





## TAREA 1.2

Historia y evolución de La Teoría de Autómatas y Lenguajes Formales

#### INSTITUTO TECNOLÓGICO DE PACHUCA

## INGENIERÍA EN SISTEMAS COMPUTACIONALES

**MATERIA:** 

Lenguajes Autómatas

**PROFESOR:** 

Rodolfo Baume Lazcano

**NOMBRE:** 

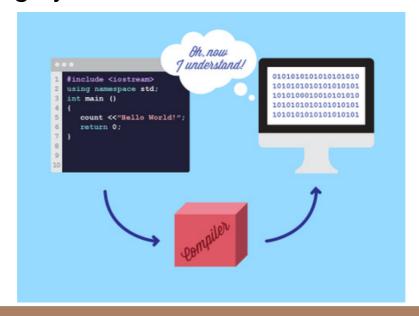
Emanuel Tolentino Santander 21200635



# INTRODUCCION

Un compilador es una herramienta fundamental en la programación que traduce código fuente de un lenguaje de programación a código ejecutable. El proceso de compilación se divide en varias fases, las fases comunes de un compilador son:

- Análisis léxico: El compilador escanea el código fuente y lo divide en tokens, como palabras clave, identificadores, operadores y símbolos, etc.
- Análisis sintáctico: Aquí, el compilador utiliza la gramática del lenguaje para analizar la estructura del código.
- Análisis semántico: En esta etapa, el compilador verifica si el código cumple con las reglas semánticas del lenguaje.







### FASES DE UN COMPILADOR





Los compiladores son programas de computadora que traducen de un lenguaje a otro. Un compilador toma como su entrada un programa escrito en lenguaje fuente y produce un programa equivalente escrito en lenguaje objeto.

ANALIZADOR LÉXICO

ANÁLISIS SINTÁCTICO ANÁLISIS SEMÁNTICO GENERACIÓN Y OPTIMIZACIÓN DE CÓDIGO INTERMEDIO

GENERACIÓN DE CÓDIGO OBJETO

TABLA DE SIMBOLOS GESTOR DE ERRORES

Detecta e informa

Lee la
secuencia de
caracteres de
izquierda a
derecha del
programa
fuente y
agrupa las
secuencias de
caracteres en
unidades con
significado

propio.

Las palabras
clave,
identificadores,
operadores,
constantes
numéricas, signos
de puntuación
como separadores
de sentencias,
llaves, paréntesis,
etc., son diversas
clasificaciones de
componentes
léxicos.

Determina si la secuencia de componentes léxicos sigue la sintaxis del lenguaje y obtiene la estructura jerárquica del programa en forma de árbol, donde los nodos son las

construcciones de

alto nivel del

lenguaje.

Realiza las
comprobacion
es necesarias
sobre el
árbol
sintáctico
para
determinar el
correcto
significado del
programa.

Las tareas básicas a realizar son: La verificación e inferencia de tipos en asignaciones y expresiones, la declaración del tipo de variables y funciones antes de su uso, el correcto uso de operadores, el ámbito de las variables y la correcta llamada

a funciones

La optimización consiste en la calibración del árbol sintáctico donde va no aparecen construcciones de alto nivel. Generando un código mejorado, ya no estructurado, más fácil de traducir directamente a código ensamblador o máquina, compuesto de un código de tres

direcciones.

Toma como entrada la representación intermedia v genera el código objeto.. Suelen implementarse a mano, y son complejos porque la generación de un buen código objeto requiere la consideración de muchos casos particulares.

Es una estructura tipo diccionario con operaciones de inserción, borrado y búsqueda, que almacena información sobre los símbolos que van apareciendo a lo largo del programa como son: - los identificadores (variables y funciones) -Etiquetas – tipos definidos por el usuario (arreglos, registros, etc.)

de errores que se produzcan durante la fase de análisis. Debe generar mensajes significativos y reanudar la traducción. Encuentra errores -En tiempo de compilación: errores léxicos (ortográficos), sintácticos (construcciones incorrectas) v semánticos (p.ej. errores de tipo) -En tiempo de ejecución: direccionamiento de vectores fuera de rango, divisiones

por cero, etc.