UN PEQUEÑO ANALIZADOR PARTE I: EL LENGUAJE

SESIÓN ANÁLISIS LÉXICO

Autómatas y Matemáticas Discretas

El Lenguaje Aemede

Vamos a diseñar un analizador para un pequeño lenguaje de programación que llamaremos Aemede.

- Un programa en Aemede comienza siempre por la palabra reservada "**inicio**" y termina por la palabra clave "**fin**".
- Todas las instrucciones terminan por el carácter de fin de sentencia "**;**".

```
inicio
  Instrucción;
  Instrucción;
  Instrucción;
  … … …
fin
```

Ejemplo:

```
inicio
  varA := 3;
  varB := 23.5;
  imprime varA + varB;
fin
```

El Lenguaje Aemede

Dos posibles instrucciones:

1. Imprimir datos por pantalla
   - Palabra clave "**imprime**" seguida de una variable, una cadena de texto, un valor numérico o una expresión aritmética

```
imprime variable;
imprime valor;
imprime texto;
```

Ejemplo:

```
imprime 27;
imprime var1;
imprime 3.74;
imprime 3 * var1;
imprime "Hola mundo";
```

El Lenguaje Aemede

Dos posibles instrucciones:

2. Asignación de valores
   - Nombre de variable, seguido del operador "**:=**" y otra variable, un valor numérico o una expresión aritmética

variable **:=** variable;
variable **:=** valor;

Ejemplo:

```
varA := 27;
var_B := 3.74;
c :=  varA * var_B;
vC :=  c;
```

El Lenguaje Aemede

Variables:

- No distinguimos tipos de variables
- El **nombre** de una variable puede contener únicamente:
  - Letras mayúsculas y minúsculas
  - Dígitos
  - Símbolo "_"
- El **nombre** de una variable:
  - Debe comenzar por una letra minúscula
  - No puede finalizar nunca por "_"

Correcto	Incorrecto
varA	3v
a4	varA_
aux	Var
var_name	_var1

El Lenguaje Aemede

Valores:

- Podemos encontrar valores **enteros** y **reales**
- Valores enteros:
  - Pueden estar precedidos opcionalmente por ‘+’ o ‘-’
- Valores reales:
  - Dos secuencias de dígitos separadas por ‘.’
  - Pueden estar precedidos opcionalmente por ‘+’ o ‘-’

Ejemplo:

27	27.54
+4	+0.54
-30	-37.01

El Lenguaje Aemede

Cadenas de texto:

- Van siempre entre comillas dobles o simples
- Pueden contener los siguientes caracteres:
  - Letras mayúsculas y minúsculas
  - Dígitos
  - Espacios en blanco
  - Símbolos +, -, *, /, (y)

Ejemplo:

“Esto es una cadena valida”
‘Con una comilla sola tambien vale’
Esta cadena es invalida porque le faltan las comillas
“No vale mezclar comillas’
“Esta cadena, tiene símbolos inválidos!”

El Lenguaje Aemede

Expresiones aritméticas:

- Pueden contener valores enteros, reales o variables
- Operadores permitidos:
  - Suma y resta: + y - respectivamente
  - Multiplicación y división: * y / respectivamente
  - Cambio de signo ó Negación: -
- Prioridad de operaciones
  - Multiplicación y división tienen prioridad sobre suma y resta
  - Se pueden introducir paréntesis para dar prioridad

Ejemplos:

```
2+(var1+4.5)
var41 + var53 * 4
var1
-(var1 * 0.51)
```

El Lenguaje Aemede

Comentarios:

- Se pueden introducir comentarios en cualquier parte del código fuente
  - La línea con comentarios empieza por //
  - El comentario puede contener cualquier secuencia de los siguientes caracteres: letras, dígitos, espacios en blanco, +, -, *, / y paréntesis
  - Tras la doble barra (//) debe haber al menos un carácter.

Ejemplos:

```
// Esta linea es un comentario valido
// Esta line@ no es un comentario_valido
// Esta, tampoco.
/ A esta le falta una barra
```

El Lenguaje Aemede

Ejemplo de programa en Aemede:

```
// Programa de ejemplo
inicio
coefA := -52;
b := 22.75;
denominador := 1 - coefA;
imprime 'El resultado final es ‘;
imprime b/denominador;
fin
```

Procesadores de lenguajes

- En el modelo más general, el proceso de compilación se divide en cuatro fases:

1. **Análisis léxico:** Recibe la cadena de texto que conforma el código fuente e identifica todas las palabras y elementos clave. Cada uno de estos elementos recibe el nombre de "lexema" y se suele representar mediante un "token".
   - *Ejemplo: Identifica que el lexema "34.85" es en realidad un REAL.*

2. **Análisis sintáctico:** Analiza que la cadena de tokens identificada en el paso anterior sigue la estructura adecuada del lenguaje.
   - *Ejemplo: Se asegura que todas las llaves que se abren estén cerradas.*
   - *Ejemplo: Se asegura de que haya un ";" al final de cada sentencia.*

Procesadores de lenguajes

- En el modelo más general, el proceso de compilación se divide en cuatro fases:

3. **Análisis semántico:** Comprueba que se cumplen las reglas de alto nivel:
   - *Ejemplo: Comprueba la compatibilidad de tipos.*
   - *Ejemplo: Comprueba que una variable tenga valor antes de usarla.*

4. **Generación de código:** Si el código es correcto, se convierte a código de bajo nivel que el procesador pueda comprender.
   - *Ejemplo: Convierte el código en Java en Ensamblador.*

Procesadores de lenguajes