARA REALIZAR ESTA ACTIVIDAD Y SUMAR PUNTOS IMPORTANTES A SU CALIFICACIÓN DE ESTA EVALUACIÓN.

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





CONSIDERACIONES.

CADA UNO DE LOS PUNTOS ANTERIORES DEBE SER DESARROLLADO CON LA PROFUNDIDAD ACORDE A UN NIVEL PROFESIONAL, Y APEGÁNDOSE COMPLETAMENTE A LAS DIRETRICES 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, SUPONDRÍA 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.**



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 , MÁS EL GRUPO (-A , -B , -C)
DOCUMENTO	: A1-ACTIVIDAD 1, P1-PRACTICA 1, R1-REPORTE 1, T1-TAREA 1, PG1-PROGRAMA, ETC. (CAMBIANDO EL NÚMERO CONSECUТИVO 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 PERÍODO SEMESTRAL EN CURSO: ENE-JUN

EJEMPLO :

SI EL TRABAJO SE SOLICITÓ EN EQUIPO.

2025-03-25_TNM_CELAYA_LAI-A_A6_EQUIPO_99_9999999_PEREZ_PEREZ_JUAN_ENE-JUN25.PDF

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

SI EL TRABAJO SE SOLICITÓ INDIVIDUALMENTE.

2025-03-25_TNM_CELAYA_LAI-A_A6_9999999_PEREZ_PEREZ_JUAN_ENE-JUN25.PDF





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.

Julián Pérez
Julián Pérez

MUY IMPORTANTE:

1. DESPUES 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 ANTICIPACIÓN 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.



ACTIVIDAD 6

análisis Semántico

Equipo:

- 21030223 Macías Serilla Diana Nathasha
- 21030104 Castañeda Pérez Cristian Eduardo
- 21031027 Aguilar Flores Abel

Profesor: Ricardo González González

Fecha: 31 / Marzo / 2025

Julián Pérez Pérez

ÁRBOL DE EXPRESIONES

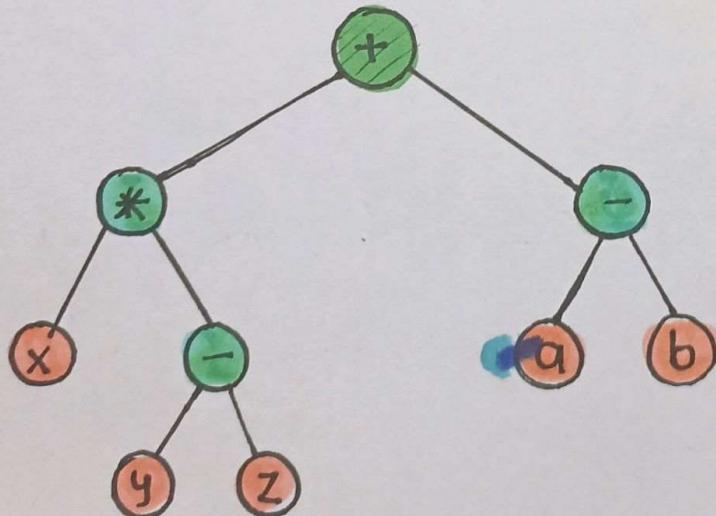
Tema 4.1

Julián Díaz

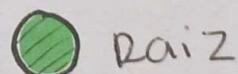
Los árboles de expresiones son estructuras de datos que representan o que definen código. Por ejemplo, introduzcamos a un árbol la siguiente expresión:

$$(x * (y - z)) + (a - b)$$

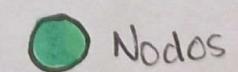
La cual tiene subexpresiones por los parentesis, el resultado es:



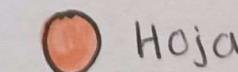
Partes de un
árbol de expresión



Raiz



Nodos internos



Hojas

Al introducir cada expresión debemos de tomar en cuenta lo siguiente:

- La raíz siempre debe ser un operador
- Las hojas siempre deben ser operandos
- Los nodos deben tener etiqueta
- Si un operador tiene mayor prioridad que la raíz se coloca como hijo, si tiene menor o igual, se coloca como padre.
- Un nodo puede contener como hijo otro subárbol que contiene una pequeña expresión.

Recorridos sobre un árbol

Recorrido o barrido es una de las operaciones más importantes sobre un árbol, quiere decir que es moverse a través de todos los nodos del árbol visitando a cada uno una única vez y en algún orden determinado.

Hay dos formas básicas de recorrer un árbol: en **amplitud** y en **profundidad**.

→ Recorrido en Amplitud

En este se recorre el árbol por niveles, del Superior a los niveles inferiores.

Recorrido en Profundidad

En este, se recorre el árbol por subárboles. Hay tres formas, y cada una tiene una secuencia distinta para analizar el árbol:

Preorden

1. RID

1. Examinar Raiz
2. Subárbol izquierdo
3. Subárbol derecho

Inorden

IRD

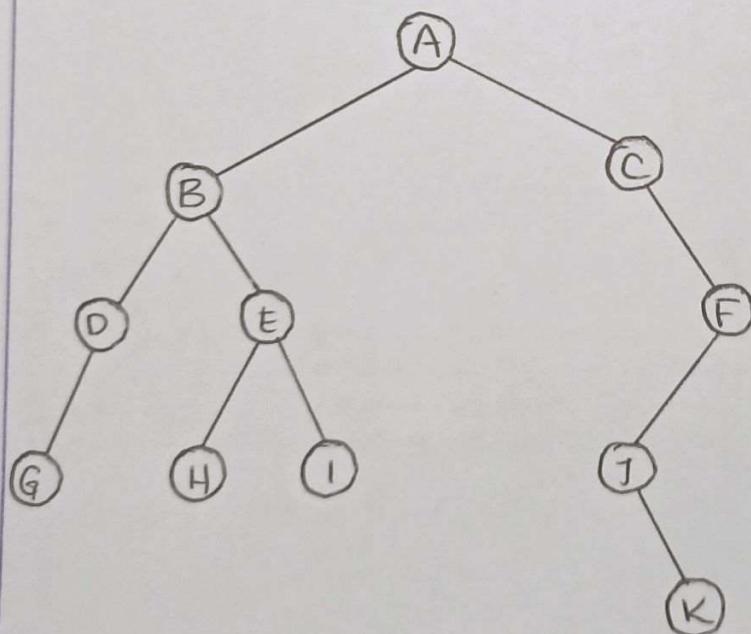
1. Subárbol izquierdo
2. Examinar Raiz
3. Subárbol derecho

Postorden

IDR

1. Subárbol izquierdo
2. Subárbol derecho
3. Examinar Raiz

Ejemplo. Del sig. árbol, realizar los recorridos antes mencionados.



Recorridos:

Por amplitud:

A B C D E F G H I J K

Por profundidad:

Preorden

ABD GEH ICJ FK

Inorden

GDBHEIACJKF

Postorden

GDHIEBKJFCA

TEMA 4.2

Acciones semánticas

DE UN ANALIZADOR
SINTÁCTICO

Guillem López

La salida del analizador sintáctico es una representación del árbol sintáctico que reconoce la secuencia de Tokens suministrada por el análisis léxico.

Las acciones semánticas se encargan de que los tipos que intervienen en las expresiones sean compatibles o que los parámetros reales de una función sean coherentes con los formales.

Entre estas acciones tenemos:

- Acceder a la tabla de símbolos
- Chequeo de tipos
- Generar código intermedio
- Generar errores cuando se producen

Este último es de los más complicados desde

el punto de vista de los compiladores, pues cuando el compilador encuentra un error debe recuperarse y seguir en búsqueda de más errores.

Los objetivos de este manejador de errores son:

- Indicar los errores de forma clara y precisa.
Aclarar el tipo de error y su localización.
- Recuperarse del error, para poder seguir examinando la entrada.
- No ralentizar significativamente la compilación.

En definitiva, realiza casi todas las operaciones de la compilación; cada operación se puede clasificar en:

- Sentencias de declaración
- Sentencias "ejecutables"
- Funciones y procedimientos
- Identificación de variables

COMPROBACIONES DE TIPOS EN EXPRESIONES

4.3

La fase de análisis semántico tiene que validar un programa, además de ser sintácticamente correcto, es además coherente con su contexto.

La comprobación de tipos en un compilador es una tarea clave dentro del análisis semántico, que implica la inferencia de tipo y comprobación para garantizar que un programa cumpla con las reglas del lenguaje. Según Louden (2004), un tipo de datos se define como un conjunto de valores y sus operaciones, representado por una expresión de tipo (como integer o float), e incluyendo estructuras más complejas como arreglos o registros. La comprobación de tipos actúa como un control semántico esencial en todas las operaciones realizadas en un programa.

La comprobación de tipos es una fase crucial en el análisis semántico. Se lleva a cabo después del análisis sintáctico y antes del análisis semántico. El resultado de la comprobación de tipos es una lista de errores y advertencias que se presentan al usuario. Los errores indican que hay algo mal en el código, mientras que las advertencias indican que algo es incorrecto pero no es un error. La comprobación de tipos también se encarga de garantizar que los tipos de datos sean consistentes entre sí y que las operaciones sean legítimas para los tipos de datos involucrados.

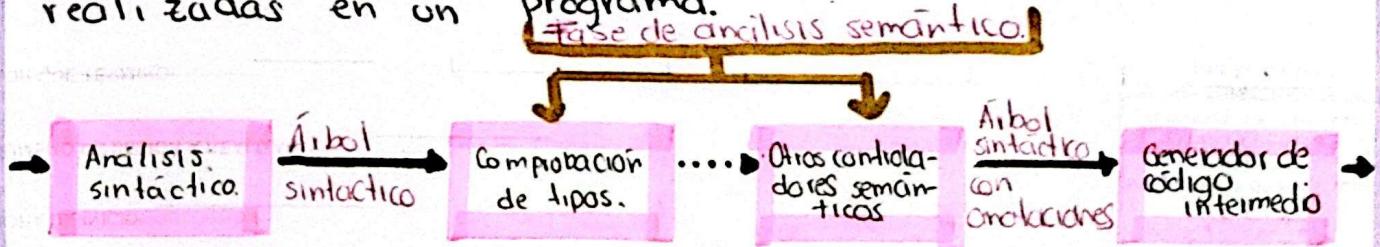


Imagen 1. Diagrama de fases

La comprobación de tipo en un compilador puede integrarse con otros controladores semánticos según las necesidades del lenguaje que se está construyendo. El momento en que se realiza depende de las decisiones del diseñador.

TIPOS

Un tipo de dato consiste en un conjunto de valores con operaciones definidas sobre ellos, como los números enteros y reales con suma, resta, división, etc.

Tareas de verificación

La comprobación de tipo puede realizarse de dos maneras:

- **Estática:** Se verifica en tiempo de compilación
- **Dinámica:** Se verifica en tiempo de ejecución

SISTEMA DE TIPOS

"Un sistema de tipos son reglas que hay que aplicar para asignar las expresiones de tipos, bien sean simples y complejas, a las distintas estructuras de un lenguaje. Por tanto, esta comprobación de las reglas se define para cada estructura del lenguaje es la comprobación de tipos."

Como nos indica Aho et al (2008) estas reglas pueden ser de dos tipos: de síntesis o de inferencia.

Por lo tanto se deduce que hay que realizar varias tareas:

1. Gestión de declaraciones: consiste en la inserción en la tabla de símbolos de los identificadores y sus tipos teniendo en cuenta el ámbito en el que se han generado.

2. Verificación de tipos: a partir de las reglas generadas, es decir, en la tabla de símbolos, verificar que se cumplen las reglas asociadas a las distintas estructuras del código fuente.

3. Inferencia de tipos: en los lenguajes que no regulan la definición de tipos, cuando hay sobrecarga de operadores o polimorfismo, hay que inferir el tipo de un dato o estructura en su función de los distintos tipos de datos que intervienen.

Como consecuencia del sistema de tipos de datos que se implementan para el lenguaje concreto; se habla de lenguajes fuertemente tipificados o débilmente tipificados.

LENGUAJES FUERTEMENTE TIPIFICADOS

Estos lenguajes aplican una verificación estricta de tipos en tiempo de compilación, evitando el uso de datos de distintos tipos sin una conversión explícita. Se consideran seguros respecto al tipo, ya que detectan errores tanto en compilación como en ejecución.

LENGUAJES DÉBILMENTE TIPIFICADOS

Ofrece mayor flexibilidad en el uso de tipos de datos, verificando solo en tiempo de ejecución. Al no realizar comprobaciones en compilación, puede permitir operaciones entre distintos tipos sin conversión explícita.

Imagen 2. Tabla comparativa de lenguajes fuertes/débiles tipificados.

REGLAS DE SÍNTESIS DE TIPOS

Estas reglas construyen el tipo de una expresión a partir de los tipos de las subexpresiones y por tanto se requiere que estas subexpresiones estén declaradas previamente.

Son las más habituales, y son de forma $E_1 \rightarrow T^* E_2$, el tipo que tenga E_1 dependerá de los tipos que tenga T y E_2 y de lo que tenga la regla respectiva para el producto cuando son del mismo tipo, por ejemplo enteros o cuando uno de ellos es real.

REGLAS DE INFERRANCIA DE TIPOS

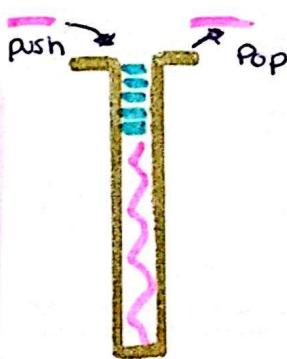
Determina el tipo de construcción a partir de las formas en que se utiliza. Este mecanismo se utilizan los lenguajes que no requieren la declaración de tipos como por ejemplo ML, que comprueban los tipos pero no es necesario que se declaren los nombres de variables.

PILA SEMANTICA EN UN ANALIZADOR SINTACTICO

4.4

La pila semántica es una estructura de datos utilizada en la fase de análisis semántico de un compilador. Se usa para almacenar información relacionada con los operadores y expresiones analizadas, permitiendo realizar comprobaciones y construir representaciones intermedias del código.

Una pila es una estructura de datos de tipo LIFO (Last In, First Out), lo que significa que el último elemento en entrar es el primero en salir.



Los operaciones que se pueden realizar con una pila son:

- PUSH (pila,elemento): Introduce un elemento en la pila. También se le conoce como poner o meter.
- POP (pila): Elimina un elemento como sacar o quitar.

de la pila. También se le conoce como vaciar la pila.

- Vacia (pila): Función booleana que indica si la pila está vacía o no.

USO DE PILA EN UN ANALIZADOR SINTACTICO.

Un analizador sintáctico es un autómata de pila que reconoce la estructura de una cadena de código.

Durante el análisis semántico, se verifican aspectos como:

- Compatibilidad de tipos (ejemplo: no sumar enteros con cadenas de texto).
- Corrección de operaciones según las reglas del lenguaje.
- Aplicación de conversiones implícitas de tipos cuando sea necesario.

REGLAS SEMÁNTICAS

Las reglas semánticas definen el significado de cada construcción válida en el lenguaje de programación. Se aplican a los nodos del círculo sintáctico y puede tener efectos como:

- Actualizar variables globales
- Imprimir valores en pantalla
- Realizar conversión de tipo.

Ejemplo

Si el compilador encuentra $x = 5 + "Hola"$, las reglas semánticas deberían detectar el error porque no se puede sumar un número entero y una cadena de texto.

MÉTODOS DE ANÁLISIS SINTÁCTICO

Los analizadores pueden trabajar de dos maneras principales:

Funciona llamando al analizador léxico para recibir símbolos de entrada y comprobar su validez según la gramática del lenguaje.

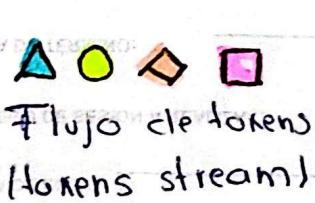
En la verificación semántica, la pila almacena información sobre los operadores y expresiones, asegurando que las reglas del lenguaje se cumplan.

Ejemplo:

Si el compilador encuentra la expresión $a + b * c$, la pila ayuda a organizar las operaciones según la precedencia de los operadores y verificar la compatibilidad de tipos.

4. OBJETIVO DE LA PILA SEMÁNTICA

El principal propósito de la pila semántica es construir el árbol de análisis sintáctico, el cual representa la estructura jerárquica de código. Sin embargo, en la práctica, se genera un árbol de sintaxis abstracta, que contiene solo la información relevante para la ejecución del programa.



Analizador
sintáctico
(Parse)

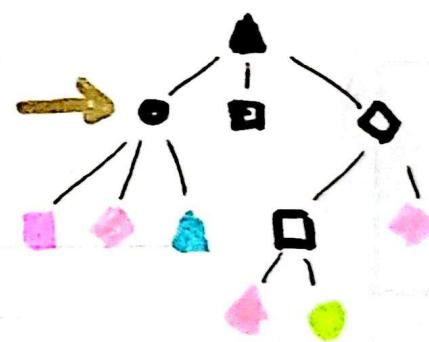


Imagen 3. Árbol
de sintaxis

Árbol de sintaxis
concreta (Parse
tree).

Analizadores descendentes (Top-Down)

- Inician desde la raíz del árbol y van descomponiendo la estructura hasta llegar a los símbolos terminales (las partes más pequeñas del código).
- Se usan en gramáticas simples y fáciles de predecir.

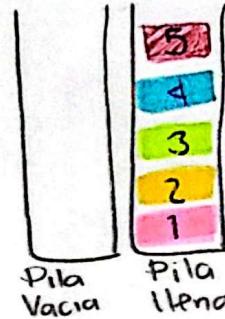
Analizadores ascendentes (Bottom-Up)

- Comienzan desde los símbolos más pequeños y van construyendo el árbol hacia arriba.
- Utilizan una pila para almacenar los símbolos y operadores mientras aplican las reglas de reducción.
- Son más potentes y pueden manejar lenguajes más complejos

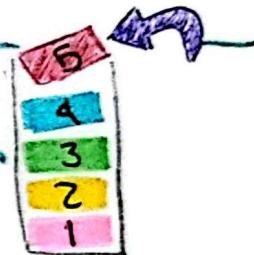
Ejemplo:

Si el código es $3+4*5$, un parser ascendente primero almacena $3, +, 4, *, 5$ en la pila y luego aplica las reglas de precedencia para procesar $4*5$ antes de sumar 3.

Objetivo teórico: Es construir un árbol de análisis sintáctico, este raramente se construye como tal, sino que las rutinas semánticas integradas van generando el árbol de Sintaxis abstracta se especifica mediante una gramática libre de contexto.



Ventajas. Los problemas de integración entre los subsistemas son su mantenimiento costoso y muchas de ellos no se solucionan hasta que la programación alcanza la fecha límite para la integración total del sistema se necesita una memoria auxiliar que nos permita guardar los datos para poder hacer la comparación



Pilas Semánticas en un Analizador Sintáctico.

Las pilas y colas son estructuras de datos que se utilizan generalmente para simplificar ciertas operaciones de programación. Estas estructuras pueden implementarse mediante arrays o listas enlazadas

Pop (para extraer un elemento)

Push (para introducir un elemento)

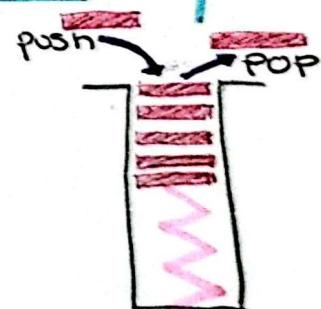


Imagen 2. Resumen de pilas semánticas.

Esquemas de Traducción

Los esquemas de traducción son una herramienta fundamental en los compiladores y traductores de lenguajes de programación. Se basa en gramáticas independientes del contexto y permiten la asociación de atributos a los símbolos gramaticales, junto con la inserción de acciones semánticas que guían el proceso de traducción.

Definición:

Un esquema de traducción es una gramática libre de contexto en la que se incluyen fragmentos de código, denominados **acciones semánticas**, que se encuentran encerradas entre llaves {} dentro del lado derecho de las producciones. Estas acciones son responsables de evaluar atributos y guiar la generación de código intermedio o final.

Características:

- **Asociación de atributos:** Cada símbolo debe tener atributos sintetizados o heredados.
- **Ejecución controlada de acciones:** Las acciones Semánticas se ejecutan en un orden preferido, generalmente siguiendo un recorrido en preorden del árbol sintáctico.
- **Complemento de las definiciones por sintaxis:** A diferencia de las definiciones dirigidas por sintaxis, en un esquema de traducción las definiciones dirigidas

Se pueden especificar de manera más explícita el orden de evaluación de atributos, y ejecución de acciones.

Proceso de uso de Un Esquema de Traducción.

Requisito para hacer un esquema de traducción.

conocer el lenguaje Símbolos gramaticales Acciones Semánticas Reglas.

para implementar un esquema de traducción, se sigue el siguiente procedimiento:

- 1º Construcción del árbol de análisis sintáctico, ignorando temporalmente las acciones semánticas.
- 2º Adición de acciones semánticas en los nodos correspondientes dentro del árbol de análisis sintáctico.
- 3º Recorrido del arbol en preorden, ejecutando las acciones en el orden adecuado

TRADUCCIÓN DESCENDENTE.

En un esquema de traducción descendente, las acciones semánticas se ejecutan en el mismo momento en que se explicarían un símbolo no terminal en el análisis sintáctico. Esto permite calcular atributos heredados antes de que los símbolos sean procesados y atributos sintetizados después de su evaluación.

Eliminación de recursiva Izquierda

Debido a que los operadores aritméticos suelen ser **asociativos por la izquierda**, se emplean gramáticas recursivas por la izquierda para manejar expresiones. Sin embargo en un esquema de traducción descendente, es necesario **eliminar la recursiva izquierda** para garantizar que los atributos se calculen en el orden correcto.

Evaluación de Atributos.

En el análisis sintáctico descendente:

- **Los atributos heredados** deben calcularse antes de expandir un símbolo.
- **Los atributos sintetizados** deben calcularse después de evaluar todos los símbolos de los que dependen.

En términos prácticos, la ejecución de las acciones semánticas sigue un recorrido en **primero - profundo** del árbol sintáctico.

4.6

Generación de la tabla de símbolos y de direcciones

Las tablas de símbolos (también conocidas como tablas de identificadores o tablas de nombres), son una estructura de datos muy importante en el proceso de compilación y traducción de lenguajes de programación.

Es utilizada para almacenar información sobre los identificadores (variables, funciones, tipos, etc.), además permite administrar los recursos asociados a las entidades que manipulará el programa.

Estructura

- Tablas Hash
- Listas enlazadas
- Estructuras jerárquicas

- Facilita la búsqueda rápida de identificadores.
- Usadas cuando hay necesidad de un mantenimiento dinámico de los símbolos.
- Permiten manejo de entornos anidados en lenguajes de programación estructurados.

• • • COMPONENTES • • •

Una tabla de símbolos típica contiene o puede contener los siguientes campos.

→ Nombre del identificador:

Es decir, nombre del símbolo (variable, función, constante, etc.).

→ Tipo de dato:

Si es entero, flotante, cadena, texto, etc.

→ Ámbito (Scope)

Indica en qué parte del programa es válido el identificador.

→ Tamaño

Espacio que ocupa en memoria.

→ Dirección de memoria.

Ubicación en memoria donde se almacena la variable o función.

• • • GENERACIÓN DE LA TABLA • • •

La tabla de símbolos se genera conforme el analizador sintáctico y semántico recorren el código fuente. A medida que se encuentran nuevas declaraciones, estas se agregan en la tabla.

► Funciones

- ① Inserción → Cuando se declara una variable o función, se agrega a la tabla con sus atributos.

Ejemplo:

int x; // se inserta en la tabla: {nombre: "x", tipo: "int", dirección: 0x1000, ...}

equipo
100%

- ② Búsqueda → Verifica si un identificador existe en el ámbito actual antes de su uso.

Ejemplo

x=5; // El compilador busca "x" en la tabla de símbolos para validar su declaración previa.

- ③ Actualización → Modifica atributos como el tipo o dirección cuando sea necesario.

- ④ Gestión de ámbitos → Se implementa mediante una pila de tablas de símbolos o estructurados anidados.

Ejemplo

{ // se agrega a una nueva tabla de ámbito local.
 int y;
} // Al salir del bloque se elimina de este ámbito.

► Asignación de direcciones

Cada identificador en la tabla se asocia con una dirección de memoria. Esta puede realizarse de las siguientes maneras.

- Asignación Estática: Dirección fija a cada identificador en tiempo de ejecución

- Asignación Dinámica: Se asignan direcciones en tiempo de ejecución, por ejemplo heap o stack.
- Espacios de memoria separados: Segmentación en código, pila y datos para optimizar el uso de memoria.

→ Ejemplo básico

Lenguaje C

```
int main () {  
    int a=10; // tabla{nombre: "a", tipo: "int", dirección: stack[-9]}  
    float b = 3.14; // tabla{nombre: "b", tipo: "float", dirección: stack[-8]}  
    return 0;  
}
```

4.7

...MANEJO DE ERRORES SEMÁNTICOS...

Julián López

El manejo de errores semánticos se refiere a la semántica del lenguaje de programación, que normalmente no está descrita por la gramática. Estos errores ocurren cuando la sintaxis del código es correcta, pero el significado no es el esperado, es decir presenta inconsistencias lógicas o de tipos que impiden su ejecución adecuada.

→ Tipos de errores semánticos

VARIABLES NO DECLARADAS

```x = 5;````



Marcará error si "x" no fue declarada.

TIPOS INCOMPATIBLES

```int a = "hola";````



Error: asignación de cadena a un entero.

ÁMBITO INCORRECTO

```{  
 int y = 5;  
}  
Print(y);````



Error: "y" no existe fuera del bloque

## Operaciones ilegales

res = 10/0;



Error: Operación inválida.

Cantidad incorrecta de argumentos en funciones

```
Void saludar(string n){
 print("Hola"+n);
}
```

```
int main(){
 Saludar();
```



Error: falta el argumento en el llamado a la función

Acceso fuera de los límites de un arreglo.

```
int num = [10, 20, 30];
Console.log(num[6]);
```



Error: intentar acceder a una posición inválida en el array.

## → Estrategias de manejo de errores

Para evitar que los errores semánticos interrumpan el proceso de compilación se implementan diferentes estrategias de detección y recuperación, por ejemplo.

| Estrategia                     | Descripción                                                                                              |
|--------------------------------|----------------------------------------------------------------------------------------------------------|
| Detección temprana de errores. | Uso de reglas estrictas que sirvan para el análisis semántico para identificar errores lo antes posible. |

| Estrategia                   | Descripción                                                                |
|------------------------------|----------------------------------------------------------------------------|
| Mensajes de error detallados | Indicar el tipo de error, la ubicación en el código y posibles soluciones. |
| Recuperación por pánico      | Ignorar tokens hasta encontrar un punto seguro en la sintaxis.             |
| Corrección automática        | Algunos compiladores pueden sugerir soluciones a errores comunes.          |
| Uso de tabla de símbolos.    | Validar identificables y tipos antes de la generación de código.           |

## → Importancia del manejo de errores semánticos.

El manejo de errores semánticos mejora la robustez y usabilidad del compilador, facilitando el desarrollo de programas sin errores y asegurando que el código generado sea correcto y eficiente.

# PRACTICA

## TEMA 4.3 COMPROBACIONES DE TIPOS EN EXPRESIONES.

### Ejercicio No.2. Uso de Pila Semántica

> Si se tiene la expresión

$$x = (5+3)^2 + \frac{(10/3)^2}{2}$$

? Encuentre su A.A.S

? Ejemplifíquelo como queda la pila completa antes de su evaluación

- A) con RID

= B) con IDR

> Si se tiene la expresión

$$x = (5+3)^2 + \frac{(10/3)^2}{2}$$

? Encuentre su A.A.S

? Ejemplifíquelo como queda la pila completa antes de su evaluación

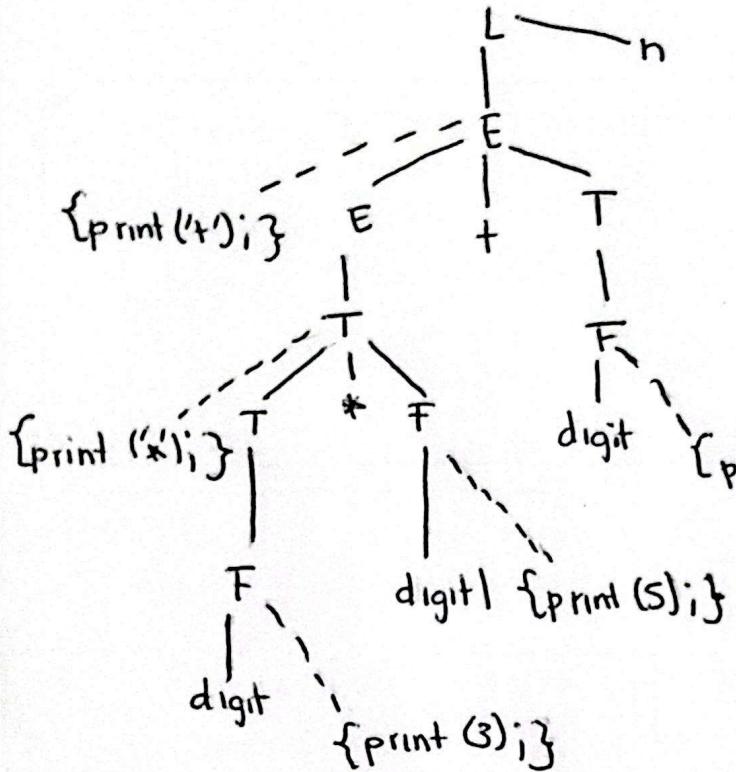
- A) con RID

= B) con IDR

# EVALUACIÓN DE LA PILA (SIVaVA).

| =  | =     | =     | =     | =     | =     | =     | =     | =      | =        |
|----|-------|-------|-------|-------|-------|-------|-------|--------|----------|
| +  | +     | +     | +     | +     | +     | +     | +     | +      | +        |
| 1  | 1     | 1     | 1     | 1     | 1     | 1     | 5.555 | 5.555  |          |
| 2  | 2     | 2     | 2     | 2     | 2     | 2     | n     | n      |          |
| n  | n     | n     | n     | 11.11 | 11.11 | 11.11 | 2     | 2      |          |
| 2  | 2     | 2     | 2     | n     | n     | n     | *     | *      |          |
| 1  | 3.33  | 3.33  | 3.33  | 2     | 2     | 2     | 3     | 3      |          |
| 3  | =     | n     | n     | *     | *     | *     | 5     | 5      |          |
| 10 | +     | 2     | 2     | 3     | 3     | 3     | x     | =      |          |
| 1  | 5.555 | =     | =     | =     | 5     | 5     |       |        | 230.55   |
| 2  | n     | +     | +     | +     | x     | x     | =     | x      |          |
| *  | 2     | 5.555 | 5.555 | 5.555 | =     | =     | +     | =      |          |
| 3  | *     | n     | n     | n     | +     | +     | 5.555 | 230.55 |          |
| 5  | 3     | 2     | 2     | 2     | 5.555 | 5.555 | 225   | x      |          |
| x  | 5     | 15    | 15    | 15    | 225   | 225   | x     |        | x=230.55 |
|    | x     | x     | +     | x     | x     | x     |       |        |          |

## Árbol con esquema de traducción o acción a realizar



Reglas utilizadas y su secuencia

1.  $L \rightarrow En$
2.  $E \rightarrow \{ \text{print}('t') ; \} E_1 + T$
3.  $E \rightarrow T$
4.  $T \rightarrow \{ \text{print}('*') ; \} T_1 * F$
5.  $T \rightarrow F$
6.  $F \rightarrow (E)$
7.  $F \rightarrow \text{digit} \{ \text{print}(\text{digit.lexval}) ; \}$

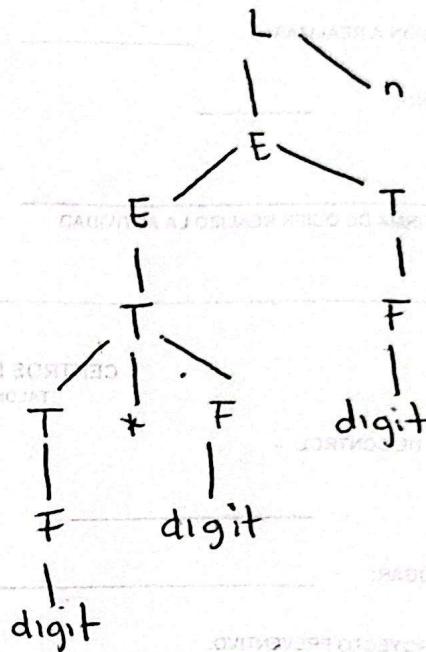
## Árbol con acciones embebidas

# 1.5 Esquemas de traducción

## Descripción

- En base al esquema de traducción anterior y basado en esas reglas, encuentre como se generaría la expresión.
- $3 * 5 + 4$

## Árbol de análisis sintáctico



(c) iscamc

Árbol Sintáctico  
sin acciones embebidas

# Práctica

→ Ejercicios Tema 4.6: Generación de Tabla de símbolos y direcciones.

Dado el siguiente código:

```
int global = 10;
void funcion(int param){
 int local = 20;
 float pi = 3.14;
}
```

- Generar la tabla de símbolos con los campos: Nombre, tipo, ámbito, dirección (simulada).

| Nombre  | Tipo   | Ámbito    | Dirección |
|---------|--------|-----------|-----------|
| global  | int    | global    | 0x1000    |
| funcion | void() | global    | 0x0000    |
| param   | int    | funcion() | 0x2000    |
| local   | int    | funcion() | 0x2004    |
| pi      | float  | funcion() | 0x2008    |

## Explicación

global → está en memoria estática

param, local y pi → son locales a funcion() y se almacenan en la pila (direcciones consecutivas)

## → Ejercicio 2

Código fuente en Python

```
x=1

def suma(a,b):
 resultado = a+b
 if resultado > 0:
 temporal = True
 return resultado
```

- Simular una pila de tablas de símbolos para manejar ámbitos anidados.

### ① Ámbito global:

|      |      |        |
|------|------|--------|
| x    | int  | global |
| suma | func | global |

### ② Ámbito de suma ():

|           |     |        |
|-----------|-----|--------|
| a         | int | Suma() |
| b         | int | Suma() |
| resultado | int | Suma() |

### ③ Ámbito del if:

|          |      |          |
|----------|------|----------|
| temporal | bool | if-block |
|----------|------|----------|

Nota: Al salir del if, se elimina temporal; al salir de suma(), se elimina a, b, resultado.

# -Tema 4.7-

## Manejo de errores semánticos

Código fuente en java

```
public class Main {
 public static void main(String [] args) {
 int numero = "5"; // Error 1
 String nombre = 10; // Error 2
 System.out.println(noDeclarada); // Error 3
 }
}
```

- **Error 1:** Asignación de **String** a **int**.

error: incompatible types: String cannot be converted to int

- **Error 2:** Asignación de **int** a **String**.

error: Incompatible types: Int cannot be converted to string

- **Error 3:** Variable no declarada

error: Cannot find symbol 'noDeclarada'

# • Uso de tabla de símbolos para validación

Código fuente (pseudocódigo)

PROGRAMA Ejemplo

ENTERO a=5

ENTERO b="Hola"

IMPRIMIR (c)

FIN

- Generar la tabla de símbolos
- hasta la línea con errores

Tabla de símbolos antes de errores

|   |        |        |
|---|--------|--------|
| a | ENTERO | global |
|---|--------|--------|

## • Detección de errores:

• Error 1: Al intentar insertar b con tipo STRING, el compilador verifica que "Hola" no es compatible con ENTERO.

• Error 2: Al buscar C en la tabla no existe

Error semántico.

# = Conclusiones =

Juanjo 100% 100%

Para concluir el presente trabajo es importante destacar que el análisis semántico es una fase fundamental en la compilación de un lenguaje de programación, ya que permite garantizar la corrección lógica y coherencia de un programa antes de su ejecución. A través de diferentes mecanismos y estructuras, como lo son árboles de expresiones, pila semántica y la tabla de símbolos es posible representar y verificar la validez del código.

Las acciones semánticas de un analizador sintáctico desempeñan un papel crucial al asociar significado a estructuras sintácticas reconocidas, permitiendo la generación de código intermedio o la construcción de estructuras de datos internas. Dentro de este contexto, la comprobación de tipos de expresiones asegura que las operaciones realizadas sean compatibles, evitando errores en tiempo de ejecución.

Así mismo el esquema de traducción permite definir la manera en que una gramática puede ser utilizada para generar código o realizar validaciones semánticas, mientras que la tabla de símbolos almacena información relevante sobre variable, funciones, entre otros, lo que facilita la organización.

Es sorprendente como funciona y todo lo que conlleva el análisis semántico, aunque para nosotros parezca de conocimientos muy importantes.

# Bibliografía y referencias

- 1.7 Manejo de errores semánticos. (2022b, febrero 17).  
<https://isclayaii.blogspot.com/2022/02/17-manejo-de-errores-semanticos.html>
- 1.6 Generación de la tabla de símbolos y de direcciones (2022, 16 de febrero). <https://isclayaii.blogspot.com/2022/02/16-generacion-de-la-tabla-de-simbolos-y.html>.
- Bill Wagner (2023, 13 marzo). Estructuras de árbol de expresión -C#. Microsoft Learn. <https://learn.microsoft.com/es-es/dotnet/csharp/advanced-topics/expression-trees-explained>.
- Acciones semánticas de un analizador sintáctico. - Bing (s.f).  
Bing. <https://www.bing/search?q=Acciones+semanticas+De+Un+analizador+sintactico> = EgS R1Z Y1ABBF6Od1YUzN AgE&FORM = ANABO1 & PC = ACTCS
- 1.3 Comprobaciones de tipos de expresiones. (2022, 13 febrero).  
<https://isclayaii.blogspot.com/2022/02/13-comprobaciones-de-tipo-en.html>.
- Esquema de Traducción. - Bing. (s.f). Bing. <https://www.bing.com/search?q=ESQUEMA+DE+TRADUCCION+BN-1crp=EgMIZGd1kgY1ABBF6DkYBgg> Od Ag E&FORM = ANABO1 & PC = ACTCS.

Julián H. Pérez

# Videos de la actividad 6

Jesús Pérez  
joseperez@alumnos.ujaen.es

Video 1. Parte 1 análisis semántico . Temas 4.1-4.4

- dar click aquí: [video](#)
- <https://youtu.be/6jbb4vZIT8c>

Video 2. Parte 2 análisis semántico. Temas 4.5-4.7

- dar click aquí: [video](#)
- <https://youtu.be/1fLOzYqb13M>

# Práctica de laboratorio

| CARRERA                                | NOMBRE DE LA ASIGNATURA  |
|----------------------------------------|--------------------------|
| INGENIERIA EN SISTEMAS COMPUTACIONALES | Lenguajes y autómatas 2. |
| <b>INTEGRANTES-EQUIPO 04</b>           |                          |
| • Aguilar Flores Abel                  | 21031027                 |
| • Casteñeda Pérez Cristian Eduardo     | 21030140                 |
| • Macías Sevilla Diana Nathasha        | 21030223                 |

| PRACTICA NO. | NOMBRE DE LA PRACTICA                                   | DURACIÓN (HORAS) |
|--------------|---------------------------------------------------------|------------------|
| 6            | Conceptos básicos del controlador de lógica programable | 2 hora(max)      |

## INTRODUCCIÓN

En el video de controladores se explora los conceptos básicos de los Controladores Lógicos Programables (PLC), dispositivos esenciales en la automatización industrial. Descubriendo su funcionamiento, sus aplicaciones y cómo revolucionaron los sistemas de control al sustituir los antiguos bancos de relés electromecánicos, aunque aún incorporan relés en su diseño.

Se analiza los fundamentos de los PLC, incluyendo operaciones lógicas como la compuerta AND, y su papel clave en la ingeniería de automatización. Contestando a las preguntas qué es un PLC o cómo funcionan las compuertas lógicas en la industria, este contenido es el punto de partida ideal para entender su importancia en los procesos automatizados moderno. Además, se conectarán estos conceptos con la teoría de lenguajes y autómatas, mostrando cómo los PLC implementan lógicas de control secuencial similares a las de un autómata finito. Veremos que las compuertas lógicas (como AND, OR, NOT) y los diagramas de escalera en

# Práctica de laboratorio

PLC siguen principios formales estudiados en lenguajes formales y teoría de la computación.



## OBJETIVO

Realizar un análisis de video "Conceptos básicos del Controlador de lógica programable" relacionándolo con la materia de lenguajes y autómatas.

## FUNDAMENTO

PLC son las siglas en inglés de Programmable Logic Controller, que en español significa Controlador Lógico Programable. Es un dispositivo electrónico que se utiliza para controlar procesos industriales y maquinaria.

### Características:

- Es una computadora industrial que se programa para realizar acciones de control automáticamente.
- Se usa en automatización desde 1970.
- Es un sistema integrado con una CPU, memoria RAM, ROM, módulos de E/S, etc.
- Se considera el cerebro de una línea de producción en la automatización industria.

En el núcleo de cada PLC hay un procesador básico de computadora que recopila varios datos de entrada y los evalúa para lograr los datos de salida deseados. Las entradas de datos pueden ser digitales o analógicas. Dado

# Práctica de laboratorio



que los usuarios pueden programar el sistema de múltiples formas para adaptarse a un determinado escenario, los PLC se adaptan a muchas aplicaciones en diversas industrias, incluidos sistemas de transporte, refinerías de petróleo, líneas de fabricación y más.

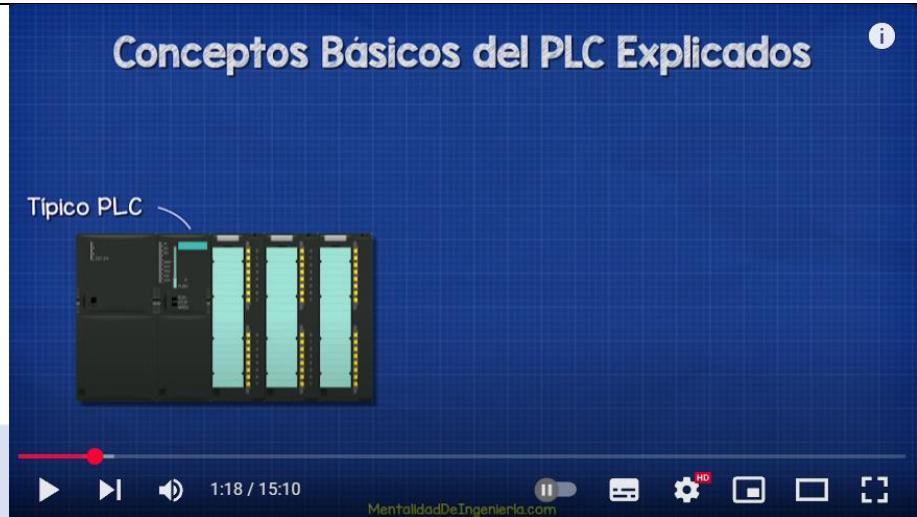
A medida que el PLC escanea entradas de múltiples fuentes, las escanea y las internaliza. Luego, ejecuta la programación del usuario para establecer las salidas deseadas. A continuación, comunica cualquier información necesaria a una red de control como Modbus o Ethernet IP. Debido a que los PLC están a cargo de los sistemas de misión crítica y generalmente hay personas cerca de las máquinas, se ejecutan una serie de diagnósticos para asegurarse de que todo esté en orden, antes de escanear las entradas nuevamente. Todo este proceso es el "ciclo de escaneo" o SCAN. Cuanto mayor sea el número de entradas, mayor será el programa del PLC y mayor será el ciclo de exploración. El ciclo de escaneo se mide en milisegundos, más comúnmente conocido como "Fast". Sin embargo, hay algunas aplicaciones en las que la velocidad no es lo suficientemente rápida. Es posible que se necesite un controlador de automatización programable o PAC. Un PAC utiliza varias CPU en un solo sistema o chasis para proporcionar procesamiento paralelo o procesamiento especializado de diferentes facetas de la aplicación.

## REQUISITOS BÁSICOS

- Una computadora
- Conexión a internet

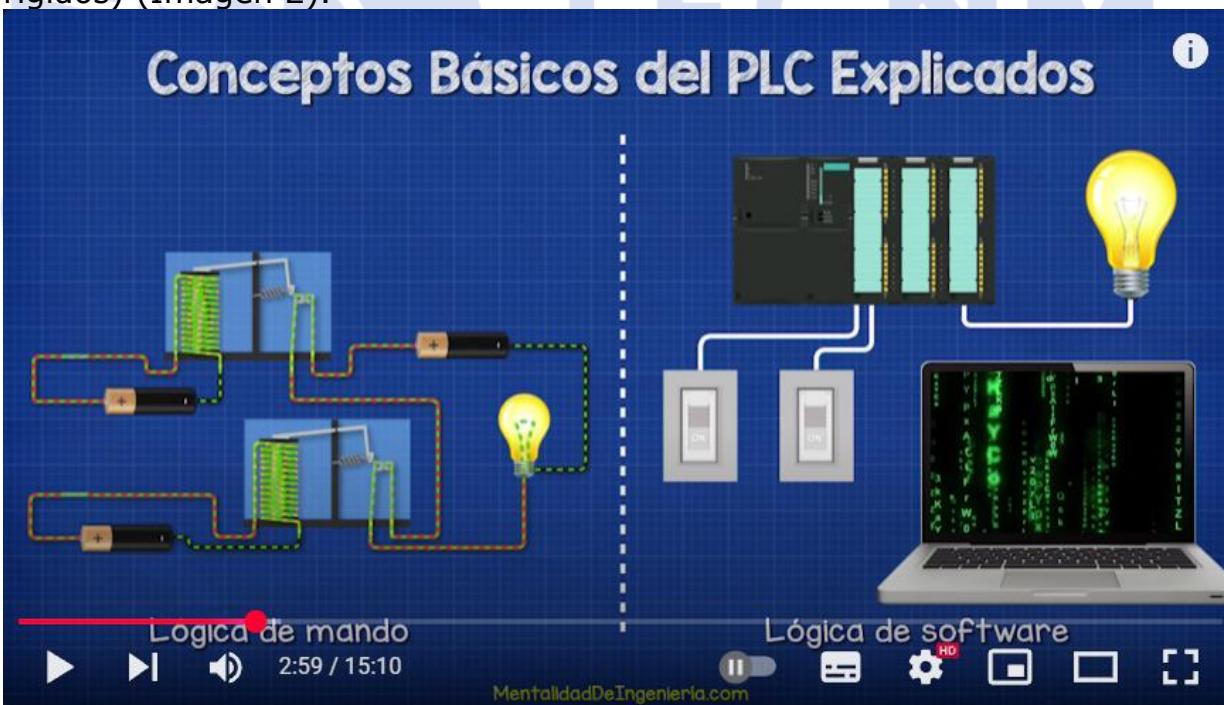
## DESARROLLO

Los Controladores Lógicos Programables (PLCs) son sistemas computarizados que automatizan procesos industriales y comerciales mediante la ejecución de programas basados en entradas (sensores) y salidas (actuadores). Su funcionamiento guarda una estrecha relación con los conceptos teóricos de Lenguajes y Autómatas 2, específicamente con autómatas finitos, lenguajes formales y sistemas de transición de estados (Imagen 1).



*Imagen 1. Introducción a los PLC's*

Los PLCs reemplazaron los antiguos sistemas de relés cableados, que operaban mediante lógica física (similar a autómatas finitos deterministas rígidos) (Imagen 2).



*Imagen 2. PLC y reles cableados*

En contraste, los PLCs implementan lógica programable, donde:

- Entradas (sensores) equivalen a símbolos de un alfabeto (ej.: un interruptor cerrado = "1", abierto = "0") (Imagen 3).

## Conceptos Básicos del PLC Explicados

Entradas analógicas

Podría convertir el voltaje en corriente usando la ley de Ohm y una resistencia.



Imagen 3. Entradas analógicas

- El programa del PLC actúa como un autómata finito: toma decisiones (transiciones de estado) basadas en reglas predefinidas (ej.: "IF temperatura\_baja AND habitación\_ocupada THEN activar\_calentón").
- Salidas (actuadores) son las acciones resultantes, análogas a las salidas de un autómata (imagen 4).

## Conceptos Básicos del PLC Explicados

Microprocesador

El microprocesador realiza la función, basándose en el valor de entrada y la lógica del programa.

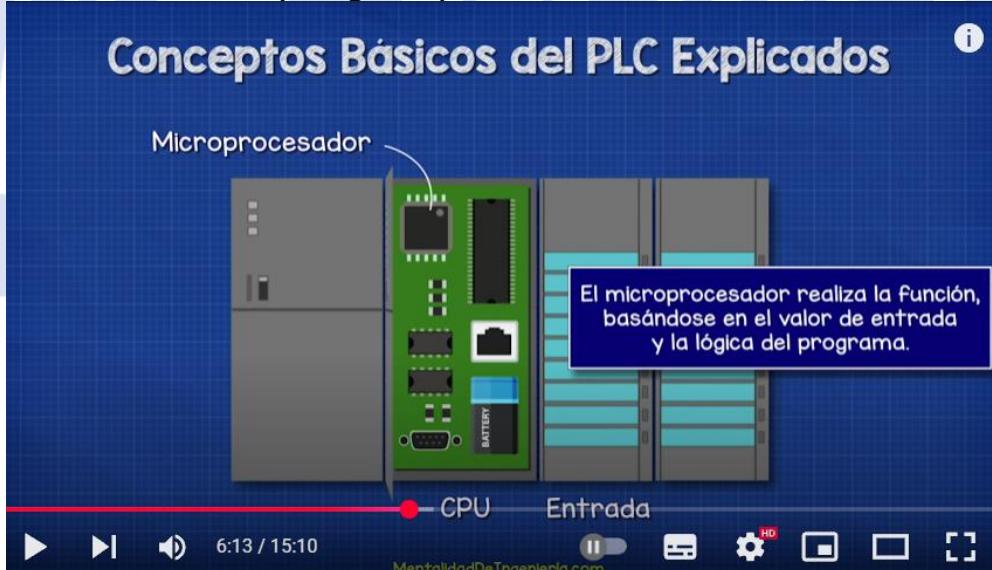


Imagen 4. Funcionamiento de los PLC

### Ejemplo en el video

# Práctica de laboratorio

Un PLC controla un calentón usando un sensor de temperatura (entrada binaria) y un sensor de movimiento. Esto refleja la evaluación de cadenas en lenguajes regulares, donde una secuencia de símbolos (entradas) debe cumplir ciertas condiciones para ser aceptada (activar salida) (Imagen 6).



Imagen 6. Ejemplo de PLC

El control PID (Proporcional, Integral, Derivativo) es una técnica avanzada utilizada en sistemas dinámicos. Los PLCs, al integrar control PID, pueden ajustar procesos como la temperatura o la velocidad de los motores con mayor precisión. Este tipo de control se utiliza cuando se requiere una regulación continua, no solo discreta, y es fundamental para mantener la estabilidad del sistema. Aunque los PLCs tradicionales usan lógica de control discreta, cuando se incorpora el control PID, se está tratando con sistemas continuos. Sin embargo, el principio sigue siendo el mismo: tomar decisiones en función de entradas (errores, cambios en el sistema) para alcanzar un estado deseado. A nivel de teoría de autómatas, el control PID puede considerarse como un sistema con una respuesta continua que ajusta su comportamiento en función de entradas dinámicas (Imagen 6).

## Conceptos Básicos del PLC Explicados

i

El PLC cumple las siguientes etapas en su funcionamiento básico:

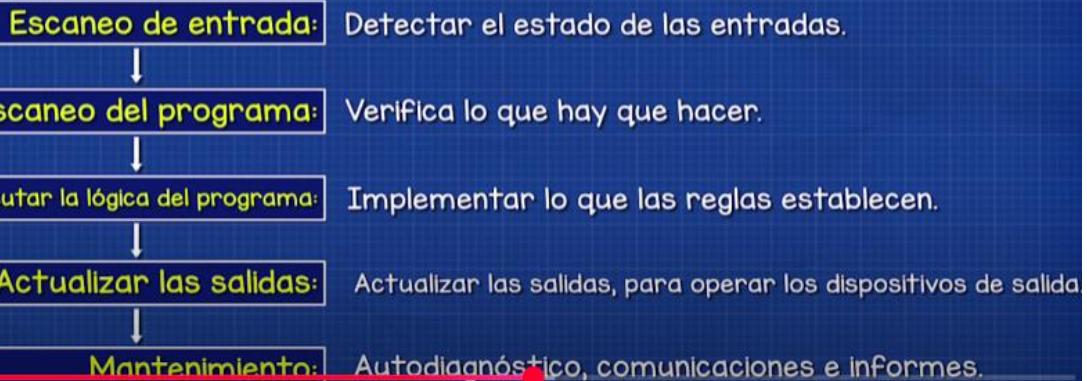


Imagen 7. Etapas del funcionamiento de los PLC

Los PLCs no funcionan de manera aislada; a menudo se necesitan comunicar entre sí o con otros dispositivos. Se mencionan protocolos de comunicación como Modbus, que permiten la interacción entre PLCs, sistemas SCADA (Control Supervisado y Adquisición de Datos), y otros equipos de automatización. Este tipo de comunicación entre dispositivos es crucial en sistemas distribuidos y refleja el uso de máquinas de estados interconectadas. Cada PLC o dispositivo puede estar en diferentes estados, pero se comunican entre sí para sincronizar su funcionamiento, lo que se puede modelar como un conjunto de autómatas trabajando juntos para alcanzar un objetivo común. (Imagen 8)

i



▶ ▶ 🔊 11:36 / 15:10 HD

▢ ▢ ▢ ▢ ▢

[MentalidadDeIngenieria.com](http://MentalidadDeIngenieria.com)

## Conceptos Básicos del PLC Explicados

Termistor

PLC

Válvula del actuador

Temperatura deseada

Temperatura actual

Controlamos la válvula usando la retroalimentación.

Cada escaneo verifica la temperatura y luego altera la posición de la válvula para lograr un funcionamiento estable.

Imagen 8. Ejemplo de control supervisado

En la parte final del video, se destaca cómo los PLCs no solo automatizan tareas repetitivas, sino que también permiten optimizar procesos industriales, ahorrar energía y reducir costos. Se enfatiza la flexibilidad de los PLCs, ya que pueden ser programados para adaptarse a diferentes tipos de sistemas y necesidades, lo que refleja. La capacidad de optimizar y adaptar procesos se alinea con el concepto de que los autómatas pueden ser flexibles en cuanto a los estados que pueden alcanzar y las transiciones que pueden realizar. La programación de los PLCs, al igual que los autómatas, se basa en cambiar entre diferentes configuraciones de control en función de las entradas que recibe. (Imagen 9)

Page 8 | 10

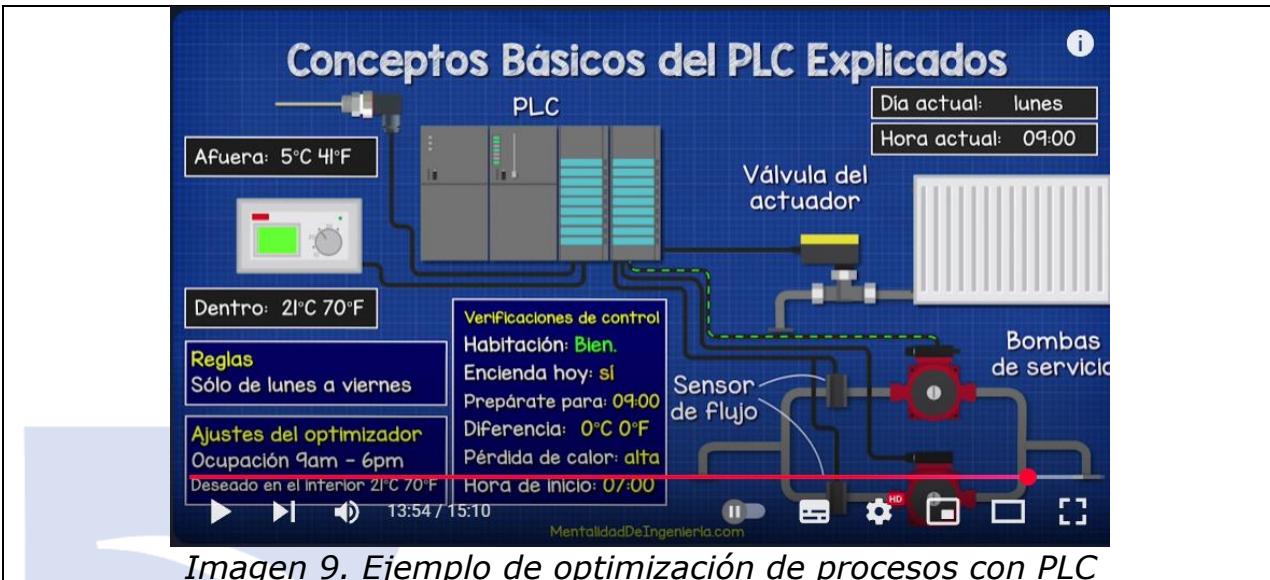


Imagen 9. Ejemplo de optimización de procesos con PLC

## INCIDENCIAS

Sin incidencias.

## OBSERVACIONES

- Los PLCs funcionan como máquinas de estados en la medida en que toman decisiones basadas en entradas y cambian de estado en función de ellas.
- A diferencia de los autómatas clásicos, que tienen un número finito de estados y transiciones bien definidas, los PLCs pueden integrar control PID y comunicación con otros sistemas, lo que permite ajustes dinámicos en tiempo real.

## CONCLUSIONES

Los PLCs y los conceptos estudiados en "Lenguajes y Autómatas" están fuertemente relacionados, ya que ambos se fundamentan en el uso de máquinas de estados finitos, donde las transiciones entre estados dependen de las señales de entrada y determinan las acciones o salidas del sistema. En el ámbito de la automatización, los PLCs operan bajo un esquema similar al de los autómatas deterministas, ejecutando instrucciones secuenciales y condicionales para controlar procesos industriales. Aunque los PLCs han evolucionado más allá de los modelos teóricos de autómatas, incorporando capacidades avanzadas como comunicación en red, control PID y programación estructurada, siguen basándose en los principios

# Práctica de laboratorio

fundamentales de lógica de control secuencial, lo que resalta la aplicabilidad práctica de la teoría de autómatas en la industria moderna.

En términos de automatización industrial, el uso de PLCs permite que sistemas complejos funcionen de manera eficiente, adaptable y controlada, garantizando una respuesta precisa a diversas condiciones operativas. Estos dispositivos aplican principios de lógica booleana para la toma de decisiones, así como conceptos de máquinas de estados para gestionar múltiples eventos y secuencias de control. Además, su programación está basada en lenguajes formales, como Ladder Diagram (LD), Structured Text (ST) o Function Block Diagram (FBD), lo que permite estructurar el comportamiento del sistema de manera clara y modular. Gracias a estas características, los PLCs son herramientas fundamentales en la automatización de procesos industriales, desde la manufactura hasta la gestión de sistemas complejos en la industria 4.0.

## BIBLIOGRAFIA

- Mentalidad De Ingeniería. (2021, 10 enero). Conceptos básicos del controlador de lógica programable [Vídeo]. YouTube.  
<https://www.youtube.com/watch?v=NPsepHRSCls>
- Industrias GSL USA (2024, 1 julio). ¿Qué es un PLC y cómo funciona? Industrias GSL.  
<https://industriasmexico.com/blogs/automatizacion/que-es-un-plc-y-como-funciona?srsltid=AfmBOopg-BwfvtqKzwTT5BiyBIVjVITOYidgBTrdGn2wY0bW0RMXAehO>