APUNTE Nº2: Introducción a los Lenguajes de Programación (LDP) II

Prof. Dr. Samuel Sepúlveda
DCI-CEIS-UFRO

2016

CONTENIDOS

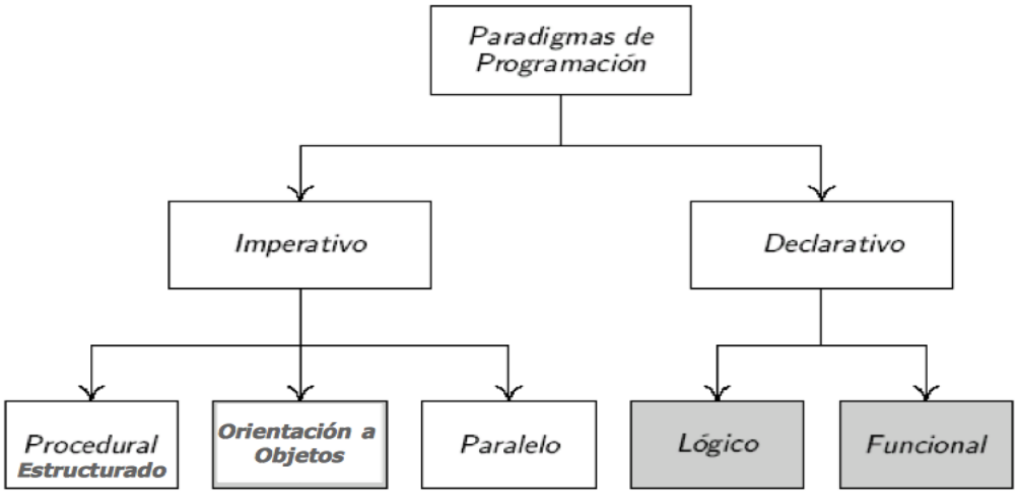
1. Paradigmas
2. Traducción
3. Niveles de los LDP
4. Actividades a desarrollar

1. PARADIGMAS

A menudo se habla que existen diversos paradigmas de programación y entonces...pero…

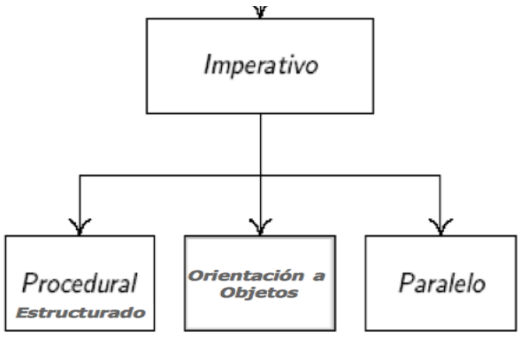
¿Qué es un paradigma?
2. m. Teoría cuyo núcleo central se acepta sin cuestionar y que suministra la base y modelo para resolver problemas y avanzar en el conocimiento; p. ej., en la ciencia, las leyes del movimiento y la gravitación de Newton y la teoría de la evolución de Darwin.
Extraído desde: Diccionario de la RAE, www.rae.es, 23ª Ed., 10/03/14.

1.1 Paradigmas de Programación



Los lenguajes imperativos están orientados al “como" mientras que los declarativos están orientados al “que".

1.2 Paradigma Imperativo

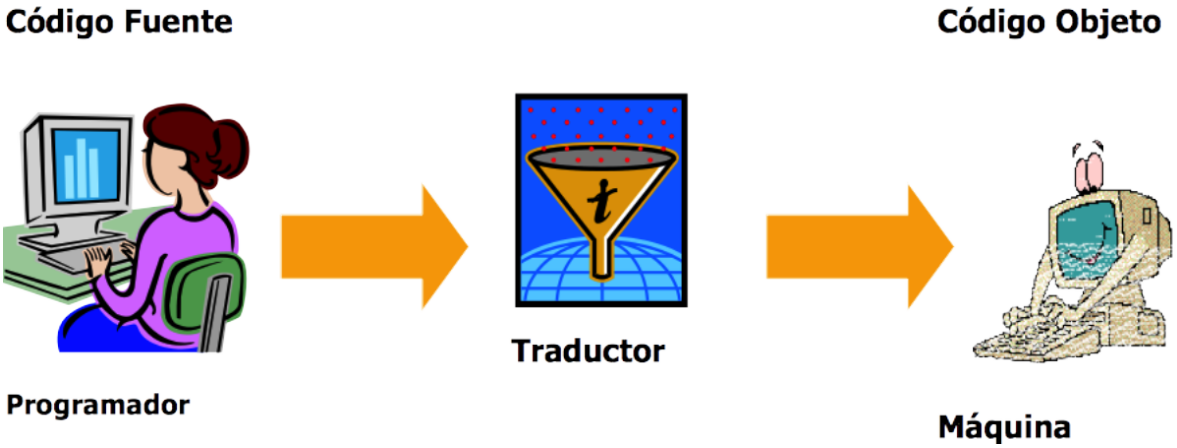


Basan su funcionamiento en un conjunto de instrucciones secuenciales, las cuales, al ejecutarse, van alterando las regiones de memoria donde residen todos los valores de las variables involucradas en el problema que se plantea resolver.

En la POO, las estructuras de datos y los algoritmos se integran en unidades, a menudo llamadas clases.
Ej: C++, Java.

2. ¿Qué es la traducción?

Un LDP se implementa construyendo un traductor, éste traduce los programas que están escritos en el LDP usado a programas en lenguaje de máquina que pueden ser ejecutados directamente por algún computador.



2.1 LDP según su traducción

Según la forma en que los lenguajes son traducidos a la máquina, se puede afirmar que existen al menos 4 tipos de traducción:

- Lenguajes Ensamblados
- Lenguajes Compilados
- Lenguajes Interpretados
- Máquinas Virtuales

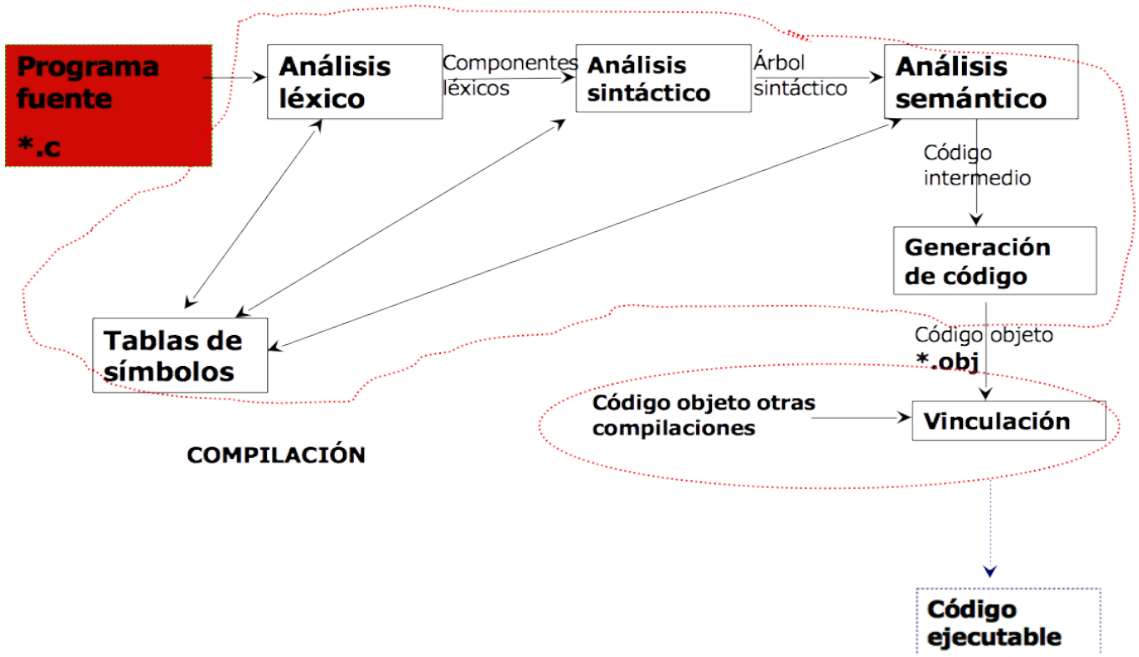
i. Lenguajes ensamblados

- Se refieren al lenguaje ensamblador (assembler)
- Una representación simbólica de las instrucciones correspondientes al lenguaje máquina de alguna arquitectura específica.

ii. Lenguajes compilados

Son aquellos que son traducidos de un lenguaje de alto nivel a lenguaje de máquina o bien a lenguaje ensamblador, produciendo un programa objeto permanente. Ejs: C, C++, Pascal.

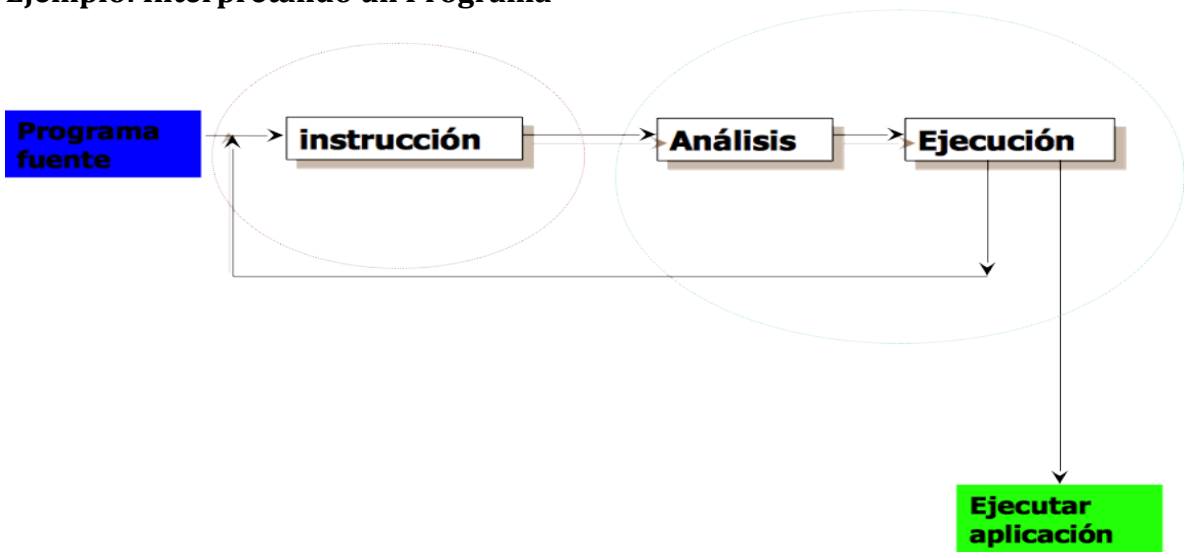
Ejemplo: Compilando un Programa (Lenguaje C)



iii. Lenguajes Interpretados

Cada instrucción es analizada y ejecutada a la vez, lo anterior ofrece mucha interacción con los desarrolladores. Pueden resultar ineficientes, cuando se desea ejecutar repetitivamente un programa. Estos tienen la particularidad, de que no producen código objeto. Ejs: PERL, JavaScript, PHP, vbScript.

Ejemplo: Interpretando un Programa



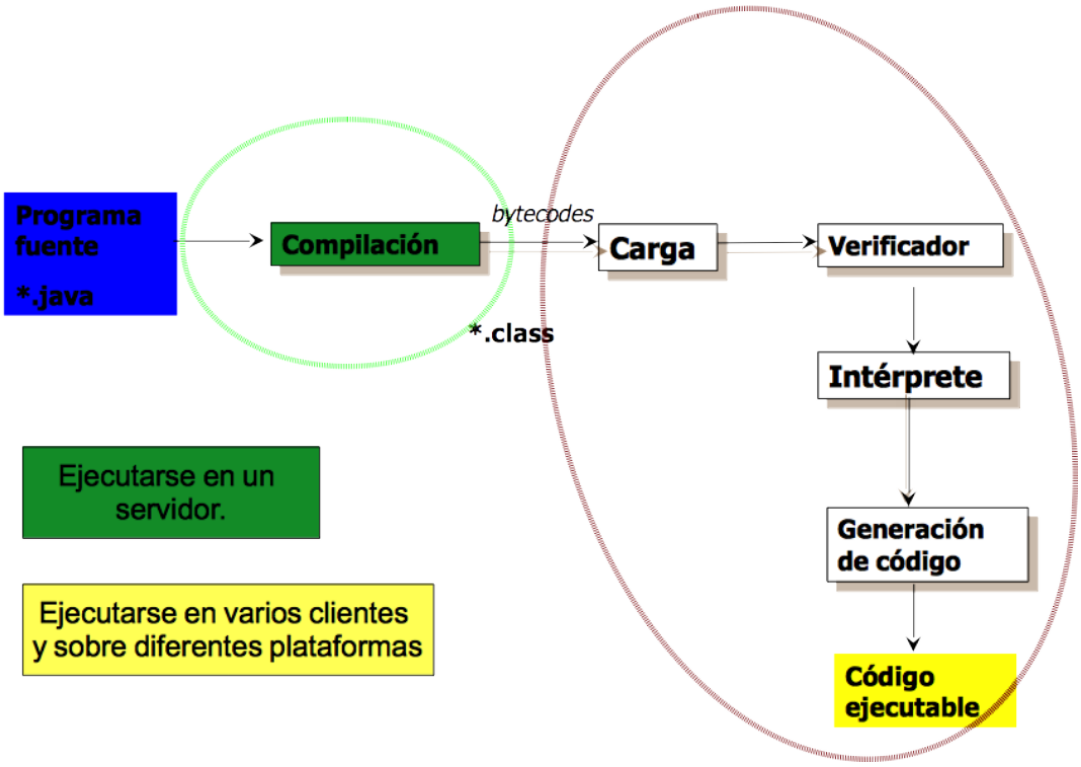
iv. Máquinas Virtuales

Generan un código intermedio entre el LDP y el lenguaje máquina del procesador.

Para su ejecución se requiere de un intérprete de ese código intermedio, la Máquina Virtual, que se encarga de pasarlo al código de máquina específico en la cual se trabaja. Ejs: Java, C#.

Si existe una máquina virtual para diversas plataformas software/hardware para ejecutar las aplicaciones desarrolladas en un lenguaje basado en máquina virtual, entonces es posible afirmar que dicho lenguaje sea multiplataforma.

Ejemplo: Compilando y ejecutando un Programa (Lenguaje Java)



3. Niveles de los LDP

A continuación se presentan los diferentes niveles de LDP y un breve análisis del impacto de estos niveles y sus características.

3.1 Lenguajes de Máquina

Concepto: Es el lenguaje que "entiende" directamente un computador.

Características

- Está en código binario (0,1)
- Muy difícil de interpretar
- Instrucciones muy relacionadas a los circuitos del procesador (que a su vez corresponden a un modelo funcionamiento de la arquitectura presente en la máquina)

3.2 Lenguaje Ensamblador

Concepto: Es el lenguaje que "más fácil" se puede traducir a lenguaje de máquina.

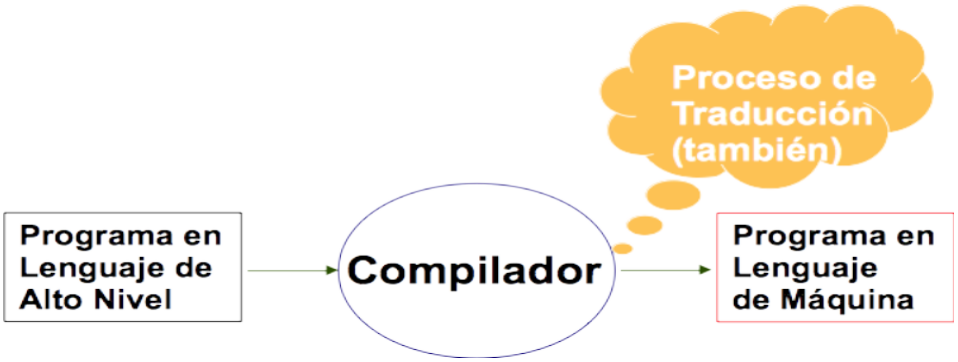
Características

- Instrucciones en base mnemónicos
- Un poco más fácil de interpretar
- Instrucciones muy relacionadas al modelo de funcionamiento de la máquina

3.3 Lenguaje de Alto Nivel

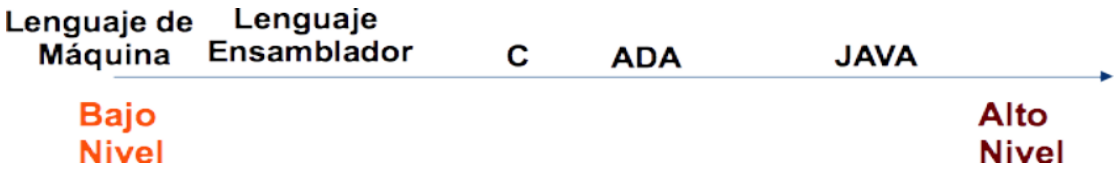
Es posible reconocer algunas ventajas en este tipo de lenguajes:

- Más Legible
- Portable
- Menor tiempo de capacitación para su uso
- Mayor productividad de programadores



3.4 Lenguajes de Alto y Bajo nivel

El concepto de "Alto Nivel" o "Bajo Nivel" es más bien continuo que discreto.



3.5 Ejemplo de Magnitud y Legibilidad

Considere un programa que debe permitir la suma de dos números y cuyo resultado se almacena en una variable. Para lo anterior se requiere almacenar el contenido de los dos números en dos variables *i, j* y almacenar el resultado de la suma *i+j* en una tercera variable *k.*

A continuación se muestra como quedaría la implementación genérica de la solución al problema anterior en un lenguaje máquina, ensamblador y de alto nivel.

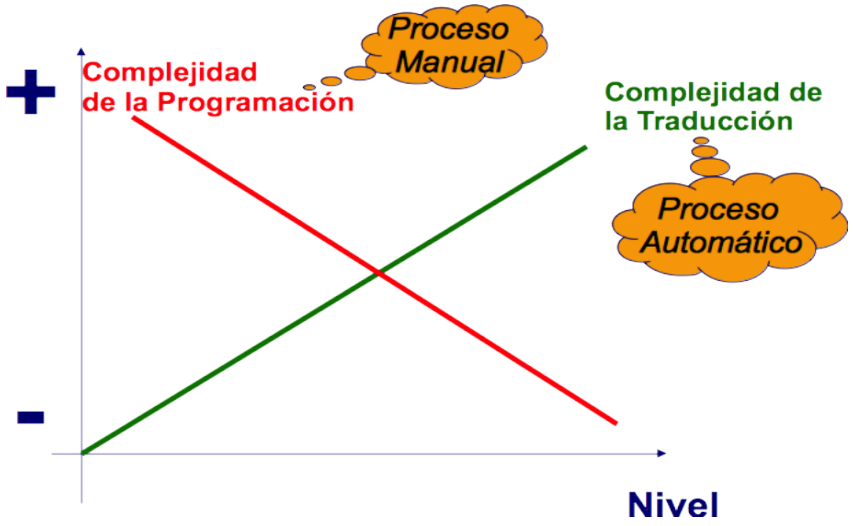
LENGUAJE		
Alto Nivel	**Ensamblador**	**Máquina**
k = i + j	Lda i Add j Str k	0100 0101 1001 0100 0101 1001 0101 0101 0010 1010 1010 1010 0000 0100 1110 0011

Lenguajes de Bajo Nivel

3.6 Lenguajes de Alto Nivel vs Bajo Nivel

A continuación se presenta una gráfica que ilustra la relación existente entre los niveles de los LDP y como aumentan en este caso la complejidad en la programación de una aplicación y su complejidad en la traducción hacia un lenguaje comprensible por la máquina.

Considerando que la programación es una tarea desarrollada por humanos (programadores) y que la traducción es una tarea automática que implica el uso de traductores (compiladores, intérpretes, máquinas virtuales, etc.), en que escenarios o bajo que condiciones usted preferiría o recomendaría el uso de lenguajes de bajo nivel o de alto nivel.



3.7 Recursos de un LDP

A continuación se ilustra mediante un ejemplo como el hardware revisado interactúa con un programa de software cuando es ejecutado por una CPU. Eso tiene sentido porque se desea solucionar un problema mediante un programa computacional.

- Modelo de Computación: Las ideas generales que sustentan las estructuras e instrucciones del LDP.
- Tipos de Datos y Operaciones: La definición de las formas aceptadas de representar elementos del problema y la definición de las operaciones sobre ellos.
- Recursos de Abstracción: Las estructuras disponibles en el Lenguaje para ayudar a la simplificación de la comprensión y la operación de los elementos del problema.

3.8 Ejemplo de los recursos de un LDP – Java

- Modelo de Computación: Basado Máquina de Von Neumann, secuencialidad de instrucciones, almacenamiento en memoria de variables, manipulación de ella mediante operaciones.
- Tipos de Datos y Operaciones: Tipos de Datos como Enteros, Reales, Caracteres, Punteros y Operaciones como asignación, suma, resta, etc...
- Recursos de Abstracción: Capacidad de definir clases, métodos, interface, packages, etc.

3.9 Ejemplo Niveles de LDP

A continuación se presenta un conjunto de ejemplos de códigos fuentes para la solución de un problema determinado, el cual ha sido desarrollado en diferentes LDP para apreciar sus diferencias a la hora de codificar (números de líneas de código, complejidad, tamaño del archivo, etc.).

Ej. Imprimir el mensaje “Hola mundo!!!” en pantalla.

- LDP C

```
#include <stdio.h>
main() {
  printf(“Hola mundo!!!”);
}
```

- LDP PERL

```
print “Hola mundo!!!”;
```

- Assembler para el compilador GCC de LDP C

```
.file "hola.c"
.section .text
LC0:
   .ascii "Hola Mundo!!!\0"
   .p2align 4
   .globl _main
_main:
   push %ebp
   movl %esp, %ebp
   subl$8, %esp
   andl$-16, %esp
subl  $12, %esp
   pushl $LC0
   call _printf
   addl $16, %esp
   movl %esp, %ebp
   popl %ebp
   ret
  .ident "GCC:
(GNU) 3.0.3"
```

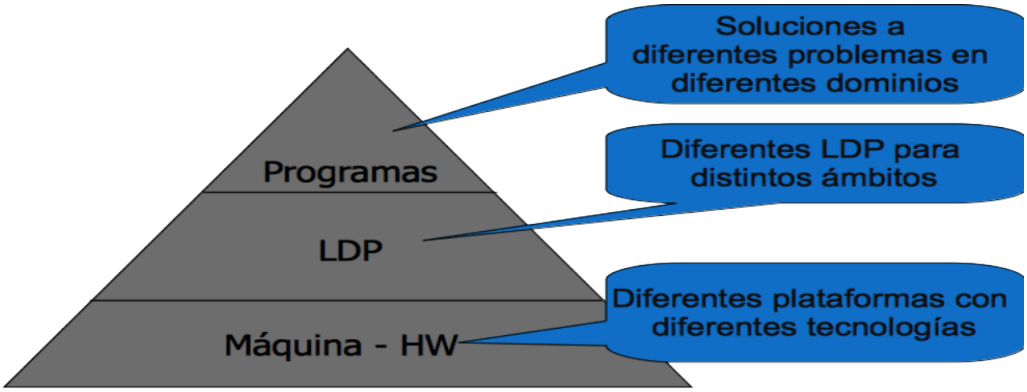
4. Actividades a desarrollar

1. Analice e infiera las ventajas y desventajas de usar lenguajes basados en compiladores, intérpretes y máquina virtual.
2. Establezca escenarios en que sea conveniente usar lenguajes de máquina, ensamblador y alto nivel. Justifique su respuesta.

3. Inferir sobre los ejemplos de código mostrados y tabla resumen, relacionar con el gráfico mostrado sobre Complejidad de Traducción v/s Complejidad de Programación.

Resumiendo

- Paradigmas de programación
- Clasificaciones de los LDP
- Mecanismos de traducción
- Niveles de los LDP



Cada capa amplía las posibilidades de la capa anterior