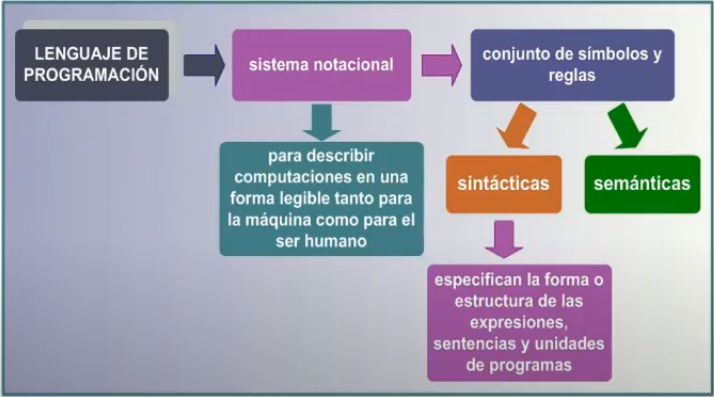
**Unidad 2 – Sintaxis y Semántica de los Lenguajes de programación**

**Sintaxis**

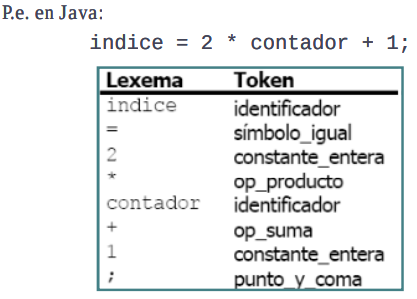


Concepto de pragmática:

* Define el uso practico que se le dan a las expresiones dentro del lenguaje.

**Concepto de Sintaxis**

* Forma o estructura de un lenguaje de programación.
* Esta dada por un conjunto de reglas que van a especificar la composición de los programas a partir de letras, digito y otros caracteres. Si una sentencia, una expresión o una unidad de programa escrita en un determinado LP esta bien formada o no. Van a describir la estructura de todos aquellos programas validos en un determinado LP.
* En muchos lenguajes de programación, la expresión y la variable deben ser de tipos compatibles (del mismo tipo), esta restricción que presentan los LP puede ser considerada una restricción sintáctica o en otro caso los autores lo consideran una restricción semántica.
* Los aspectos sintácticos influyen fuertemente en la facilidad de escritura de los programas y en su legibilidad posterior.
* Definida por reglas:
  + Léxicas:
    - Establecen el conjunto de caracteres del lenguaje de programación y la forma en que se combinan esos caracteres para formar palabras validas.
    - Ej Pascal considera que son iguales los identificadores escritos en mayúscula y minúscula, mientras que C los considera distintos.
  + Sintácticas:
    - Van a especificar la forma de las estructuras de los enunciados, de las sentencias. Va a indicar que cadenas de caracteres pertenecientes a un lenguaje son validas respecto de su estructura.
* Para especificar la sintaxis de un lenguaje de programación existe cierta terminología:
  + Lenguaje:
    - Conjunto de cadenas de caracteres válidos pertenecientes a algún alfabeto.
  + Sentencia:
    - Cada una de estas cadenas.
  + Lexemas:
    - Unidades sintácticas de mas bajo nivel.
    - Ej. +, \*, else, begin, end, contador, suma, etc. (Operadores, palabras reservadas e identificadores).
    - Un programa es una cadena de lexemas.
  + Token:
    - Agrupan categorías de lexemas.
    - Ej. Identificadores, símbolos de operador, palabras reservadas.
    - Pueden describirse usando expresiones regulares



* + - El analizador léxico es el primero que hace este tipo de reconocimiento.
* El propósito de la sintaxis es proporcionar una notación que facilite la comunicación entre el programador y el traductor del LP.

**Criterios generales de sintaxis**

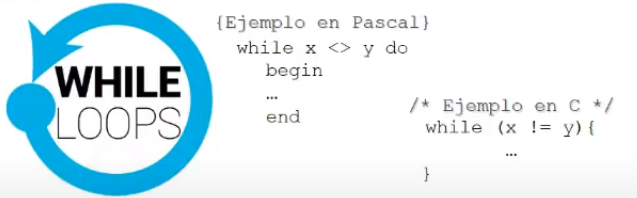
* Legibilidad:
  + Un programa es legible si la estructura subyacente del algoritmo que se utiliza para resolver un problema y los datos que el programa representa quedan expuestos al inspeccionar el texto del programa.
  + Un programa legible se dice que es autodocumentable (entendible si lo leo no siendo el autor del programa y sin documentación que lo respalde).
  + La legibilidad mejora a través de una sintaxis en la cual las diferencias sintácticas reflejen diferencias sintácticas subyacentes, es decir que no se utilicen sentencias o expresiones para representar estructuras que son sistemáticamente diferentes.
* Facilidad de Escritura:
  + Esta mejora a través de el uso de estructuras sintácticas concisas y regulares (que tengan una uniformidad).
  + Ej. las convenciones sintácticas implícitas como Fortran que el tipo de una variable va a depender de la inicial del nombre del identificador hacen que los programas sean mas fáciles de escribir pero dificultan su lectura, ya que debo conocer otras reglas de lenguaje para conocer su tipo.
  + Una sintaxis es redundante si comunica lo mismo de mas de una manera. Por ejemplo la suma en C que puede representar de distintas manera. Esto hace que se facilite la lectura del programa y permite buscar errores en la traducción, pero hace que los programas se dificulten en su escritura.
* Facilidad de Verificación:
  + Relacionado con la Legibilidad y la Facilidad de Escritura (usuario).
  + Entender cada enunciado de un lenguaje no resulta muy difícil, pero el proceso para crear programas correctos es mucho mas difícil. Es necesario utilizar otras técnicas para probar que un programa es correcto o no. Ej. yo puedo entender como se utiliza un enunciado repetitivo pero al usarlo el resultado obtenido no es el que esperaba, entonce el programa no resulta correcto.
* Facilidad de Traducción:
  + Criterio relacionado con las necesidades del traductor que va a procesar el programa escrito en un determinado LP.
  + La clave para una traducción fácil es la regularidad de la estructura.
  + Muchas veces la forma de lograr la regularidad va en decremento de la legibilidad y la facilidad de escritura.
* Carencia de Ambigüedad:
  + La definición de un lenguaje proporciona idealmente un significado único para cada construcción sintáctica que se puede escribir.
  + Construcción ambigua: dos o mas posibles interpretaciones distintas. Surge en la interacción de distintas estructuras. Ej el anidamiento de la sentencia if, no es posible determinar a que if corresponde un else.
  + La sintaxis de un lenguaje de programación debe carecer de ambigüedad.

**Elementos sintácticos básicos de un LP**

* Conjunto de caracteres que van a conformar el LP. La elección de este conjunto es el primer paso para proyectar la sintaxis del lenguaje. Letras mayúsculas y minúsculas, dígitos, caracteres especiales (+, -, etc). El orden de los elementos va a depender de la implementación y de la tabla que se utilice
* Identificadores:
  + Cadena de letras y dígitos, que en la mayoría de los LP comienzan con una letra.
  + La sintaxis para construir los nombres de los identificadores es general para la mayoría de los LP y la inclusión de caracteres especiales como un guion mejoran la legibilidad.
  + Si el lenguaje restringe la longitud de los identificadores, restringe la legibilidad.
* Símbolos de los operadores:
  + Casi todos los lenguajes utilizan el + y el – para representar las dos operaciones básicas aritméticas, pero para los otros tipos de operadores no existe uniformidad dentro de los lenguajes. Ej Fortran para la igualdad se utiliza .eque y no el igual.
* Palabras claves y palabras reservadas:
  + Palabra clave: identificador que se utiliza como una parte fija en la sintaxis de un enunciado. Ej if.
  + Palabra reservada: no se puede usar como identificador.
  + Mejora la detección de errores durante la traducción.
  + La dificultad con este tipo de palabras se presenta cuando es necesario ampliar el LP para incluir enunciados nuevos usando palabras reservadas
* Palabras pregonadas:
  + Palabras opcionales que se insertan en los enunciados para mejorar la legibilidad.
  + Ej. en un enunciado goto, hay lenguajes que consideran a la palabra to como opcional, no es obligatorio su uso.
* Comentarios:
  + Su inclusión en el texto del programa facilita la documentación.
  + Los LP tienen distintas formas de permitir su uso.
  + Ej. en C se utiliza /\* \*/.
* Espacios en blanco:
  + La regla sobre el uso de este tipo de elementos varia de lenguaje en lenguaje
  + En algunos LP tienen un papel sintáctico importante, porque se usan como separadores.
* Delimitadores y corchetes:
  + Delimitador:
    - Elemento sintáctico que se usa para señalar el principio o el fin de algún enunciado o expresión.
    - Se utilizan para mejorar la legibilidad y para simplificar el análisis sintáctico.
    - Propósito general: eliminar ambigüedades de una construcción sintáctica en particular.
  + Corchetes: delimitadores apareados. En un LP son los paréntesis o los pares begin end.
* Formatos de campos libres y fijos:
  + Una sintaxis se dice que es de campo libre si los enunciados de los programas se pueden escribir en cualquier parte de un renglón de entrada.
  + Una sintaxis es de campo fijo si hay que utilizar una posición en particular dentro del renglón para comenzar a escribir los enunciados del programa.
* Expresiones:
  + Son funciones que acceden a objetos de datos en un programa y devuelven un valor.
  + Se pueden considerar como bloques sintácticos a partir de los cuales se constituyen posteriormente enunciados y programas.
  + En los lenguajes imperativos constituyen las operaciones básicas que van a permitir que un enunciado modifique el estado de la maquina
* Enunciados:
  + En los lenguajes imperativos el componente sintáctico mas destacado.
  + La sintaxis de los enunciados tiene un efecto decisivo sobre la regularidad, la legibilidad y la facilidad de escritura del LP.
  + Ciertos LP utilizan un formato único para los enunciados aunque la mayoría de ellos utilizan diferentes estructuras sintácticas para cada tipo de enunciado.

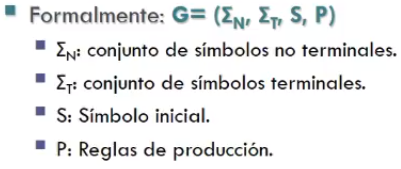
**Sintaxis abstracta y sintaxis concreta**

* Algunas construcciones del lenguaje que tienen la misma estructura conceptual pero difieren en su apariencia a nivel léxico.
* Cuando dos construcciones difieren solo a nivel léxico, tienen la misma **sintaxis abstracta (**estructura**)**, pero difieren a nivel de **sintaxis concreta (**forma en que se escriben**).**
* Conceptualmente la sintaxis concreta es irrelevante pero pragmáticamente puede llegar a afectar la usabilidad del LP y la legibilidad de los programas

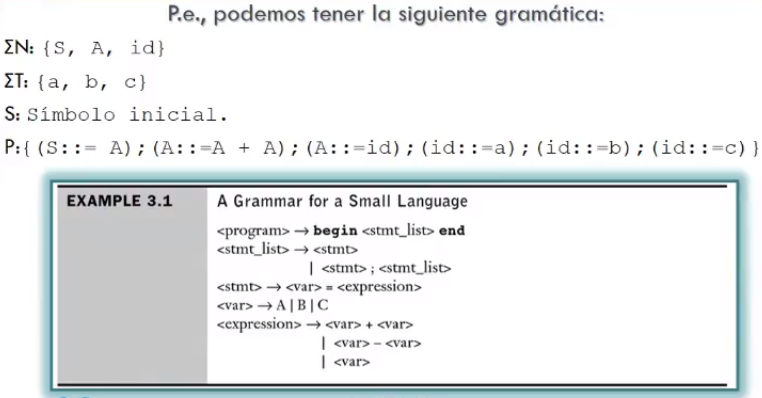


**Métodos formales**

* Gramáticas libres de contexto:
  + Gramática: conjunto de reglas que definen las construcciones validas que puede aceptar un lenguaje de programación.

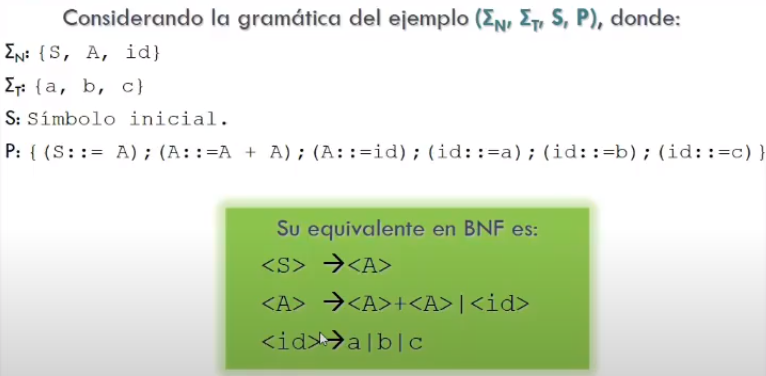


* + Derivación:
    - Aplicación repetida de reglas de producción sintácticas.
    - Comienza por el símbolo inicial y termina con una sentencia.
    - Una cadena de símbolos en la derivación es una forma sentencia.
    - Una sentencia es una forma sentencial que solo tiene símbolos terminales.
    - Derivación por izquierda: se expande el no terminal que esta mas a la izquierda en cada forma sentencial.
    - Derivación por derecha: se expande el no terminal que esta mas a la derecha en cada forma sentencial.
  + Una gramática libre de contexto (tipo 2) es aquella donde la parte derecha de la producción puede tener un símbolo no terminal y pueden emplearse como generadoras de programas validos comenzado por el símbolo inicial y sustituyendo en cada paso un símbolo no terminal por su definición hasta que todos los símbolos sean terminales.

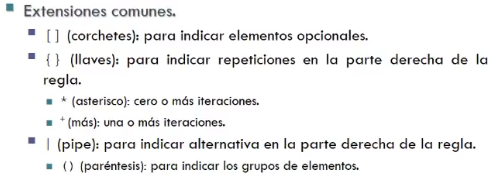


* BNF
  + Notación formal para describir la sintaxis de un LP.
  + Inventada por John Backus para describir Algol 58, y modificada por Peter Naur para la descripción de Algol 60.
  + Fuertemente ligada a la lingüística y a la teoría de los lenguajes formales.
  + Equivalente a una gramática libre de contexto.
  + Metalenguaje, permite describir LP.
  + Lenguaje objeto: va a definir al metalenguaje.
  + Lenguaje sujeto: nombrar el lenguaje de programación que esta siendo descripto.
  + Nos permite identificar una descripción que nos va a servir para generar el conjunto de todas las cadenas que constituyen programas en el lenguaje que se describe y decidir si ese lenguaje es valido de acuerdo a esa descripción dada.



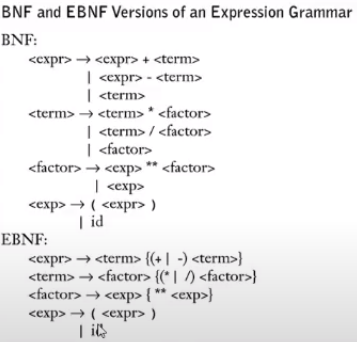


* + Metasimbolos: símbolos que aparecen en la especificación y que no forman parte del lenguaje sujeto . Ej la flecha, el pipe, símbolos mayor o menor que encierran metavariables (palabras que se usan para denotar secuencias de símbolos y palabras del lenguaje sujeto)
* EBNF
  + Una definición en BNF de un lenguaje puede contener cientos de reglas. Una BNF pura puede generar descripciones excesivamente verborragicas. Con la finalidad de reducir el tamaño e incrementar la claridad de las especificaciones el metalenguaje puede llegar a enriquecerse con recursos adicionales.
  + Extensión de BNF que mejora la legibilidad y facilita la escritura.

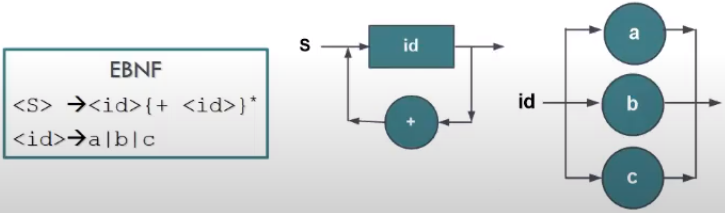


* + Todo esto no aumenta el poder expresivo de la BNF, solo permite especificaciones mas reducidas y claras
  + El problema que traen es que los corchetes, llaves, etc., se confundan con los símbolos del lenguaje que se esta tratando de definir

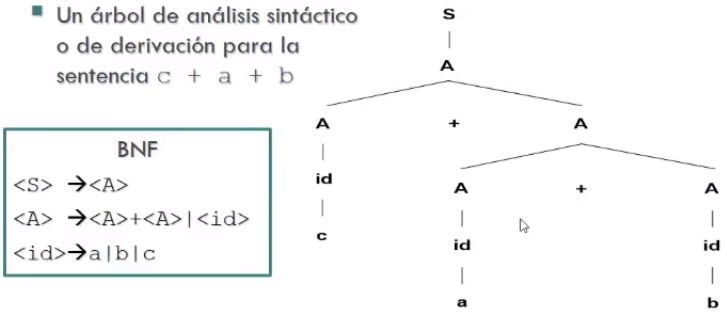




* Diagramas sintácticos:
  + Representación visual equivalente pero mas sencilla de distinguir que la información que se describe con reglas BNF o una EBNF.
  + Cada regla esta representada por un camino donde los símbolos no terminales se representan con rectángulos y los símbolos terminales se representan con círculos.
  + El símbolo no terminal es definido con un diagrama de transición que tiene una linea de entrada y una linea de salida

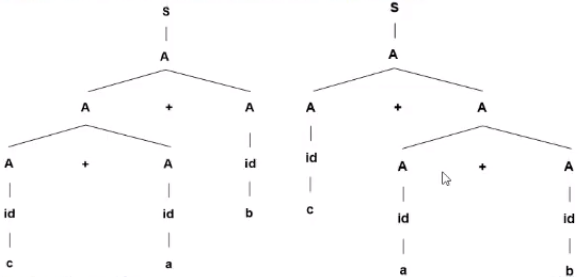


* + Una cadena de palabras es un programa valido si estas pueden ser generadas cruzando el diagrama de sintaxis desde la entrada hasta la salida:
    - Si se encuentra un terminal, la palabra debe estar dentro de la cadena que esta reconociendo.
    - Cuando se encuentra un no terminal, debe reconocerse atravesando el diagrama para ese no terminal.
    - Cuando se encuentra una bifurcación, se puede cruzar por cualquiera de las lineas.
* Arboles de análisis sintáctico:
  + Representación grafica de las derivaciones de una gramática.
  + Se utilizan para la construcción de traductores.
  + Un analizador sintáctico usa las reglas de la definición de un lenguaje para determinar si una cadena es un programa valido.
  + La estructura sintáctica de una cadena puede representarse como un árbol sintáctico o árbol de derivación una vez que atraviesa el analizador sintáctico.
  + Hojas: símbolos y palabras del lenguaje.
  + Nodos internos: metavariables.
  + Raíz: Axioma.



**Ambigüedad**

* Una cadena que esta asociada a mas de un árbol de derivación se dice que es una cadena ambigua y la regla que permite construirla también se denomina ambigua.
* Una gramática es ambigua si existe alguna sentencia que admite mas de un árbol sintáctico diferente.
* En ocasiones, la ambigüedad no afecta la semántica y puede tolerarse, como por ejemplo este caso. Pero hay casos que se producen derivaciones que no son semánticamente equivalentes, por lo tanto en esos casos la ambigüedad debe quedar eliminada del lenguaje y de las reglas que permiten definir esas cadenas dentro del lenguaje.



**Beneficios y restricciones de las definiciones formales de las sintaxis de un LP**

* Beneficios:
  + Constituyen una referencia para usuarios (necesitan un documento al cual recurrir para obtener la información detallada y concisa de la sintaxis o de como se escriben las sentencias en un determinado LP) y para implementadores (especificaciones formales facilitan la estandarización dado que van a eliminar los dialectos que surgen de las diferentes implementaciones)
  + Prueba de programas (Las herramientas de verificación y de testeo requieren que los constructores del lenguaje estén formalizados de manera rigurosa) y de implementaciones (para probar la correctitud de un programa los requerimiento tienen que haber sido especificados rigurosamente).
  + Favorece la implementación automática , en la actualidad existen herramientas para automatizar parte de la construcción de compiladores e interpretes.
  + Diseño del LP, este debe asegurar cierto nivel de naturalidad, de simplicidad, de claridad conceptual, y de poder expresivo, cosas difíciles de notar si no se posee una descripción formal del LP.
* Restricciones:
  + A la complejidad de aprender, comprender y usar un LP, se agrega la complejidad de aprender, comprender y usar un metalenguaje que se utiliza para la descripción formal de un LP.
  + Distinguir los elementos del LP que se esta describiendo, de los elementos del metalenguaje que se esta usando para describirlo.
  + Existen algunas facilidades que aun no han sido modeladas por ningún método de definición formal de sintaxis de un LP, como la concurrencia, excepciones, etc. Ademas hay cosas que autores afirman que pertenecen a la semántica y otros que pertenecen a la sintaxis.