

# Apuntes de Algoritmos

## Apuntes complementarios para la asignatura Fundamentos de Programación (PRO111)

Material preparado por Dr. Gabriel Astudillo Muñoz,  
Escuela de Ingeniería Informática

### 1. Algoritmo

Un **algoritmo** es una secuencia finita de pasos para resolver un problema. Los pasos deben estar muy bien definidos, y tienen que describir, sin ambigüedades, cómo llegar desde el inicio hasta el final.

### 2. Componentes de un algoritmo

Conceptualmente, un algoritmo tiene tres componentes:

1. La **entrada**: son los datos sobre los que el algoritmo opera.
2. El **proceso**: es el conjunto de pasos finitos y bien definidos, que hay que seguir. Estos pasos o instrucciones son los que transforman los datos de salidas en los datos de entrada.
3. La **salida**: es el resultado que entrega el algoritmo.

El proceso es una secuencia de instrucciones, que debe ser realizada en orden. El proceso también puede tener ciclos o bucles (grupos de sentencias que son ejecutadas varias veces) y condicionales (grupos de sentencias que sólo son ejecutadas bajo ciertas condiciones).

### 3. Diseño de algoritmos

Para la representación y el diseño de algoritmos, es usual utilizar dos herramientas: diagramas de flujo y pseudocódigo. Son herramientas distintas pero complementarias. Ambas sirven para representar la lógica y la estructura de un algoritmo de manera visual y comprensible, aunque utilizan diferentes enfoques para lograrlo.

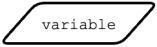
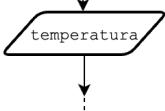
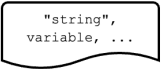
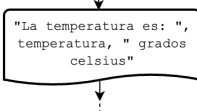
**Diagrama de flujo:** Es una representación gráfica de un algoritmo que utiliza símbolos y líneas para representar las operaciones y el flujo de control del programa. Los símbolos representan diferentes tipos de operaciones (como asignaciones, decisiones, bucles, etc.), mientras que las líneas conectan estos símbolos para mostrar el flujo de ejecución del algoritmo. Los diagramas de flujo son útiles para visualizar la estructura y el flujo de un algoritmo de manera intuitiva, lo que facilita la comprensión y la comunicación entre los miembros del equipo de desarrollo.

**Pseudocódigo:** Es una representación textual de un algoritmo que utiliza un lenguaje de programación simplificado y estructurado, cercano al lenguaje humano. El pseudocódigo describe paso a paso las operaciones que el algoritmo debe realizar, utilizando una sintaxis simple y clara. Es útil para planificar y diseñar algoritmos antes de su implementación en un lenguaje de programación específico.


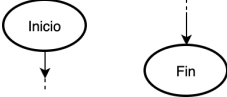

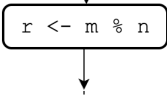

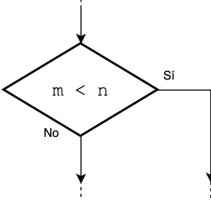
### 3.1. Diagramas de flujo

Una de las formas de representar en forma gráfica un algoritmo, es el uso de Diagramas de Flujo, los que están definidos en la norma ISO 5807-1985<sup>1</sup>. Se componen en tres conjuntos de símbolos: 1) Símbolos para el tratamiento de datos, 2) Símbolos de procesos y 3) Símbolos de línea.

#### 3.1.1. Símbolos de datos


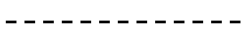
Símbolo	Uso	Ejemplo	Descripción
	Sirve para especificar valor de variables. Dicho valor se obtiene a través de una entrada manual de datos, por ejemplo, vía teclado o formulario web. El dato ingresado es almacenado en la variable indicada en el interior del símbolo.		Indica que el dato ingresado es almacenado en la variable <b>temperatura</b> .
	Este símbolo representa una acción de mostrar. Lo que se muestra (por ejemplo, en la pantalla del computador) es lo que se indica en el interior del símbolo.		Indica que se mostrará el string "La temperatura es ", luego el valor de la variable <b>temperatura</b> seguido del string " grados celsius".

#### 3.1.2. Símbolos de procesos

Símbolo	Uso	Ejemplo	Descripción
	Especifica el inicio o el fin del algoritmo		
	Mostrar datos para que puedan ser leídos por humanos. El medio puede ser la pantalla, impresora, página web, etc.		Determina el residuo de la división entera entre <b>m</b> y <b>n</b> . El resultado se lo asigna a <b>r</b> .
	Este símbolo representa una decisión. Esta decisión se toma en base al valor lógico de la expresión lógica que se indica en el interior del símbolo. Si la condición lógica es verdadera, toma la dirección de la rama "Sí". En caso contrario, el flujo toma la dirección de la rama "No"		La condición lógica que se evalúa es que si el contenido de la variable <b>m</b> es menor que <b>n</b> . Si esto se cumple, el flujo sigue por la línea indicada por "Sí". En caso contrario, el flujo sigue por la línea indicada por "No".

<sup>1</sup> <https://www.iso.org/standard/11955.html> Information processing - Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts

### 3.1.3. Símbolos de línea.

Símbolo	Uso
	Una flecha sólida representa el flujo de datos.
	Una línea segmentada sirve para colocar un comentario en un símbolo para aclarar su funcionamiento.

### 3.2. Pseudocódigo

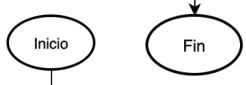
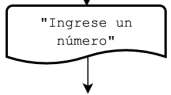
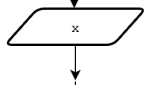
Un pseudocódigo es una representación de un algoritmo, que utiliza un lenguaje de descripción simple y estructurado, que se asemeja al lenguaje de programación pero es menos formal y más cercano al lenguaje humano.

El objetivo principal del pseudocódigo es ayudar a los programadores a planificar y organizar la lógica de un algoritmo antes de implementarlo en un lenguaje de programación específico. El pseudocódigo no está atado a la sintaxis de ningún lenguaje de programación en particular, lo que lo hace útil como una herramienta de diseño independiente del lenguaje<sup>2</sup>.

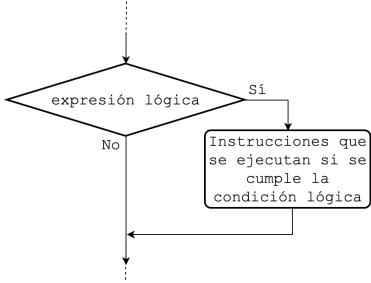
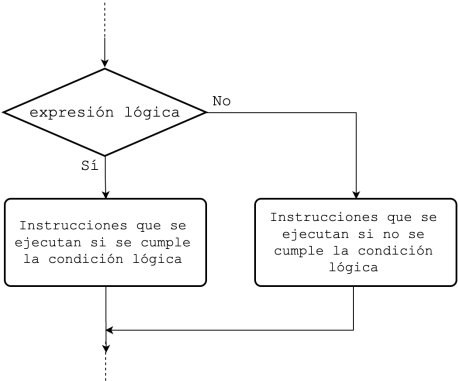
Un pseudocódigo generalmente incluye:

1. **Instrucciones:** Descripciones paso a paso de las operaciones que el algoritmo debe llevar a cabo.
2. **Estructuras de control:** Como bucles, condicionales y estructuras de decisión para controlar el flujo del programa.
3. **Variables y asignaciones:** Para representar datos y operaciones sobre ellos.
4. **Comentarios:** Notas explicativas que ayudan a comprender el código.

### 3.3. Relación entre Diagramas de Flujo y Pseudocódigo

Símbolo	Pseudocódigo
	<b>START</b>  <b>END</b>
	<b>WRITE</b> ("Ingrese un número")
	<b>READ</b> (x)

<sup>2</sup> En este documento, se opta por escribir las palabras reservadas del pseudocódigo en inglés. En clases se ha utilizado palabras en español, pero su traducción es trivial.

Símbolo	Pseudocódigo
	<pre> <b>IF</b> (condición lógica) <b>THEN</b>     Instrucciones que se     ejecutan si se cumple     la condición lógica <b>END IF</b> </pre>
	<pre> <b>IF</b> (expresión lógica) <b>THEN</b>     Instrucciones que se     ejecutan si se cumple     la condición lógica <b>ELSE</b>     Instrucciones que se     ejecutan si no se cum-     ple la condición ló-     gica <b>END SI</b> </pre>

#### 4. Estructuras de control: condicionales

##### 4.1. Estructura de decisión simple: IF - THEN

Permite que un bloque de instrucciones (puede ser sólo una) sea ejecutado si y sólo si una **expresión lógica** es **verdadera**. Si la expresión lógica es **falsa** se salta ese bloque de instrucciones y se ejecuta la primera sentencia ejecutable que está fuera de la estructura.

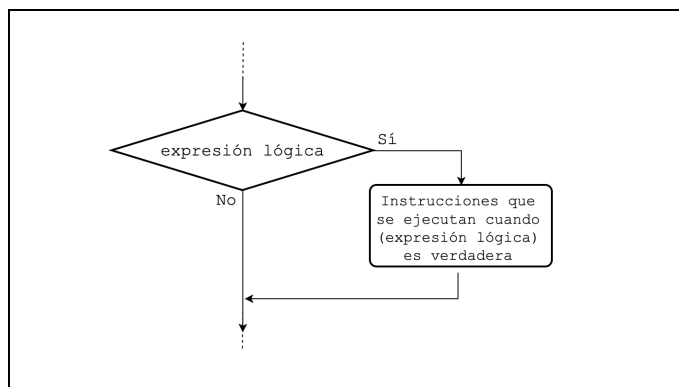
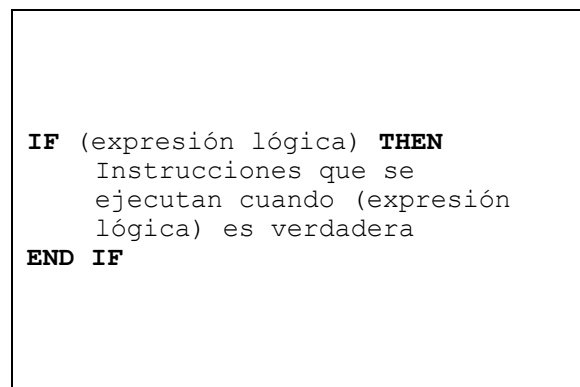
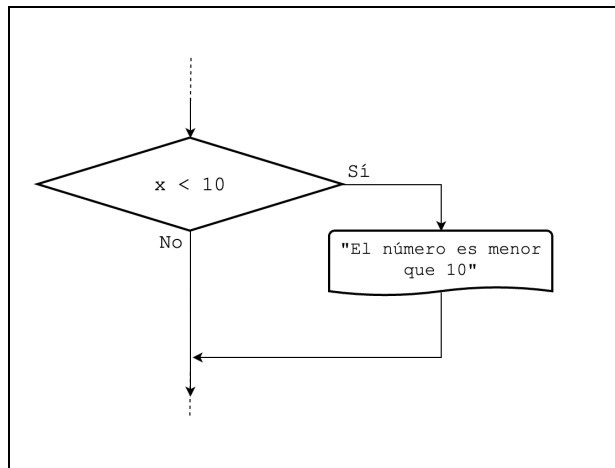


Diagrama de Flujo de la estructura  
IF-THEN

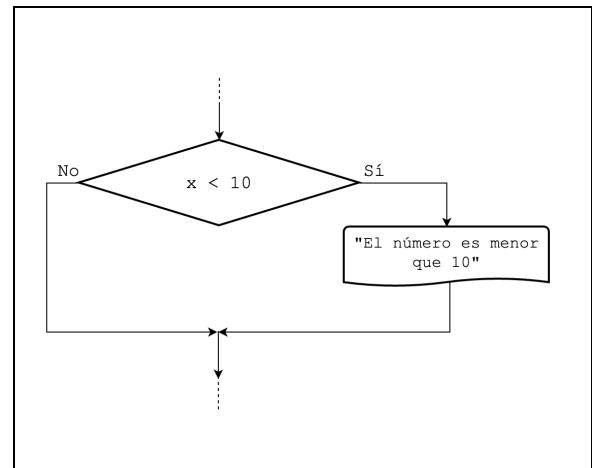


Pseudocódigo estructura  
IF-THEN

**Observación:** según la norma ISO 5807:1985 no hay reglas específicas sobre la posición de la rama "Sí" y "No" en el símbolo de decisión. Esto significa que, por ejemplo, los diagrama a) y b) que se muestran son equivalentes.



a) Diagrama de Flujo de la estructura IF-THEN



b) Diagrama de Flujo de la estructura IF-THEN

Cómo se verá más adelante, ambos códigos representan el pseudocódigo:

```

IF x < 10 THEN
    WRITE("El número es menor que 10")
END IF
  
```

#### 4.2. Estructura de decisión doble: IF - THEN - ELSE

Permite que un bloque de instrucciones (puede ser sólo una) sea ejecutado si y sólo si la **expresión lógica** es verdadera. Si la expresión lógica es falsa se salta ese bloque de instrucciones y se ejecuta un segundo bloque de instrucciones.

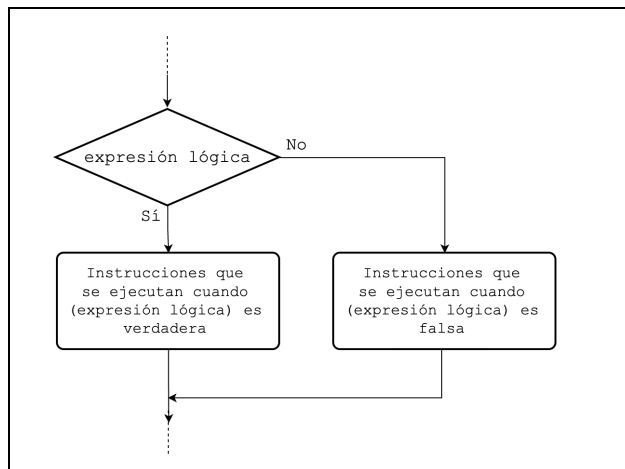


Diagrama de Flujo de la estructura IF-THEN-ELSE

```

IF (expresión lógica) THEN
    Instrucciones que se ejecutan
    cuando (expresión lógica) es
    verdadera
ELSE
    Instrucciones que se ejecutan
    cuando (expresión lógica) es
    falsa
END IF
  
```

Pseudocódigo estructura IF-THEN-ELSE

## 5. Ejemplo de algoritmo

La Figura 1 muestra un diagrama de flujo que determina si un número ingresado es mayor o menor igual que 2.

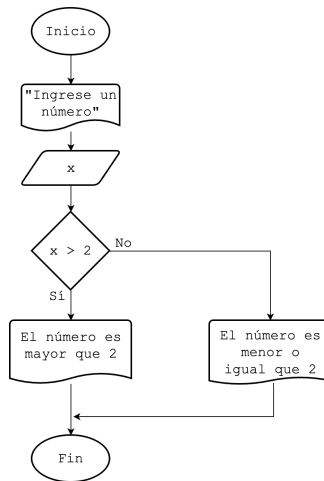


Figura 1



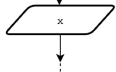
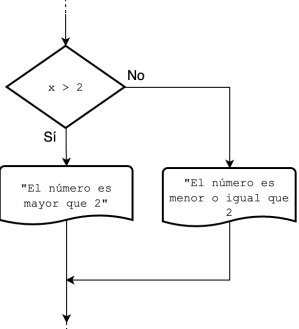

Explicación de los símbolos utilizados:

Tabla 1

Símbolo	Explicación
	Representa el inicio del algoritmo
	La figura indica que, al llegar a este punto, se mostrará el string "Ingrese un número" en la pantalla.
	La figura indica que el dato ingresado es almacenado en la variable <b>x</b> .
	La condición lógica que se evalúa es que si el contenido de la variable <b>x</b> es mayor que 2. Si esto se cumple, el flujo sigue por la línea indicada por "Sí". En caso contrario, el flujo sigue por la línea indicada por "No".

Símbolo	Explicación
	Representa el fin del algoritmo

El pseudocódigo que representa el algoritmo descrito por el diagrama de flujo de la Figura 1 se puede obtener directamente a través de la Tabla 1.

Símbolo	Descripción	Pseudocódigo
	Inicio del algoritmo	<b>START</b>
	Mostrar "Ingrese un número" en pantalla.	<b>WRITE</b> ("Ingrese un número")
	Leer desde teclado y almacenarlo en la variable <b>x</b> .	<b>READ</b> (x)
	La condición lógica que se evalúa es que si el contenido de la variable x es mayor que 2. El tipo de decisión se ajuste a una estructura <b>IF-THEN-ELSE</b> (ver sección 4.2).	<b>IF</b> x > 2 <b>THEN</b> <b>WRITE</b> ("El número es mayor que 2") <b>ELSE</b> <b>WRITE</b> ("El número es menor o igual que 2") <b>END IF</b>
	Fin del algoritmo	<b>END</b>

Adicionalmente, se observa que se necesita declarar la variable x. Para fines del ejemplo, se asume que es del tipo flotante (FLOAT). Finalmente, el pseudocódigo es:

```

VAR
    x: FLOAT
INPUT:
    x
OUTPUT:
    Dos mensajes:
    a) si el número es mayor que 2.
    b) si el número es menor o igual que 2.
START
    WRITE ("Ingrese un número")
    READ (x)
    IF x>2 THEN
        WRITE ("El número es mayor que 2")
    ELSE
        WRITE ("El número es menor o igual que 2")
    END IF
END

```

## **6. Implementación en un lenguaje de programación**

En preparación...



## 7. Ejemplos de Algoritmo

### 7.1. Resolver ecuaciones cuadráticas

#### 7.2. Problema

Determinar los valores de  $x$  que satisfacen la ecuación cuadrática de la forma  $ax^2 + bx + c = 0$ .

#### 7.3. Antecedentes

Del álgebra se sabe que la solución es:

$$x = \frac{-b \pm \sqrt{D}}{2a}, \text{ donde } D = b^2 - 4ac$$

Según el valor de  $D$  (discriminante), las raíces pueden ser de tres tipos.

a) Caso  $D > 0$

Existen dos raíces reales y distintas ( $x_1$  y  $x_2$ ):

$$x_1 = \frac{-b + \sqrt{D}}{2a} \text{ y } x_2 = \frac{-b - \sqrt{D}}{2a}$$

b) Caso  $D = 0$

Existen dos raíces reales e iguales ( $x_1 = x_2$ ):

$$x_1 = x_2 = \frac{-b}{2a}$$

c) Caso  $D < 0$

Existen dos raíces complejas y conjugadas ( $x_1$  y  $x_2$ ):

$$x_1 = \frac{-b}{2a} + i \frac{\sqrt{-D}}{2a} \text{ y } x_2 = \frac{-b}{2a} - i \frac{\sqrt{-D}}{2a}$$

#### 7.4. Análisis

Para obtener las soluciones, es necesario calcular el discriminante  $D$ , el que depende los coeficientes  $a$ ,  $b$  y  $c$ . Por lo tanto, las entradas deben ser estos coeficientes. Las salidas son las raíces calculadas. El proceso consta de las siguientes tareas: a) el cálculo de  $D$  b) determinar las raíces de la ecuación según el valor del discriminante  $D$ .

#### 7.5. Solución

##### 7.5.1. Algoritmo

**Entradas:**  $a, b$  y  $c \in \mathbb{R}$ , donde  $ax^2 + bx + c = 0$

**Salida:** Raíces  $x_1$  y  $x_2$  de la ecuación cuadrática.

Ecuación\_Cuadrática( $a, b, c$ ):

❶ Definición de variables

**D:** variable real que almacena el discriminante de la ecuación

$$D \leftarrow b^2 - 4ac$$

**x1, x2:** variables reales que almacenarán el valor de las raíces de la ecuación

2 Si  $D \geq 0$ , entonces

$$\text{Determinar } x1: x1 \leftarrow \frac{-b + \sqrt{D}}{2a}$$

$$\text{Determinar } x2: x2 \leftarrow \frac{-b - \sqrt{D}}{2a}$$

Mostrar x1 y x2.

4 Si  $D < 0$ , entonces

$$\text{Determinar } x1: x1 \leftarrow \frac{-b}{2a} + i \frac{\sqrt{-D}}{2a}$$

$$\text{Determinar } x2: x2 \leftarrow \frac{-b}{2a} - i \frac{\sqrt{-D}}{2a}$$

Mostrar x1 y x2.

### 7.5.2. Diagrama de Flujo

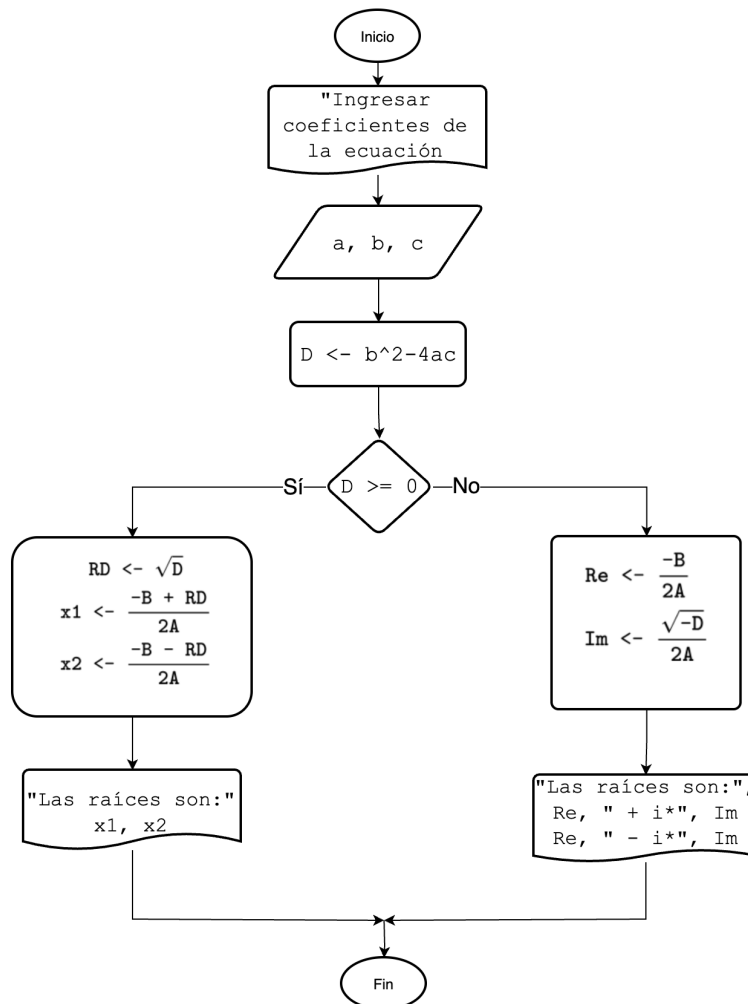


Figura 2 Diagrama de flujo para determinar las raíces reales o complejas de la ecuación cuadrática  $ax^2 + bx + c = 0$

### 7.5.3. Pseudocódigo

```

VAR
    a, b, c : FLOAT
    x1, x2  : FLOAT
INPUT:
    a,b,c
OUTPUT:
    x1, x2
START
    WRITE("Ingresar coeficientes de la ecuación (a,b,c):")
    READ(a,b,c)
     $D \leftarrow b^2 - 4ac$ 
    IF  $D \geq 0$  THEN
         $RD \leftarrow \sqrt{D}$ 
         $x1 \leftarrow \frac{-b+RD}{2a}$ 
         $x2 \leftarrow \frac{-b-RD}{2a}$ 
        WRITE("Las raíces son:")
        WRITE("x1=", x1)
        WRITE("x2=", x2)
    ELSE
         $Re \leftarrow \frac{-b}{2a}$ 
         $Im \leftarrow \frac{\sqrt{-D}}{2a}$ 
        WRITE("Las raíces son:")
        WRITE("x1=", Re, " +i*", Im)
        WRITE("x2=", Re, " -i*", Im)
    END IF
END

```

### 7.5.4. Implementación en Python3

Para el cálculo de la raíz cuadrada, se utilizó la función `sqrt()` de la librería `math`. Otra opción es elevar a 0.5.

```

import math

print("Ingresar coeficientes de la ecuación (a,b,c):")
a = float(input())
b = float(input())
c = float(input())

D = b**2 - 4*a*c

if D >= 0:
    RD = math.sqrt(D)
    x1 = (-b+RD)/(2*a)
    x2 = (-b-RD)/(2*a)
    print("Las raíces son:")
    print("x1=", x1)
    print("x2=", x2)
else:
    Re = -b/(2*a)
    Im = math.sqrt(-D)/(2*a)
    print("Las raíces son:")
    print("x1=", Re, " +i*", Im)
    print("x2=", Re, " -i*", Im)

```

**Observación:** en esta implementación, se asume que es posible realizar la conversión de **string** a **float** en las variables **a**, **b** y **c**. Una implementación más elaborada debería utilizar las instrucciones `try-except`.

