

Clase 1

CRITERIOS PARA EVALUAR LENGUAJES DE PROGRAMACIÓN	2
• SIMPLE Y LEGIBLE	2
• CONFIABILIDAD:	2
• SOPORTE.....	2
• ABSTRACCIÓN.....	2
• ORTOGONALIDAD:	2
• EFICIENCIA	2
SINTAXIS.....	2
CARACTERÍSTICAS QUE DEBE CUMPLIR:	3
ELEMENTOS:	3
• Alfabeto o conjunto de caracteres	3
• Operadores:	3
• Palabra clave y palabra reservada	3
o Palabra reservada	3
o Palabra clave:.....	3
• Comentarios y uso de blancos	3
ESTRUCTURA SINTÁCTICA	3
• Vocabulario o words	3
• Expresiones.....	3
• Sentencias.....	3
REGLAS LÉXICAS Y SINTÁCTICAS	3
• Léxicas.....	3
• Sintéticas:	3
TIPOS DE SINTAXIS	3
• Abstracta	3
• Concreta.....	3
• Pragmática	3
¿COMO DEFINIR LA SINTAXIS?	4
• BNF: Backus Naun Form.....	4
• EBNF:	4
• Describir una gramática de forma gráfica (CONWAY):.....	4
GRAMÁTICA	4
FORMAS O MÉTODOS DE ANÁLISIS SINTÁCTICOS	5
• Árbol de reconocimiento	5
o Método botton-up.....	5
o Método top-dow	5
• Arboles de derivación	5
PRODUCCIONES RECURSIVAS	5
SUBGRAMATICAS	6
GRAMÁTICAS LIBRES DE CONTEXTO Y SENSIBLES AL CONTEXTO	6
• Gramática libre de contexto	6
• Gramática sensible al contexto.....	6
SEMÁNTICA (DINÁMICA).....	6

Conceptos de paradigmas de lenguaje

En la teoría vamos a ver análisis vertical, eso quiere decir los conceptos más importantes (de forma abstracta). Cosas relacionadas con la semántica.

En la *práctica* vamos a ver los conceptos de teoría aplicados en un lenguaje, lo que vendría ser un análisis horizontal.

Al analizar un lenguaje logramos conocerlo más a fondo, como sus debilidades y fortalezas. A su vez el análisis de los lenguajes me puede servir para elegir cual es el mejor para una situación es específico y poder crear hasta yo uno que desee, teniendo en cuenta todo y cada uno de los pasos a seguir.

Criterios para evaluar lenguajes de programación

Los explicados a continuación son solo algunos de los tantos criterios que hay.

- **Simple y legible:** Enfocándose a la forma o los recursos que ese lenguaje tiene para poder escribir programas y a la hora de leerlo. Nos fijamos en si es fácil, difícil, complejo, etc.
 - Cosas que afectan: Si el lenguaje tiene muchos elementos o abarcas muchas cosas entonces es posible que el programador solo aprenda un subconjunto de las misma y no aproveche al máximo su potencial. También si se abusar de operadores sobrecargados o si el código se escribe distinto pero el resultado es el mismo.
- **Confiability:** Se refiere en cuando a que tan seguro es programar en este lenguaje. Relacionado con la seguridad del lenguaje a la hora de evitar y manejar errores.
- **Soporte:** Si el soporte es accesible para cualquiera que quiera instalarlo o si por ejemplo se puede interpretar en distintas plataformas.
- **Abstracción:** Definir y usar estructuras u operadores de tal manera que me permita ignorar los detalles.
- **Ortogonalidad:** Significa que distintos aspectos del lenguaje pueden ser usados en cualquier combinación, y que esas combinaciones tienen sentido. Además, que el significado de un aspecto determinado es consistente, sin importar con que otros aspectos es combinado.
- **Eficiencia:** Se tienen en cuenta varios aspectos. Como el tiempo y espacio, el esfuerzo humano y que tan optimizable puede ser.

Sintaxis

Es el conjunto de reglas que de cómo se componen las letras, dígitos y otros caracteres para formar los programas. Como se forma una palabra, como esa palabra se combina para formar una sentencia válida para el lenguaje y como esa sentencia valida forma un programa valido. Lo usamos los programadores para escribir programas correctos y aquellos que quieran crear compiladores.

Características que debe cumplir:

- Legibilidad
- Verificabilidad
- Traducción
- Falta de ambigüedad.

Elementos:

- **Alfabeto o conjunto de caracteres:** Con los que va a trabajar el lenguaje.
- **Identificadores:** Cada lenguaje define una forma de definir la palabra claves, por ejemplo, que se pueda poner letras y dígitos, pero que comience con una letra en especial o que tenga cierta longitud.
- **Operadores:** Que operadores se definen. En la mayoría de lenguaje están de acuerdo con los operadores aritméticos, pero no es así con la comparación o relacionales cada lenguaje define sus propios operadores.
- **Palabra clave y palabra reservada:**
 - *Palabra reservada:* Es una palabra clave que además no pueden ser usadas por el programador como identificador de otra entidad.
 - *Palabra clave:* Son palabras que tiene un significado dentro de un contexto. Se utilizan para lo que fue creada pero también se puede utilizar como nombre de variable u otra cosa.
- **Comentarios y uso de blancos:** Hace un uso más legible del programa. Se debe definir por cada lenguaje.

Estructura sintáctica

- **Vocabulario o words:** Conjunto de caracteres y palabras necesarias para construir expresiones, sentencias y programas.
- **Expresiones:** Llamadas a funciones que devuelven un resultado.
- **Sentencias:** Se componen de expresiones y palabras. El conjunto de sentencias arma un programa.

Reglas léxicas y sintácticas:




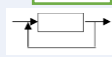

- **Léxicas:** El análisis más básico. Conjunto de reglas que determinan si la palabra (Word) se identifica con un identificador, una sentencia, un operador, etc.
- **Sintéticas:** Conjunto de reglas que se fija si las palabras están bien combinadas para formar sentencias válidas.

Tipos de sintaxis

- **Abstracta:** como está formada, su estructura.
- **Concreta:** me fijo en la estructura y cosas léxicas.
- **Pragmática:** Cómo repercute la sintaxis en el uso.

¿Como definir la sintaxis?

Se necesita dar una descripción formal de definir todas las combinaciones de la sintaxis. Una de las formas es hacerlo con lenguaje natural, pero tiene ambigüedades. Otra forma es BNF (usado en libros y en lenguajes) o usando diagramas sintéticos.

- **BNF: Backus Naun Form:** Es una notación formal que usa símbolos para describir la sintaxis. Se dice que es un metalenguaje porque sirve para definir sintaxis otros lenguajes. Es una gramática libre de contexto. Es decir, no importa si tiene sentido, sino si la estructura es correcta
 - Nombrar un símbolo no terminal => <>
 - Para decir "se define como" => ::=
 - Opciones => |
- **EBNF:** Es la BNF extendida, en la cual se agregar meta símbolos. Me permite realizar una notación más sencilla.
 - [] → van los elementos optativos
 - (|) → elementos opcionales
 - {} → repetición, puede ir acompañada con un "*" o "+"
 - * → 0 o más veces.
 - + → al menos una vez o más.
- **Describir una gramática de forma gráfica (CONWAY):** Es un gráfico sintáctico o carta sintáctica, en la que cada diagrama tiene una entrada, una salida y el camino determina el análisis. Cada diagrama representa una regla o producción. Si se encuentra un camino desde una entrada hasta la salida siguiendo una sentencia entonces es válida.
 - Terminales → 
 - Flujo → 
 - No terminales → 
 - Repetición → 
 - Selección → 

Gramática: Todas las reglas posibles para la sintaxis que estoy describiendo. Está compuesta por 4-tupas

$G = (N, T, S, P)$

N=Conjunto de símbolos no terminales (cosas que voy definiendo en la producción)

T=Conjunto de símbolos terminales (no se definen)

S=símbolos distinguidos de la gramática que pertenecen a N → es el punto de partida

P= Conjunto de producción. → todas las producciones, defino todos los conjuntos de símbolos no terminales hasta llegar a representar todos los conjuntos de nodos no terminales

```
G = (N, T, S, P)

Palabras del lenguaje:
Juan, Ana, tiene, compra, canta, un, una, canción, manta, perro.

T = {Juana, Ana, tiene, compra, canta, un, una, canción, manta, perro}
N = {<oración>, <sujeito>, <predicado>, <sustantivoPropio>, <articuloIndeterminado>,
<sustantivoComún>, ...}
S = {<oración>}
P = {
  <oración> ::= <sujeito><predicado>
  <sujeito> ::= <sustantivoPropio> | <articuloIndeterminado><sustantivoComún>
  <sustantivoPropio> ::= Juan | Ana
  <articuloIndeterminado> ::= un | una
  <sustantivoComún> ::= manta | perro | canción
  <predicado> ::= <verbo><objetoDirecto>
  <verbo> ::= compra | tiene | canta
  <objetoDirecto> ::= <articuloIndeterminado><sustantivoComún>
}
```

Formas o métodos de análisis sintácticos

Pasa saber si es una oración sintácticamente correcta o no, necesito métodos de reconocimiento o análisis. Para ello se puede armar un árbol sintáctico o arboles de derivación.

- **Árbol de reconocimiento:** Se puede construir de dos maneras:
 - **Método bottom-up:** De abajo hacia arriba. Se puede hacer de izquierda a derecha o de derecha a izquierda
 - **Método top-down:** De arriba hacia abajo voy haciendo el match. Esto se puede hacer de izquierda a derecha o de derecha a izquierda.
- **Arboles de derivación:** Lo vas descomponiendo hasta que llego a un ejemplo.
Normalmente los compiladores utilizan el método bottom-up de izquierda a derecha.

Producciones recursivas: Son las que haces que el conjunto de sentencias descripto sea infinito. Por ejemplo:

<natural> ::= <dígito> | <dígito> <natural>

<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Sino tendría que en natural poner todas las combinaciones de dígitos posibles y eso sería un embole.

Para que no sean ambiguas tengo que definir cualquiera de las siguiente (Según que elija es como se va a interpretar):

- Regla recursiva por la izquierda
- Regla recursiva por la derecha

Subgramaticas

Definición de gramáticas de cosas como muy de bajo nivel como identificadores, números, etc. Normalmente no se definen porque alguien ya lo hizo y es igual en todos los lados.

Gramáticas libres de contexto y sensibles al contexto:

- Gramática libre de contexto es aquella en que no realiza un análisis de contexto.
- Gramática sensible al contexto analiza por ejemplo si un identificador está definido dos veces, ósea, que está mirando el contexto.

Semántica (dinámica)

Conjunto de reglas para saber qué pasa cuando se ejecuta, si va a ser válido o no (que produce la frente a la ejecución de esa estructura).