Herramientas Computacionales para la Astroinformática

Cristian A. Vega-Martínez (oficina: IMIP, Académicos 2)
Facundo A. Gómez

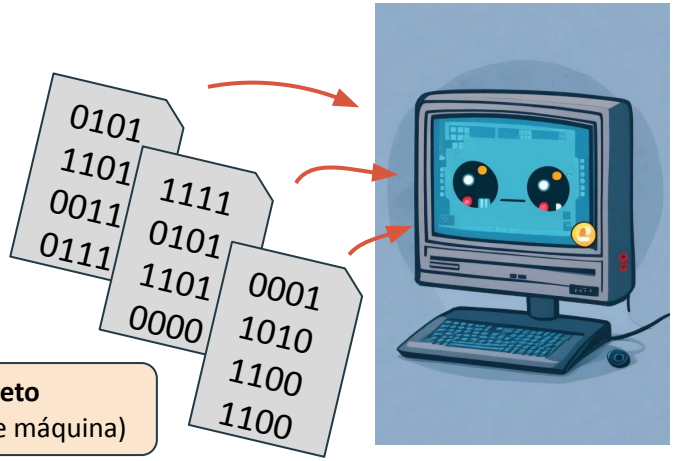
UNIDAD II
Herramientas Programación Avanzada

Lenguajes de programación

Lenguaje de programación

Un lenguaje de programación es un **lenguaje formal** (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que permite a una persona escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de **algoritmos** con el fin de controlar el comportamiento físico o lógico de un sistema informático, para que de esa manera se puedan obtener diversos **tipos de datos** o **ejecutar determinadas tareas**.

Deben ser traducidos a lenguaje de máquina para que puedan ser interpretados por una computadora.



Código objeto
(binario o lenguaje máquina)

Lenguajes de programación

CONSIDERACIONES

- Tienen su propia **sintaxis** (definida por cada lenguaje).
- Están asociados a **paradigmas de programación** (e.g. imperativa, declarativa, orientada a objetos).
- Pueden ser **compilados** o **interpretados**.
- Tienen asociado un **nivel de abstracción**.

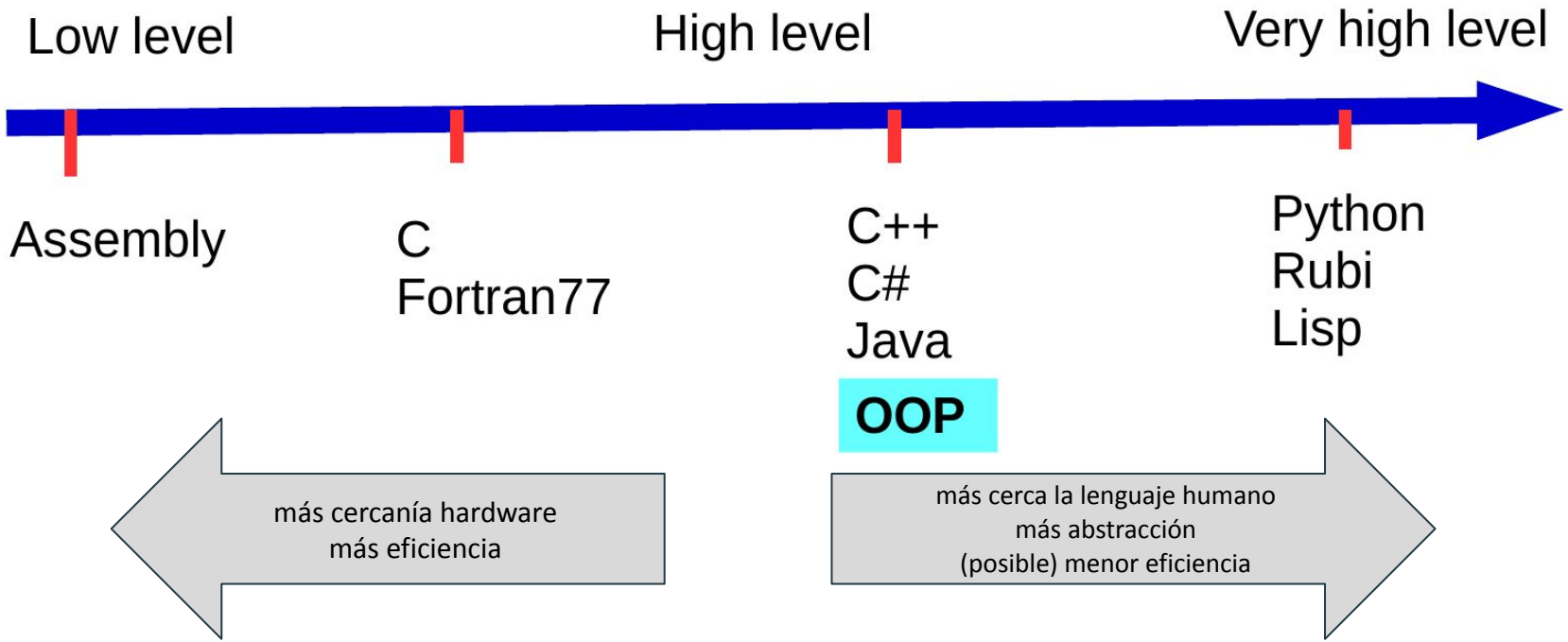
Nivel de abstracción

Se refiere al grado en que el lenguaje permite trabajar de manera independiente del hardware o del sistema operativo subyacente. En otras palabras, indica cuán cercano o lejano está un lenguaje de las instrucciones de máquina que el hardware puede ejecutar directamente.
Se puede entender como el nivel de complejidad con que una persona interactúa con el sistema.

Ejemplo: calcular una suma `SUM(x, x = 1..N)`

- bajo nivel: `r = 0;  r = r + 1;  r = r + 2;  r = r + 3; ... r = r + N`
- alto nivel: `for(i=1, r=0; i<=N; i++) r += i;`
- más alto: `sum(1, N)`

Nivel de abstracción



Bajo nivel: Ensamblador (Assembly)

- Código fuente indica directamente las acciones a ser realizadas por el procesador.
- Está muy vinculado con las posibilidades del hardware, sintaxis simple, **alta eficiencia**.
- Se usa típicamente en el desarrollo de sistemas operativos y drivers.

Ejemplo:

```
MOV AX, 1
ADD AX, 2
```

Este código mueve el valor 1 a un registro (AX) y luego suma 2 a ese registro.

Ejemplo: algoritmo de alto nivel

```
def reduce_image(image, bias, dark, flat):
    """ Rutina de reducción de imágenes astronómicas"""

    result = image.copy()
    result -= bias
    result -= (dark - bias)
    result /= (flat - bias - dark)

    return result
```

Ejemplo: algoritmo de alto nivel

```
def vestir(user):
    """ Rutina para vestir un usuario"""

    user.put.underwear()
    user.put.pants()
    user.put.shirt()
    user.put.tie()
    user.put.hat()
```

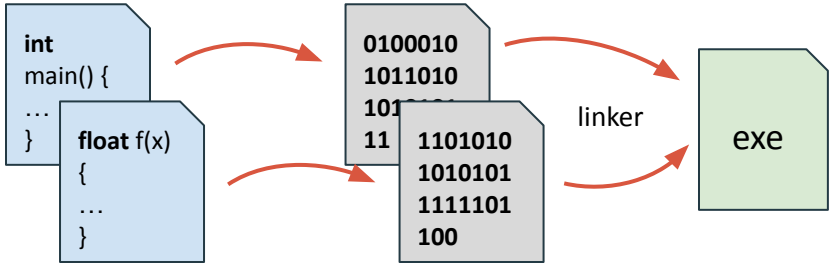
Nivel de abstracción

Nivel	Ventajas	Desventajas
Bajo Nivel	- Control total del hardware. - Eficiencia en la ejecución.	- Difícil de aprender y mantener. - Dependiente del hardware.
Nivel Medio	- Equilibrio entre control y abstracción. - Relativamente eficiente	- Menos fácil de usar que los de alto nivel. - Requiere conocimiento sobre gestión de recursos.
Alto nivel	- Fácil de aprender y usar - Independiente del hardware.	- Menos control sobre el hardware. - Puede ser menos eficiente en la ejecución.

Lenguaje compilado

Es el proceso mediante el cual un **código fuente** escrito en un lenguaje de programación se *traduce* a un **lenguaje de máquina** (código objeto) que puede ser entendido y ejecutado directamente por un ordenador. Este proceso es llevado a cabo por un programa especializado conocido como **compilador**. Considera:

- Preprocesado (recolección)
- Verificado léxico/sintáctico/semántico
- Optimización
  +
- Enlazador (linker)



El enlazador combina los archivos creados, incluyendo la conexión con las **librerías necesarias** para que el programa funcione.

Lenguaje interpretado

Un **intérprete** es un programa que ejecuta instrucciones escritas en un lenguaje de programación de alto nivel de forma directa, sin necesidad de un proceso previo de compilación. A diferencia de un compilador, que traduce todo el código fuente a un archivo ejecutable antes de la ejecución, un intérprete traduce y ejecuta el código línea por línea en tiempo real.

Ejemplo: 

Permite trabajar con tipos de datos dinámicos y realizar depuración en tiempo real.

Ejemplos: *Hola mundo* en C y Python

```c
#include <stdlib.h>
#include <stdio.h>

int main( ) {

  printf("Hola mundo");
  return EXIT_SUCCESS;

}
```

Requiere ser compilado (GCC) y luego ejecutado.

```python
#!/usr/bin/env python
print("Hola mundo")
```

Puede ser ejecutado directamente con el intérprete.

Tema para profundizar

Investigue sobre la historia y características de **C y C++**, buscando responder las siguientes preguntas:

- ¿En qué se diferencian C y C++?
- ¿Cuál antecede al otro?
- ¿Tienen el mismo paradigma / nivel de abstracción?
- ¿Tiene alguna (des)ventaja uno por sobre el otro?