Sintaxis

1. ¿Qué es la sintaxis?

Es un conjunto de reglas que definen como componer letras, dígitos y otros caracteres para formar los programas.

2. ¿Cuál es la utilidad de definir y conocer la sintáxis de un lenguaje? ¿Quiénes se benefician?

La definición de la sintáxis de un lenguaje de programación proporcionan mecanismos para que una persona o una computadora pueda decir si el programa es válido. Esto, a primera vista, implicaría que solo el programador se beneficia de esto. Lo cierto es que también estas reglas sirven para el desarrollo de compiladores e intérpretes, por lo que, si no se hiciera tanta énfasis como se hace, el desarrollo de éstos no sería posible.

3. ¿Cuáles son las características de la sintaxis?

Se caracteriza por brindar soluciones a características tales como:

- Legibilidad
- Verificabilidad
- Traducción
- Falta de ambigüedad

Para esto, la sintáxis establece reglas que definen cómo deben combinarse las componentes básicas, llamadas *word*, para formar sentencias y programas.

4. ¿Cuáles son los elementos de la sintaxis?

- Palabra claves o palabra reservadas
- Operadores
- Comentarios y uso de blancos
- Identificadores
- Alfabeto o conjunto de caracteres

- **Palabras claves o palabras reservadas**: las palabras clave son palabras que tienen un significado dentro de un contexto que no pueden ser usadas por el programador como identificador de otra entidad. Algunos ejemplos comunes son *if, for, while,* etc. Algunas de sus ventajas son:
    - Permitir al compilador y al programador expresarse claramente.
    - Hacer los programas más legibles y permitir una rápida traducción.
- **Operadores**: son los operadores de suma, resta, etc. La mayoría de los lenguajes utilizan +, -, *, etc. En los otros operadores (DIV, MOD, etc)no hay tanta uniformidad.
- **Identificadores**: por lo general, es una cadena de letras y dígitos, que deben comenzar con una letra que representan los distintos tipos de datos del lenguaje. Si se restringe la longitud, se pierde legibilidad.
- **Alfabeto o conjunto de caracteres**: es el conjunto de elementos estructurales del lenguaje. Hay tres tipos:
    - Caracteres alfabéticos (letras minúsculas y mayúsculas).
    - Caracteres numéricos (0 al 9).
    - Caracteres especiales (&,%,#, /…).

5. ¿Cuál es la estructura sintáctica?

- **Vocabulario o words**: conjunto de caracteres y palabras necesarias para construir expresiones, sentencias y programas. Ej: identificadores, operadores, palabras claves, etc. Las words no son elementales, se construyen a partir del alfabeto.
- **Expresiones**: son funciones o bloques sintácticos básicos que a partir de un conjunto de datos devuelven un resultado. Son los que forman los programas o sentencias.
- **Sentencias**: son las unidades ejecutables más pequeñas de un programa, en otras palabras, una línea de código cualquiera. Especifican y controlan el flujo y orden de ejecución del programa. Hay sentencias simples, estructuradas y anidadas.

6. ¿Cuáles son las reglas sintácticas y léxicas?

- **Reglas léxicas**: conjunto de reglas para formar las *word*, a partir de los caracteres del alfabeto. Ejemplo: En C el símbolo de distinto es != en Pascal <>.
- **Reglas sintácticas**: conjunto de reglas que definen como formar las expresiones y sentencias. Ejemplo: el if en C no lleva *then*; en Pascal, si.

7. ¿Qué tipos de sintaxis existen?

- **Abstracta**: se refiere básicamente a la estructura.
- **Concreta**: se refiere básicamente a la parte léxica.
- **Pragmática**: se refiere básicamente al uso práctico.

8. ¿Cómo defino una sintaxis?

Se necesita una descripción finita para definir un conjunto infinito (conjunto de todos los programas bien escritos). Hay varias formas de lograr esto:

- Usando lenguaje natural.
- Usando gramática libre de contexto como BNF(Backus y Naun).
- Diagramas sintácticos (Como BNF pero más intuitivo).

9. ¿Qué es BNF?

Es una notación formal para describir la sintaxis. Es también un metalenguaje (algo asi como un lenguaje que se usa para hablar acerca de otro lenguaje). Como tal, utiliza metasímbolos. Define las reglas por medio de producciones.

10. ¿Qué es la gramática?

Es un conjunto de reglas finita que define un conjunto infinito de posibles sentencias válidas en el lenguaje. Una gramática está formada por una 4-tupla

- Conjunto de símbolos no terminales(N).
- Conjunto de símbolos terminales(T).
- Símbolo distinguido de la gramática que pertenece a N(S).
- Conjunto de producciones (P).

11. ¿Qué es un arbol sintáctico?

Es una representación de todos los string válidos que se puedan armar con los terminales. Éste es el método de reconocimiento que usan los compiladores o intérpretes en el que permiten determinar si un string dado es válido o no en el lenguaje. Esto es lo que se llama *parsing* o *parse* y consiste en realizar un árbol sintáctico para cada sentencia que analize el compilador/intérprete.

12. ¿Qué es una gramática ambigua?

Ocurre cuando una sentencia puede derivarse de mas de una forma, esto es, hay mas de una representación posible usando la gramática libre de contexto(BNF, por ejemplo).

13. Aparte de BNF ¿Existen otras gramáticas?

Si, por ejemplo EBNF(Extended BNF) o los diagramas de Conway. Algunas ni siquiera son libres de contexto.

Tip para diferenciar las gramaticas libre de contexto y aquellas que no lo son:

- Si es libre de contexto, entonces la semántica estática no está se involucrada.
- Si es sensible al contexto, entonces si está involucrada.

14. ¿Qué son los diagramas de Conway?

Es un grafo sintáctico o carta sintáctica en el que cada diagrama representa una regla o producción y tiene una entrada y una salida. El camino determina el análisis. Para que una sentencia sea válida, debe haber un camino desde la entrada hasta la salida que la describa. Se visualiza y entiende mejor que BNF o EBNF.