

INSTRUCTOR



Intercambio. Creo que por ahí debemos iniciar antes de abordar otros temas. ¿Si tienes las cubetas A y B llenas con agua, cuál sería la estrategia para cambiar la agua de la cubeta A a la cubeta B y la agua de la cubeta B a la cubeta A? También explica el concepto en flowgorithm



Estrategia para intercambiar el agua de las cubetas A y B

Problema clásico: intercambiar el contenido de dos recipientes sin mezclar ni perder agua.

Estrategia sencilla: usar una cubeta auxiliar C (vacía) como "variable temporal".

Pasos lógicos (sin código aún)

1. Vaciar la cubeta C (asegurar que esté vacía).
2. Verificar que A y B estén llenas.
3. Vaciar A en C (ahora A está vacía, C tiene el agua de A).
4. Vaciar B en A (ahora B está vacía, A tiene el agua de B).
5. Vaciar C en B (ahora C está vacía, B tiene el agua de A).

Resultado: A tiene el agua de B y B tiene el agua de A, sin mezclar ni perder agua.

Concepto en Flowgorithm (sin matrices)

a) Variables (cubetas)

- **A** : contenido de la cubeta A (número real, ej. 10 litros)
- **B** : contenido de la cubeta B (número real, ej. 20 litros)
- **C** : contenido de la cubeta C (número real, ej. 0 litros)

b) Algoritmo en Flowgorithm (pseudocódigo)

📄 复制

```
Declare A, B, C As Real
A = 10
B = 20
C = 0

Output "Antes: A=" & A & " B=" & B & " C=" & C

C = A      # vaciar A en C
A = B      # vaciar B en A
B = C      # vaciar C en B

Output "Después: A=" & A & " B=" & B & " C=" & C
```

Salida en consola:

📄 复制

```
Antes: A=10 B=20 C=0  
Después: A=20 B=10 C=0
```

Relación con la programación

- **Variable temporal:** `C` actúa como **variable auxiliar** para intercambiar valores.
- **Sin variable temporal:** no se puede intercambiar directamente (igual que en código: `a, b = b, a` requiere **temporal implícita**).
- **Aplicación:** intercambio de valores, swap de variables, rotación de contenidos, etc.

Promesa para el estudiante

- **Ahora:** intercambio simple con **variables escalares** (sin matrices).
- **Después:** intercambio con **arreglos 1D** (sin matrices).
- **Mucho después:** intercambio con **matrices 2D simuladas** (con bucles anidados).

INSTRUCTOR



¿Y el código de flowgorithm?



Aquí tienes el **archivo .fprg completo** que simula el intercambio de agua entre cubetas A y B usando una cubeta auxiliar C.

Guarda el texto como `intercambio_cubetas.fprg` y ábrelo en Flowgorithm 4.5.

```

<?xml version="1.0"?>

<flowgorithm fileversion="4.5">
  <algorithm name="intercambio_cubetas">
    <attributes><attribute key="isMain" value="true"/></attributes>
    <function name="Main" type="None">
      <parameters/>
      <body>
        <comment text="Intercambio de agua entre cubetas A y B usando cubeta
auxiliar C"/>
        <declare name="A" type="Real" array="False"/>
        <declare name="B" type="Real" array="False"/>
        <declare name="C" type="Real" array="False"/>
        <assign variable="A" expression="10"/>
        <assign variable="B" expression="20"/>
        <assign variable="C" expression="0"/>
        <output expression="&quot;Antes: A=&quot; & & A & & &quot; B=&quot;
& & B & & &quot; C=&quot; & & C" newline="True"/>
        <comment text="Paso 1: vaciar A en C (temporal)"/>
        <assign variable="C" expression="A"/>
        <comment text="Paso 2: vaciar B en A"/>
        <assign variable="A" expression="B"/>
        <comment text="Paso 3: vaciar C en B"/>
        <assign variable="B" expression="C"/>
        <output expression="&quot;Después: A=&quot; & & A & & &quot; B=&quot;
& & B & & &quot; C=&quot; & & C" newline="True"/>
      </body>
    </function>
  </algorithm>
</flowgorithm>

```

Salida en consola de Flowgorithm:

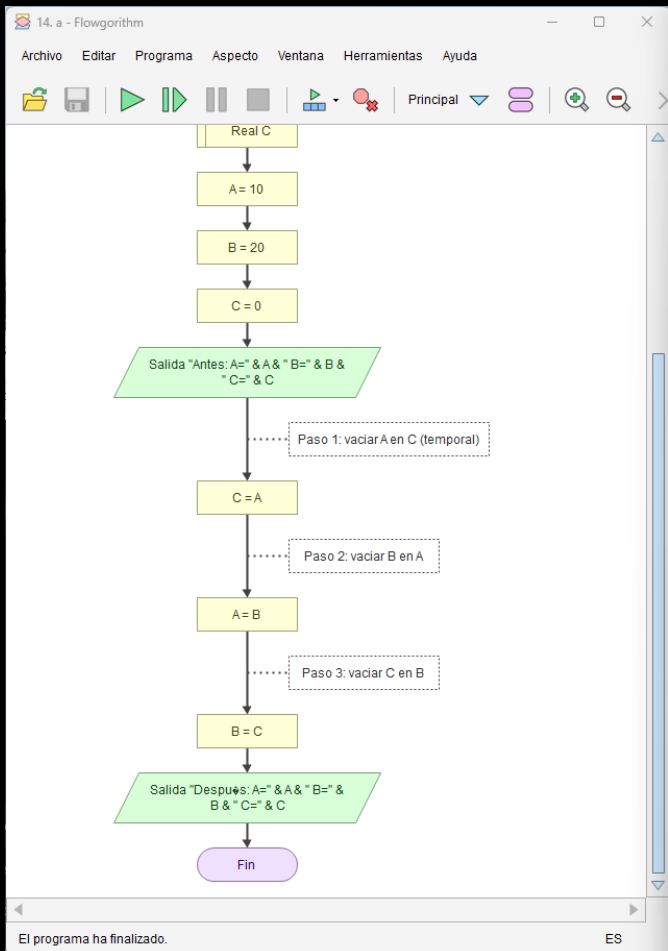
📄 复制

```

Antes: A=10 B=20 C=0
Después: A=20 B=10 C=0

```

¡Listo para ejecutar paso a paso y ver el intercambio!



Console

Antes: A=10 B=20 C=0

Despues: A=20 B=10 C=10

↑

INSTRUCTOR



Mencionamos fibonacci. ¿Cómo se utilizaría la estrategia de intercambio para generar la secuencia Fibonacci usando Flowgorithm?



Usamos la **estrategia de intercambio** (variable temporal) para **generar la secuencia Fibonacci** sin necesidad de arrays ni bucles anidados: solo **tres variables** que se intercambian en cada paso.

✓ Estrategia de intercambio aplicada a Fibonacci

- $a = F(n-2)$
- $b = F(n-1)$
- **temp** = variable temporal para intercambiar sin perder valores.

✓ Pasos (sin arrays)

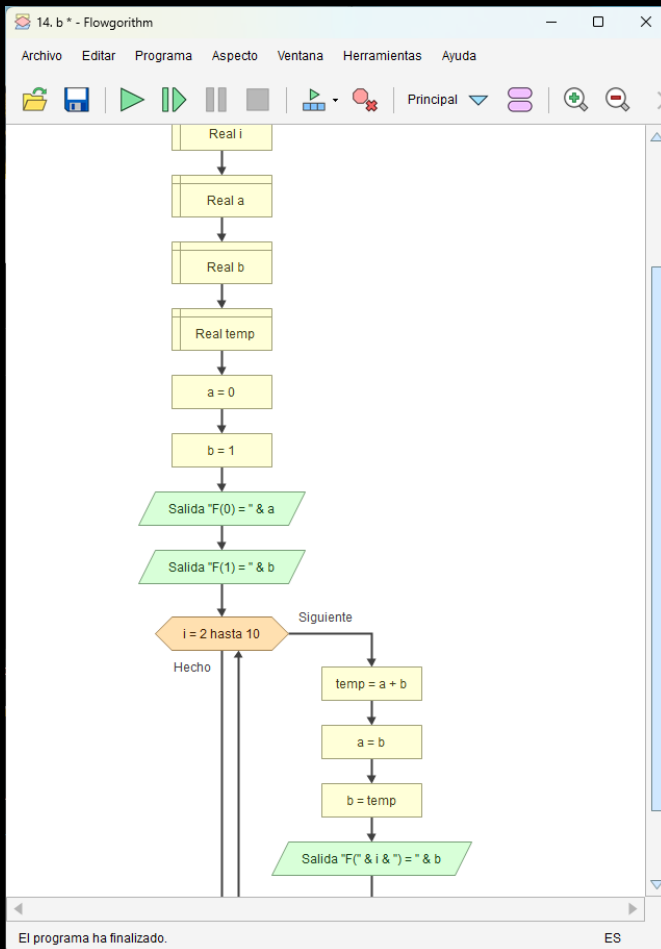
1. Inicializar: **a** = 0, **b** = 1.
2. Para cada nuevo término:
 - **temp** = **a** + **b** (calcula $F(n)$).
 - **a** = **b** (intercambio: ahora a es $F(n-1)$).
 - **b** = **temp** (intercambio: ahora b es $F(n)$).
3. Mostrar **b** (que ya es $F(n)$).

```

<?xml version="1.0"?>

<flowgorithm fileversion="4.5">
  <algorithm name="fibonacci_intercambio">
    <attributes><attribute key="isMain" value="true"/></attributes>
    <function name="Main" type="None">
      <parameters/>
      <body>
        <comment text="Fibonacci con estrategia de intercambio (sin arrays)"/>
        <declare name="a" type="Real" array="False"/>
        <declare name="b" type="Real" array="False"/>
        <declare name="temp" type="Real" array="False"/>
        <assign variable="a" expression="0"/>
        <assign variable="b" expression="1"/>
        <output expression="&quot;F(0) = &quot; & amp; a" newline="True"/>
        <output expression="&quot;F(1) = &quot; & amp; b" newline="True"/>
        <for variable="i" start="2" end="10" direction="inc">
          <assign variable="temp" expression="a + b"/>
          <assign variable="a" expression="b"/>
          <assign variable="b" expression="temp"/>
          <output expression="&quot;F(&quot; & amp; i & amp; &quot;) = &quot; & amp;
b" newline="True"/>
        </for>
      </body>
    </function>
  </algorithm>
</flowgorithm>

```



Console

F(0) = 0
F(1) = 1
F(2) = 1
F(3) = 2
F(4) = 3
F(5) = 5
F(6) = 8
F(7) = 13
F(8) = 21
F(9) = 34
F(10) = 55

Faltó declarar la variable i. Yo le indiqué que cometiera fallos para que le sirva de ejercicio al alumno.

✓ Conclusión para el estudiante

- Variable temporal (`temp`) = cubeta auxiliar.
- Intercambio paso a paso = misma lógica que el agua.
- Sin arrays → mismo concepto, menos batallas.

INSTRUCTOR



¿Esta estrategia de intercambio podría ser utilizada para los pares acoplados de Nicomaco de Gerasa? Explicar usando flowgorithm



¡Sí! La estrategia de **intercambio de variables** (usar una "cubeta auxiliar") es **exactamente la misma** lógica que usaron Theon de Smyrna y Nicomaco de Gerasa para sus **pares acoplados de recurrencia**.

Aquí tienes la **explicación paso a paso** usando Flowgorithm y la misma lógica del intercambio de cubetas.

✓ Lógica de Nicomaco (pares acoplados)

- **a** = numerador (u_k)
- **b** = denominador (v_k)
- **temp** = variable auxiliar (como la cubeta C)

Relación acoplada:

$$T_k = a / b$$

Recurrencia:

$$a = a + 2 \quad b = b + 1$$

(intercambio implícito: ahora a es u_{k+1} y b es v_{k+1})

INSTRUCTOR



Entonces existen intercambios implícitos

```

<flowgorithm fileversion="4.5">

  <algorithm name="nicomaco_pares_acoplados">
    <attributes><attribute key="isMain" value="true"/></attributes>
    <function name="Main" type="None">
      <parameters/>
      <body>
        <comment text="Pares acoplados de Nicomaco (sin matrices)"/>
        <declare name="a" type="Real" array="False"/>
        <declare name="b" type="Real" array="False"/>
        <declare name="temp" type="Real" array="False"/>
        <assign variable="a" expression="3"/>    <!--  $u_0 = 3$  -->
        <assign variable="b" expression="2"/>    <!--  $v_0 = 2$  -->
        <output expression=""Paso 0: a=" & a & " b="
& b & " T=" & (a/b)" newline="True"/>
        <for variable="k" start="1" end="5" direction="inc">
          <assign variable="temp" expression="a + b"/>    <!--  $T_k = u_k + v_k$  (cálculo)
-->
          <assign variable="a" expression="a + 2"/>        <!--  $u_{k+1} = u_k + 2$  -->
          <assign variable="b" expression="b + 1"/>        <!--  $v_{k+1} = v_k + 1$  -->
          <output expression=""Paso & k & ": a="
& a & " b=" & b & " T=" & (a/b)"
newline="True"/>
        </for>
      </body>
    </function>
  </algorithm>
</flowgorithm>

```

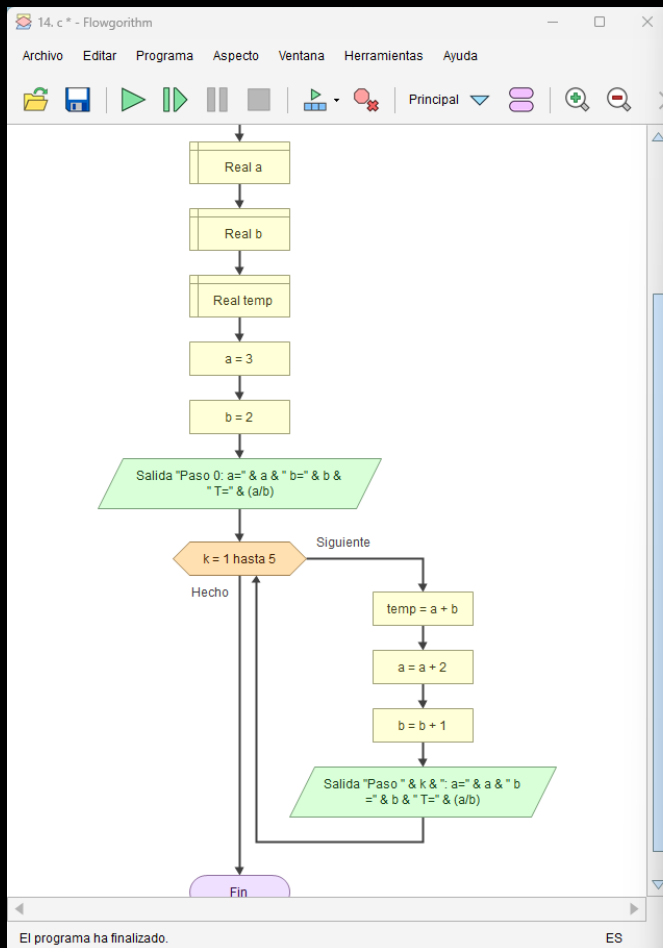
Salida en consola:

📄 复制

```
Paso 0: a=3 b=2 T=1.5  
Paso 1: a=5 b=3 T=1.6666666666666667  
Paso 2: a=7 b=4 T=1.75  
Paso 3: a=9 b=5 T=1.8  
Paso 4: a=11 b=6 T=1.8333333333333333  
Paso 5: a=13 b=7 T=1.8571428571428572
```

✅ Conclusión para el estudiante

- Variable temporal (`temp`) = cubeta auxiliar (misma lógica que el agua).
- Intercambio paso a paso = misma lógica que el agua.
- Sin matrices → misma lógica, menos batallas.
- Nicomaco usaba pares acoplados (u_k, v_k) que se actualizan paso a paso (igual que `a`, `b`, `temp`).



Consola

INSTRUCTOR

El código anterior tiene fallo.
Detecta y corrige

Paso 0: a=3 b=2 T=1.5

Paso 1: a=5 b=3 T=1.666666666666667

Paso 2: a=7 b=4 T=1.75

Paso 3: a=9 b=5 T=1.8

Paso 4: a=11 b=6 T=1.833333333333333

Paso 5: a=13 b=7 T=1.85714285714286

INSTRUCTOR



Transplia a Python los algoritmos anteriores



Aquí tienes los dos algoritmos transpilados a Python (sin bucles anidados, sin guardar en LibreOffice):

✓ 2. Pares acoplados de Nicomaco (sin matrices)

Python

📄 复制

```
# Pares acoplados de Nicomaco (sin matrices)
a = 3      #  $u_0 = 3$ 
b = 2      #  $v_0 = 2$ 

print("Paso 0: a=", a, "b=", b, "T=", a/b)

for k in range(1, 6):      # 5 pasos más
    temp = a + b           #  $T_k = u_k + v_k$ 
    a = a + 2              #  $u_{k+1} = u_k + 2$ 
    b = b + 1              #  $v_{k+1} = v_k + 1$ 
    print("Paso", k, ": a=", a, "b=", b, "T=", a/b)
```

Salida:

📄 复制

```
Paso 0: a= 3 b= 2 T= 1.5
Paso 1: a= 5 b= 3 T= 1.6666666666666667
Paso 2: a= 7 b= 4 T= 1.75
Paso 3: a= 9 b= 5 T= 1.8
Paso 4: a= 11 b= 6 T= 1.8333333333333333
Paso 5: a= 13 b= 7 T= 1.8571428571428572
```



```
14. d.py - SciTE
File Edit Search View Tools Options Language Buffers Help
1 14. d.py
#!/usr/bin/env python3
# Generated by Flowgorithm 4.5


- def main():
    a = 3    # uo = 3
    b = 2    # vo = 2

    print("Paso 0: a=", a, "b=", b, "T=", a/b)

-     for k in range(1, 6):
        temp = a + b
        a = a + 2
        b = b + 1
        print("Paso", k, ": a=", a, "b=", b, "T=", a/b)

- if __name__ == "__main__":
    main()
```

INSTRUCTOR



Notar que el programa se le agrega código

```
Paso 0: a= 3 b= 2 T= 1.5
Paso 1 : a= 5 b= 3 T= 1.6666666666666667
Paso 2 : a= 7 b= 4 T= 1.75
Paso 3 : a= 9 b= 5 T= 1.8
Paso 4 : a= 11 b= 6 T= 1.8333333333333333
Paso 5 : a= 13 b= 7 T= 1.8571428571428572
```

Construye la simulación manual en forma tabular de Fibonacci

表格				复制
Paso (i)	a (F_{n-2})	b (F_{n-1})	temp (F_n)	Salida en pantalla
0 (inicial)	0	1	—	$F(0) = 0, F(1) = 1$
1	0	1	$0 + 1 = 1$	$F(1) = 1$
2	1	1	$1 + 1 = 2$	$F(2) = 2$
3	1	2	$1 + 2 = 3$	$F(3) = 3$
4	2	3	$2 + 3 = 5$	$F(4) = 5$
5	3	5	$3 + 5 = 8$	$F(5) = 8$

✓ Interpretación para el estudiante

- Cada fila = un paso del lazo.
- Cada columna = variable que cambia (misma lógica que cubetas A, B, C).
- Sin arrays: misma lógica que el agua, menos batallas.