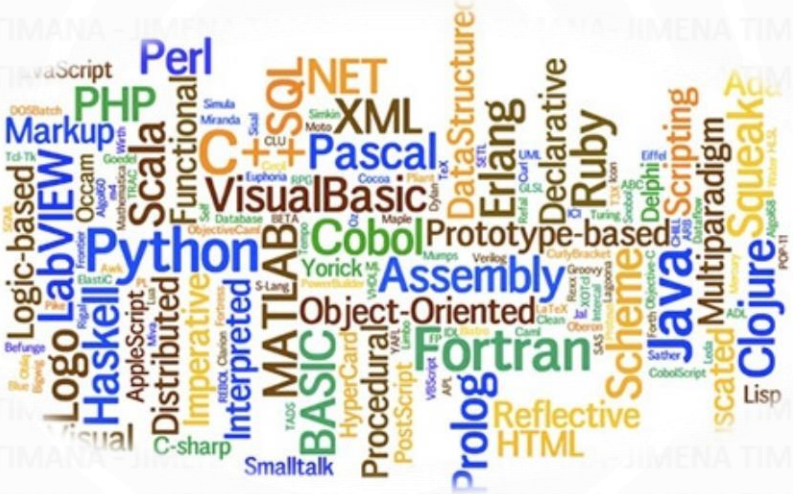


ESTRUCTURAS DE LENGUAJES

CAPITULO 2 – INTRODUCCIÓN A LOS LENGUAJES DE PROGRAMACIÓN

LECTURA EN CASA

MSc. Jimena Adriana Timaná Peña

AGENDA

2.1 Razones para estudiar conceptos de Lenguajes de Programación

2.2 Dominios de Programación

2.3 Criterios de Evaluación para los Lenguajes y Trade-offs del diseño de Lenguajes

¿POR QUÉ ESTUDIAR LENGUAJES DE PROGRAMACIÓN?

PORQUÉ ESTUDIAR LENGUAJE DE PROGRAMACIÓN

1. Acrecentar el vocabulario con construcciones útiles sobre programación

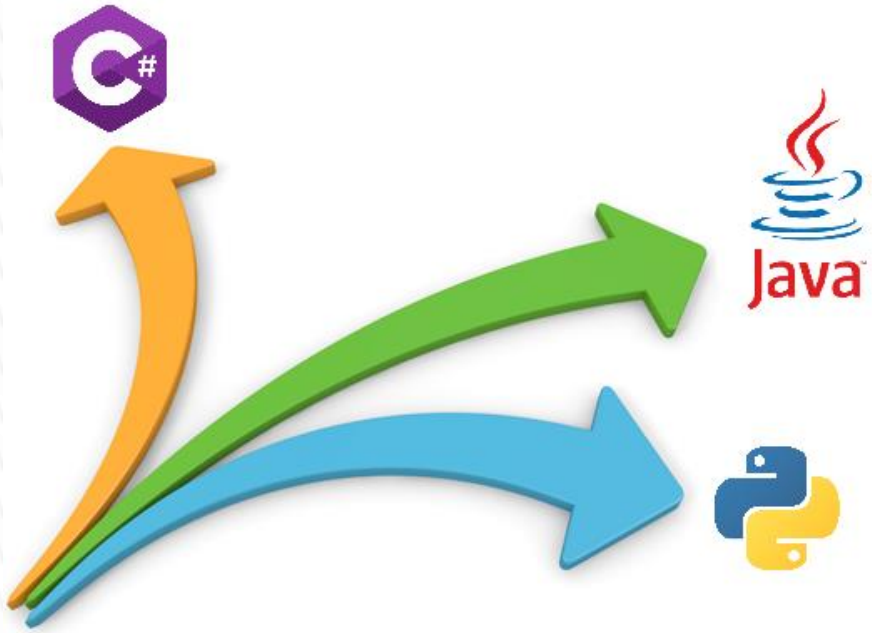
- La familiaridad con un único lenguaje = efecto de **restricción** (pensamiento limitado para dar soluciones a problemas). "Si lo único que conoces es trabajar con un martillo, todo lo verás con cabeza de puntilla"

- Es importante el "**estudio profundo**" de las construcciones suministradas por los lenguajes y cómo se aplican esas construcciones a la solución de un problema. De igual manera es importante conocer cómo la máquina ejecuta las instrucciones del lenguaje para aprender a usar el lenguaje de forma más inteligente.

- Conocer una amplia gama de Lenguajes (bueno?/malo?) → reconocer y diferenciar sintaxis/semántica, "ver la diferencias entre los diferentes conceptos que surgen a nivel de los lenguajes" > "vocabulario enriquecido".

PORQUÉ ESTUDIAR LENGUAJE DE PROGRAMACIÓN

2. Hacer posible una mejor elección del L.P



- Plantear soluciones de diferentes puntos de vista.
- Conocer fortalezas y debilidades de los L.P.
- > conocimiento profundo diversos lenguajes → elección adecuada para proyecto en particular < esfuerzo codificación requerida.

PORQUÉ ESTUDIAR LENGUAJE DE PROGRAMACIÓN

3. Mejorar la habilidad para desarrollar algoritmos eficientes

- Todos los lenguajes incluyen características que dependiendo de su uso (correcto/incorrecto) pueden afectar (+/-) la solución a un problema (tiempo, costo, tiempo de ejecución, etc.)

¿Qué es lo Importante?

- determinar si una situación particular de programación realmente amerita su uso.
- Ejemplo: Recursión

PORQUÉ ESTUDIAR LENGUAJE DE PROGRAMACIÓN

4. Facilitar el aprendizaje de un nuevo lenguaje

- Lenguajes con estructuras similares.
- Lenguajes basados en lenguajes anteriores.

5. Contribuciones en el diseño de un nuevo lenguaje o características

- Si un programador está familiarizado con diversas construcciones, conceptos, aspectos de sintácticos, semánticos en caso de diseñar un nuevo lenguaje o ampliar sus características podrá aportar en cuanto a mejoras, simplificar/mejorar aspectos de diseño, etc.

DOMINIOS DE PROGRAMACIÓN

DOMINIOS DE PROGRAMACIÓN

El uso apropiado de un lenguaje depende del dominio de aplicación

- Aplicaciones Científicas
- Aplicaciones Empresariales
- Inteligencia Artificial
- Programación de Sistemas
- Aplicaciones Web

DOMINIOS DE PROGRAMACIÓN

- **Aplicaciones Científicas**
- Aplicaciones Empresariales
- Inteligencia Artificial
- Programación de Sistemas
- Aplicaciones Web

solución diversas ecuaciones matemáticas, análisis numérico, generación de estadísticas, cálculos aritméticos con altos requerimientos computacionales(punto flotante).

FORTRAN (1er Lenguaje para aplicaciones científicas)

Lenguajes más recientes: lenguaje M (Matlab) y Lenguaje R

Lenguaje R: (1993) Uno de los lenguajes más utilizados en investigación por la comunidad estadística. Muy popular en el campo de la minería de datos, big data, la investigación biomédica, la bioinformática y las matemáticas financieras.

DOMINIOS DE PROGRAMACIÓN

- Aplicaciones Científicas
- **Aplicaciones Empresariales**
- Inteligencia Artificial
- Programación de Sistemas
- Aplicaciones Web

Aquí entran los lenguajes que permiten procesamiento de un volumen de datos muy grande, manejo de inventario, nóminas, etc.
Brindan la posibilidad de generación de reportes

COBOL (1959) (acrónimo de **CO**mmon **B**usiness-**O**riented **L**anguage, Lenguaje Común Orientado a Negocios)

4GL (adaptados dominio específico) + novedosas características (PL/SQL), Java, C#, etc

DOMINIOS DE PROGRAMACIÓN

- Aplicaciones Científicas
- Aplicaciones Empresariales
- **Inteligencia Artificial**
- Programación de Sistemas
- Aplicaciones Web

Las áreas de la IA son muy extensas por lo tanto habrá una gran variedad de lenguajes para trabajarlas.

Áreas de la IA

Áreas básicas:

Representación del conocimiento. Pruebas automáticas de teoremas y matemática simbólica.

Resolución de problemas. Búsqueda. Problemas de optimización combinatorios y de itinerarios.

Áreas específicas:

Procesamiento del Lenguaje Natural

Robótica

Programación y Aprendizaje Automático

Sistemas Expertos

Percepción y reconocimiento de patrones.

Agentes autónomos

Lógica Difusa

DOMINIOS DE PROGRAMACIÓN

- Aplicaciones Científicas
- Aplicaciones Empresariales
- **Inteligencia Artificial**
- Programación de Sistemas
- Aplicaciones Web

Sin embargo, hay principalmente dos lenguaje que han sido muy utilizados a lo largo de la historia en las investigaciones de la IA: LISP y ProLog

Lisp

- 1958
- Las listas constituyen las estructuras de datos básicas de LISP, de ahí su nombre ***List P**rocessing*
- Todo el código de programación de LISP se escribe en *expresiones S*, o lo que es lo mismo, listas entre paréntesis.
- Sintaxis basada en notación prefija (+ 2 3)

Prolog

- 1970
- PROLOG ("PROgrammation en LOGique") fue creado para hacer deducciones a partir de texto.
- Está orientado a resolver problemas usando el cálculo de predicados

DOMINIOS DE PROGRAMACIÓN

- Aplicaciones Científicas
- Aplicaciones Empresariales
- Inteligencia Artificial
- **Programación de Sistemas**
- Aplicaciones Web

Hoy en día varios de los lenguajes para programación de sistemas embebidos o programación de microcontroladores/IoT están basados/influenciados/ desarrollados generalmente en C o en C++.

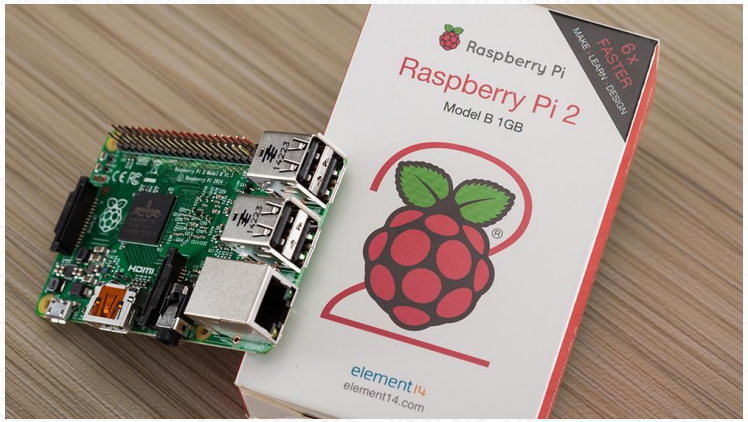
Dichos lenguajes pueden acceder al S.O y al hardware subyacente. (trabajan a Bajo nivel).

¿En qué lenguaje se programa el Internet de las Cosas (IoT) ?

DOMINIOS DE PROGRAMACIÓN

- Aplicaciones Científicas
- Aplicaciones Empresariales
- Inteligencia Artificial
- **Programación de Sistemas**
- Aplicaciones Web

Primero: Se selecciona el hardware: (microcomputadores) como Arduino, Raspberry Pi 2, Intel Galileo, etc



Segundo: elegimos el LP → podemos optar por **lenguajes clásicos** conocidos y con el que se pueden obtener buenos resultados o por **lenguajes más específicos** que pueden sacar toda la "esencia" al dispositivo (aunque la curva de aprendizaje puede resultar algo complicada)

Lenguajes para programar en IoT:

- C / C++
- JAVA (IDE Eclipse IoT)
- Python (lenguaje estrella de Raspberry Pi 2)
- Go / Rust (curva de aprendizaje dura)
- Ensamblador

DOMINIOS DE PROGRAMACIÓN

- Aplicaciones Científicas
- Aplicaciones Empresariales
- Inteligencia Artificial
- Programación de Sistemas
- **Aplicaciones Web**

Lenguajes creados inicialmente para el desarrollo de páginas web (HTML) → estáticas

Códigos dentro de documentos HTML

Lenguajes de Scripting(JavaScript, PHP)

Lenguajes para el lado del cliente(JavaScript, VBScript) y del lado del servidor(PHP, ASP(Active Server Pages), JavaServer Pages)

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES

- Se debe examinar cuidadosamente los conceptos fundamentales de varias construcciones y capacidades de los lenguajes de programación. También debemos evaluar sus **características**, enfocándose en su impacto sobre los procesos de desarrollo del software incluyendo el mantenimiento. Para hacer esto, necesitamos un conjunto de **criterios de evaluación**.

	CRITERIOS		
Característica	**Legibilidad**	**Escritura**	**Confiabilidad**
Simplicidad/ ortogonalidad	*	*	*
Sentencias de control	*	*	*
Tipos de datos	*	*	*
Diseño de sintaxis	*	*	*
Soporte para abstracción		*	*
Expresividad		*	*
Tipo de chequeo			*
Manejo de excepciones			*
Aliasing			*

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Legibilidad → "la facilidad con que los programas puedan ser leídos y entendidos"

Importante: este criterio debe ser considerado en el contexto del dominio del problema.

Característica: **Simplicidad**

- > número de construcciones básicas + difícil de aprender

- *Java*

```
class Hola {
        public static void main(String argument os[]} {
                System.out.println("Hola, mundo");
        }
}
```

- *C*

```
void main(){
        printf ("Hola, mundo");
}
```

- *Pascal*

```
Program Hola;
Begin
        writeln('Hola, mundo');
End.
```

- *PYTHON*

```
print('hola mundo')
```

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Característica: **Simplicidad**

- Esta característica se ve afectada cuando se permite definir nuevas construcciones

- **Multiplicidad**
  - ❖ cuando hay varios caminos para realizar una operación particular
  - ❖ Ejemplo: incremento

```
count = count + 1
count += 1
count++
```

- **Sobrecarga de operadores**
  - ❖ Símbolo de operador con más de un significado. Ejemplo: El + para suma y concatenar

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Ejemplo: APL (***A P**rograming **L**anguage*) 1962, basado en la notación matemática desarrollada por el matemático Kenneth Iverson.

- Presenta un número considerable de operadores para trabajar con sistemas de ecuaciones, vectores y matrices y utiliza un conjunto especial de caracteres no presentes en el código ASCII. Es un lenguaje de programación excepcionalmente compacto que permite implementar complejos procedimientos aritméticos y lógicos con poco código. Sin embargo, al ser tan compacto, el código es difícil de leer, comprender, mantener y depurar. Necesita pegatinas en el teclado para poder ver los operadores (símbolos propios de APL) que se asignan a cada tecla y cada sentencia siempre se ejecuta de derecha a izquierda.

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

APL puede resolver un sistema de ecuaciones en una sola línea de código

Por ejemplo:

$7x + 4y + 2z = 4$

$6x + 8y + 9z = 7$

$4x + 2y + 1z = 2$

Basta ejecutar UNA ÚNICA sentencia de APL, cuya sintaxis es:

4 7 2 [÷] 3 3 ⍴ 7 4 2 6 8 9 4 2 1

Donde el operador ρ (rho) formatea la lista de números en una matriz de 3X3 El operador de [÷] calcula la inversa de la matriz y la multiplica por el vector 4 7 2, generando la solución para cada variable (x, y, z) que en este caso será 0, 1.1 y -0.2 respectivamente.

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Característica: Ortogonalidad

- En un LP significa que un conjunto pequeño de construcciones primitivas pueden combinarse en un número relativamente pequeño de modos o formas para construir estructuras de datos y de control del lenguaje. Adicionalmente, cada posible combinación de primitivos es legal y significativa ”
- “Combinar varias características de un lenguaje en todas las combinaciones posibles, de manera que todas ellas tengan significado”.
- “las construcciones del lenguaje no deben comportarse de manera diferente en contextos diferentes, por lo que las *restricciones* que dependen del contexto son **NO ortogonales**”.

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Característica: **Ortogonalidad**

Ejemplos:

- Suponga un lenguaje que dispone de:
  - Expresión (producir valor)
  - Enunciado condicional (evalúa expresión): V o F
  - Estas dos características del lenguaje, la expresión y el enunciado condicional son **ortogonales** si se puede usar y evaluar cualquier expresión dentro del enunciado condicional.
- En C existe una ***No ortogonalidad*** en el paso de parámetros: C pasa todos los parámetros por valor excepto los arreglos o arrays que se pasan por referencia.
- En C y C++ los valores de todos los tipos de datos, excepto los tipos de arreglos o matrices pueden ser devueltos de una función.

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Característica: **Ortogonalidad**

- Como ejemplos de la falta de ortogonalidad en un lenguaje de alto nivel, considere las siguientes reglas y excepciones en C. Aunque C tiene dos tipos de tipos de datos estructurados: arrays (vectores y matrices) y registros (estructuras), los registros pueden ser retornados por funciones pero los arrays no.

- Un miembro o un atributo de una estructura puede ser cualquier tipo de dato, excepto una estructura del mismo tipo.

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Característica: **Ortogonalidad**

- La ortogonalidad está estrechamente relacionada con la simplicidad: cuanto más ortogonal es el diseño de un lenguaje, menos excepciones requieren las reglas del lenguaje. Menos excepciones significan un mayor grado de regularidad en el diseño, lo que hace que el lenguaje sea más fácil de aprender, leer y comprender.

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Característica: **Sentencias de Control**

- La revolución de la programación estructurada en los 70s fue en parte una reacción a la pobre legibilidad causada por las inadecuadas sentencias de control de algunos lenguajes de los 5s y 60s.
- En particular se hizo ampliamente conocido el uso indiscriminado de las sentencias *goto* las cuales reducían considerablemente la legibilidad de los programas.
- Un programa que puede leerse sin problema de arriba a abajo es mucho más fácil de entender que un programa que requiera que el lector salte de una declaración a otra declaración no adyacente para seguir el orden de ejecución.

```
while (incr <  20) {
   while (sum <= 100) {
      sum += incr;
   }
   incr++;
}
```

```
loop1:
  if (incr >= 20) go to out;
loop2:
 if (sum >  100) go to next;
  sum += incr;
  go to loop2;
next:
  incr++;
  go to loop1;
out:
```

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: LEGIBILIDAD

Característica: Tipos de datos

- La adecuada facilidad para definir tipos de datos en un lenguaje es otro aporte significante dentro de la legibilidad.
- Por ejemplo, suponga que un tipo numérico es usado como un indicador bandera porque no hay un tipo de dato booleano en el lenguaje.
- Que significado tendría entonces la variable timeOut = 1 ? No es del todo claro, a diferencia de timeOut = true

Característica: Diseño de Sintaxis

- Palabras especiales (no necesariamente entre + palabras reservadas + legibilidad)
- Identificadores: Fortran 95 permitía que palabras especiales como Do y End fueran utilizadas como nombres de variables.
- Omisión de llaves → legibilidad en manos del programador con la indentación

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: ESCRITURA

Facilidad de Escritura→ "que tan fácil el lenguaje puede utilizarse para crear programas para un dominio de problema específico"

Característica: **Simplicidad y ortogonalidad**

- Lenguajes con diferentes construcciones: programadores no se familiarizan con todas ellas → ventaja y desventaja a la vez

Pocas construcciones, pocas primitivas = pocas reglas para combinarlas.

Soporte a la abstracción: habilidad de definir y usar estructuras y operaciones "complejas" de modo que los detalles sean ignorados.

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: CONFIABILIDAD

Confiabilidad → "desempeño conforme a las especificaciones"

Característica: **Chequeo de tipos**

- La "verificación o chequeo de tipos" simplemente está probando los "errores de tipo" en un programa dado, ya sea por el compilador o durante la ejecución del programa. La "verificación de tipos" es un factor importante en la confiabilidad del lenguaje. La comprobación de tipo en tiempo de ejecución es costosa, es más deseable verificar el tipo, en tiempo de compilación.

Característica: **Manejo de excepciones**

- Habilidad para interceptar errores en tiempo de ejecución.

Característica: **Aliasing**

- Característica Peligrosa
- Presencia de 2 o + métodos distintos de referencia para la misma locación de memoria

CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES: CONFIABILIDAD

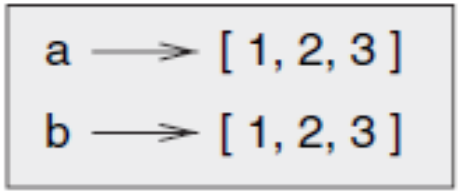
Característica: Aliasing

- Situación en Python

```
>> a = [1,2,3]
>> b = [1,2,3]

>> id(a)
32029272
>> id(b)
29521720
```

Se podría ver como:

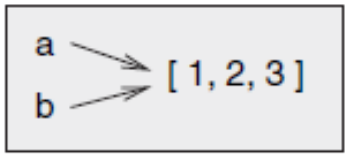


a y b tienen el mismo valor, pero no se refieren al mismo objeto.

En este ejemplo se puede decir que las dos listas son *equivalentes* porque tienen los mismos elementos, pero no son *idénticas* porque no son el mismo objeto.

Si dos objetos son idénticos ellos también son equivalentes pero si ellos son equivalentes no necesariamente son idénticos.

Si se asigna b = a, ambas variables refieren al mismo objeto.



Los cambios hechos a un alias afectan al otro:

```
>> b[0] = "hola"
>> print (a)
["hola", 2, 3]
```

OTROS CRITERIOS DE EVALUACIÓN PARA LOS LENGUAJES

- ***Portable*** → *la facilidad con la que los programas se pueden mover de una implementación a otra.*

- ***Generalidad*** → *la aplicabilidad a una amplia gama de aplicaciones o áreas*

- ***Bien definido*** → *lo completo y la precisión de la documentación oficial de la definición del lenguaje*

- ***Otros criterios? Como el costo?***

PREGUNTAS

- Que trade-offs podemos manejar a nivel de diseño de lenguajes?

- Ejemplos

Trade-offs →compensaciones

TRADE-OFFS DEL DISEÑO DE LENGUAJES

- **Confiabilidad VS costo de ejecución**
  - Java: todas las referencias a los elementos de un array serán verificadas para asegurar que el índice o índices estén en el rango legal →Adiciona mucho costo de ejecución en programas que contengan arrays bastante largos.
  - C no requiere la verificación del rango de índice, por lo que los programas C se ejecutan más rápido que los programas Java semánticamente equivalentes, aunque los programas Java son más confiables.

TRADE-OFFS DEL DISEÑO DE LENGUAJES

- **Escritura VS Lectura**
  - APL
    - Provee muchos operadores, símbolos, primitivas, realiza cálculos complejos. Sin embargo la legibilidad es muy pobre.