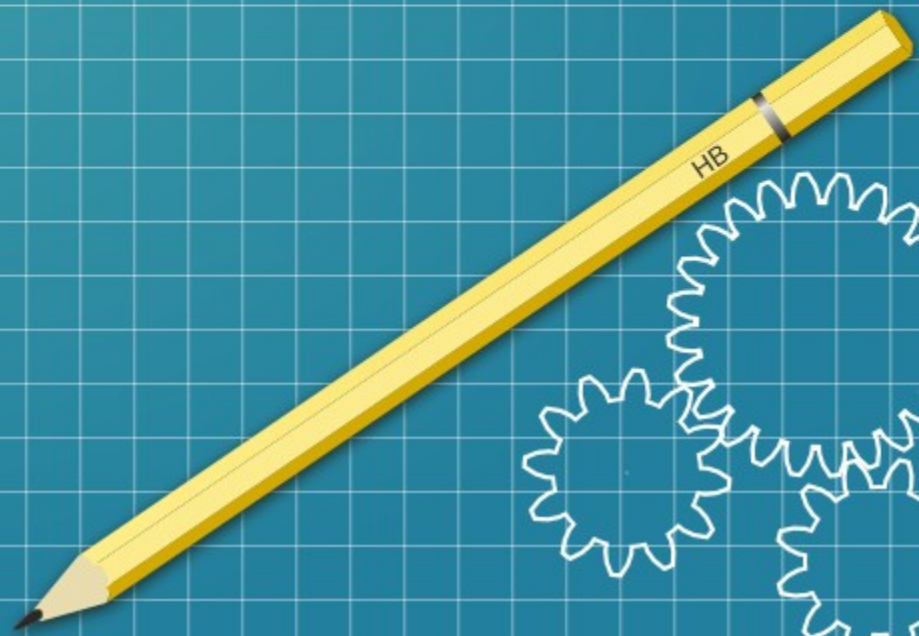
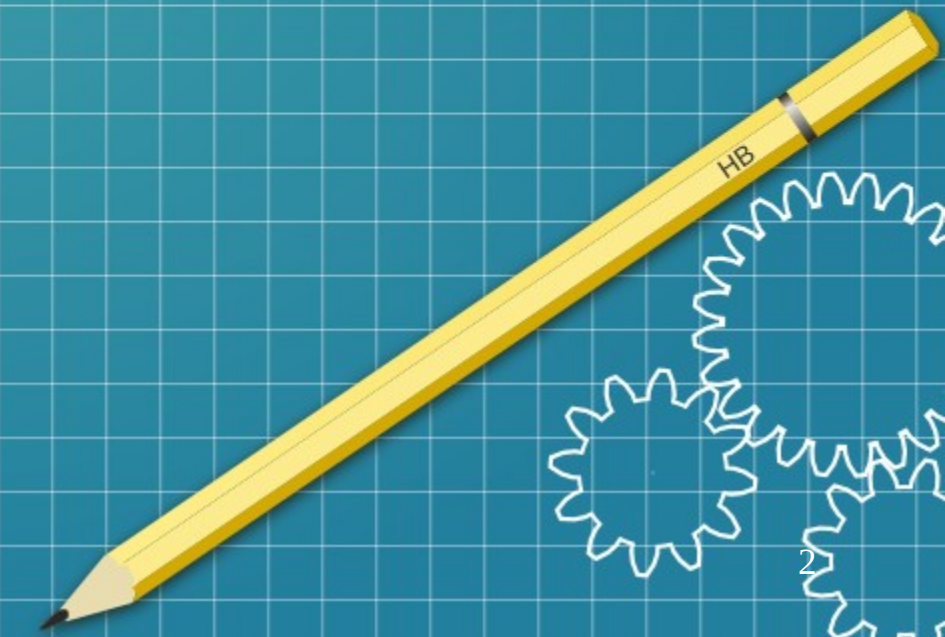
Resolución de problemas en programación

UD- 0



Paradigmas de programación

- Principales enfoques aplicados a la hora de diseñar lenguajes y diseñar programas.

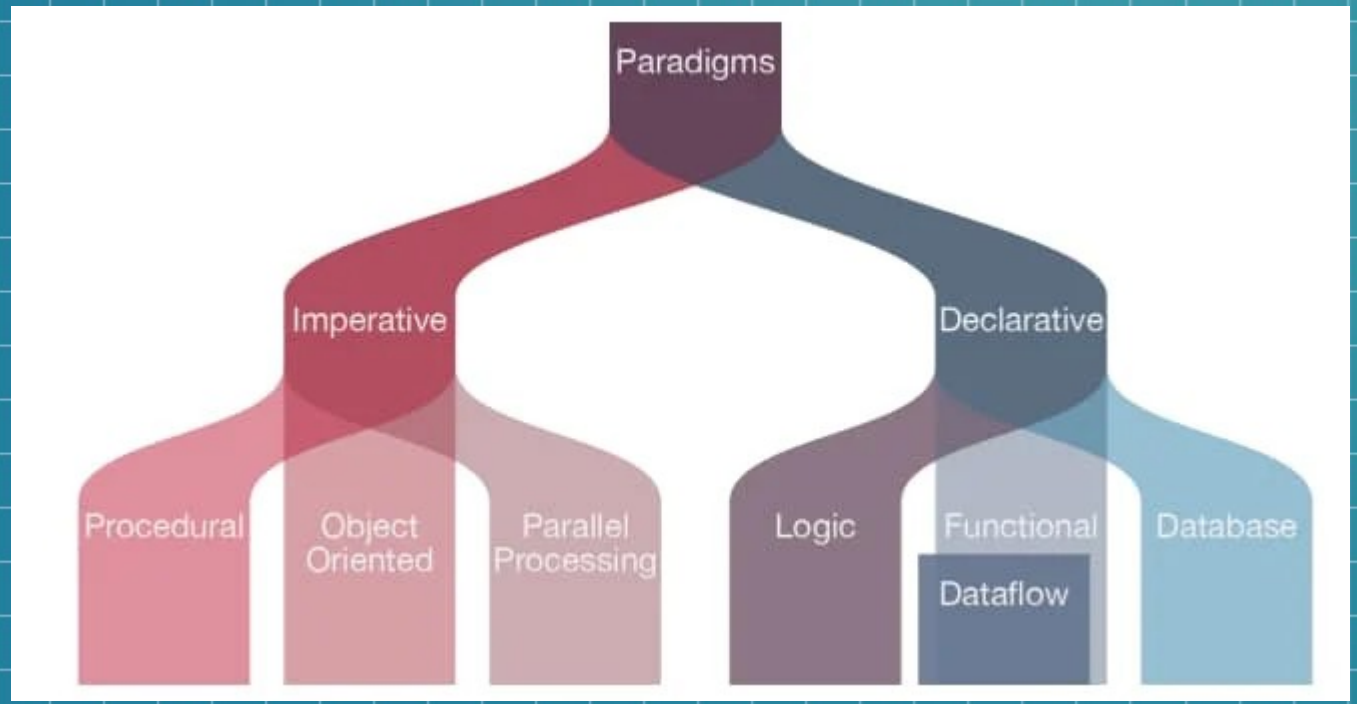


Paradigmas de programación (II)

- Programación imperativa
- Programación funcional
- Programación lógica
- Programación Orientada a Objetos

https://homes.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html

Paradigmas de programación (III)



https://keepcoding.io/blog/paradigmas-de-programacion/

Paradigmas de programación (II)

- Programación imperativa
  - Lenguaje C
- Programación Orientada a Objetos
  - Java

https://homes.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html

Paradigmas de programación (II)

- Programación funcional:
  - Haskell:

haskell Copy code

```haskell
-- Definición de una función recursiva para sumar los elementos de una
sumaLista :: [Int] -> Int
sumaLista [] = 0
sumaLista (x:xs) = x + sumaLista xs

-- Ejemplo de uso de la función con una lista de números
ejemploLista = [1, 2, 3, 4, 5]
resultado = sumaLista ejemploLista
```

Paradigmas de programación (II)

- Programación funcional:
  - Python funcional:

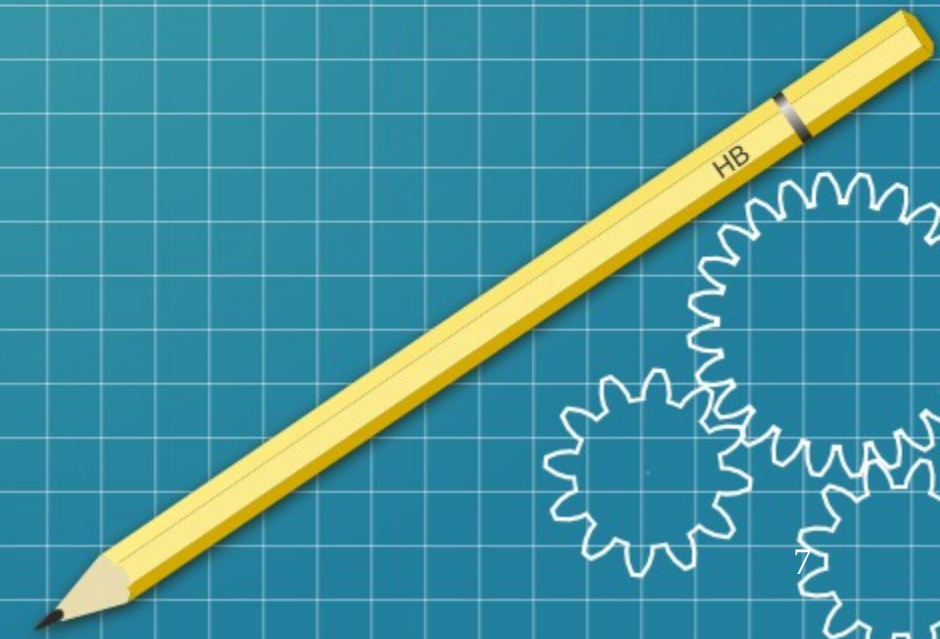
python Copy code

```python
# Definición de una función para duplicar un número
def duplicar(x):
    return x * 2

# Lista de números
numeros = [1, 2, 3, 4, 5]

# Aplicar la función duplicar a cada elemento de la lista usando map
numeros_duplicados = list(map(duplicar, numeros))

print(numeros_duplicados)  # Imprimir la lista con los números duplicad
```



Paradigmas de programación (II)

- Programación funcional:
  - Python funcional:
    - En este ejemplo, la función duplicar simplemente multiplica por 2 el valor que recibe como argumento. Luego, la función map se utiliza para aplicar la función duplicar a cada elemento de la lista numeros, generando una nueva lista llamada numeros_duplicados que contiene cada número duplicado.

    - El uso de map, filter, reduce y la capacidad de trabajar con funciones como objetos de primera clase en Python son ejemplos de cómo el lenguaje permite un enfoque más funcional en la escritura de código, aunque el paradigma predominante en Python sea el imperativo.

Resolución general de problemas

- Análisis
- Diseño
- Programación / Codificación
- Pruebas
- Implantación
- Mantenimiento
- Finalización

https://programacionpro.uno/tecnicas-de-resolucion-de-problemas-en-programacion/

Enfoques de programación

- Divide y vencerás: Divide un problema en subproblemas más pequeños y manejables, resuelve cada uno por separado y luego combina las soluciones para obtener la respuesta final. (Arriba - Abajo)

- Programación dinámica: Divide un problema en subproblemas, pero a diferencia de "divide y vencerás", resuelve cada subproblema solo una vez y guarda su solución para no tener que resolverlo nuevamente. (Abajo – Arriba).

- Algoritmos ávidos (greedy): Toma la mejor decisión en cada paso localmente con la esperanza de llegar a una solución óptima global. No siempre garantiza la mejor solución, pero a menudo es rápido y fácil de implementar.

- Búsqueda y exploración: Utiliza algoritmos de búsqueda como búsqueda en profundidad (DFS) o búsqueda en anchura (BFS) para encontrar soluciones. Son útiles para problemas de grafos y árboles.

- Fuerza bruta: Prueba todas las posibles soluciones para un problema, aunque sea poco eficiente. A veces es útil para problemas pequeños o cuando no hay un enfoque más óptimo.

- Algoritmos heurísticos: Encuentra soluciones aproximadas para problemas difíciles en un tiempo razonable, aunque no garantizan la solución óptima.

Enfoques de programación

- Divide y vencerás:

python Copy code

```python
def fibonacci(n):
    # Casos base
    if n <= 1:
        return n

    # Recursión para calcular el número de Fibonacci
    return fibonacci(n - 1) + fibonacci(n - 2)

# Probamos la función con n = 5
resultado = fibonacci(5)
print("El número de Fibonacci en la posición 5 es:", resultado)
```

https://platzi.com/tutoriales/2218-pensamiento-logico-2020/9581-enfoque-descendente-para-la-resolucion-problemas-escritura-de-algoritmos/

Enfoques de programación

- **Programación dinámica:**

```python
def fibonacci(n):
    # Creamos un arreglo para almacenar los resultados intermedios
    fib = [0, 1]

    # Calculamos los números de Fibonacci desde el tercero hasta el n-é
    for i in range(2, n + 1):
        fib.append(fib[i - 1] + fib[i - 2])

    # El resultado estará en la posición n del arreglo
    return fib[n]

# Probamos la función con n = 5
resultado = fibonacci(5)
print("El número de Fibonacci en la posición 5 es:", resultado)
```

Propuesta para resolver problemas programación

- Diseñar un ejemplo a mano
- Describir los pasos de lo que se hizo
- Encontrar PATRONES se repitan
- Comprobarlo a mano
- Traducirlo a código
- Diseñar y ejecutar test unitarios
- Depurar test unitarios.

https://sumn2u.medium.com/a-seven-step-approach-to-solving-programming-problems-59c4687d6c85