PROGRAMACION II

UNIDAD I

PARADIGMAS Y LENGUAJES DE PROGRAMACION

Lenguaje de programación

Cualquier lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador

Características:

- Define un proceso que se ejecuta en un computador
- Es de alto nivel, cercano a los problemas que se quieren resolver (abstracción)
- Permite construir nuevas abstracciones que se adapten al dominio que se programa

Elementos:

- Expresiones primitivas que representan las entidades más simples del lenguaje
- Mecanismos de combinación con los que se construyen elementos compuestos a partir de elementos más simples
- Mecanismos de abstracción con los que dar nombre a los elementos compuestos y manipularlos como unidades

¿Qué es un Paradigma?

Proviene del griego *paradeigma*, que significa “ejemplo” o “modelo”.

Conjunto de creencias que asumimos, lo que sirve como base o filtro para nuestra percepción e interpretación de la realidad.

¿Qué es Programación?

Es la acción de escribir un conjunto de instrucciones escritas en un determinado lenguaje mediante una computadora para la ejecución de una serie de operaciones, con el objetivo de resolver un problema que se ha definido previamente.

¿Qué es un Paradigma de Programación?

Provee la visión y métodos de un programador en la construcción de un programa. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de problemas.

Los lenguajes de programación son basados en uno o más paradigmas.

Paradigmas de programación

Define un conjunto de reglas, patrones y estilos de programación que son usados por un grupo de lenguajes de programación

Clasificación:

- **Paradigma funcional**
- **Paradigma lógico**
- **Paradigma imperativo o procedural**
- **Paradigma orientado a objetos**

Tipos de paradigmas de programación:

1. Paradigma Imperativo.
2. Paradigma Declarativo.
   - Paradigma Funcional.
   - Paradigma Lógico.
3. Paradigma Estructurado.
4. Paradigma Orientado a Objetos.

2. Paradigma Declarativo.

Las sentencias describen al problema que se quiere solucionar, pero no las instrucciones necesarias para solucionarlo. Esto último se realizará mediante mecanismos internos de inferencia de información a partir de la descripción realizada.

El paradigma declarativo es un término que agrupa los siguientes paradigmas de programación:

- **Paradigma funcional**. Todo se resuelve por medio de la evaluación de funciones matemáticas.

```
predecesor(x)= x-1, si x>0
sucesor(x)= x+1
suma(x,0)= x
suma(x, y)= sucesor(suma(x, predecesor(y)))

?- suma(3,2)
```

- **Paradigma lógico**. Consiste en declarar una serie de hechos (elementos, objetos, etc.) y reglas (relación entre objetos que cumplen unas propiedades), que servirán para generar un resultado.

Paradigma funcional

Se basa en el concepto de función que describe una relación entre una entrada y una salida.

El concepto de estado o variable no existe.

Características:

- La computación se realiza mediante la evaluación de expresiones
- Definición de funciones
- Funciones como datos primitivos
- Valores sin efectos laterales, no existe la asignación
- Programación declarativa

Lenguajes: LISP, Scheme, Haskell, Scala

Ejemplo en Scheme:

```
(define (doble x)
   (* x 2))

(define (cuadrado x)
   (* x x))

(define (incremento x)
   (+ x 1))

(doble 4)
8
```

Paradigma Lógico

Los problemas se modelan utilizando la lógica formal llegando a una conclusión por medio de hechos y reglas

Se Aplican reglas de la lógica para inferir conclusiones a partir de datos

Características:

- **Definición de reglas**
- **Unificación de términos**
- **Programación declarativa**
- **Mecanismos de inferencia automática**
- **Visión lógica de la computación**

Lenguajes: Prolog, Mercury

Paradigma Lógico: ejemplo en Prolog

```
padrede('juan', 'maria)
padrede('pablo', 'juan')
padrede('pablo', 'marcela')
padrede('carlos', 'debora')
hijode(A,B) :- padrede(B,A)
abuelode(A,B) :- padrede(A,C), padrede(C,B)
hermanode(A,B) :- padrede(C,A) , padrede(C,B), A \== B
?- hermanode('juan', 'marcela').
yes
?- hermanode('carlos', 'juan')
No
?- abuelode('pablo', 'maria')
yes
?- abuelode('maria', 'pablo')
No
```

•Paradigma lógico.

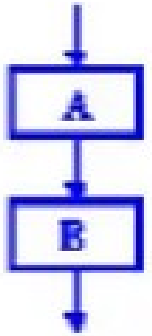
```
Mujer(Rosa)
Mujer(Marta)
Mujer(Laura)
Padres(Rosa, Carlos, Pilar)
Padres(Marta, Carlos, Pilar)
Padres(Laura, Carlos, Pilar)
Hermanas(X, Y):- mujer(X), mujer(Y), padres(X, P, M), padres(Y, P, M)

?- hermanas(Rosa, Marta)
?- hermanas(Rosa, X)
```

3. Paradigma Estructurado.

Es una técnica para escribir programas de una manera más fácil, siguiendo tres estructuras:

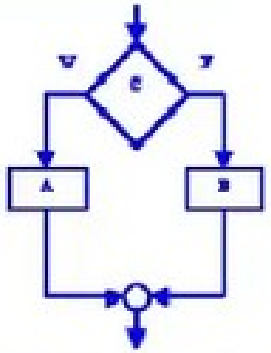
3. Paradigma Estructurado.

- Estructura Secuencial




```
INPUT x
INPUT y
auxiliar= x
x= y
y= auxiliar
PRINT x
PRINT y
```

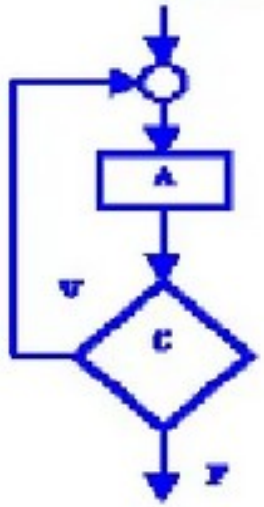
- Estructura Selectiva




```
IF a > b THEN
   PRINT a ; " es mayor que " ; b
ELSE
   PRINT a ; " no es mayor que " ; b
END IF
```

3. Paradigma Estructurado.

•Estructura Repetitiva (ó Iterativa)




```
a= 0
b= 7
DO WHILE b > a
   PRINT a
   a= a + 1
LOOP
```

Se basa un *Teorema del programa estructurado*, el cual afirma que cualquier programa, puede ser elaborado utilizando únicamente las tres estructuras básicas.

1. Paradigma Imperativo o por procedimientos.

Describe la programación como una secuencia instrucciones o comandos que ejecutan una tras otra y además cambian el estado de un programa.

```
leer(x)
leer(y)
resultado = x + y
escribir(resultado)
```

Paradigma imperativo

La computación se realiza cambiando el estado del programa por medio de sentencias que definen pasos de ejecución

Características:

- Definición de procedimientos
- Definición de tipos de datos
- Chequeo de tipos en tiempo de compilación
- Cambio de estado de variables
- Pasos de ejecución de un proceso

Lenguajes Fortran, C, Basic, js

Ejemplo Js:

```
function fibonacci(n) {
  var actual, ant1, ant2;
  if (n === 0) {
    actual = 0;
  } else if (n === 1) {
    actual = 1;
  } else {
    ant1 = ant2 = 1;
    for (i = 2; i < n; i++) {
      actual = ant1 + ant2;
      ant2 = ant1;
      ant1 = actual;
    }
  }
  return actual;
}
```

4. Paradigma Orientado a Objetos.

Es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos.

Los programas están compuestos por objetos que tienen propiedades y pueden llevar acabo ciertas operaciones.

Surge de la evolución de la programación estructurada y trata de amoldarse al modo de pensar del hombre y no al de la máquina.

Una de las ideas fundamentales del paradigma de programación orientada a objetos es el concepto de objeto como una entidad que engloba datos y acciones.

¿Qué es un Objeto?

Todo lo que vemos a nuestro alrededor puede ser considerado un objeto (una computadora, un teléfono celular, un árbol, un automóvil, etc).

Un objeto es una unidad que contiene datos y operaciones que operan sobre esos datos.

Todo objeto del mundo real tiene dos partes características y comportamiento.

La Programación orientada a objetos trabaja de esta manera: todo el programa está construido en base a diferentes componentes (objetos), cada uno tiene un rol específico en el programa y todos los componentes pueden comunicarse entre ellos de formas predefinidas.

Todo objeto del mundo real tiene dos partes características y comportamiento.

Paradigma Orientado a Objetos

La POO se basa en la idea natural de un mundo lleno de objetos y la resolución de problemas se realiza mediante el modelado de esos objetos.

Características:

- **Definición de clases y herencia**
- **Objetos como abstracción de datos y procedimientos**
- **Polimorfismo y chequeo de tipos en tiempo de ejecución**

Lenguajes: C++, Java, Php

Paradigma OO: Ejemplo en Java

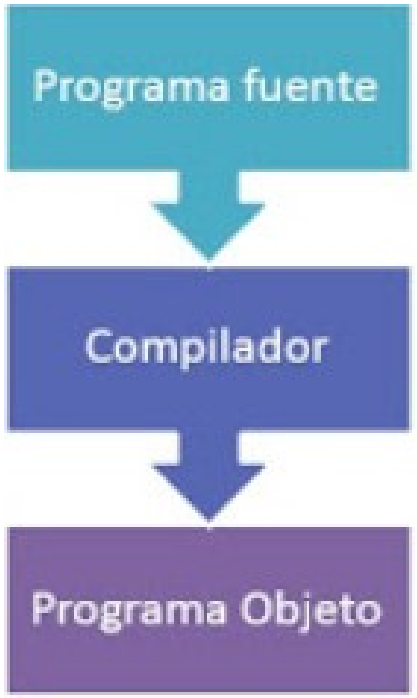
```
public class Bicicleta {
   public int marcha;
   public int velocidad;
   public Bicicleta(int velocidadInicial, int marchaInicial) {
   marcha = marchaInicial;
   velocidad = velocidadInicial;
   }
   public void setMarcha(int nuevoValor) { marcha = nuevoValor;}
   public void frenar(int decremento) {velocidad -= decremento;}
   public void acelerar(int incremento) {velocidad += incremento;}
 }
public class MountainBike extends Bicicleta {
  public int alturaSillin;
  public MountainBike(int alturaInicial, int velocidadInicial, int marchaInicial) {
  super(velocidadInicial, marchaInicial); alturaSillin = alturaInicial;
  }
 public void setAltura(int nuevoValor) { alturaSillin = nuevoValor;}
 }
```

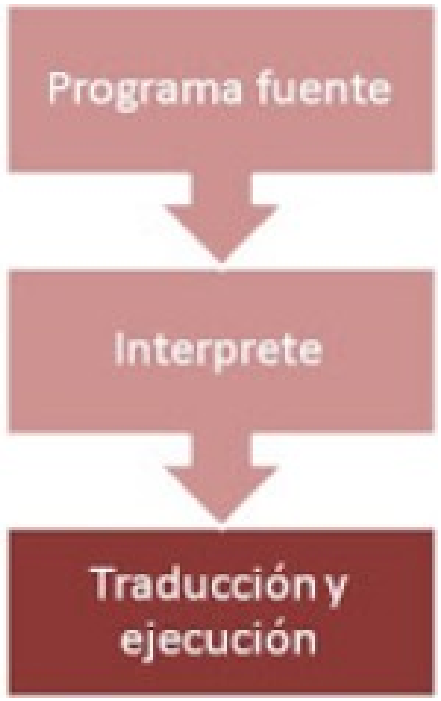
Un objeto tiene tres características:

- Comportamiento: ¿Qué se puede hacer con ese objeto?
- Estado: ¿Cómo reacciona el objeto?
- Identidad: ¿Cómo se distingue un objeto de otro que tiene el mismo comportamiento y estado?

COMPILADORES E INTERPRETES

- **INTERPRETE**
  - **Dado un Programa Fuente, en un lenguaje de alto nivel, realiza la traducción y ejecución instrucción a instrucción.**
- **COMPILADOR**
  - **Dado un Programa Fuente, en un lenguaje de alto nivel realiza la traducción completa generando un Programa Objeto**





EN AMBOS CASOS SE DEBE LEER UNA ENTRADA Y ENTENDERLA

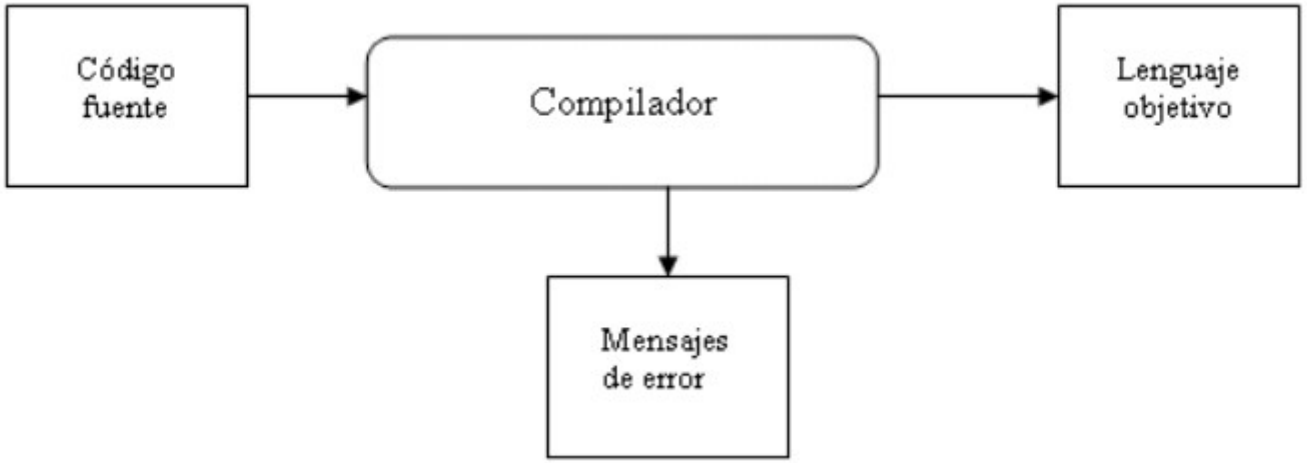
INTERPRETES

- Se realiza la traducción y ejecución de forma simultanea
- Se lee el código fuente y se va ejecutando al mismo tiempo

Lenguajes interpretados: Ruby, Python, PHP, Perl

COMPILADORES

- Se lee el código fuente y se traduce o convierte a otro lenguaje.
- Se muestran los errores existentes en el código fuente.



Lenguajes Compilados : C, C++, Fortran, Cobol

COMPILADORES: ETAPAS

- **Edición.** Esta fase consiste en escribir el programa empleando algún lenguaje y un editor.
  - Como resultado se obtiene el código fuente.
- **Compilación.** En esta fase se traduce el código fuente a código máquina.
  - Si no se produce ningún error se obtiene el código objeto.
  - En caso de errores la traducción no se realiza y se informan esos errores.
- **Linkeo.** Esta fase consiste en unir el código objeto a librerías del lenguaje
  - Como resultado se obtiene el programa ejecutable.
    Existen dos tipos de linkeo:
    - **Linkeo estático**: El código objeto de las librerías se añaden al código objeto del programa generando el archivo ejecutable.
    - **Linkeo dinámico**: El código objeto de las librerías no se añade al código objeto del programa, se cargan y se ligan en el momento en que se necesiten.