

# **Pseudocódigo y diagramas de flujo**

**Fundamentos de Computación**

**Diego Caro**

**2021-1**

# Resultados de aprendizaje

- Identificar operaciones y estructuras de control en pseudocódigo
- Describir algoritmos cotidianos en pseudocódigo
- Identificar el nivel de detalle mínimo para poder escribir un buen pseudocódigo

**¿Sabes hacer panqueques?**

## Panqueques con manjar



## Ingredientes

- 1/2 litro de leche
- 1 huevo
- 1 taza de harina sin polvos de hornear
- 3 cucharadas soperas de aceite
- 1 pizca de sal

## Preparación

1. Mezcla en un bowl la harina, la sal y los huevos.
2. Añade la leche poco a poco, hasta lograr la consistencia deseada. La mezcla debe quedar ni muy líquida ni muy dura.
3. Calienta una sartén pequeña de teflón con una pizca de aceite, y a continuación vierte una pequeña mezcla a la sartén.
4. Da vuelta varias veces hasta que se dore el panqueque.
5. Si se deseas, puedes llenar con manjar.

**Imagina que la receta es para alguien que  
solo ha cocinado huevos  
¿Cómo reescribirías la receta?**

[https://jamboard.google.com/d/1nMzx-rYJ2\\_OPB-MsAFOoMsvOI1dYkZLFChp\\_9Yp-l1k/edit?usp=sharing](https://jamboard.google.com/d/1nMzx-rYJ2_OPB-MsAFOoMsvOI1dYkZLFChp_9Yp-l1k/edit?usp=sharing)

## Panqueques con manjar



## Ingredientes

- 1/2 litro de leche
- 1 huevo
- 1 taza de harina sin polvos de hornear
- 3 cucharadas soperas de aceite
- 1 pizca de sal

## Preparación

1. Mezcla en un bowl la harina, la sal y los huevos.
2. Añade la leche poco a poco, hasta lograr la consistencia deseada. La mezcla debe quedar ni muy líquida ni muy dura.
3. Calienta una sartén pequeña de teflón con una pizca de aceite, y a continuación vierte una pequeña mezcla a la sartén.
4. Da vuelta varias veces hasta que se dore el panqueque.
5. Si se deseas, puedes rellenar con manjar.

# ¿Qué es pseudocódigo?

**Algoritmo:** secuencia de pasos finita para resolver un problema o realizar alguna computación.

- **Pseudocódigo:** descripción formal de los pasos de un **algoritmo**.
  - **Objetivo:** ser leído por personas (en vez de computadores)
  - ¿Por qué usar pseudocódigo en vez de código Python/Java/C/C++ (etc...)?
    - Balancear lo comprensible e informal del lenguaje natural (español o inglés) con la precisión del código.
    - Si la descripción está en español, podría estar en demasiado alto nivel, y entonces podría ser imposible de programar.
    - El objetivo es proveer una descripción de alto nivel para facilitar análisis y eventualmente programarlo, evitando detalles de implementación.

# Ejemplo 1: promedio notas

- Escriba un algoritmo que imprima el promedio final de un estudiante. La nota final es calculada como el promedio final de cuatro notas.
- Idea (estrategia de solución)
  - Preguntar por las cuatro notas
  - Calcular el promedio sumando las notas y dividendo por 4
  - Imprimir el promedio

Pseudocódigo:

1. **read** C1, C2, C3, C4
2.  $P = (C1 + C2 + C3 + C4)/4$
3. **print** P

Usamos notación matemática

Pasos bien definidos

# Ejemplo 2: producto punto

- Calcule e imprima el producto punto entre los vectores A y B en  $\mathbb{R}^3$
- Idea:
  - Leer las 3 componentes del vector A y del vector B
  - Calcular producto punto como la suma de los productos de cada componente.
  - Imprimir el producto punto

Pseudocódigo:

1.  $a = [a_1, a_2, a_3]$  vector en  $\mathbb{R}^3$
2.  $b = [b_1, b_2, b_3]$  vector en  $\mathbb{R}^3$
3.  $p = a_1b_1 + a_2b_2 + a_3b_3$
4. **print p**

Usamos notación matemática

Pasos bien definidos

# Ejemplo 3: producto punto en $\mathbb{R}^N$

- Calcule e imprima el producto punto entre los vectores A y B en  $\mathbb{R}^N$
- Idea:
  - Leer las  $N$  componentes del vector A y del vector B
  - Calcular producto punto como la suma de los productos de cada componente.
  - Imprimir el producto punto

1.  $a = [a_1, a_2, \dots, a_N]$  vector en  $\mathbb{R}^N$
2.  $b = [b_1, b_2, \dots, b_N]$  vector en  $\mathbb{R}^N$
3.  $p = \sum_{i=1}^N a_i b_i$
4. **print p**

Usamos notación matemática

Pasos bien definidos

# Condicionales

- Condicionales: ejecutar un conjunto de instrucciones solo si se cumple alguna condición.
- Ejemplo: Escriba un algoritmo que determine el promedio final de un estudiante. **Indique si le estudiante aprueba o no el curso.** La nota final es calculada como el promedio final de cuatro notas.

## Idea (estrategia de solución)

- Preguntar por las cuatro notas
- Calcular el promedio sumando las notas y dividendo por 4
- Si el promedio es menor que 4
  - Imprimir “Reprueba”
- En caso contrario
  - Imprimir “Aprueba”

Caso nota menor que 4

Caso nota mayor o igual que 4

## Pseudocódigo:

1. **read** C1, C2, C3, C4
2.  $P = (C1 + C2 + C3 + C4)/4$
3. **if**  $P < 4$ :
4.     **print** “Reprueba”  
Instrucciones que se ejecutan
5. **else**:
6.     **print** “Aprueba”  
Instrucciones que se ejecutan

# Ejemplo: mínimo

- Calcular el mínimo valor entre dos números

## Idea (estrategia de solución)

- Preguntar por los dos números
- Si el primer número es menor que el segundo,
  - El mínimo es el primer número
- En caso contrario
  - El mínimo es el segundo número
- Imprimir numero menor

1. **read A, B**
2. **if A < B:**
3.     **min = A**
4. **else:**
5.     **min = B**
6. **print min**

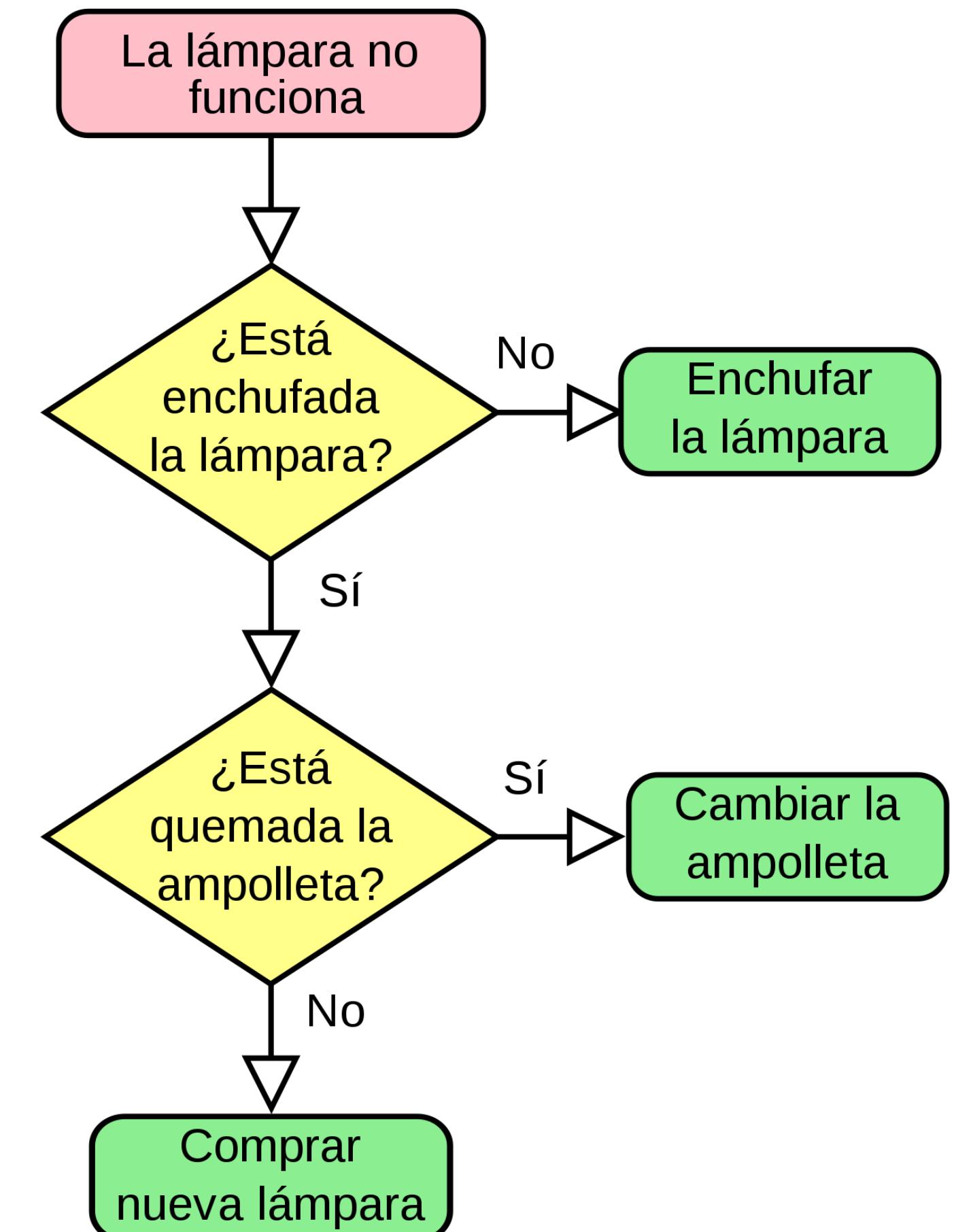
Instrucciones que se ejecutan solo si  $A < B$  es verdadero

Instrucciones que se ejecutan en caso contrario

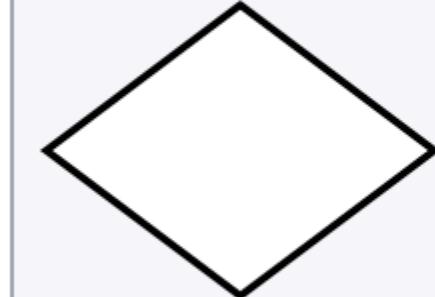
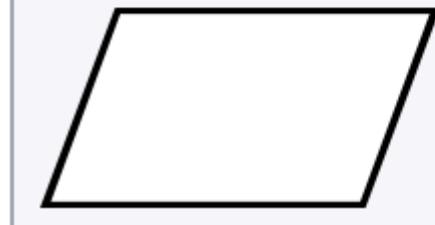
# Diagramas de flujo

- Representación gráfica de un algoritmo.
  - (o de pseudocódigo)

```
1. if no está enchufada la lámpara  
2.   print "Enchufar lámpara"  
3. if está quemada la ampolleta  
4.   print "Cambiar ampolleta"  
5. else  
6.   print "Comprar lámpara nueva"
```



# Estándar ISO 5807:1985

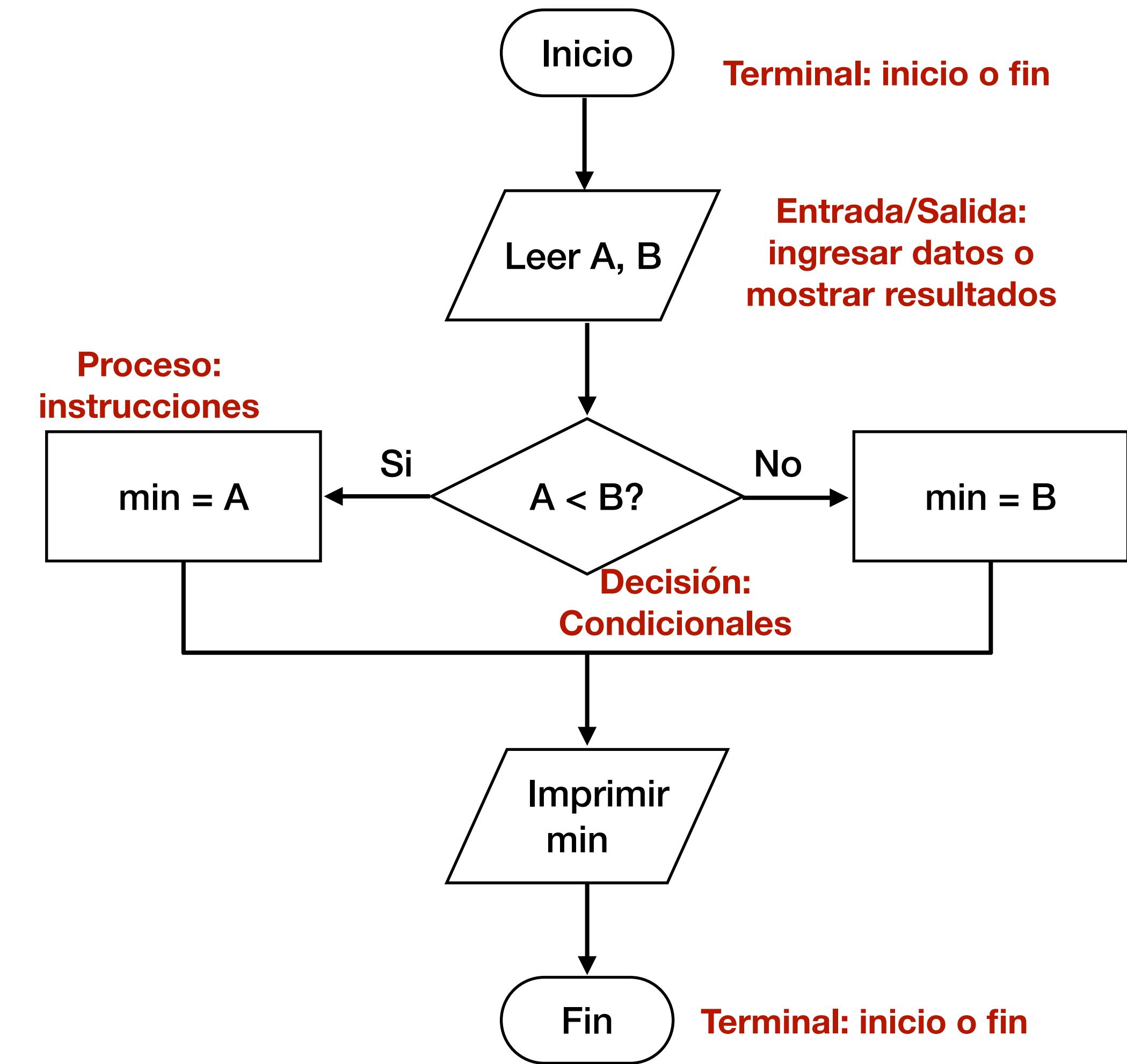
Forma ANSI/ISO	Nombre	Descripción
	Línea de flujo (Flecha) <sup>4</sup>	Muestra el orden de operación de los procesos. Una línea saliendo de un símbolo y apuntando a otro. <sup>3</sup> Las fechas se agregan si el flujo no es el estándar de arriba hacia abajo, de izquierda a derecha. <sup>4</sup>
	Terminal <sup>3</sup>	Indica el inicio o fin de un programa o subprocessos. Se representa como un stadium, <sup>3</sup> ovalo o rectángulo redondeado. Usualmente contienen la palabra "Inicio" o "Fin", o alguna otra frase señalando el inicio o fin de un proceso, como "presentar consulta" o "recibir producto".
	Proceso <sup>4</sup>	Representa un conjunto de operaciones que cambiar el valor, forma o ubicación de datos. Representado como un rectángulo. <sup>4</sup>
	Decisión <sup>4</sup>	Muestra una operación condicional que determina cuál de los dos caminos tomará el programa. <sup>3</sup> La operación es comúnmente una pregunta de sí/no o una prueba de verdadero/falso. Representada como un rombo.(rombo). <sup>4</sup>
	Entrada/Salida <sup>4</sup>	Indica el proceso de hacer entrar o salir datos, <sup>4</sup> en la forma de ingresar datos o mostrar resultados. Representado como un paralelogramo. <sup>3</sup>

# Ejemplo: mínimo con diagrama de flujo

- Calcular el mínimo entre dos números

Pseudocódigo

```
1. read A, B  
2. if A < B:  
3.   min = A  
4. else:  
5.   min = B  
6. print min
```



# Ejemplo: imprimir la tabla del 2

Idea 1 (estrategia de solución)

- **print** el resultado de  $2 \times 1$
- **print** el resultado de  $2 \times 2$
- **print** el resultado de  $2 \times 3$
- **print** el resultado de  $2 \times 4$
- **print** el resultado de  $2 \times 5$
- **print** el resultado de  $2 \times 6$
- **print** el resultado de  $2 \times 7$
- **print** el resultado de  $2 \times 8$
- **print** el resultado de  $2 \times 9$
- **print** el resultado de  $2 \times 10$

**¡Demasiado repetitivo!**



¿Qué pasaría si me piden la tabla del 3?  
¿Del 4? ¿Del 5?

# Repetición: Ciclo Mientras

- Repetición: Ciclo **while** (mientras)
  - repetir instrucciones mientras una condición se cumpla

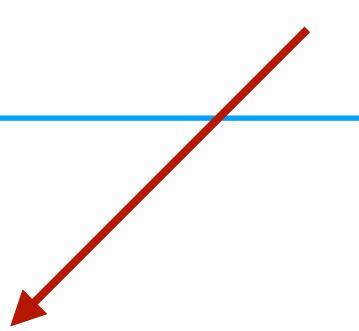
## Idea 2 (estrategia de solución)

- Crear numero con valor 1
- **Mientras** número sea menor o igual que 10
  - Imprimir 2 por el número
  - Sumar 1 a número
  - Repetir

## Pseudocódigo

```
1. n = 1
2. while  $n \leq 10$ :
3.   t = 2n
4.   print t
5.   n = n + 1
```

Condición que  
se debe cumplir



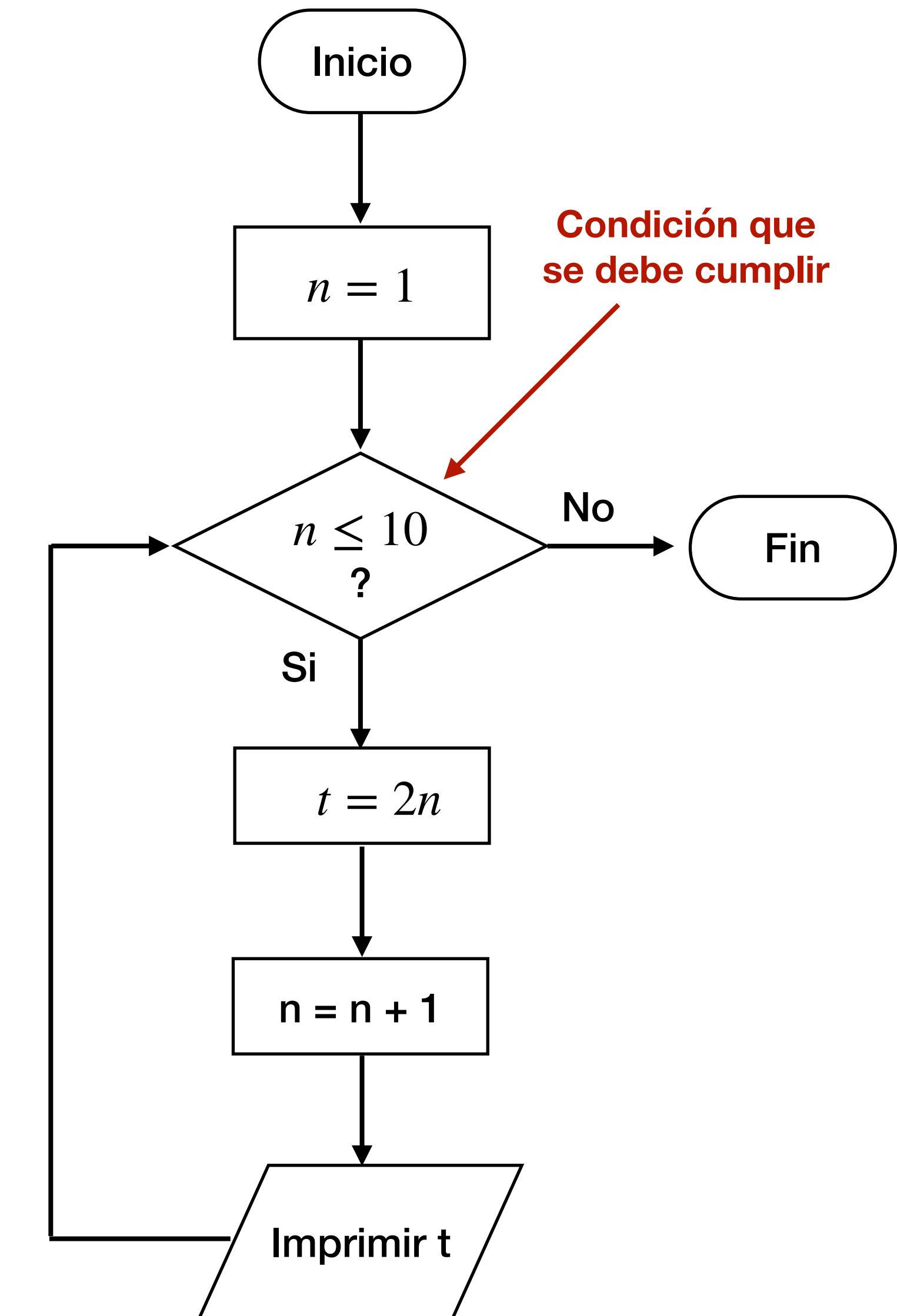
# Repetición: Ciclo Mientras

- Ciclo **while**: repetir instrucciones mientras una condición se cumpla

Pseudocódigo

```
1. n = 1  
2. while n ≤ 10:  
3.     t = 2n  
4.     print t  
5.     n = n + 1
```

Condición que se debe cumplir



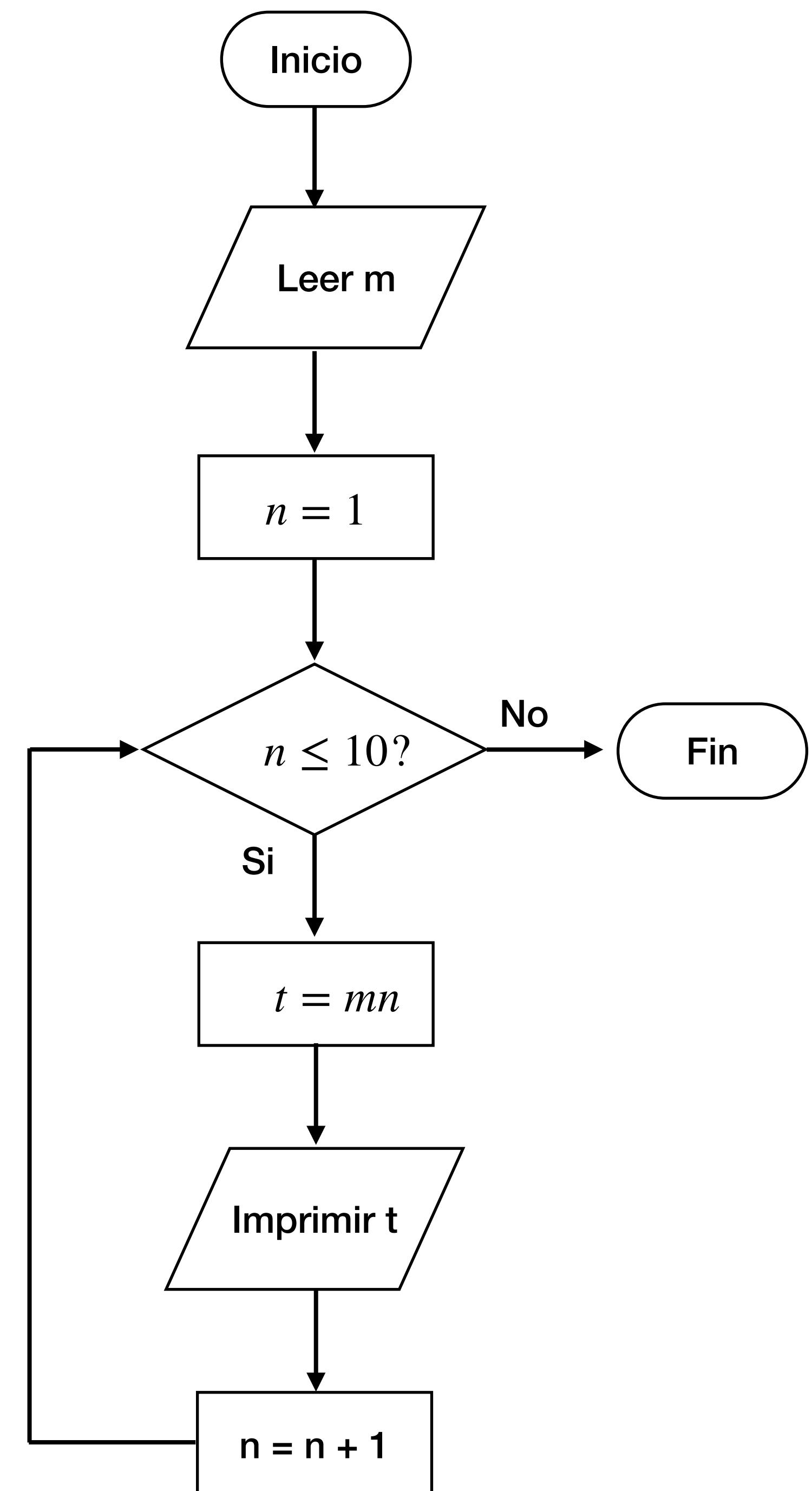
# Ejemplo: Imprimir tabla de un número dado

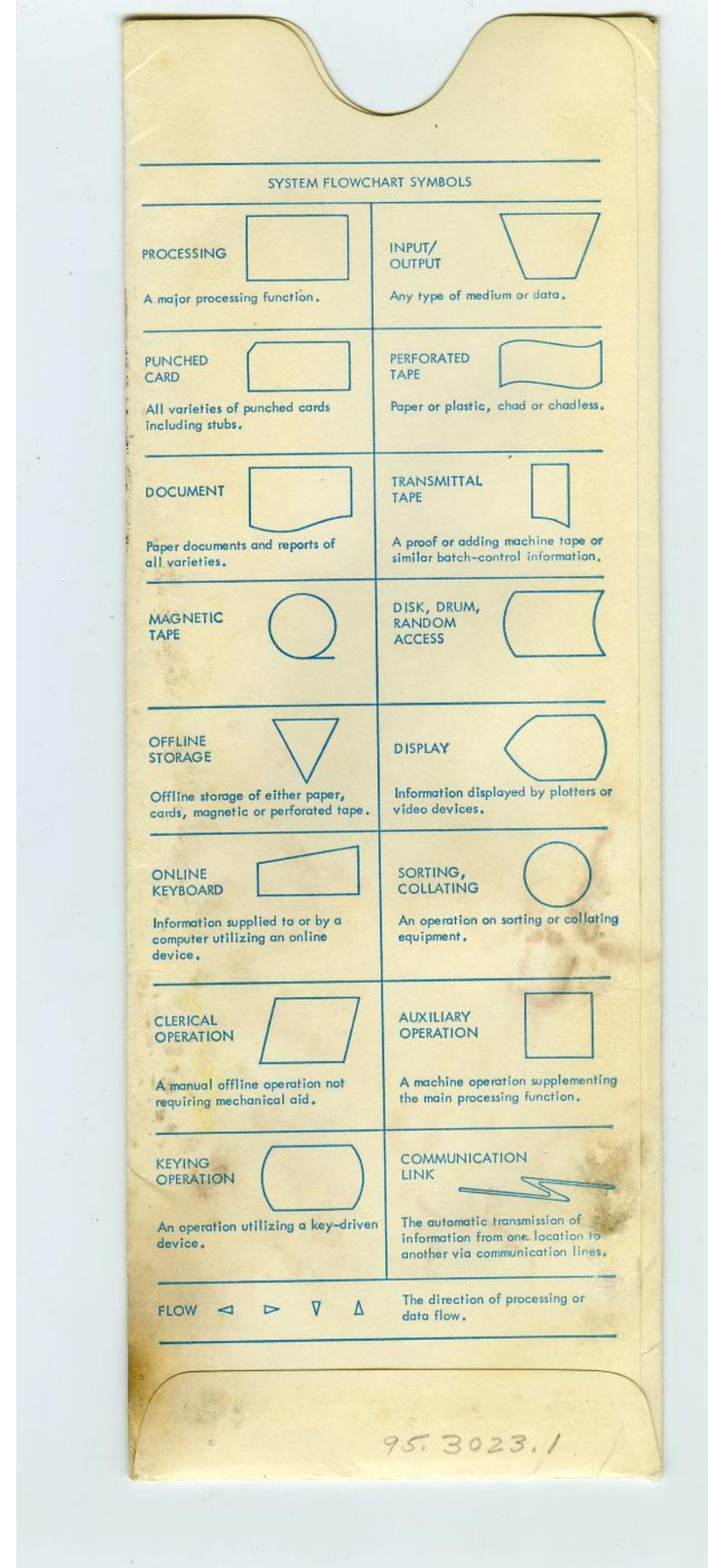
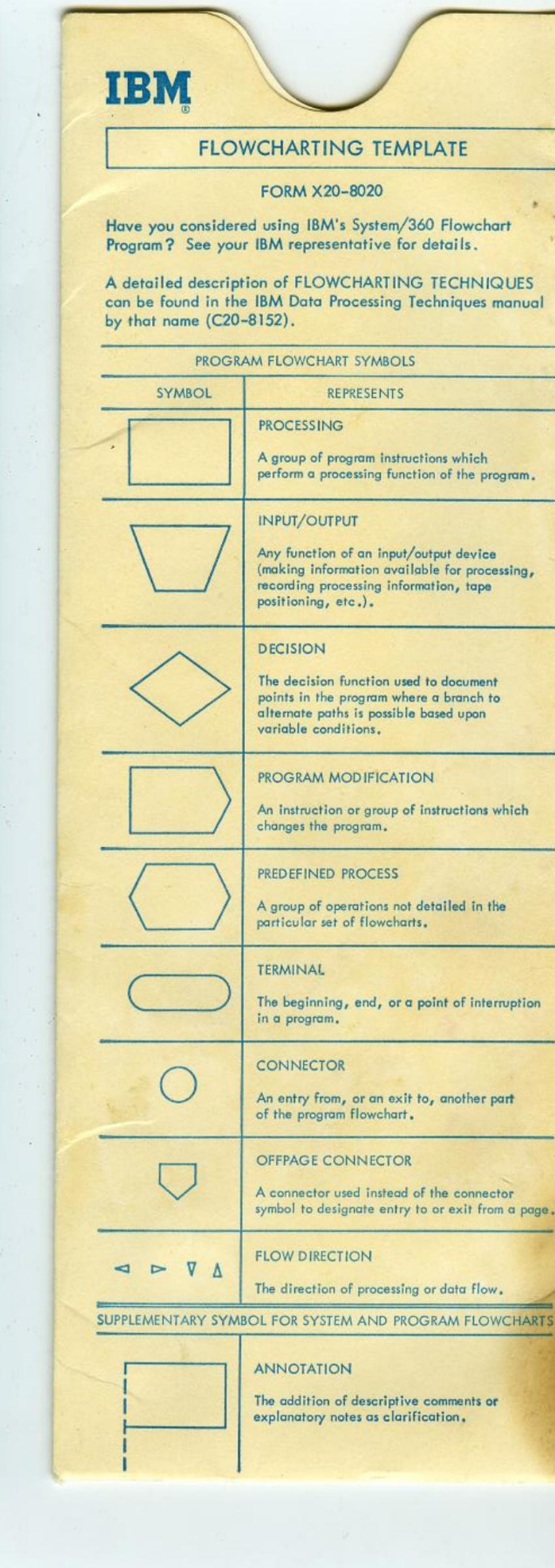
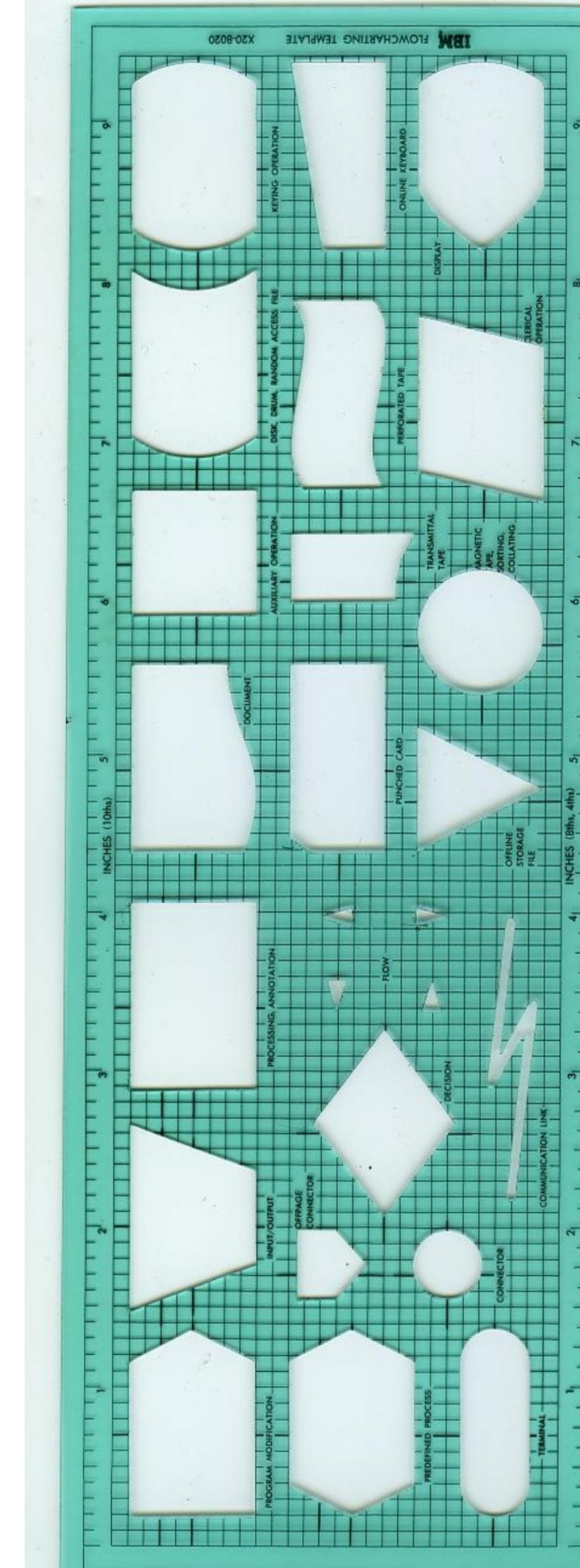
## Idea (estrategia de solución)

- Preguntar por número  $m$
- Crear numero con valor 1
- **Mientras** número sea menor o igual que 10
  - Imprimir  $m$  por el número
  - Sumar 1 a número
  - Repetir

## Pseudocódigo

```
1. read m  
2. n = 1  
3. while  $n \leq 10$ :  
4.     t = mn  
5.     Imprimir t  
6.     n = n + 1
```





# ¿Y realmente se usa el pseudocódigo?

SIAM J. COMPUT.  
Vol. 22, No. 5, pp. 935–948, October 1993

© 1993 Society for Industrial and Applied Mathematics  
003

Knowl Inf Syst  
DOI 10.1007/s10115-015-0908-6



REGULAR PAPER

## SUFFIX ARRAYS: A NEW METHOD FOR ON-LINE STRING SEARCHES\*

UDI MANBER<sup>†‡</sup> AND GENE MYERS<sup>†§</sup>

**Abstract.** A new and conceptually simple data structure, called a suffix array, for on-line string searches is introduced in this paper. Constructing and querying suffix arrays is reduced to a sort and search paradigm that employs novel algorithms. The main advantage of suffix arrays over suffix trees is that, in practice, they use three to five times less space. From a complexity standpoint, suffix arrays permit on-line string searches of the type, “Is

```
l ← lcp(APos[0], W)
r ← lcp(APos[N-1], W)
if l = P or wl ≤ aPos[0]+l then
    LW ← 0
else if r < P or wr ≤ aPos[N-1]+r then
    LW ← N
else
    { (L, R) ← (0, N - 1)
        while R - L > 1 do
            { M ← (L + R)/2
                if l ≥ r then
                    if Lcp[M] ≥ l then
                        m ← l + lcp(APos[M]+l, Wl)
                    else
                        m ← Lcp[M]
                else
                    if Rcp[M] ≥ r then
                        m ← r + lcp(APos[M]+r, Wr)
                    else
                        m ← Rcp[M]
                if m = P or wm ≤ aPos[M]+m then
                    (R, r) ← (M, m)
                else
                    (L, l) ← (M, m)
            }
        }
    }
    LW ← R
}
```

FIG. 3. An  $O(P + \log N)$  search for  $L_W$ .

<https://doi.org/10.1137%2F0222058>

[https://es.wikipedia.org/wiki/Arreglo\\_de\\_sufijos](https://es.wikipedia.org/wiki/Arreglo_de_sufijos)

## ¡Compartir ideas!

## Compressed $k^d$ -tree for temporal graphs

Diego Caro<sup>1</sup> · M. Andrea Rodríguez<sup>1</sup> ·  
Nieves R. Brisaboa<sup>2</sup> · Antonio Fariña<sup>2</sup>

**Algorithm:** range( $l, n, u, R, z$ ) returns the set of active cells in region  $R$ .

**Output:** Cells inside the region defined by  $R$ .

```
if region(u, u + n) ∩ region(B) is empty then return
if Ti[z] = 1 then
    if l = depth-1 then
        | output u;
    else if Bi[rank(Ti, z)] = 1 then
        | p = rank(Bi, rank(Ti, z));
        | output Ai[p];
    if l = -1 then
        | z' = 0;
    else
        | z' = (rank(Ti, z - 1) - rank(Bi, rank(Ti, z - 1))) × kd;
for i = 0 to kd - 1 do
    for j = 0 to d - 1 do
        | u'_j = uj + n/k × (i/kj mod k)
        | range(l + 1, n/k, u', R, z' + i)
```

Fig. 7 Algorithm for orthogonal range search in the  $ck^d$ -tree

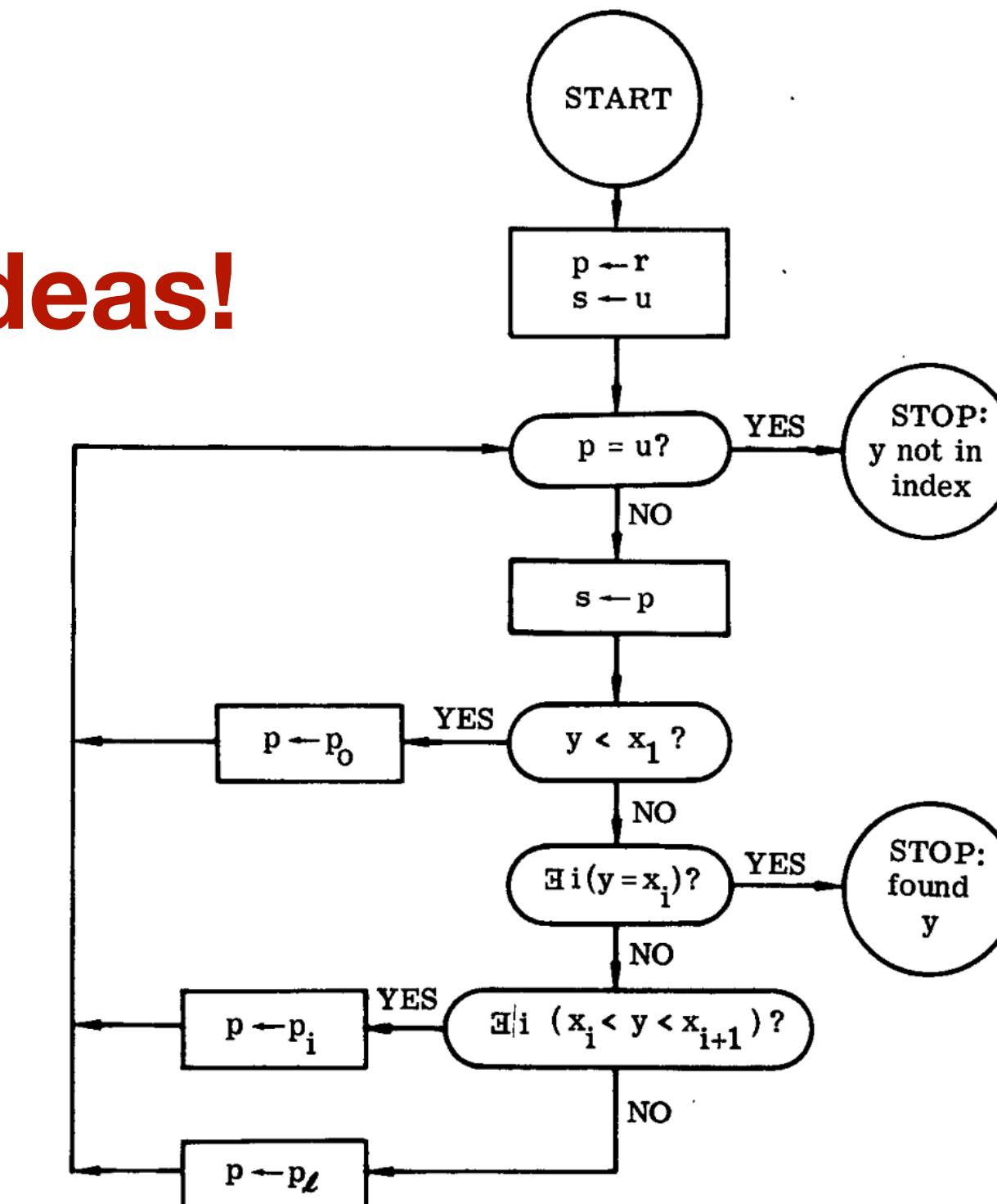
<https://link.springer.com/article/10.1007/s10115-015-0908-6>

# ¿Y realmente se usan los diagramas de flujo?

ORGANIZATION AND MAINTENANCE OF LARGE  
ORDERED INDICES  
by  
R. Bayer  
and  
E. McCreight

Este artículo científico es fundacional. Los sistemas de archivos  
**NTFS, EXT4, HFS** y otros usan esta idea! También es el origen  
de los índices para bases de datos.

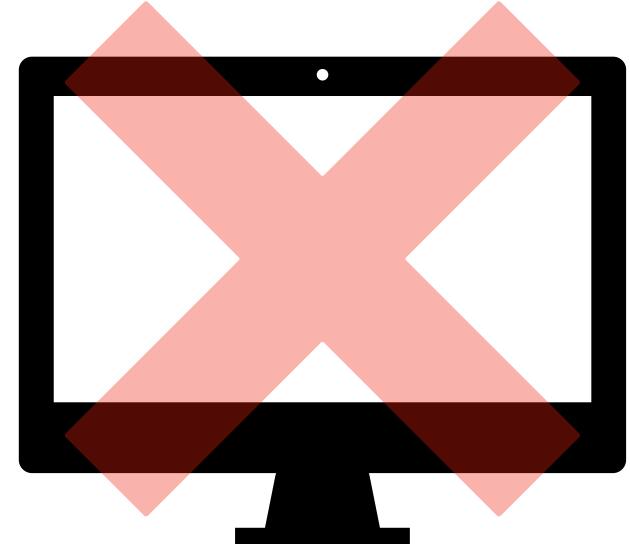
¡Compartir ideas!



**Pseudocódigo**

**Diagramas de Flujo**

**Pensar  
+  
Comunicar**



# Algunas reglas

- Atención: estamos usando pseudocódigo basado en Python
  - Eso NO QUIERE DECIR que podemos usar las funcionalidades de Python
  - Por ejemplo, podemos escribir notación matemática siempre y cuando sea legible
    - **Regla 1:** si la expresión matemática es compleja, divídela en pasos. Recuerda que el objetivo es comunicar!
    - **Regla 2:** la sintáxis se parece a Python, pero NO ES PYTHON. Por lo tanto, no puedes suponer que tu pseudocódigo lo entenderá cualquier persona, no todos saben Python.

# Actividad asincrónica

- Ver video Introducción a la programación
  - <https://youtu.be/8GneSctTItA>

