Teoría- CLASE 1

Los lenguajes de programación:

Los lenguajes de programación son herramientas que usamos para comunicarnos con las máquinas y también con otras personas.

Objetivos de la materia:

Introducir, analizar y evaluar los conceptos más importantes de los lenguajes de programación.

- Adquirir habilidad de apreciar y evaluar lenguajes, identificando los conceptos más importantes.
- Habilidad para elegir, diseñar, implementar o utilizar un lenguaje.
- Enfatizar la abstracción como mejor forma de manejar la complejidad de objetos y fenómenos.

¿Para qué?

- Aumentar la capacidad para producir software
- Mejorar el uso del lenguaje
- Elegir mejor un lenguaje
- Facilitar el aprendizaje de nuevos lenguajes
- Facilitar el diseño e implementación de lenguajes

CRITERIOS PARA EVALUAR LENGUAJES DE PROGRAMACIÓN:

Se mencionan algunos ejemplos, hay más en la bibliografía.

- Simplicidad y legibilidad
- Claridad en los bindings
- Confiabilidad
- Soporte
- ...

1. SIMPLICIDAD Y LEGIBILIDAD:

DEBERÍA:

- Poder producir programas fáciles de escribir y de leer.
- Resultar fáciles a la hora de aprenderlo o enseñarlo.

NO DEBERÍA (algunos ejemplos):

- **Tener muchas componentes elementales** (estructuras de datos, de control, etc.):
  - Los programadores solo logran aprenderse un subconjunto de componentes, por ser demasiados, y así se pierde potencialidad.
- **Mismo concepto semántico, distinta sintaxis:**
  - Ejemplo: i++, i = i+1, i+= 1, etc.
- **Distinto concepto semántico, misma notación sintáctica.**
  - Ejemplo: Acceder a la posición 1 de un arreglo "arreglo(1)" y poner 1 como parámetro en una rutina rutina(1) tienen la misma notación, y hacen cosas distintas.
- **Abuso de operadores sobrecargados.**
  - Ejemplo: "+" que puede concatenar, o sumar, o etc., etc.

2. CONFIABILIDAD:

- Chequeo de tipos: Encontrar los errores antes, reduce los errores en la ejecución y hace menos costoso repararlos.
- Manejo de excepciones: La habilidad para interceptar errores en tiempo de ejecución, tomar medidas y continuar.

3. SOPORTE:

- Que sea accesible para cualquiera que quiera usarlo o instalarlo (interprete de dominio público).
- Que pueda ser implementado en distintas plataformas.

4. ABSTRACCIÓN:

- Capacidad de definir y usar estructuras u operadores complicadas de manera que sea posible ignorar muchos de los detalles.
    - Abstracción de procesos y de datos.

5. ORTOGONALIDAD:

«Lenguaje que pueda y tenga sus componentes y que sus componentes puedan ser combinadas y que eso no cause errores» - Cita textual de lo que dijo la profe.

Lenguaje cuyas componentes puedan ser combinadas libremente sin causar errores.

Ejemplo:

C: "`return <expresion>`", donde `<expresion>` puede ser cualquier tipo de datos EXCEPTO un array o una función. Por lo tanto, no es ortogonal.

6. EFICIENCIA:

- Tiempo y espacio
- Esfuerzo humano
- Optimizable

SINTAXIS Y SEMÁNTICA

- **Sintaxis**: Conjunto de reglas que definen cómo componer letras, dígitos y otros caracteres para formar los programas.
    - La forma de escribir.
- **Semántica**: Conjunto de reglas para dar significado a los programas sintácticamente válidos.
    - Lo que hará.

La definición de la sintaxis y la semántica permiten a una persona o computadora pueda afirmar:

1. Que el programa es válido
2. Si lo es, qué significa

Ejemplo:

Pascal	C
`v: array [1..10] of Integer;`	`int v[10];`

1. **Análisis sintáctico:** Hay muchas diferencias.
2. **Análisis semántico:** Sería un error asumir que es lo mismo, dado que los arreglos en C inician en 0 y en Pascal está declarado desde 1, por lo que al hacer v[10] va a dar error.

CARACTERÍSTICAS:

- Debe ayudar al programador a escribir programas correctos sintácticamente.
- Establece las reglas para que el programador se comunique con el procesador.
- Debe contemplar soluciones a características tales como:
    - Legibilidad
    - Verificabilidad
    - Traducción
    - Falta de ambigüedad

SINTAXIS:

Elementos de la sintaxis

- Alfabeto o conjunto de caracteres.
- Identificadores (nombres de variables, de rutinas, etc.)
- Operadores
- Palabras clave y palabras reservadas
- Comentarios y espacios en blanco.

1. ALFABETO:

Antes de ASCII, había problemas para intercambiar información. ASCII es una tabla estandarizada que usa 1 Byte para representar 128 caracteres (usa un bit de paridad). La tabla solo contemplaba el idioma inglés.

Posteriormente surgió ASCII Extendido, eliminando el Bit de paridad y se lograban 256 caracteres. Incorporaba el español.

Además, existían otras tablas: Latin-1, Cyrillic, Arabic, etc.

Luego surgió UNICODE. Los primeros 256 caracteres son los de ASCII Extendido y luego los demás conjuntos de caracteres.

Es importante saber cuál es el alfabeto que usa el lenguaje.

2. IDENTIFICADORES:

Reglas para armar los nombres para los identificadores.

Algunos restringen cantidad de caracteres, tipos de caracteres, etc.

Ejemplo: Los identificadores pueden tener letras, números y caracteres especiales, pero SIEMPRE debe empezar por una letra.

3. OPERADORES:

Elección de los operadores.

4. COMENTARIOS:

Permitir hacer comentarios. Hacen más legible el programa.

5. PALABRA CLAVE Y PALABRA RESERVADA:

Las palabras clave son palabras que tienen un significado dentro de un contexto (las palabras que PERTENECEN al lenguaje).

Las palabras reservadas son palabras clave que no pueden ser usadas por el programador para algo distinto que lo que fue creado.

Importante saber si hay palabras reservadas, cuáles, etc.

ESTRUCTURA SINTÁCTICA:

Vocabulario o "words":

Conjunto de caracteres y palabras necesarias para construir expresiones, sentencias y programas. Ejemplo: Identificadores, palabras clave, etc.

- No son elementales, se construyen a partir del alfabeto.
- A las Palabras también se las llama Lexemas.

Expresiones:

Funciones que, a partir de un conjunto de datos devuelven un resultado.

Sentencias:

Fui a tomar agua xd

REGLAS LÉXICAS Y SINTÁCTICAS:

- **Reglas léxicas**: Conjunto de reglas para formar las palabras.
    - Cuando encuentro un elemento entre dos espacios en blanco (un lexema), analizo si coincide con un identificador, o una expresión, etc. Si no coincide con nada, significa que hay un error.
    - Ejemplo: ¿Distingue entre mayúsculas y minúsculas?
- **Reglas sintácticas:** Conjunto de reglas que definen cómo formar las expresiones y sentencias.
    - Reviso que las palabras estén bien combinadas.
    - Ejemplo: Si en Pascal pongo un IF, analiza si hay un THEN. En C, no.

TIPOS DE SINTAXIS:

- Abstracta: La estructura.
- Concreta: La parte léxica.
- Pragmática: Cómo repercute la sintaxis en el uso.

Cuando se pide un análisis, se puede hacer en alguno de esos tipos.

CÓMO DEFINIR LA SINTAXIS:

Se necesita una descripción finita para definir un conjunto prácticamente infinito (conjunto de todos los programas bien escritos).

Debe cubrir TODAS las formas en las que se puede hacer algo (ej: Definir una variable).

- En Fortran usaron Lenguaje natural.
- En los '60 usaron "*gramática libre de contexto*" (por Backus y Naun: BNF). Es la más usada en los manuales y libros de lenguajes de programación.
- También se puede encontrar Diagramas sintácticos (son equivalentes a BNF, pero más intuitivos)

BACKUS NAUN FORM (BNF):

Es una notación formal que usa símbolos para describir la sintaxis.

Es una gramática libre de contexto. Es decir, no importa si tiene sentido, sino si la estructura es correcta.

GRAMÁTICA:

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

Terminales (T): Palabras del lenguaje.

No terminales (N): <Palabra> deben definirse.

Axioma (S) = Desde donde voy a empezar a definir. El punto de partida de mis producciones

Producción (P): Definición de los No terminales. Deben definirse todos.

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

N = Conjunto de símbolos no terminales.

T = Conjunto de símbolos terminales.

S = Símbolo distinguido de la gramática N

P = Conjunto de producciones.

EJEMPLO

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

Importante: Que haya dos definiciones iguales, no significa que sean lo mismo.

Por ejemplo, aunque el final de Sujeto coincide con la definición de Objeto directo, el sujeto no es un Objeto directo, puede ser un sustantivopropio o un artículo indeterminado + sustantivo común. Que, aunque coincide con la definición de objeto directo, no implica que sea uno (o algo así entendí, yo también pensaba que se podía sustituir esa parte por ObjetoDirecto xD)