El Pensamiento Algorítmico

Materia: Algoritmos y Programación

Prof. Dr. BARSEKH-ONJI Aboud

Facultad de Ingeniería
Universidad Anáhuac México

16 de enero de 2026

Agenda

¿Qué es un Algoritmo?

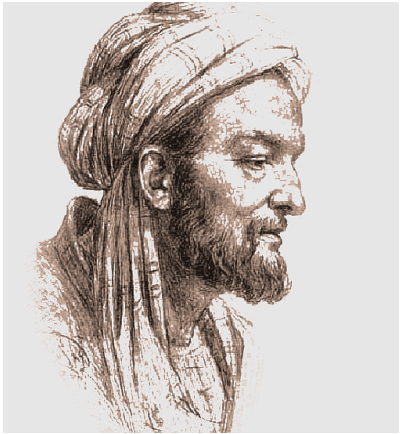
Definición Intuitiva: Una Receta

En su esencia, un **algoritmo** es una *receta*: un conjunto de instrucciones bien definidas, ordenadas y finitas que describen la secuencia de operaciones necesarias para resolver un problema específico o realizar una tarea.

Definición Formal

Un algoritmo es una secuencia finita de instrucciones precisas y no ambiguas, cada una ejecutable en tiempo y con esfuerzo finitos, diseñada para resolver una clase particular de problemas.

¿Qué es un Algoritmo?



Origen del Término

Deriva del nombre del matemático persa del siglo IX, **Muhammad ibn Musa al-Khwarizmi**, cuyo trabajo sentó las bases del álgebra. Su nombre latinizado era *Algoritmi*.

Componentes de un Algoritmo

Flujo Básico

Entrada → Proceso → Salida

Entrada (Input): Son los datos iniciales que se le proporcionan al algoritmo para que opere. Puede tener cero o más entradas.

Componentes de un Algoritmo

Flujo Básico

Entrada → Proceso → Salida

Entrada (Input): Son los datos iniciales que se le proporcionan al algoritmo para que opere. Puede tener cero o más entradas.

Proceso (Process): Es la secuencia de pasos, cálculos y operaciones que transforman los datos de entrada en una solución.

Componentes de un Algoritmo

Flujo Básico

Entrada → Proceso → Salida

Entrada (Input): Son los datos iniciales que se le proporcionan al algoritmo para que opere. Puede tener cero o más entradas.

Proceso (Process): Es la secuencia de pasos, cálculos y operaciones que transforman los datos de entrada en una solución.

Salida (Output): Es el resultado final que el algoritmo produce. Todo algoritmo debe tener al menos una salida.

Características Fundamentales de los Algoritmos

Criterios de Efectividad

Para que una secuencia de instrucciones sea considerada un algoritmo válido, debe cumplir con las siguientes características indispensables:

Características Fundamentales de los Algoritmos

Preciso: Cada paso debe ser definido de manera clara y sin ambigüedad. No debe haber lugar a interpretaciones.

Características Fundamentales de los Algoritmos

Preciso: Cada paso debe ser definido de manera clara y sin ambigüedad. No debe haber lugar a interpretaciones.

Definido: Si se ejecuta el algoritmo varias veces con la misma entrada, el resultado de salida debe ser **siempre** el mismo.

Características Fundamentales de los Algoritmos

Preciso: Cada paso debe ser definido de manera clara y sin ambigüedad. No debe haber lugar a interpretaciones.

Definido: Si se ejecuta el algoritmo varias veces con la misma entrada, el resultado de salida debe ser **siempre** el mismo.

Finito: Un algoritmo debe terminar después de un número finito de pasos. No puede entrar en un bucle infinito.

Características Fundamentales de los Algoritmos

Preciso: Cada paso debe ser definido de manera clara y sin ambigüedad. No debe haber lugar a interpretaciones.

Definido: Si se ejecuta el algoritmo varias veces con la misma entrada, el resultado de salida debe ser **siempre** el mismo.

Finito: Un algoritmo debe terminar después de un número finito de pasos. No puede entrar en un bucle infinito.

Legible: El algoritmo debe ser comprensible y su lógica debe poder ser entendida. Cada paso debe ser lo suficientemente básico como para poder ser realizado, en principio, con lápiz y papel.

Agenda

Herramientas de Diseño de Algoritmos

Del Problema al Código

Antes de escribir una sola línea de código, es fundamental plasmar la lógica de nuestra solución de una manera clara y estructurada. Las herramientas de diseño nos permiten concentrarnos en la lógica sin preocuparnos por la sintaxis de un lenguaje específico.

El Puente entre el Humano y la Máquina

Las dos herramientas más utilizadas en el desarrollo de software son:

- **Pseudocódigo:** Una descripción textual y estructurada.
- **Diagramas de Flujo:** Una representación gráfica del proceso.

Pseudocódigo: Hablando el Lenguaje de la Lógica

Definición

El **pseudocódigo** (del griego *pseudo*, 'falso') es una descripción de alto nivel, compacta e informal, del principio operativo de un algoritmo.

Características Principales

- Utiliza las convenciones estructurales de un lenguaje de programación, pero está diseñado para la **lectura humana**, no para la máquina.
- Su principal ventaja es la **flexibilidad**: no está atado a ninguna sintaxis estricta, permitiendo al programador expresar la lógica de forma clara y concisa.

Estructura y Palabras Clave Comunes

Convenciones Estructurales

Aunque no hay un estándar universal, se suelen adoptar ciertas palabras clave que imitan a los lenguajes de programación estructurados.

- **INICIO / FIN:** Delimitan el algoritmo.
- **LEER / OBTENER:** Para recibir datos de entrada.
- **ESCRIBIR / MOSTRAR:** Para presentar datos en la salida.
- **ASIGNAR / HACER** o $\leftarrow$: Para asignar un valor a una variable.

Estructuras de Control

Condicionales

IF (*una condición* is True)
THEN...(Haz esto)
ELSE...(Haz esto)
END_IF
Se utiliza para tomar decisiones y ejecutar diferentes bloques de código basados en si una condición es verdadera o falsa.

Estructura y Palabras Clave Comunes

Bucles Condicionales

WHILE (*una condición* is True)
DO SOMTHING...(Haz esto)
END_WHILE
Repite un bloque de código continuamente mientras una condición especificada siga siendo verdadera.

Estructura y Palabras Clave Comunes

Bucles Controlados por Contador

FOR...(inicio) **TO**...(fin)
DO SOMTHING...(Haz esto)
END_FOR
Repite un bloque de código un número predeterminado de veces.

Ejemplo 1: Área y Perímetro de un Rectángulo.

Descripción

Un algoritmo secuencial clásico. Se leen dos valores, se realizan dos cálculos distintos y se muestran los resultados. El flujo es lineal y directo.

Ejemplo 1:Área y Perímetro de un Rectángulo.

Algorithm 1 Área y Perímetro de un Rectángulo

1: **INICIO**
2: **LEER** L, B ▷ Largo y Ancho
3: **ASIGNAR** AREA $\leftarrow$ L * B
4: **ASIGNAR** PERIMETRO $\leftarrow$ 2 * (L + B)
5: **ESCRIBIR** AREA, PERIMETRO
6: **FIN**

Ejemplo 2: Calcular el Área de un Círculo

Descripción

Este es un algoritmo secuencial simple que sigue los pasos de entrada, proceso y salida sin ninguna bifurcación o bucle.

Ejemplo 2: Calcular el Área de un Círculo

Algorithm 2 Calcular el Área de un Círculo

1: **INICIO**
2: **DEFINIR** la constante PI ← 3.14159
3: **ESCRIBIR** 'Por favor, ingrese el radio del círculo:'
4: **LEER** radio
5: **ASIGNAR** area ← PI * (radio * radio)
6: **ESCRIBIR** 'El área del círculo es: ', area
7: **FIN**

Ejemplo 3: Sumar los primeros 100 números enteros.

Descripción

Aquí utilizamos un bucle `For` para realizar una tarea repetitiva un número fijo de veces. Introducimos el concepto de un *acumulador* (`suma`).

Ejemplo 3: Sumar los primeros 100 números enteros.

Algorithm 3 Sumar los Primeros 100 Enteros Positivos

```
1: INICIO
2: ASIGNAR suma ← 0
3: for contador ← 1 to 100 do
4:     ASIGNAR suma ← suma + contador
5: end for
6: ESCRIBIR 'La suma total es: ', suma
7: FIN
```

Ejemplo 4: Conversión de Grados Celsius a Fahrenheit.

Descripción

Un ejemplo práctico de un algoritmo secuencial que aplica una fórmula matemática para la conversión de unidades. Demuestra la importancia de usar tipos de datos adecuados (números reales o flotantes) para cálculos precisos.

Ejemplo 4: Conversión de Grados Celsius a Fahrenheit.

Algorithm 4 Convertir Celsius a Fahrenheit

1: **INICIO**
2: **ESCRIBIR** 'Ingrese la temperatura en grados Celsius:'
3: **LEER** gradosCelsius
4: **ASIGNAR** gradosFahrenheit $\leftarrow$ (gradosCelsius * 9/5) + 32
5: **ESCRIBIR** gradosCelsius, '°C equivalen a ', gradosFahrenheit, '°F.'
6: **FIN**

Ejemplo 5: Calcular el factorial de un número.

Descripción

El factorial de un entero no negativo n, denotado como $n!$, es el producto de todos los enteros positivos menores o iguales a n. Este algoritmo utiliza un bucle `FOR` cuyo rango depende de la entrada del usuario y un acumulador que multiplica en lugar de sumar.

Ejemplo 5: Calcular el factorial de un número.

Algorithm 5 Cálculo del Factorial

```
1: INICIO
2: ESCRIBIR 'Ingrese un número entero no negativo para calcular su factorial:'
3: LEER numero
4: if numero < 0 then
5:     ESCRIBIR 'Error: El factorial no está definido para números negativos.'
6: else
7:     ASIGNAR factorial ← 1
8:     for i ← 1 to numero do
9:         ASIGNAR factorial ← factorial * i
10:    end for
11:    ESCRIBIR 'El factorial de ', numero, ' es: ', factorial
12: end if
13: FIN
```

Ejemplo 6: Encontrar el mayor de tres números

Descripción

Este ejemplo es excelente para ilustrar el poder de las decisiones anidadas. Se comparan los números por pares para determinar cuál es el mayor de todos. El diagrama de flujo visualiza claramente las múltiples rutas que puede tomar la ejecución.

Ejemplo 6: Encontrar el mayor de tres números.

Algorithm 6 Encontrar el Mayor de Tres Números

```
1: INICIO
2: LEER num1, num2, num3
3: if num1 > num2 then
4:     if num1 > num3 then
5:         ESCRIBIR 'El mayor es: ', num1
6:     else
7:         ESCRIBIR 'El mayor es: ', num3
8:     end if
9: else
10:    if num2 > num3 then
11:        ESCRIBIR 'El mayor es: ', num2
12:    else
13:        ESCRIBIR 'El mayor es: ', num3
14:    end if
15: end if
16: FIN
```

Ejemplo 7: Encontrar el máximo de una lista de números

Descripción

Este algoritmo muestra cómo recorrer una colección de datos (una lista o arreglo) para encontrar un valor específico. Se inicializa una variable `maximo` con el primer elemento y luego se compara con los demás, actualizándola si se encuentra un valor mayor.

Ejemplo 7: Encontrar el máximo de una lista de números

Algorithm 7 Encontrar el Máximo en una Lista

```
INICIO
DEFINIR listaNumeros ← [15, 9, 27, 4, 12, 33, 8]
if la lista está vacía then
    ESCRIBIR 'La lista no puede estar vacía.'
else
    ASIGNAR maximo ← listaNumeros[0]          ▷ Asumimos el primero como máximo inicial
    for cada numero EN listaNumeros do
        if numero > maximo then
            ASIGNAR maximo ← numero
        end if
    end for
    ESCRIBIR 'El número más grande en la lista es: ', maximo
end if
FIN
```

Ejemplo 8: Validar una contraseña simple.

Descripción

Este algoritmo solicita al usuario una contraseña y utiliza un bucle `MIENTRAS` para asegurarse de que cumple con un requisito mínimo (en este caso, tener al menos 8 caracteres). Es un ejemplo fundamental de validación de entrada.

Ejemplo 8: Validar una contraseña simple.

Algorithm 8 Validación de Contraseña Simple

```
1: INICIO
2: ASIGNAR contrasenaValida ← FALSO
3: while NO contrasenaValida do
4:     ESCRIBIR 'Cree una contraseña (mínimo 8 caracteres):'
5:     LEER contrasena
6:     if LONGITUD(contrasena) >= 8 then
7:         ASIGNAR contrasenaValida ← VERDADERO
8:         ESCRIBIR 'Contraseña creada exitosamente.'
9:     else
10:        ESCRIBIR 'Error: La contraseña es demasiado corta. Inténtelo de nuevo.'
11:    end if
12: end while
13: FIN
```

Tarea: Calculadora de promedios y asignación de nota final.

Descripción

Este algoritmo es más interactivo y robusto. Permite al usuario ingresar un número indeterminado de calificaciones hasta que introduce un valor centinela (-1) para finalizar. Luego, calcula el promedio, se asegura de no dividir por cero si no se ingresaron calificaciones, y finalmente asigna una nota cualitativa (letra) basada en el promedio numérico. Este ejemplo combina bucles controlados por eventos, acumuladores, contadores y condicionales anidados.