# Segundo Parcial

## Arreglos en general


*Explicación: La mayoría de lenguajes tienen la posibilidad de poder trabajar arreglos de n dimensiones, sin embargo flowgorithm no cuenta con esa potencialidad. Flowgorithm puede simular arreglos de n dimensiones utilizando arreglos de una única dimensión.*

INSTRUCTOR

Como se puede utilizar un arreglo de una sola dimensión para simular arreglos de 1, 2, 3, 4 y 5 dimensiones

## Simulando cualquier dimensión con un solo arreglo 1D

**Idea clave:**
*Aplanar* el índice multidimensional con una **fórmula de desplazamiento** que **respeta el orden de filas, columnas, páginas, etc.**

---

### 1. Fórmula general (column-major, C-style)

Para un arreglo $A[d_1][d_2]...[d_n]$ almacenado en **un solo vector 1D** de tamaño $T = d_1 \times d_2 \times ... \times d_n$ :

$$\text{índice 1D} = i_1 \cdot (d_2 \cdot d_3 \cdot \ldots \cdot d_n) + i_2 \cdot (d_3 \cdot \ldots \cdot d_n) + \ldots + i_n$$

En código:

Python 复制

```
idx = i1 * stride[0] + i2 * stride[1] + ... + in * stride[n-1]
```

donde $stride[k]$ = producto de las dimensiones desde k+1 hasta la final .

## 2. Ejemplos concretos (y su índice 1D)

表格 复制

| Dim | Dimensiones | Fórmula C (row-major) | E |
|---|---|---|---|
| 1D | [d] | i | |
| 2D | [d1][d2] | i * d2 + j | |
| 3D | [d1][d2][d3] | i * d2 * d3 + j * d3 + k | |
| 4D | [d1][d2][d3][d4] | i * d2 * d3 * d4 + j * d3 * d4 + k * d4 + l | |
| 5D | [d1][d2][d3][d4][d5] | i * d2 * d3 * d4 * d5 + j * d3 * d4 * d5 + k * d4 * d5 + l * d5 + m | |

## 3. Ejemplo completo en Flowgorithm (3D → 1D)

xml 复制

```xml
<!-- Dimensión 3D: d1 × d2 × d3 -->
<declare name="d1" type="Integer" array="False"/><assign variable="d1"
expression="4"/>
<declare name="d2" type="Integer" array="False"/><assign variable="d2"
expression="3"/>
<declare name="d3" type="Integer" array="False"/><assign variable="d3"
expression="2"/>

<declare name="A" type="Real" array="True" size="24"/>   <!-- 4×3×2 = 24 -->

<!-- Rellenar consecutivamente 0...23 -->
<declare name="cont" type="Integer" array="False"/><assign variable="cont"
expression="0"/>

<for variable="i" start="0" end="d1 - 1">
  <for variable="j" start="0" end="d2 - 1">
    <for variable="k" start="0" end="d3 - 1">
      <assign variable="A[i * d2 * d3 + j * d3 + k]" expression="cont"/>
      <assign variable="cont" expression="cont + 1"/>
    </for>
  </for>
</for>
```

### *4. Transpilación rápida a Python (3D → 1D)*

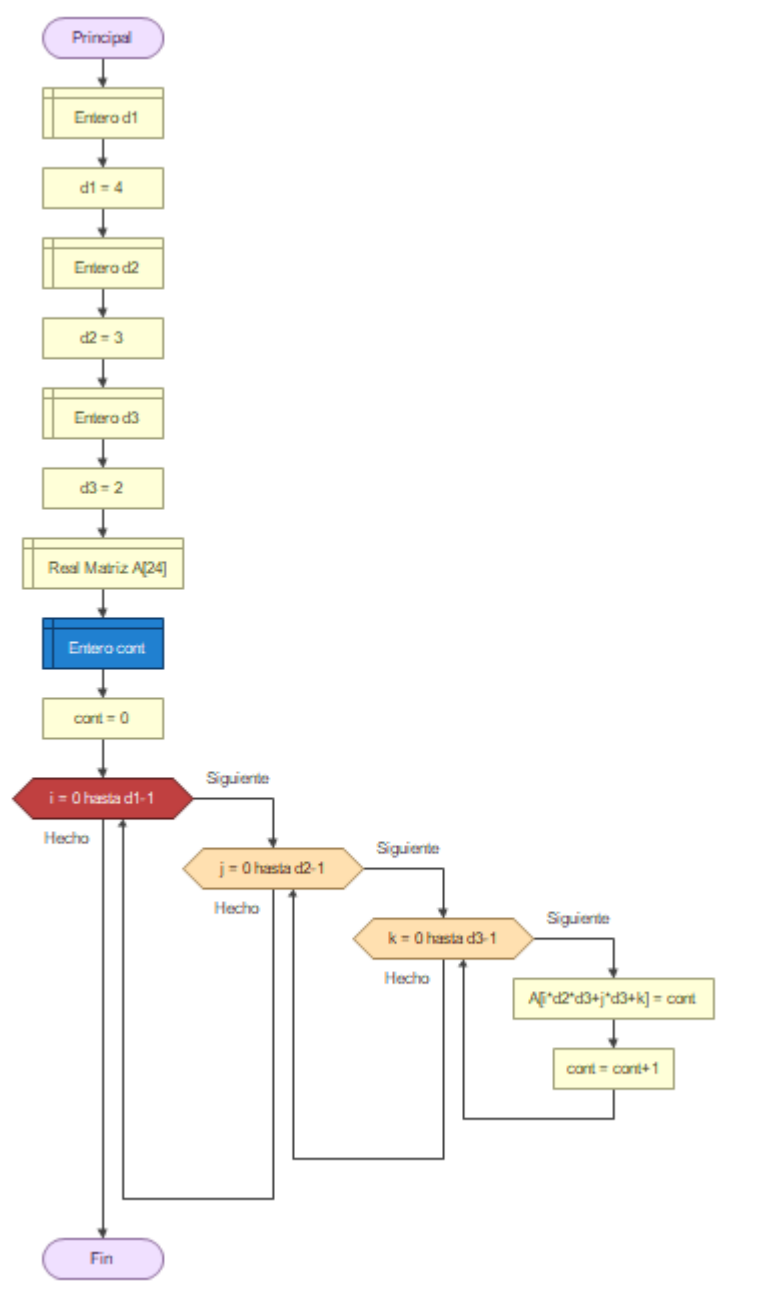
Python 复制

```python
d1, d2, d3 = 4, 3, 2
A = [0.0] * (d1 * d2 * d3)

for i in range(d1):
    for j in range(d2):
        for k in range(d3):
            idx = i * d2 * d3 + j * d3 + k
            A[idx] = float(i * 100 + j * 10 + k)   # ejemplo cualquiera

print("A 3D aplanado:", A)
```

**5. Conclusión**

Con **una sola dimensión y la fórmula de stride** puedes **simular cualquier número de dimensiones** sin cambiar la estructura del lenguaje.

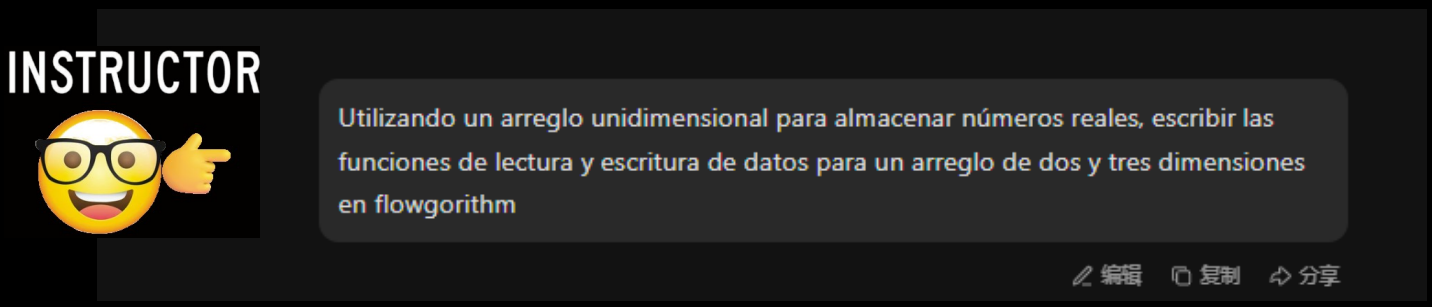
*El programa anterior de Kimi (flowgorithm) contiene errores.*

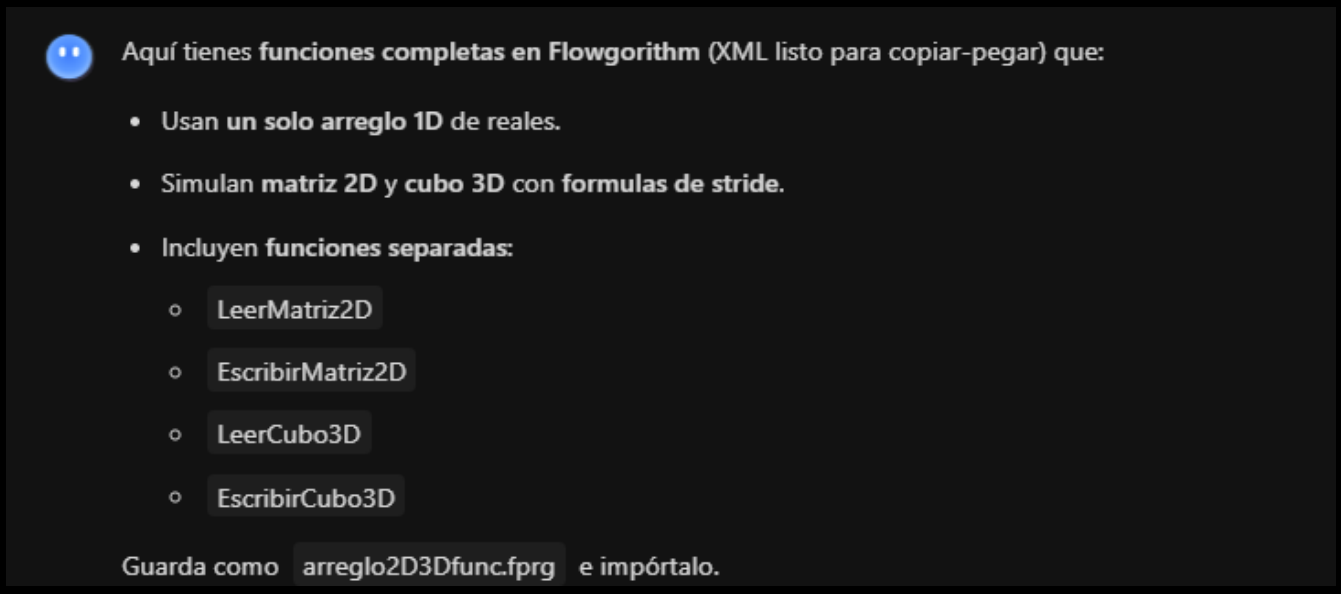



# INSTRUCTOR

*Modificar el programa de flowgorithm anterior para que funcione de manera correcta*

*Explicación: La mayoría de las veces que se utilizan arreglos en programas se utilizan arreglos de 1, 2 o incluso 3 dimensiones. Es raro ver lógica de programas que usen más de tres dimensiones.*



xml 复制

```xml
<?xml version="1.0"?>
<flowgorithm fileversion="3.0">
  <!-- ============================================================
       FUNCIONES: leer/escribir 2D y 3D usando UN SOLO ARREGLO 1D
       Autor: Kimi
       ============================================================ -->
  <function name="LeerMatriz2D" type="None">
    <parameter name="A" type="Real" array="True"/>
    <parameter name="filas" type="Integer" array="False"/>
    <parameter name="cols" type="Integer" array="False"/>
    <body>
      <for variable="i" start="0" end="filas - 1">
        <for variable="j" start="0" end="cols - 1">
          <output text="Elemento [" newline="False"/>
          <output expression="i" newline="False"/>
          <output text="," newline="False"/>
          <output expression="j" newline="False"/>
          <output text="]: " newline="False"/>
          <declare name="temp" type="Real" array="False"/>
          <input variable="temp"/>
          <assign variable="A[i * cols + j]" expression="temp"/>
        </for>
      </for>
    </body>
  </function>
```

```
<function name="EscribirMatriz2D" type="None">
  <parameter name="A" type="Real" array="True"/>
  <parameter name="filas" type="Integer" array="False"/>
  <parameter name="cols" type="Integer" array="False"/>
  <body>
    <output text="Matriz 2D:" newline="True"/>
    <for variable="i" start="0" end="filas - 1">
      <for variable="j" start="0" end="cols - 1">
        <output expression="A[i * cols + j]" newline="False"/>
        <if expression="j &lt; cols - 1">
          <then><output text="," newline="False"/></then>
        </if>
      </for>
      <output text="" newline="True"/>
    </for>
  </body>
</function>
```

```
<function name="LeerCubo3D" type="None">
  <parameter name="A" type="Real" array="True"/>
  <parameter name="d1" type="Integer" array="False"/>
  <parameter name="d2" type="Integer" array="False"/>
  <parameter name="d3" type="Integer" array="False"/>
  <body>
    <for variable="i" start="0" end="d1 - 1">
      <for variable="j" start="0" end="d2 - 1">
        <for variable="k" start="0" end="d3 - 1">
          <output text="Elemento [" newline="False"/>
          <output expression="i" newline="False"/>
          <output text="," newline="False"/>
          <output expression="j" newline="False"/>
          <output text="," newline="False"/>
          <output expression="k" newline="False"/>
          <output text="]: " newline="False"/>
          <declare name="temp" type="Real" array="False"/>
          <input variable="temp"/>
          <assign variable="A[i * d2 * d3 + j * d3 + k]" expression="temp"/>
        </for>
      </for>
    </for>
  </body>
</function>
```

```
<function name="EscribirCubo3D" type="None">
  <parameter name="A" type="Real" array="True"/>
  <parameter name="d1" type="Integer" array="False"/>
  <parameter name="d2" type="Integer" array="False"/>
  <parameter name="d3" type="Integer" array="False"/>
  <body>
    <output text="Cubo 3D:" newline="True"/>
    <for variable="i" start="0" end="d1 - 1">
      <output text="Capa " newline="False"/>
      <output expression="i" newline="True"/>
      <for variable="j" start="0" end="d2 - 1">
        <for variable="k" start="0" end="d3 - 1">
          <output expression="A[i * d2 * d3 + j * d3 + k]" newline="False"/>
          <if expression="k &lt; d3 - 1">
            <then><output text="," newline="False"/></then>
          </if>
        </for>
        <output text="" newline="True"/>
      </for>
      <output text="" newline="True"/>
    </for>
  </body>
</function>
```

```
  <!-- ================  PROGRAMA PRINCIPAL  ================ -->
  <function name="Main" type="None">
    <parameter/>
    <body>
      <!-- ---------- 2D ---------- -->
      <declare name="n2" type="Integer" array="False"/>
      <output text="Orden 2D n: "/>
      <input variable="n2"/>
      <declare name="Mat2D" type="Real" array="True" size="10000"/>
      <call function="LeerMatriz2D"><argument name="Mat2D"/><argument name="n2"/>
<argument name="n2"/></call>
      <call function="EscribirMatriz2D"><argument name="Mat2D"/><argument
name="n2"/><argument name="n2"/></call>

      <!-- ---------- 3D ---------- -->
      <declare name="n3" type="Integer" array="False"/>
      <output text="Orden 3D n: "/>
      <input variable="n3"/>
      <declare name="Cub3D" type="Real" array="True" size="100000"/>
      <call function="LeerCubo3D"><argument name="Cub3D"/><argument name="n3"/>
<argument name="n3"/><argument name="n3"/></call>
      <call function="EscribirCubo3D"><argument name="Cub3D"/><argument
name="n3"/><argument name="n3"/><argument name="n3"/></call>
    </body>
  </function>
</flowgorithm>
```

# INSTRUCTOR

*El programa anterior tiene algunos errores- Corregir*

# INSTRUCTOR

El código anterior contiene las siguientes líneas:

```
      <declare name="n2" type="Integer" array="False"/>
      <output text="Orden 2D n: "/>
      <input variable="n2"/>
      <declare name="Mat2D" type="Real" array="True" size="10000"/>
      <call function="LeerMatriz2D"><argument name="Mat2D"/><argument
name="n2"/><argument name="n2"/></call>
```

Llamando a la función `LeerMatriz2D` con argumentos n2 y n2. Para matrices cuadradas. Modificar el código para que solucione matrices de cualquier orden.

De igual forma, contiene las líneas:

```
      <declare name="n3" type="Integer" array="False"/>
      <output text="Orden 3D n: "/>
      <input variable="n3"/>
      <declare name="Cub3D" type="Real" array="True" size="100000"/>
      <call function="LeerCubo3D"><argument name="Cub3D"/><argument
name="n3"/><argument name="n3"/><argument name="n3"/></call>
```

Para solucionar cubos uniformes en “i”, “j” y “k”. Modificar el código para que solucione cubos de cualquier orden.