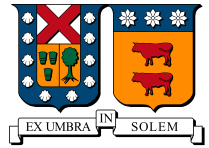


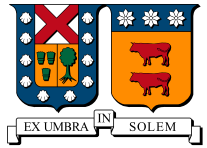
Introducción a Lenguajes de Programación

Roberto Díaz
roberto.diazu@usm.cl

Universidad Técnica Federico Santa María
Departamento de Informática – Santiago, Chile

2021-2

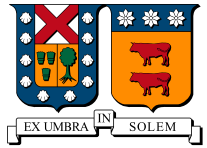
¿Qué es un lenguaje de programación?



- Un *modelo computacional* es una colección de valores y operaciones
- Una *computación* es la aplicación de una secuencia de operaciones a un valor para producir otro valor
- Un *programa* es una especificación de una computación
- Un *lenguaje de programación* es una notación para escribir programas

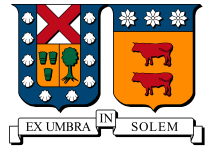
Aaby Anthony (2004). Introduction to Programming Languages.

¿Por qué estudiar lenguajes?



- Incrementar capacidad de expresar ideas.
- Mejor base de conocimiento para elegir lenguajes apropiados.
- Incrementar habilidad de aprender nuevos lenguajes.
- Mejor comprensión del significado de la implementación.
- Mejor uso de lenguajes ya conocidos.
- Mejor visión del avance de la informática y la computación.

Dominios de Programación

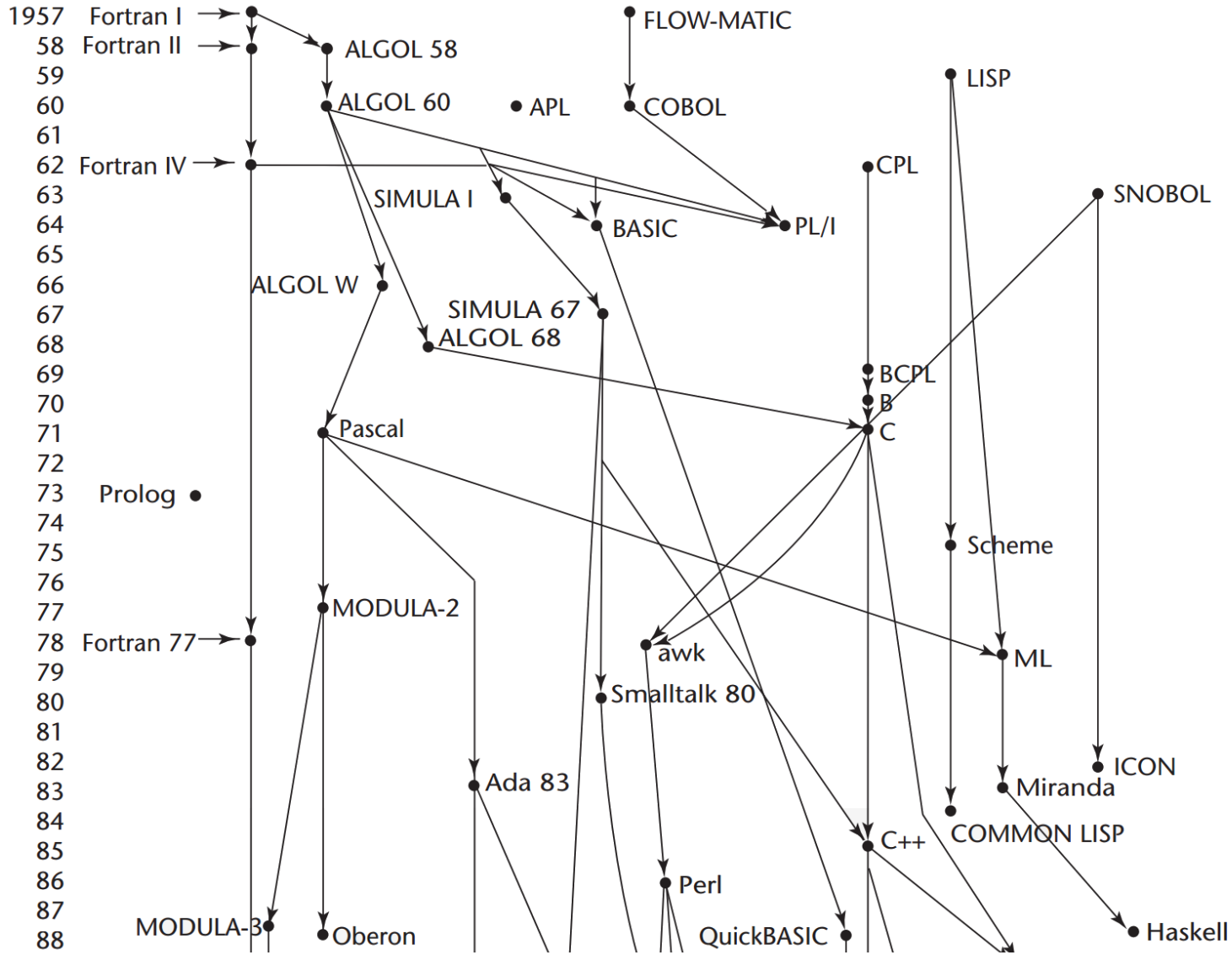


- Aplicaciones de negocio (sistemas de información)
- Aplicaciones científicas y de ingeniería
- Programación de sistemas
- Inteligencia artificial
- Aplicaciones Web
- Lenguajes de propósitos especiales (p.e. SQL, Simula, XML, etc.)

Evolución de Lenguajes

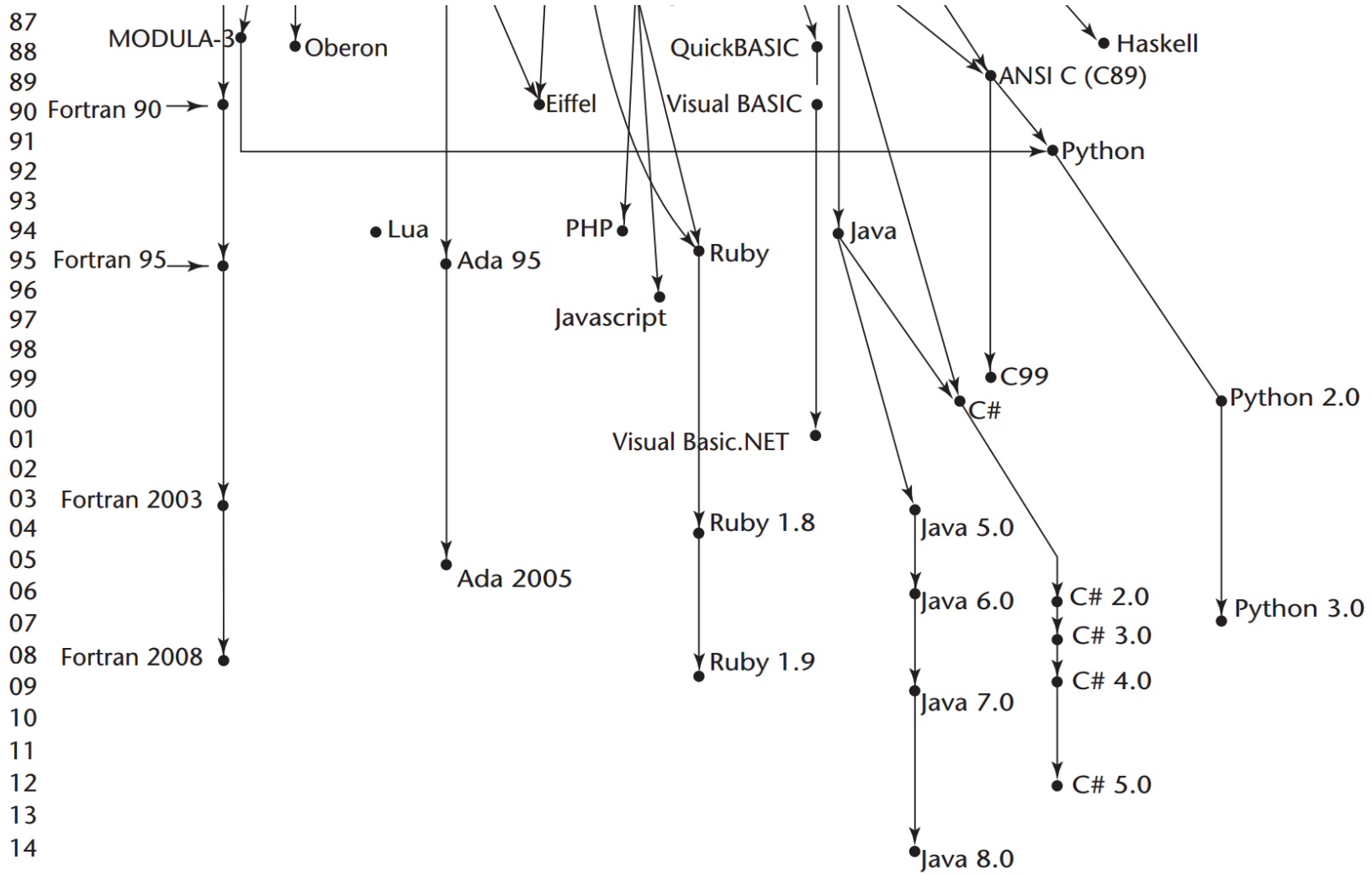
- **Lenguajes de Máquina**: primeros programas; Babbage (1837), Turing (1936), Zuse (1941), ENIAC (1946)
- **Lenguajes de Ensamblaje (assembly)**: código simbólico y herramientas de software; IBM (1954)
- **Primeros Lenguajes de Alto Nivel**: independiente de máquina; FORTRAN (1957), LISP (1958), COBOL (1959)
- **Lenguajes Estructurados**: ALGOL (1960), ALGOL 68, PASCAL (1970), C (1972) y ADA (1979)
- **Lenguajes Orientados a Objetos**: Simula (1967), Smalltalk (1980), C++ (1983), Eiffel (1986), Java (1995)
- **Lenguajes de Scripting**: JCL (1964), RUNCOM (1964), SH (1971); GREP (1973), AWK (1977); TCL (1988); PERL (1987), PYTHON (1991); JavaScript (1995), PHP (1995), Ruby (1995)

Evolución de Lenguajes



SEBESTA, Robert W. Concepts of programming languages 11th Edition. Pearson Education Limited, 2016.

Evolución de Lenguajes



SEBESTA, Robert W. Concepts of programming languages 11th Edition. Pearson Education Limited, 2016.

Paradigmas de Programación

- **Imperativo**: Basado en Máquina de von Neumann. Ejecución secuencial, variables de memoria, asignación y E/S. Típicamente procedural. Dominan principalmente por mejor desempeño. Ejemplos: Fortran, Algol, Pascal y C.
- **Orientado a Objetos**: Conjunto de objetos (piezas) que interactúan controladamente intercambiando mensajes. Normalmente extienden el paradigma imperativo. Ejemplo: Smalltalk, C++ y Java.
- **Funcional**: Basado en cálculo Lambda. Usa funciones y recursión. Ejemplos: LISP, Scheme, Haskell.
- **Lógico**: Basada en cálculo de predicados (lógica simbólica). Está fundamentalmente basado en reglas. Ejemplo: PROLOG.
- **Declarativo**: Se declara lo que se quiere hacer, no cómo. Es más abstracto al no especificar un algoritmo. Ejemplos: SQL y PROLOG.

Algoritmo MCD

```c
int mcd(int a, int b) { //C
  while (a != b) {
    if (a > b) a = a - b;
    else b = b - a;
    }
  return a;
}
```

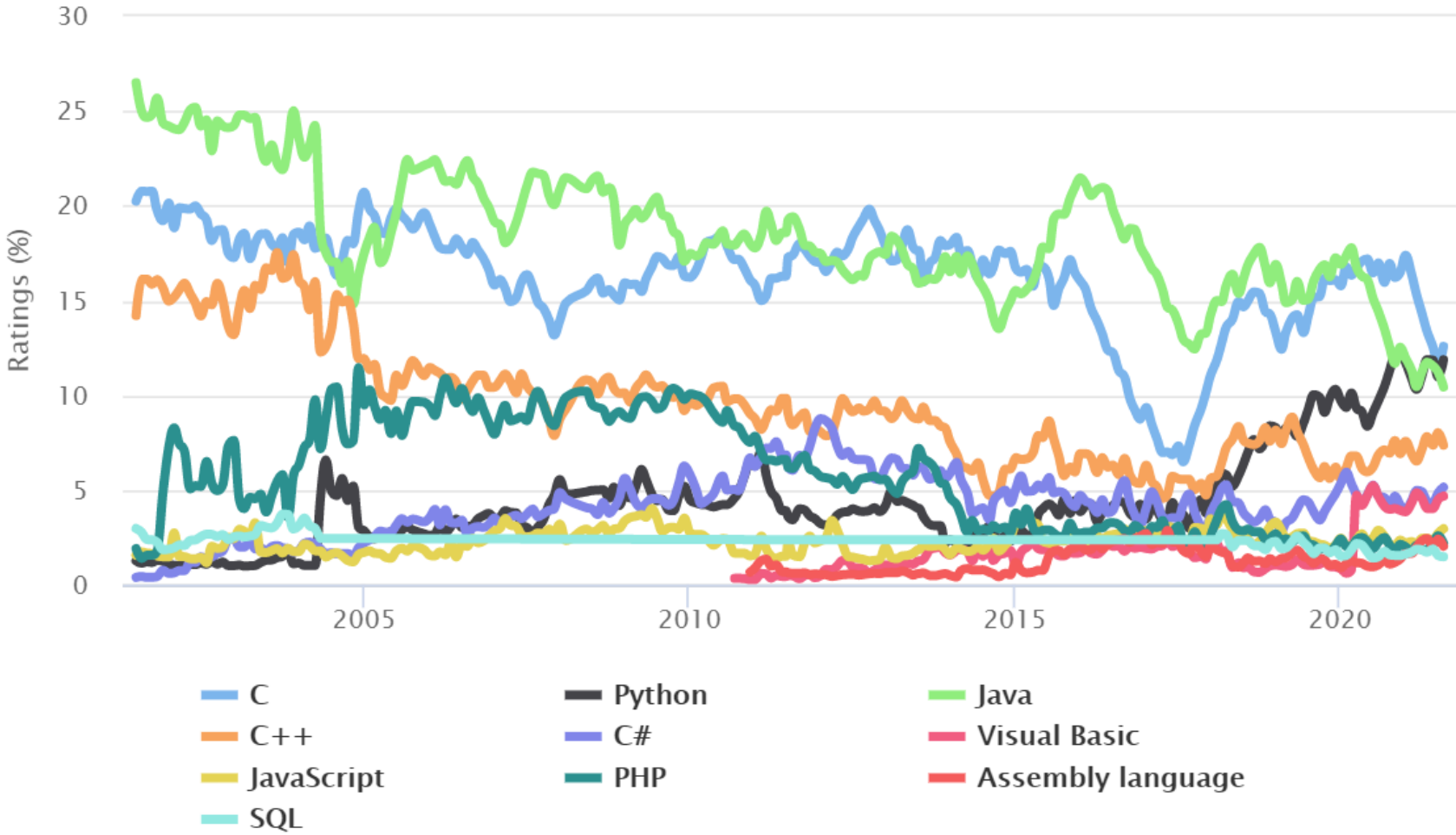
```scheme
(define mcd ;;Scheme
  (lambda (a b)
    (cond
      ((= a b)  a )
      ((> a b)  (mcd (- a b) b) )
      (else     (mcd (- b a) a) )
    )
  )
)
```

```prolog
mcd(A,B,G) :- A = B, G = A. %Prolog
mcd(A,B,G) :- A > B, C is A-B, mcd(C,B,G).
mcd(A,B,G) :- B > A, C is B-A, mcd(C,A,G).
```

Otros modelos de programación

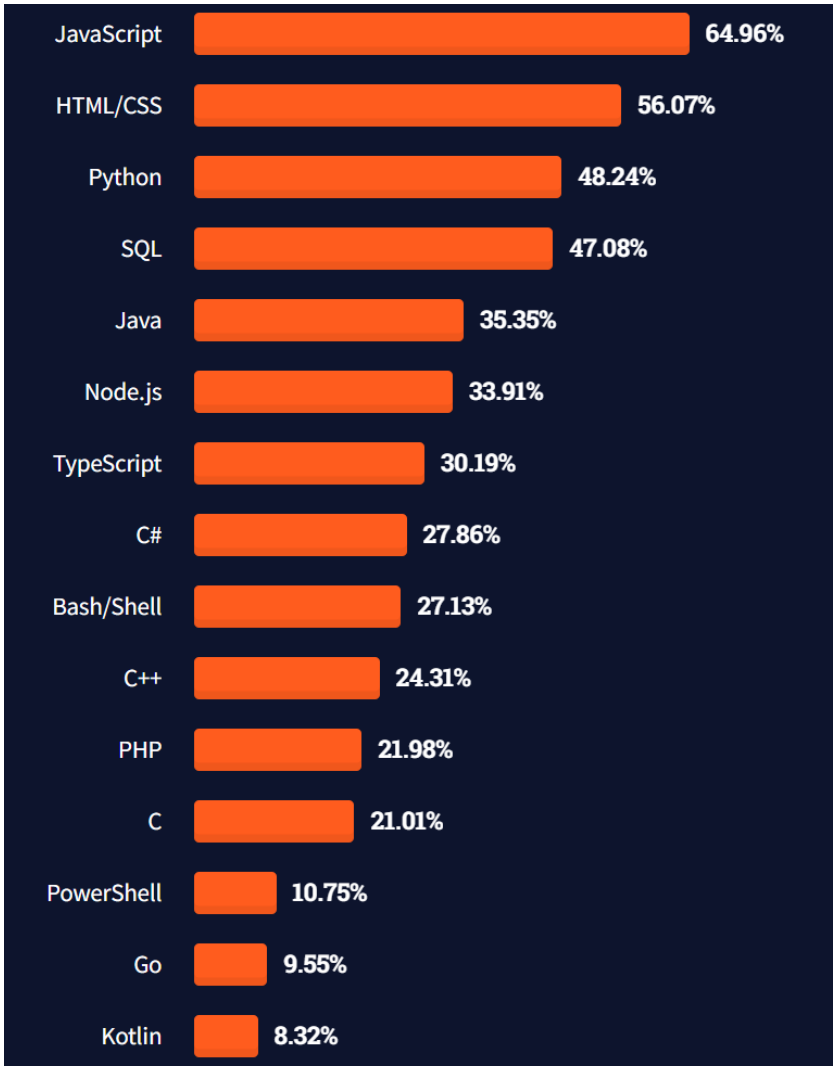
- **Programación basada en Eventos**: flujo del control está determinado por eventos que procesa el manejador de eventos. Ejemplos: Interfaces gráficas, manejo de interrupciones, sistema de sensores.
- **Programación Concurrente**: Conjunto de procesos cooperativos que se pueden ejecutar en paralelo. Se requiere sincronización en el acceso a recursos compartidos. Ejemplos: Sistema operativo, sistema distribuido.
- **Programación Visual**: Permite crear programa manipulando objetos gráficos. Normalmente se integra con otros lenguajes. Ejemplos: Ingeniería de software, Kodu (juegos), LabVIEW (ingeniería).
  - No se debe confundir con un ambiente de programación visual (e.g. Visual Studio)

Popularidad de Lenguajes



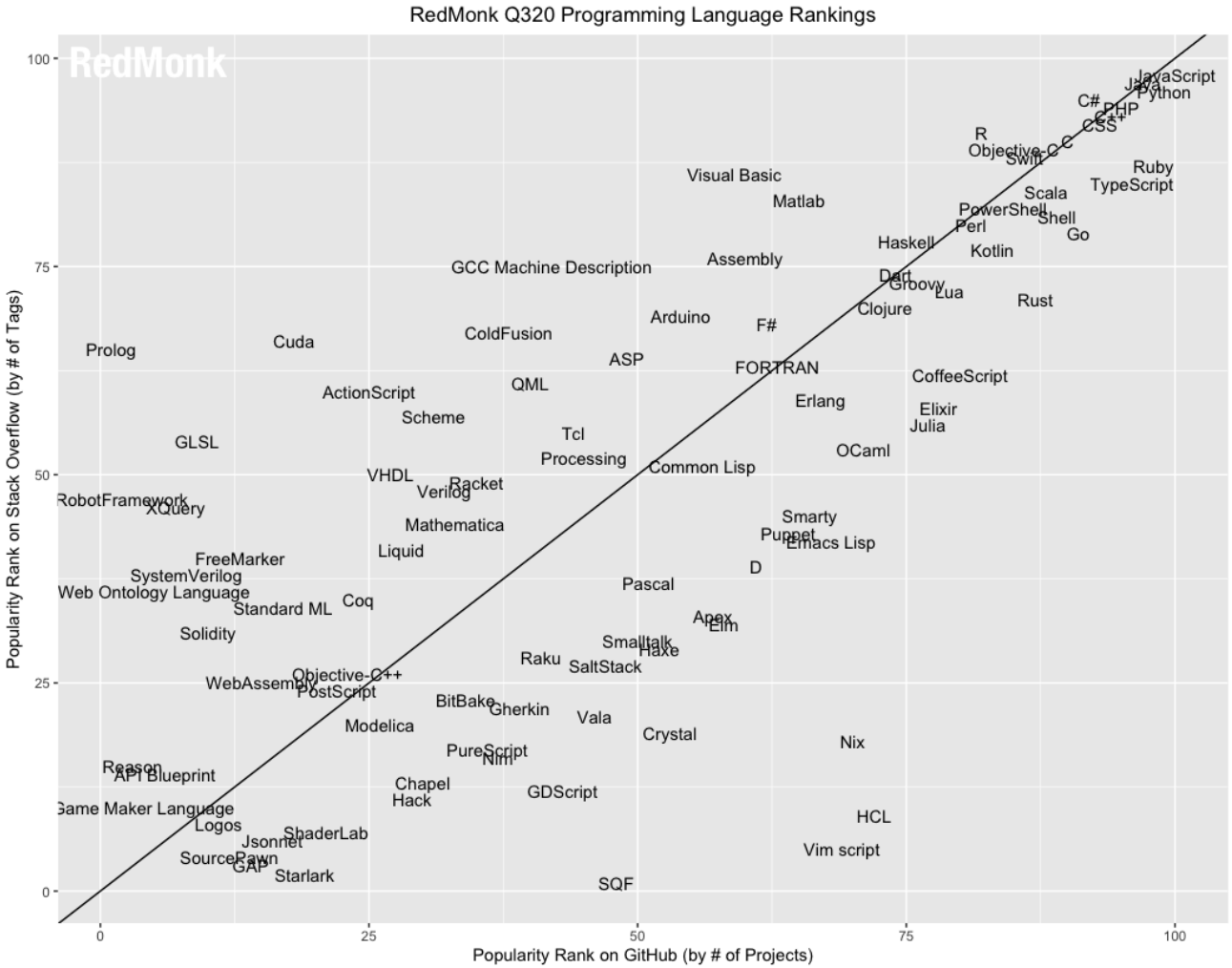
TIOBE Index for August 2021 (`https://www.tiobe.com/tiobe-index/`)

Popularidad de Lenguajes



Stack Overflow Developer Survey 2021 (`https://insights.stackoverflow.com/survey/2021`)

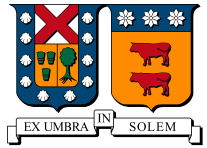
Popularidad de Lenguajes



1. JavaScript
2. Python
2. Java
4. PHP
5. CSS
5. C++
5. C#
6. Typescript
7. Ruby
8. C

The Programming Language Rankings: June 2021 (`https://redmonk.com/sogrady/2020/07/27/language-rankings-6-20/`)

Popularidad de Lenguajes



- The most popular programming languages in 2021
  `https://darly.solutions/the-most-popular-programming-languages-in-2021/`
- PYPL PopularitY of Programming Language
  `https://pypl.github.io/PYPL.html`
- Languish - Programming Languages Trends
  `https://tjpalmer.github.io/languish/`
- Github Language Stats
  `https://madnight.github.io/githut`

Abstracciones

- **Datos**
  - **Primitivos**: Tipos de datos básicos y variables (e.g. enteros y reales, caracteres)
  - **Simples**: Tipo de datos no estructurado, que puede ser primitivo o definido por el usuario en base a uno primitivo
  - **Estructurados**: permiten agrupar o componer conjuntos de datos en una unidad (e.g. arreglo, registro, archivo de texto). Define nuevos tipos de datos

Abstracciones

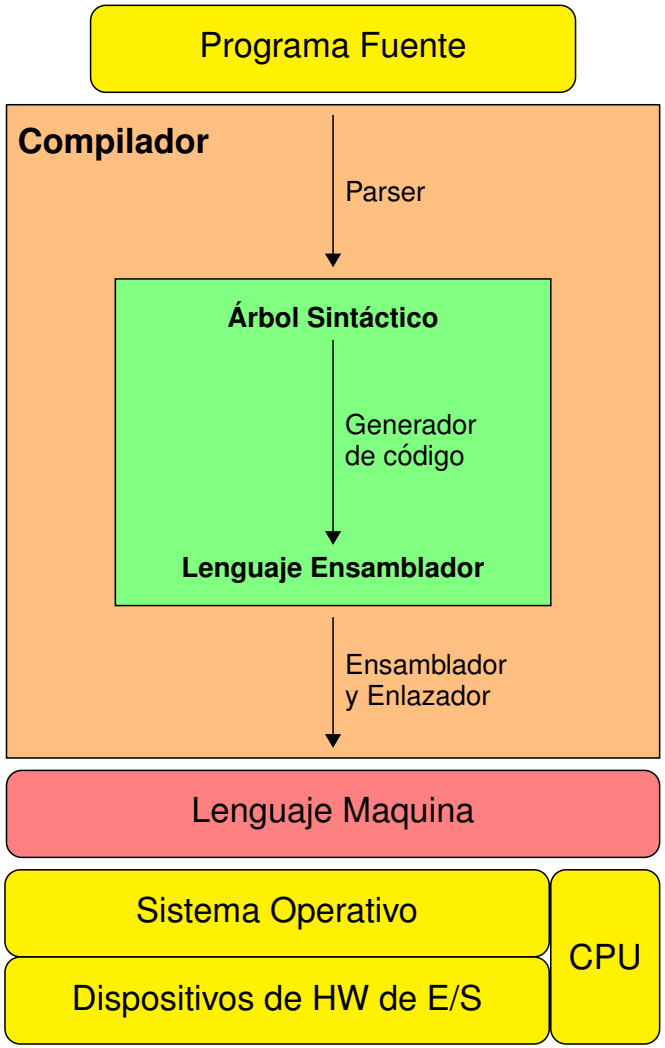
- **Control**
  - **Sentencias**: abstrae conjunto de instrucciones
  - **Estructuras de control**: secuenciación, condición y repetición (e.g. if, while e iterador)
  - **Abstracción procedural**: permite invocar un procedimiento con un nombre y parámetros (e.g. procedimientos, subprogramas y funciones)
  - **Concurrencia**: permite computación paralela (e.g. procesos, hebras y tareas). Se introducen también abstracciones de comunicación
- **Tipos de Datos Abstractos**
  - Agrupa datos y operaciones relacionadas en una unidad (e.g. módulos y clases). Abstrae el tipo de dato con sus operaciones, de la implementación del tipo

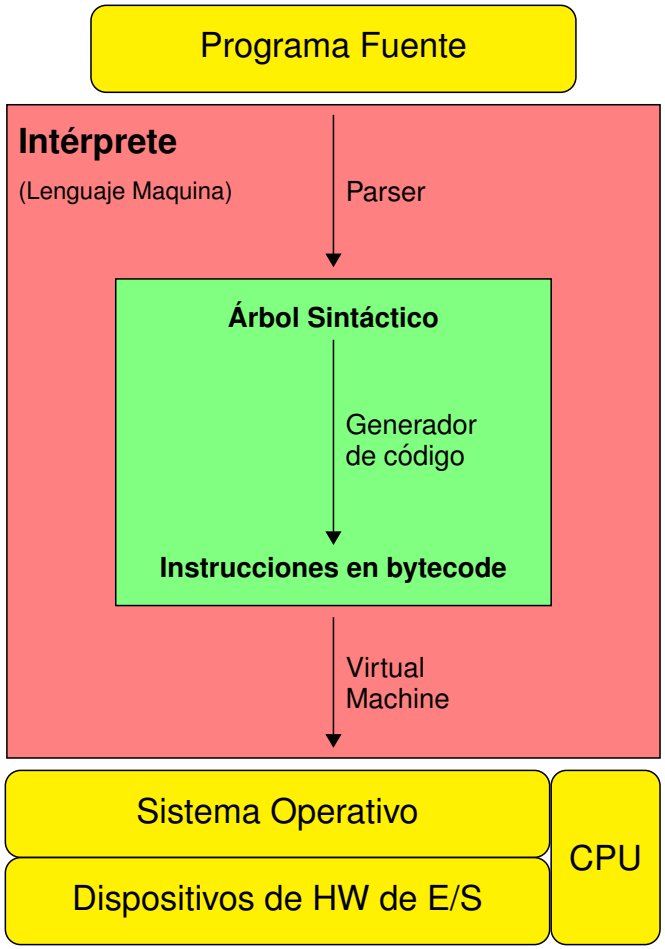
Métodos de Implementación

- **Compilación** (e.g. C, C++)
  Se traduce a lenguaje de máquina para su posterior ejecución (interpretación directa por el procesador)
- **Interpretación** (e.g. LISP, Python)
  Una máquina virtual interpreta directamente el código fuente durante la ejecución.
- **Híbrido** (e.g. Java, C#)
  Se compilan a un lenguaje intermedio, que luego es interpretado

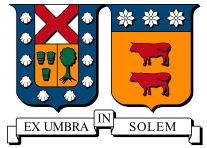
Observación: No confundir preprocesamiento con híbrido

Compilación vs Interpretación





Programación en Grande



- **Modularización**: Necesidad de descomponer los programas en unidades de desarrollo más pequeñas (piezas o módulos de software). Apoya el concepto de ocultamiento de información, que reduce carga cognitiva.
- **Compilación Separada o Independiente**: Módulos de un programa se pueden traducir aparte. Facilita mantención.
- **Reutilización**: Módulos se pueden reutilizar para diferentes programas (e.g. Bibliotecas, módulos y paquetes).
- **Ambientes de Desarrollo**: Se tienen ambientes de desarrollo con diferentes tipos de herramientas y facilidades para apoyar el proceso en todo el ciclo de vida del software.

Criterios de Evaluación

Característica	Facilidad de lectura	Facilidad de escritura	Confiabilidad
Simplicidad	●	●	●
Ortogonalidad	●	●	●
Tipos de Datos	●	●	●
Diseño de Sintaxis	●	●	●
Soporte de Abstracción		●	●
Expresividad		●	●
Chequeo de Tipos			●
Manejo de Excepciones			●
Restricción de alias			●

SEBESTA, Robert W. Concepts of programming languages. Pearson Education India, 1993.

Aspectos de Diseño

- **Arquitectura**: Máquina objetivo donde se ejecuta. Mayoría de computadores siguen basados en modelo de von Neumann por lo que a veces los lenguajes no calzan con su modelo.
- **Estándares**: Lenguajes populares tienden a estandarizarse, haciéndolos más portables. Se incorpora la estandarización de los tipos de datos primitivos (e.g. enteros, caracteres) y bibliotecas (e.g. STL). Hace más pesado el proceso de definición e innovación.
- **Sistemas legados**: Mantener compatibilidad hacia atrás. Permite mantener código legado (e.g. Cobol y C++). Hace más complejo el diseño de los lenguajes.

Tendencias

Lenguajes tienden a ser multi-paradigma y a especializarse en determinados tipos de problemas.

Fuerte auge de lenguajes de programación para el área de ciencia de datos.

Existe un gran movimiento en el desarrollo de lenguajes para programación de aplicaciones Web y móviles.

Lenguajes basados en SQL siguen dominando el área de base de datos.

Referencias

- Capítulos I y II de [Sebesta, 2015]
- Capítulos I y II de [Louden, 2011]
- Capítulo I de [Tucker, 2006]
- Capítulo I de [Kent, 2019]