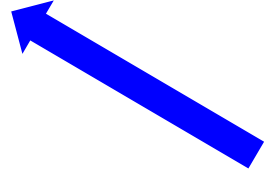


# Análisis Sintáctico

Procesadores de Lenguajes

# Contextualización

- Un procesador de lenguaje:
    - **Analiza** un mensaje recibido
    - **Actúa** en consecuencia
  - Dentro de las fases de análisis:
    - Analizar los **elementos del lenguaje**
    - Analizar su **combinación**
    - Analizar su **significado**
- Analizador sintáctico** 



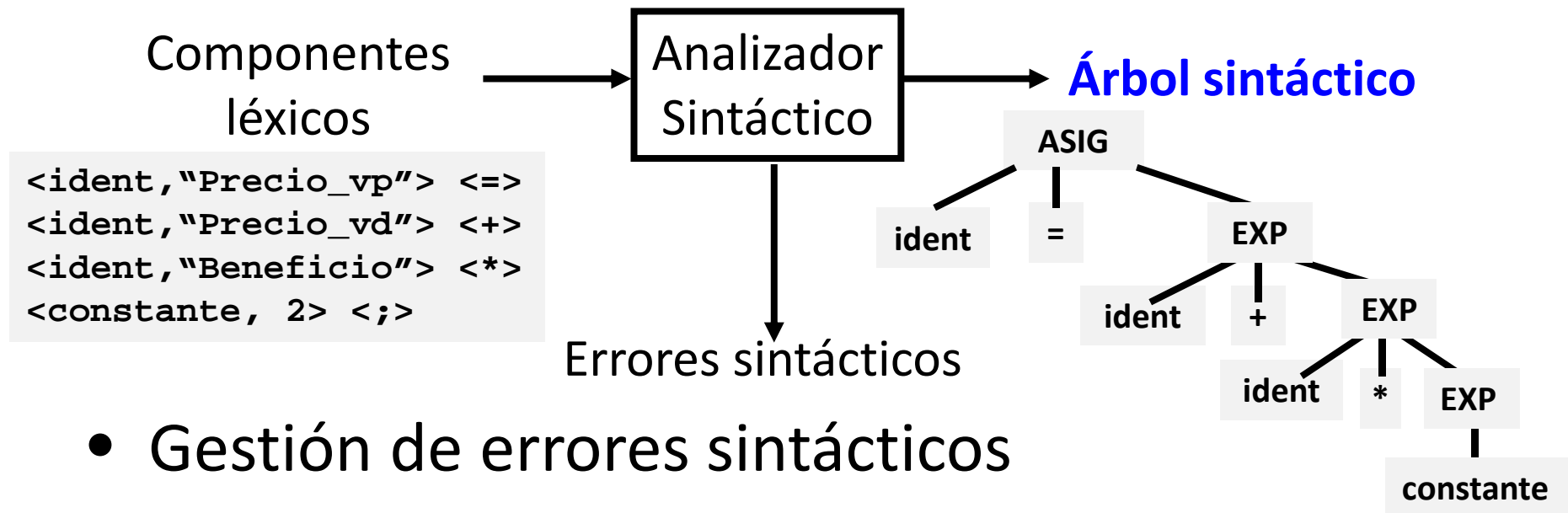
# Contenidos

- **Funciones del analizador sintáctico**
- Conexión con el analizador léxico
- Errores sintácticos
- Fundamentos teóricos
- Analizadores descendentes
- Analizadores ascendentes
- Desarrollo de analizadores sintácticos



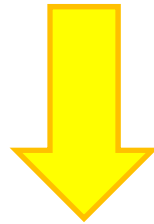
# Funciones del analizador sintáctico

- Comprobación de la **combinación** de los elementos del lenguaje
- Combinación = **corrección sintáctica**



# Funciones del analizador sintáctico

- Construcción del **árbol sintáctico**



- Formas de construcción = tipos de analizadores sintácticos:
  - **Descendentes**: de la raíz a las hojas
  - **Ascendentes**: de las hojas a la raíz



# Contenidos

- Funciones del analizador sintáctico
- **Conexión con el analizador léxico**
- Errores sintácticos
- Fundamentos teóricos
- Analizadores descendentes
- Analizadores ascendentes
- Desarrollo de analizadores sintácticos



# Conexión con el analizador léxico

- Dos formas básicas:

- **Buffer** intermedio de “n” tokens

El analizador léxico rellena el buffer, el sintáctico lo procesa, vuelve el léxico ...

- El analizador léxico es una **subrutina** del analizador sintáctico

Cuando el analizador sintáctico necesita un token se lo pide al léxico

**¿Cuál elegimos?**



# Contenidos

- Funciones del analizador sintáctico
- Conexión con el analizador léxico
- **Errores sintácticos**
- Fundamentos teóricos
- Analizadores descendentes
- Analizadores ascendentes
- Desarrollo de analizadores sintácticos





# Errores sintácticos

- Detección de errores:
  - Propiedad del **prefijo viable**
  - Informar con **claridad** y **exactitud**
    - Ubicación: línea/columna
    - Posible causa
  - Detectar **no implica** localizar:
    - ¿qué **begin** no tiene **end**?

```
PROGRAM Pepito
...
BEGIN
...
      BEGIN
      ...
END
```



# Errores sintácticos

- Recuperación:
  - El procesador **no debe parar** por un error
  - Reconocer la **mayor cantidad** de errores posible
  - **Objetivo:** volver a un estado conocido
  - **Mal hecha** tiene el efecto contrario:
    - detección de **errores inexistentes**
  - ¿podemos **adivinar** lo que quería hacer el programador?



# Errores sintácticos

- Estrategias de recuperación:
  - **Modo pánico:** **ignorar** componentes hasta asegurar que se puede seguir analizando
  - **A nivel de frase:** Corrección local. **Sustitución** de la cadena por otra que permita continuar
  - **Producción de error:** **especificación** que construya cadenas incorrectas
  - **Corrección global:** Mínimo número de cambios



# Contenidos

- Funciones del analizador sintáctico
- Conexión con el analizador léxico
- Errores sintácticos
- **Fundamentos teóricos**
- Analizadores descendentes
- Analizadores ascendentes
- Desarrollo de analizadores sintácticos



# Fundamentos teóricos

- Sintaxis de lenguajes incluye **estructuras recursivas**

Ejemplo: parejas **BEGIN - END**

- Especificación mediante **gramáticas libres de contexto** o de tipo 2:
  - **Símbolos**: Terminales (T) y No terminales (NT)
  - Símbolo inicial o **Axioma**
  - **Producciones**: formación de cadenas del lenguaje



# Fundamentos teóricos

- **Reconocimiento** de una cadena  
= transformación por **aplicación** de producciones  
= **construcción** del árbol sintáctico

```
ASIG ::= id "=" EXP
EXP  ::= id "+" EXP
      | id "*" EXP
      | cte "+" EXP
      | cte "*" EXP
      | id
      | cte
```

```
id = id + id * cte
```

```
ASIG
id = EXP
id = id + EXP
id = id + id * EXP
id = id + id * id
```

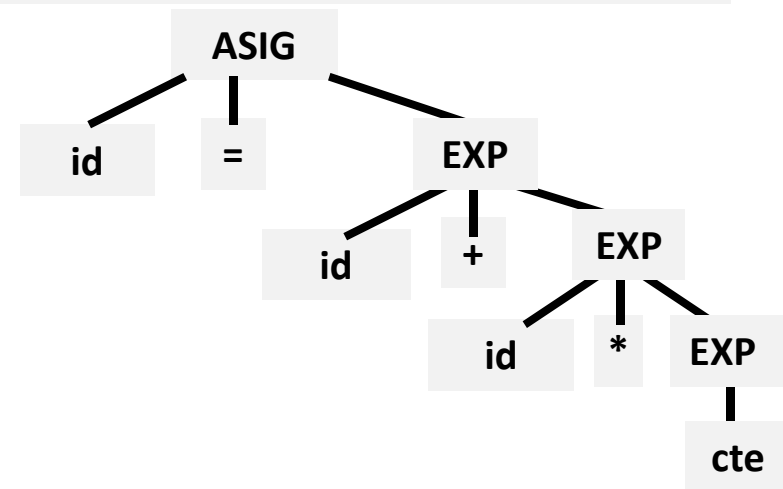


# Fundamentos teóricos

- **Reconocimiento** de una cadena  
= transformación por **aplicación** de producciones  
= **construcción** del árbol sintáctico

```
ASIG ::= id "=" EXP
EXP  ::= id "+" EXP
      | id "*" EXP
      | cte "+" EXP
      | cte "*" EXP
      | id
      | cte
```

id = id + id \* cte



# Fundamentos teóricos

- **Ejercicios** de gramáticas:

1.  $L = \{ a^3 b^n c^n \mid n > 0 \}$

2.  $L = \{ a^n b^{2m} c^m \mid n, m \geq 0 \}$

3. Paréntesis balanceados





# Fundamentos teóricos

- Implementación de **reconocedores** de gramáticas de contexto libre mediante **autómatas con pila**:

AFD

- + Alfabeto de pila
- + Información en **transiciones** sobre inserción y extracción de la pila
- + Finalización por **vaciado de pila**



# Fundamentos teóricos

- **Ejercicios** de autómatas:

1.  $L = \{ a^3 b^n c^n \mid n > 0 \}$

2.  $L = \{ a^n b^{2m} c^m \mid n, m \geq 0 \}$

3. Paréntesis balanceados

