Universidad Nacional de Asunción
Facultad Politécnica

ESTRUCTURA DE LOS LENGUAJES

Criterios de evaluación de lenguajes de programación

Dr. Christian von Lücken

OBJETIVOS

1. Analizar un conjunto de criterios de evaluación de los lenguajes de programación
2. Analizar el espectro de los lenguajes de programación

CONTENIDO

1. Criterios de evaluación de los lenguajes de programación
2. El espectro de los lenguajes de programación, clasificación, métodos de implementación

CRITERIOS DE EVALUACIÓN DE LENGUAJES DE PROGRAMACIÓN

¿QUÉ HACE A UN LENGUAJE EXITOSO?

- Poder expresivo: principalmente las facilidades de abstracción
- Facilidad de uso para un novato
- Facilidad de implementación
- Compiladores y herramientas excelentes
- Economía, sponsor, inercia

No existe un factor único que determine que un lenguaje sea bueno
Se deben considerar diferentes puntos de vista, tanto del implementador como del programador

CRITERIOS DE EVALUACIÓN DE LOS LENGUAJES SEGÚN SEBESTA

1. Legibilidad (Readability)
2. Facilidad de escritura (Writability)
3. Confiabilidad
4. Costo
5. Otros: portabilidad, generalidad, buena definición

CRITERIOS Y CARACTERÍSTICAS

Table 1.1 Language evaluation criteria and the characteristics that affect them

	CRITERIA		
Characteristic	READABILITY	WRITABILITY	RELIABILITY
Simplicity	•	•	•
Orthogonality	•	•	•
Data types	•	•	•
Syntax design	•	•	•
Support for abstraction		•	•
Expressivity		•	•
Type checking			•
Exception handling			•
Restricted aliasing			•

CRITERIOS DE EVALUACIÓN DE LOS LENGUAJES SEGÚN SEBESTA

1. **Legibilidad (Readability)**
2. Facilidad de escritura (Writability)
3. Confiabilidad
4. Costo
5. Otros: portabilidad, generalidad, buena definición

LEGIBILIDAD

Facilidad con la que se pueden leer y comprender los programas

- en los primeros tiempos, la eficiencia y la legibilidad de la máquina eran lo más importante
- década de 1970 - ciclo de vida del software: codificación (pequeño) + mantenimiento (grande)

La legibilidad es importante para el mantenimiento

CARACTERÍSTICAS QUE CONTRIBUYEN A LA LEGIBILIDAD

- Simplicidad general
  - Es malo el poseer demasiadas/muy pocas características
  - Es mala la multiplicidad de características
- Ortogonalidad
  - Facilidad de leer y aprender
  - El significado es independiente del contexto
- Declaraciones de control
- Tipos y estructuras de datos
- Diseño de la sintaxis

FACTORES QUE AFECTAN LA LEGIBILIDAD SIMPLICIDAD GENERAL

- Problemas potenciales:
  - Patrón de aprendizaje: cuando se utiliza un lenguaje muy grande se tiende a aprender un subconjunto del mismo e ignorar el resto
  - Multiplicidad de características: tener más de una manera de realizar una operación
  - Sobrecarga de operadores: un solo símbolo tiene más de un significado
- Lo más simple no significa lo mejor y puede hacer que los programas sean difíciles de leer

  – Lenguajes ensambladores: carecen de declaraciones de control complejas, por lo que la estructura del programa es menos obvia

SIMPLICIDAD GENERAL
MULTIPLICIDAD DE CARACTERÍSTICAS MÍNIMA

- Multiplicidad de características: tener más de una manera de realizar una operación
- En Java

```
count = count + 1

count += 1

count ++

++count
```

SIMPLICIDAD GENERAL
SOBRECARGA DE OPERADORES MÍNIMA

- Sobrecarga de operadores: un solo símbolo tiene más de un significado
- Esto puede conducir a reducir la legibilidad si a los usuarios se les permite crear los suyos propios y no lo hacen con sensatez.
- Ejemplo:

• usar + para la suma de números enteros y coma flotante es aceptable y contribuye a la simplicidad

• pero si un usuario define + para que signifique que la suma de todos los elementos de dos matrices unidimensionales no lo es, es diferente de la suma de vectores

FACTORES QUE AFECTAN LA LEGIBILIDAD ORTOGONALIDAD

- Un conjunto relativamente pequeño de construcciones primitivas pueden ser combinadas en un conjunto relativamente pequeño de formas para construir las estructuras de datos y de control del lenguaje.
- Todas las combinaciones posibles de primitivas deberían ser legales y significativas.
- El significado de las características del lenguaje deben ser independientes del contexto en el cual aparecen en el programa
- Ejemplo:

– Cuatro tipos de datos primitivos: entero, flotante, doble y carácter

– Operadores de dos tipos: matriz y puntero

– Si los dos operadores de tipo se pueden aplicar a sí mismos y a los cuatro tipos de datos primitivos, se puede definir una gran cantidad de estructuras de datos.

– Sin embargo, si no se permitiera que los punteros apunten a matrices, muchas de esas posibilidades útiles se eliminarían

FACTORES QUE AFECTAN LA LEGIBILIDAD
ORTOGONALIDAD

- La ortogonalidad está relacionada con la simplicidad:

+ ortogonalidad = - excepciones en las reglas del lenguaje

- excepciones = + regularidad en el diseño

+ regularidad = más fácilidad de aprender, leer y entender

EJEMPLO DE NO ORTOGONALIDAD EN C

- Una función puede retornar un valor de cualquier tipo excepto el tipo array o función.
- Un miembro de una estructura puede ser de cualquier tipo excepto void o una estructura del mismo tipo
- Un elemento de un array puede ser de cualquier tipo excepto void
- Los parámetros a funciones son pasados por valor excepto si son arrays en cuyo caso se pasan por referencia
- En la expresión a + b, el significado de b depende de si es que es o no un tipo puntero (esto es un ejemplo de una dependencia de contexto)

ORTOGONALIDAD (OTROS EJEMPLOS)

Ejemplo del ensamblador de VAX (diseño ortogonal) tiene una instrucción de adición para enteros de 32-bit

ADDL op1 op2

los parámetros pueden referir a un registro o una posición de memoria.

Ensamblador de las mainframes de IBM (falta de ortogonalidad), hay dos instrucciones ADD análogas (A y AR):

- op1 registro y op2 posición de memoria
- op1 y op2 registros.

Más restrictivo
Menor facilidad de escritura
Menor legibilidad

ORTOGONALIDAD (DEMASIADA)

Algol 68 diseñado para ser muy ortogonal:

Intuition leads one to ascribe certain advantages to orthogonality: the reduction in the number of special rules or exceptions to rules should make a language easier "to describe, to learn, and to implement" — in the words of the Algol 68 report. On the other hand, strict application of the orthogonality principle may lead to constructs which are conceptually obscure when a rule is applied to a context in an unusual combination. Likewise the application of orthogonality may extend the power and generality of a language beyond that required for its purpose, and thus may require increased conceptual ability on the part of those who need to learn and use it

"On Orthogonality in Programming Languages", ACM SIGPLAN Notices, July 1979 (pág 18), B.T. Denvir

Explosión de combinaciones

FACTORES QUE AFECTAN LA LEGIBILIDAD

ORTOGONALIDAD (DEMASIADA)

ALGOL 68 es el lenguaje más ortogonal:

- Cada construcción tiene un tipo
- La mayoría de las construcciones producen valores
- Esto permite tener estructuras extremadamente complejas
- Ejemplo: un condicional puede aparecer a lado izquierdo de una sentencia se asignación puesto que este produce una posición

```
(if (A<B) then C else D) := 3
```

Los lenguajes funcionales ofrecen una buena combinación de simplicidad y ortogonalidad

FACTORES QUE AFECTAN LA LEGIBILIDAD
TIPOS DE DATOS Y ESTRUCTURAS

- Los tipos primitivos/intrínsecos deben ser adecuados.

– Si no hay ningún tipo booleano disponible, se puede definir una bandera como un número entero:

`found = 1` (en lugar de `found = true`)

Puede significar que `found` es tratado como booleano o que es 1

- Primeras versiones de C sin tipo booleano, forzando al programador a usar un entero para representar true/false (0 es falso, todo lo demás es true)
- ¿Qué pasa con esta sentencia?

```
if (k = 5) { ... } else { ... }
```

FACTORES QUE AFECTAN LA LEGIBILIDAD
TIPOS DE DATOS Y ESTRUCTURAS

– Una matriz de tipo de registro es más legible que un conjunto de matrices independientes

- Los primeros FORTRAN no tenían construcciones para record/struct, entonces los campos no pueden ser encapsulados en una única estructura (que podía ser referenciada con un nombre).

FACTORES QUE AFECTAN LA LEGIBILIDAD CONSIDERACIONES DE DISEÑO DE LA SINTAXIS

- *Forma de los identificadores:*

- Restringir la longitud de los identificadores puede afectar la legibilidad:
  - Los identificadores FORTRAN77 pueden tener como máximo 6 caracteres.
  - ANSI BASIC: un identificador es un solo carácter o un solo carácter seguido de un solo dígito.
- La disponibilidad de caracteres de concatenación de palabras (p. ej., _) es buena para la legibilidad.
  - en COBOL los nombres pueden incluir guiones lo que puede confundirse con una resta

FACTORES QUE AFECTAN LA LEGIBILIDAD CONSIDERACIONES DE DISEÑO DE LA SINTAXIS

- *Palabras especiales:*
  - Palabras tales como while, if, end, class, etc., tienen un significado especial dentro de un programa. ¿Estás palabras estan reservadas para estos fines o pueden usarse como nombres de variables o subprogramas también?
  - La manera en que se señala el comienzo / final de una declaración compuesta puede afectar la legibilidad

– En PASCAL y C, se usa end o } para finalizar una declaración compuesta. Es difícil saber qué se termina.

–ADA usa `end if` y `end loop` para terminar una selección y un bucle, respectivamente

FACTORES QUE AFECTAN LA LEGIBILIDAD
CONSIDERACIONES DE DISEÑO DE LA SINTAXIS

- *Forma y significado*: la semántica de una construcción sintáctica debería ser evidente desde su forma.
  - Una buena elección de palabras especiales ayuda a esto
  - También ayuda si una forma sintáctica significa lo mismo en todos los contextos, en lugar de diferentes cosas en diferentes contextos (Uso de static en c)

FACTORES QUE AFECTAN LA LEGIBILIDAD CONSIDERACIONES DE DISEÑO DE LA SINTAXIS

Ejemplo de lenguajes esotéricos:

- INTERCAL (https://en.wikipedia.org/wiki/INTERCAL)
- Omgrofl (https://esolangs.org/wiki/Omgrofl)

- https://esolangs.org/wiki/Hello_world_program_in_esoteric_languages

CRITERIOS DE EVALUACIÓN DE LOS LENGUAJES SEGÚN SEBESTA

1. **Legibilidad (Readability)**
2. **Facilidad de escritura (Writability)**
3. Confiabilidad
4. Costo
5. Otros: portabilidad, generalidad, buena definición

FACILIDAD DE ESCRITURA

- Qué tan fácilmente un lenguaje puede ser utilizado para crear programas en un dominio específico
- La mayoría de las características que afectan la facilidad de lectura afectan también la facilidad de escritura
- Se deben evaluar los lenguajes en el contexto de la aplicación para la cual fueron desarrollados

Factores:

- Simplicidad y ortogonalidad
- Soporte para la abstracción
- Expresividad

FACTORES QUE AFECTAN LA ESCRITURA SIMPLICIDAD Y ORTOGONALIDAD

- Un número pequeño de construcciones primitivas y un conjunto consistente de reglas para combinarlas (ortogonalidad) es mejor que tener un gran número de primitivas

- Demasiada ortogonalidad puede ir también en detrimento de la facilidad de la escritura, cuando los errores de la programación no pueden ser detectados puesto que prácticamente cualquier combinación de primitivas es legal

FACTORES QUE AFECTAN LA ESCRITURA SOPORTE PARA LA ABSTRACCIÓN

- Abstracción
  - capacidad de definir y utilizar estructuras complicadas u operaciones de forma tal que los detalles pueden ser ignorados
  - clave en el diseño de lenguajes de programación modernos.
- **Abstracción de procesos:** ejemplo utilización de un subprograma para implementar un algoritmo
- **Abstracción de datos:** el lenguaje debería proporcionar facilidades para acercar la solución del problema al dominio del problema

FACTORES QUE AFECTAN LA ESCRITURA
SOPORTE PARA LA ABSTRACCIÓN: ABSTRACCIÓN DE PROCESOS

- El ejemplo más simple de abstracción son los subprogramas (p. ej., métodos).
- Usted define un subprograma, luego lo usa ignorando cómo funciona realmente.
- Elimina la replicación del código
- Ignora los detalles de implementación - p.ej. algoritmo de clasificación

FACTORES QUE AFECTAN LA ESCRITURA
SOPORTE PARA LA ABSTRACCIÓN: ABSTRACCIÓN DE DATOS

- Como ejemplo de abstracción de datos, un árbol se puede representar de manera más natural utilizando punteros en los nodos.
- En FORTRAN77, donde los tipos de puntero no están disponibles, un árbol se puede representar utilizando 3 matrices paralelas, dos de las cuales contienen los índices de la descendencia y la última contiene los datos.

FACTORES QUE AFECTAN LA ESCRITURA
EXPRESIVIDAD

- Tener formas más convenientes y breves de expresar los cómputos
- Por ejemplo en C:

```
count++;
```

Es más conveniente y expresivo que

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

Con `for` es más fácil escribir ciclos que con `while`

CRITERIOS DE EVALUACIÓN DE LOS LENGUAJES SEGÚN SEBESTA

1. **Legibilidad (Readability)**
2. **Facilidad de escritura (Writability)**
3. **Confiabilidad**
4. Costo
5. Otros: portabilidad, generalidad, buena definición

CONFIABILIDAD

Un programa es confiable si se comporta de acuerdo a sus especificaciones bajo todas las condiciones.

Factores:

- Chequeo de tipos
- Manejo de excepciones
- Aliasing
- Legibilidad y facilidad de escritura

FACTORES QUE AFECTAN LA CONFIABILIDAD
CHEQUEO DE TIPOS

- El chequeo de tipos es la verificación de errores de tipo ya sea al momento de la compilación o durante la ejecución de un programa
- El chequeo en tiempo de compilación es deseable: cuanto antes se detecten los errores serán menos costosos de reparar

FACTORES QUE AFECTAN LA CONFIABILIDAD
CHEQUEO DE TIPOS

- El chequeo de tipos es la verificación de errores de tipo ya sea al momento de la compilación o durante la ejecución de un programa
- El chequeo en tiempo de compilación es deseable: cuanto antes se detecten los errores serán menos costosos de reparar

FACTORES QUE AFECTAN LA CONFIABILIDAD
CHEQUEO DE TIPOS

- El C original no verifica tipos ni en el tiempo de compilación, ni de ejecución.
- La version actual require que todos los parámetros tengan verificación de tipo
- Por ejemplo, el programa siguiente en C original se compila y ejecuta

```
foo (float a) {
printf ("a: %g and square(a): %g\n", a,a*a);
}
main () {
char z = 'b';
foo(z);
}
```

SALIDA: a: 98 and square(a): 9604

FACTORES QUE AFECTAN LA CONFIABILIDAD
MANEJO DE EXCEPCIONES

Manejo de excepciones

- El manejo de excepciones se refiere a la habilidad de un programa de interceptar errores de tiempo de ejecución, tomar medidas correctivas y luego continuar

Aliasing

- Es tener más de una referencia, método o nombre para la misma celda de memoria

CRITERIOS DE EVALUACIÓN DE LOS LENGUAJES SEGÚN SEBESTA

1. **Legibilidad (Readability)**
2. **Facilidad de escritura (Writability)**
3. **Confiabilidad**
4. **Costo**
5. Otros: portabilidad, generalidad, buena definición

COSTO

Tipos de costos:

1. **Costo de entrenar a los programadores:** Función de simplicidad y ortogonalidad, experiencia de los programadores.
2. **Costo de escribir programas:** función de la facilidad de escritura

Nota: estos dos costos se pueden reducir en un buen entorno de programación

3. **Costo de compilar programas:** costo del compilador y tiempo de compilación

- Los compiladores Ada de primera generación eran muy costosos

COSTO

4. Costo de ejecutar programas: si un lenguaje requiere muchas verificaciones de tipos en tiempo de ejecución, los programas escritos en ese lenguaje se ejecutarán lentamente.

– Compensación entre costo de compilación y costo de ejecución.

– Optimización: disminuye el tamaño o aumenta la velocidad de ejecución.

- Sin optimización, se puede reducir el costo de compilación.
- El esfuerzo adicional de compilación puede resultar en una ejecución más rápida.
- Más adecuado en un entorno de producción, donde los programas compilados se ejecutan muchas veces

COSTO

5. **Costo del sistema de implementación:** Si es costoso o solo se ejecuta en hardware costoso, no se usará ampliamente.
6. **Costo de confiabilidad:** importante para sistemas críticos como una planta de energía o una máquina de rayos X
7. **Costo de mantenimiento de los programas:** Para correcciones, modificaciones y adiciones.

- Función de legibilidad.
- Por lo general, y desafortunadamente, el mantenimiento lo realizan personas distintas a los autores originales del programa.
- Para programas grandes, los costos de mantenimiento son de 2 a 4 veces los costos de desarrollo.

CRITERIOS DE EVALUACIÓN DE LOS LENGUAJES SEGÚN SEBESTA

1. **Legibilidad (Readability)**
2. **Facilidad de escritura (Writability)**
3. **Confiabilidad**
4. **Costo**
5. **Otros: portabilidad, generalidad, buena definición**

OTROS: PORTABILIDAD, GENERALIDAD, BUENA DEFINICIÓN

- **Portabilidad:** la facilidad con la que los programas se pueden mover de una implementación a otra
- **Generalidad:** La aplicabilidad a una amplia gama de aplicaciones
- **Bien definido:** la integridad y precisión de la definición oficial del idioma.

COMPENSACIONES DE DISEÑO DE LENGUAJE

- **Confiabilidad versus costo de ejecución**

Ejemplo: Java exige que se verifique la indexación adecuada de todas las referencias a los elementos de la matriz, lo que conduce a un aumento de los costos de ejecución

- **Legibilidad frente a capacidad de escritura**

Ejemplo: APL proporciona muchos operadores potentes para matrices (y una gran cantidad de símbolos nuevos), lo que permite escribir cálculos complejos en un programa compacto pero a costa de una mala legibilidad.

- **Capacidad de escritura (flexibilidad) versus confiabilidad**

Ejemplo: los punteros de C++ son potentes y muy flexibles, pero poco fiables

EL ZEN DE PYTHON

- El Zen de Python es una colección de 20 principios de software que influyen en el diseño del Lenguaje de Programación Python, de los cuales 19 fueron escritos por Tim Peters en junio de 1999

    - https://es.wikipedia.org/wiki/Zen_de_Python

1. Bello es mejor que feo.
2. Explícito es mejor que implícito.
3. Simple es mejor que complejo.
4. Complejo es mejor que complicado.
5. Plano es mejor que anidado.
6. Espaciado es mejor que denso.
7. La legibilidad es importante.
8. Los casos especiales no son lo suficientemente especiales como para romper las reglas.
9. Sin embargo la practicidad le gana a la pureza.
10. Los errores nunca deberían pasar silenciosamente.
11. A menos que se silencien explícitamente.
12. Frente a la ambigüedad, evitar la tentación de adivinar.
13. Debería haber una, y preferiblemente solo una, manera obvia de hacerlo.
14. A pesar de que eso no sea obvio al principio a menos que seas Holandés.
15. Ahora es mejor que nunca.
16. A pesar de que nunca es muchas veces mejor que *ahora* mismo.
17. Si la implementación es difícil de explicar, es una mala idea.
18. Si la implementación es fácil de explicar, puede que sea una buena idea.
19. Los espacios de nombres son una gran idea, ¡tengamos más de esos!

DOMINIOS DE PROGRAMACIÓN

- **Aplicaciones científicas** (primeras computadoras digitales –1940)
  - – Gran aritmética de punto flotante, eficiencia de ejecución, arreglos y matrices, bucles de conteo
  - – Ejemplos: FORTRAN, ALGOL 60, C
- **Aplicaciones de Negocios** (década de 1950)
  - – Producir informes elaborados, números decimales y datos de caracteres
  - – Ejemplos: COBOL (década de 1960), hojas de cálculo, procesadores de texto, bases de datos (SQL)
- **Inteligencia artificial**
  - – Programación simbólica (nombres en lugar de números, listas enlazadas en lugar de matrices)
  - – Ejemplos: LISP(1959),PROLOG(principios de los 70)

DOMINIOS DE PROGRAMACIÓN

- **Programación de Sistemas**
  - – Software del sistema: Sistema operativo y todas las herramientas de apoyo a la programación de un sistema informático
  - – Ejecución eficiente y rápida, funciones de bajo nivel para controladores de dispositivos periféricos
  - – Ejemplos: PLS/2(IBM Mainframe), BLISS (Digital), C (Unix)
- **Lenguajes de secuencias de comandos**
  - – Se pone en un archivo la lista de comandos (Script) a ejecutar.
  - – Ejemplos: sh, csh, tcsh, awk, gawk, tcl, perl, javascript
- **Lenguajes de propósito especial**
  - – Ejemplos: RPG (Business Reports), SPICE (Simulación de Circuitos Electrónicos), SPSS (Estadísticas), Latex (Preparación de documentos). HTML, XML (prog. web)

CATEGORÍAS DE LENGUAJES

- **Imperativo**

– Las características centrales son las variables, las instrucciones de asignación y la iteración.

– Incluir lenguajes que soporten la programación orientada a objetos

– Incluir lenguajes de script

– Incluir los lenguajes visuales

– Ejemplos: C, Java, Perl, Visual BASIC .NET, C++

- **Funcional**

– El medio principal para hacer cálculos es aplicando funciones a parámetros dados

– Ejemplos: LISP, Scheme,, ML, F#

CATEGORÍAS DE LENGUAJES

- **Lógica**

– Basado en reglas (las reglas se especifican sin ningún orden en particular)

– Ejemplo: Prolog

- **Híbrido de marcado/programación**

– Lenguajes de marcado extendidos para soportar algo de programación

– Ejemplos: JSTL, XSLT

“HELLO WORLD” EN DIFERENTES LENGUAJES

https://www.geeksforgeeks.org/hello-world-in-30-different-languages/

RESUMEN

- El estudio de los lenguajes de programación es valioso por varias razones:

– Aumentar nuestra capacidad de utilizar diferentes construcciones

– Permitirnos elegir lenguajes de forma más inteligente

– Facilita el aprendizaje de nuevos lenguajes

- Los criterios más importantes para evaluar los lenguajes de programación incluyen:

– Legibilidad, capacidad de escritura, confiabilidad, costo

BIBLIOGRAFIA

Concepts of Programming Languages, 11/E. Robert W. Sebesta, ISBN: 978-0-13-394302-3. 2016. Pearson.