LENGUAJES DE PROGRAMACIÓN

Equipo de Profesores:
Jorge Díaz Matte - José Luis Martí Lara
Wladimir Ormazabal Orellana

Unidad 1

Introducción a los Lenguajes de Programación


Programming
Clojure
.NET
AJAX
PowerShell
Assembly
Scheme
MATLAB
VimL
XML
CODING
Cuda
Groovy
Delphi
WebDNA
Active Server Page
Visual Basic
Perl
R
Cobol
OpenEdge ABL
Mathematica
Scala
Ada
Swift
Prolog
Forth
Rexx
Python
JavaScript
Pascal
PHP
Haskell
Dart
Shell
BASH
Objective-C
Gambas
TCL
Java
GO
HTML
Transact-SQL
C
Ruby
Processing
SAS
Bourne
open/source
Erlang
VBScript
ActionScript
ARDUINO
C++
Scratch
Liberlab
Logo
C#
Synergy
SQL
Smalltalk
Abap
Elixir
Lisp
CoffeScript
Fortran
AvtiveVFP

¿Porqué estudiar Lenguajes de Programación?

- Incremento de la capacidad de expresar ideas
- Mejor base de conocimiento para elegir lenguajes apropiados
- Incremento de la 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

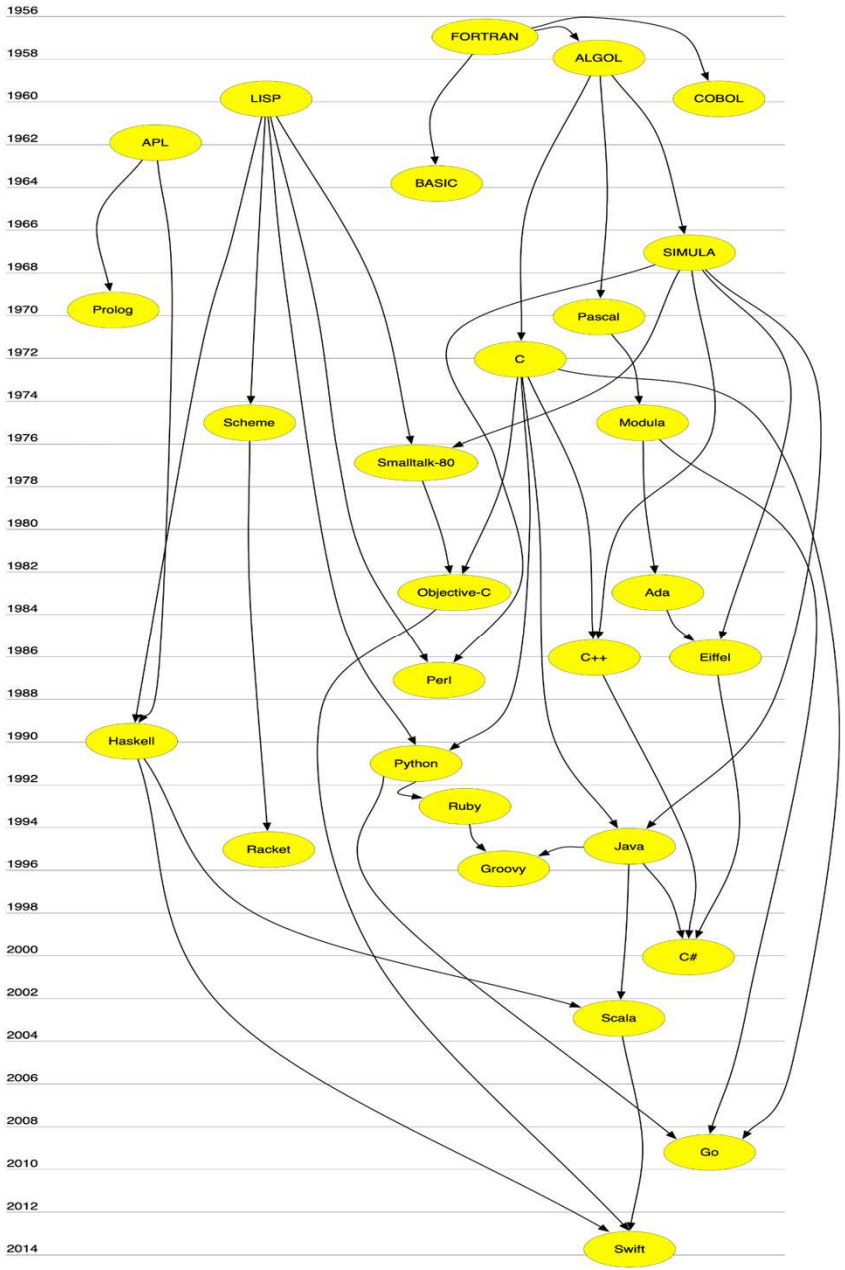
Dominios de Programación

- Aplicaciones de negocio
- Aplicaciones científicas y de ingeniería
- Programación de sistemas
- Ciencia de datos
- Inteligencia artificial
- Aplicaciones web

Evolución de los Lenguajes de Programación

- Lenguajes de Máquina: primeros programas; Babbage (1837), Turing (1936), Zuse (1941), ENIAC (1946)
- Lenguajes de Ensamblaje: 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 Objeto: 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)

Genealogía de los Lenguajes de Programación



https://domingogallardo.github.io/apuntes-lpp/teoria/tema01-historia-lenguajes-programacion/tema01-historia-lenguajes-programacion.html

Paradigmas de Programación

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

Ejemplo: Máximo común denominador (MCD)

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

Lenguaje C

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

Lenguaje Scheme

```
mcd(A,B,G) :- A = B, G = A.
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).
```

Lenguaje Prolog

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: Sistemas operativos, sistemas distribuidos.
- 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 (ej.: Visual Studio)

Popularidad de los Lenguajes de Programación

1 JavaScript
2 Python
3 Java
4 PHP
5 C#
6 TypeScript
7 CSS
8 C++
9 Ruby
10 C
11 Swift
12 Go
13 R
14 Shell
14 Objective-C

Feb 2024	Feb 2023	Change	Programming Language	Ratings	Change
1	1		Python	15.16%	-0.32%
2	2		C	10.97%	-4.41%
3	3		C++	10.53%	-3.40%
4	4		Java	8.88%	-4.33%
5	5		C#	7.53%	+1.15%
6	7	^	JavaScript	3.17%	+0.64%
7	8	^	SQL	1.82%	-0.30%
8	11	^	Go	1.73%	+0.61%
9	6	v	Visual Basic	1.52%	-2.62%
10	10		PHP	1.51%	+0.21%
11	24	^^	Fortran	1.40%	

https://tecsify.com/blog/top-lenguajes-2024/

Worldwide, Aug 2024 :

Rank	Change	Language	Share	1-year trend
1		Python	29.6 %	+1.7 %
2		Java	15.51 %	-0.3 %
3		JavaScript	8.38 %	-1.0 %
4		C#	6.7 %	-0.0 %
5		C/C++	6.31 %	-0.2 %
6	↑	R	4.6 %	+0.2 %
7	↓	PHP	4.35 %	-0.6 %
8		TypeScript	2.93 %	-0.1 %
9		Swift	2.76 %	+0.1 %
10	↑	Rust	2.58 %	+0.5 %
11	↓	Objective-C	2.4 %	+0.2 %
12		Go	2.14 %	+0.2 %

https://pypl.github.io/PYPL.html

Abstracciones (1/2)

Datos

- Primitivos: tipos de datos básicos y variables (ej.: enteros y reales, caracteres).
- Simples: tipos de datos no estructurados, que pueden ser primitivos o definidos por el usuario en base a uno primitivo.
- Estructurados: permiten agrupar/componer conjuntos de datos en una unidad (ej.: arreglo, registro, archivo de texto). Define nuevos tipos de datos.

Abstracciones (2/2)

Control

- Sentencias: abstraen un conjunto de instrucciones.
- Estructuras de control: secuenciación, condición y repetición (ej.: if, while e iterador).
- Abstracción procedural: permite invocar un procedimiento con un nombre y parámetros (ej.: procedimientos, subprogramas y funciones).
- Concurrencia: permite computación paralela (ej.: procesos, hebras y tareas). Se introducen también abstracciones de comunicación.

Tipos de Datos Abstractos

- Agrupa datos y operaciones relacionadas en una unidad (ej.: módulos y clases). Abstrae el tipo de dato con sus operaciones, de la implementación del tipo.

Modelos de Implantación

Compilación

- Se traduce a lenguaje de máquina para su posterior ejecución (interpretación directa por el procesador). Ejemplos: C, C++.

Interpretación

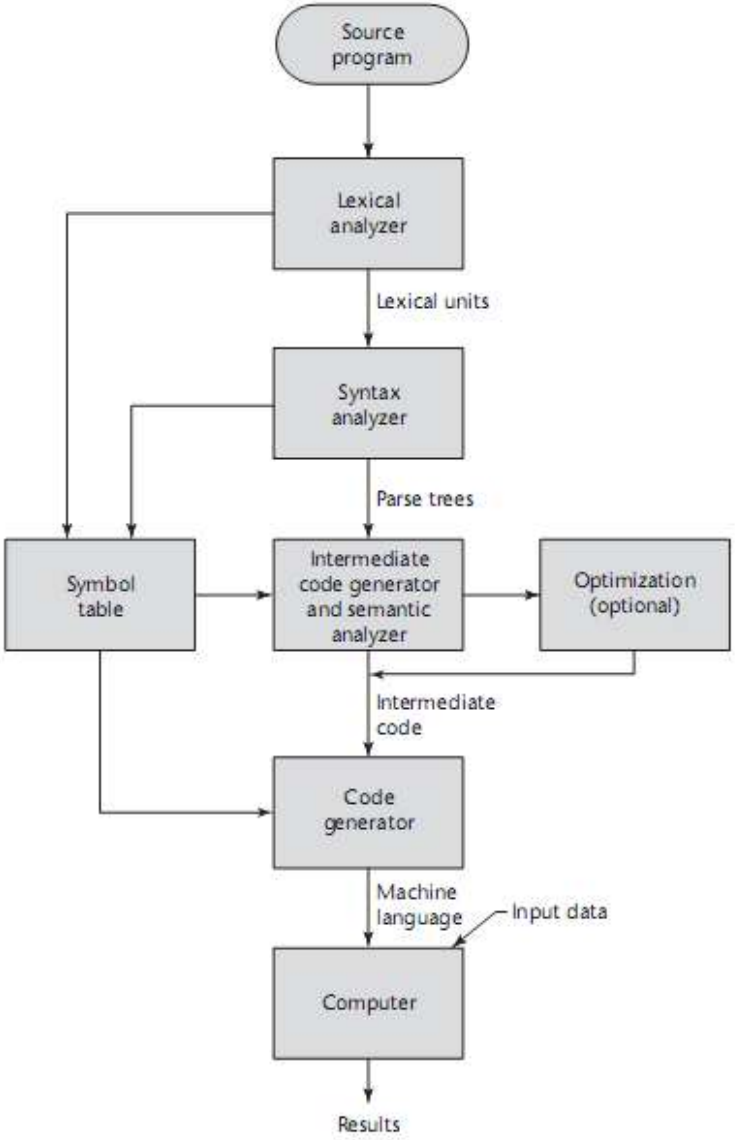
- Una máquina virtual interpreta directamente el código fuente durante la ejecución. Ejemplos: LISP, Python.

Esquema Híbrido

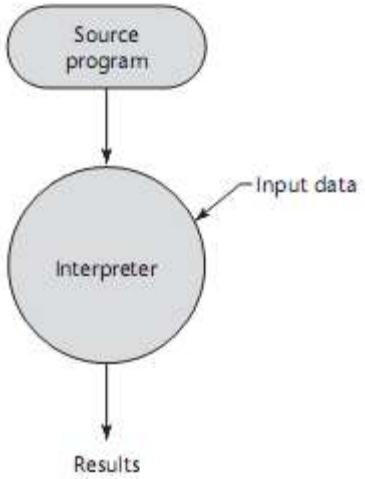
- Se compila a un lenguaje intermedio, que luego es interpretado por una máquina virtual. Ejemplos: Java, C#.

Observación: No confundir preprocesamiento con híbrido.

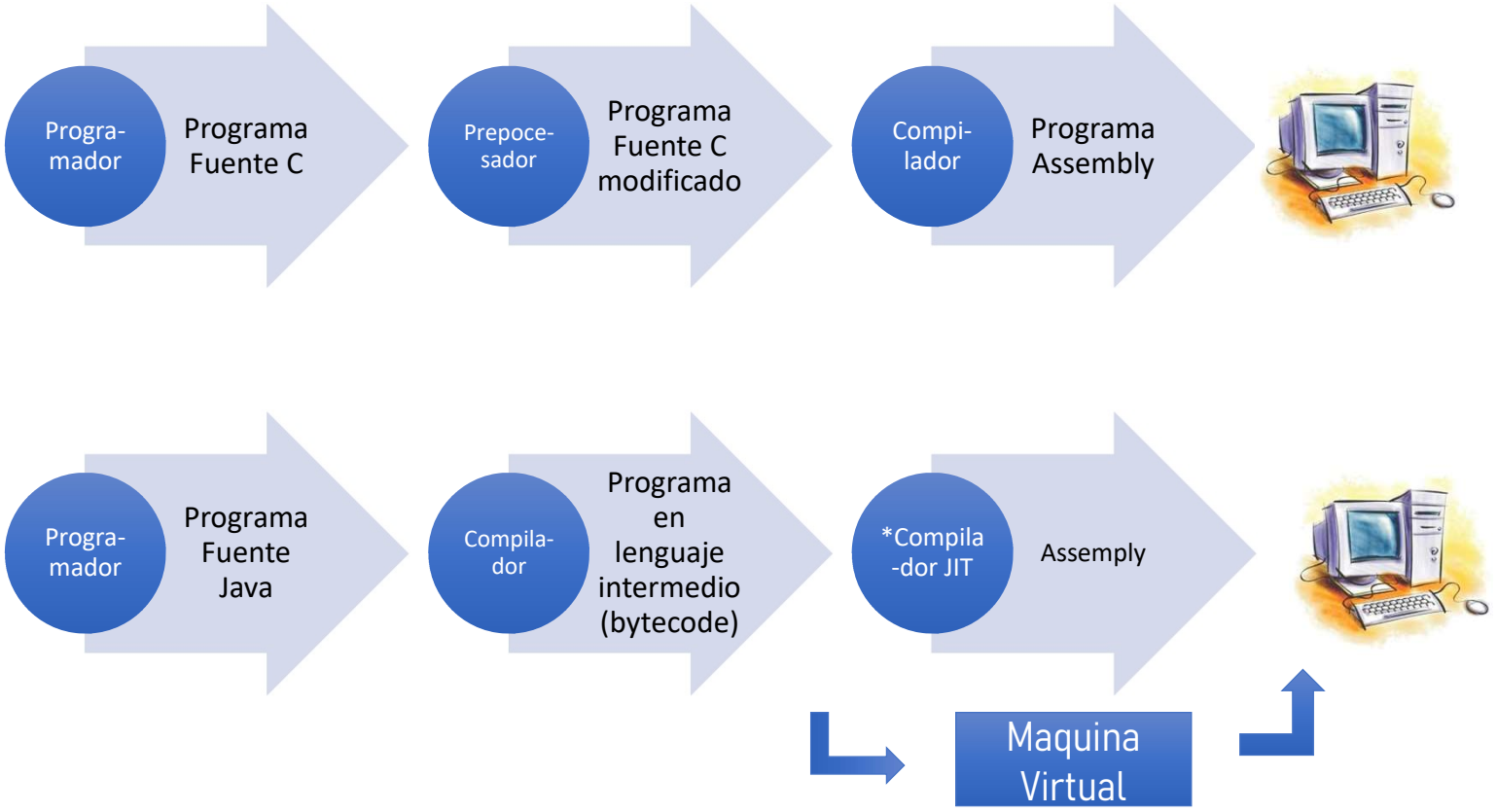
Compilación



Interpretación



Ejemplos de C y de Java



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 (ej.: 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.

Aspectos de Diseño

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

Tendencias

- Lenguajes de programación tienden a ser multiparadigmas (ej.: Python) y a especializarse en determinados tipos de problemas.
- Fuerte auge de lenguajes de programación para el área de ciencia de datos (ej.: Python, Scala).
- Existe un gran movimiento en el desarrollo de lenguajes para programación de aplicaciones Web y móviles (Ej: JavaScript).

Unidad 1

Introducción a los
Lenguajes de Programación

FIN