

Universidad Distrital Francisco José de Caldas

Pseudocódigo y Pruebas de Escritorio

Materia:

Algoritmos y Programación en C++

Docente:

Karen de los Ríos

Alumno:

Jireth Soffia Yañez Gutiérrez

Matrícula: 20241135051

Bogotá, 18 de septiembre de 2025

Índice

1. Definiciones	2
1.1. Pseudocódigo	2
1.2. Prueba de escritorio	2
2. Análisis y comentario de ejemplos	2
2.1. Ejemplo 1: Determinar si un número es positivo o negativo	2
2.2. Ejemplo 2: Obtener el mayor de dos números distintos	2
2.3. Ejemplo 3: Calcular el factorial de un número	2
3. Conclusión	3
4. Referencias	3

1. Definiciones

1.1. Pseudocódigo

Un pseudocódigo es una forma de expresar, mediante texto estructurado, los pasos que se deben seguir para resolver un problema; en otras palabras, es la representación escrita de un algoritmo. Aunque no es un lenguaje de programación formal, utiliza una sintaxis propia basada en palabras reservadas y reglas simples que permiten describir con claridad las instrucciones. Gracias a ello, el pseudocódigo actúa como un puente entre el lenguaje cotidiano y los lenguajes de programación, facilitando la comprensión, el diseño y la posterior codificación de soluciones en un programa real.

1.2. Prueba de escritorio

Una prueba de escritorio es una forma de verificar un algoritmo antes de programarlo, simulando en papel cada paso que seguiría la computadora. Se hace con una tabla donde se anotan los valores de las variables y los resultados en cada iteración, lo que permite comprobar si la lógica es correcta y detectar posibles errores de manera sencilla.

2. Análisis y comentario de ejemplos

2.1. Ejemplo 1: Determinar si un número es positivo o negativo

Problema: consiste en identificar si un número real es positivo o negativo, siempre que no sea cero.

Comentario: Para resolverlo, el algoritmo pide un número al usuario y, mediante una condición, compara su valor: si es mayor que cero se clasifica como positivo, y si es menor que cero se clasifica como negativo. Este ejercicio es útil porque introduce la idea básica de las estructuras condicionales (si-entonces), además de mostrar la importancia de establecer restricciones claras, en este caso la de no aceptar el número cero. Gracias a la prueba de escritorio, se pueden simular distintos valores y confirmar que el algoritmo funciona correctamente en cada situación.

2.2. Ejemplo 2: Obtener el mayor de dos números distintos

Problema: plantea comparar dos números reales distintos para determinar cuál de ellos es mayor.

Comentario: El algoritmo primero solicita al usuario dos valores y luego utiliza una condición para compararlos: si el primer número es mayor, se muestra como resultado; en caso contrario, el segundo número será el mayor. La restricción de que los números no sean iguales es importante porque evita confusiones en la salida. Este ejemplo refuerza el uso de estructuras condicionales y demuestra, con la prueba de escritorio, que el algoritmo funciona correctamente con distintos tipos de valores, ya sean positivos, negativos o muy cercanos entre sí.

2.3. Ejemplo 3: Calcular el factorial de un número

Problema: consiste en calcular el factorial de un número entero no negativo, lo que significa multiplicar ese número por todos sus antecesores hasta llegar a 1.

Comentario: El algoritmo comienza pidiendo un valor al usuario y establece la restricción de no aceptar números negativos. Para resolverlo, se usan variables acumuladoras: una llamada contador, que va aumentando paso a paso, y otra llamada factorial, que guarda el resultado de las multiplicaciones sucesivas. Este ejemplo es más complejo porque introduce el concepto de iteración, es decir, repetir un proceso varias veces, además de mostrar cómo validar los datos de entrada y cómo se van actualizando las variables en cada paso, algo que puede comprobarse fácilmente con la prueba de escritorio.

3. Conclusión

En conclusión estudio del pseudocódigo y de las pruebas de escritorio permite comprender de manera ordenada y sencilla la lógica que existe detrás de los algoritmos. El pseudocódigo, al ser un lenguaje intermedio entre el lenguaje natural y los lenguajes de programación, facilita la expresión de las ideas sin preocuparse todavía por la sintaxis de un compilador. Por su parte, la prueba de escritorio constituye una herramienta esencial para validar la correcta ejecución de los pasos, ya que simula manualmente el comportamiento de las variables y ayuda a detectar errores lógicos antes de llevar el algoritmo a código real.

Los ejemplos analizados determinan la positividad de un número, obtención del mayor de dos valores y cálculo del factorial que muestran la importancia de las estructuras condicionales e iterativas, así como del control en los datos de entrada. Estos casos prácticos no solo refuerzan los conceptos teóricos, sino que también evidencian cómo las herramientas vistas son aplicables a problemas diversos, desde los más simples hasta los que requieren procesos repetitivos.

4. Referencias

Facultad de Ingeniería. (2025). *Manual de prácticas del Laboratorio de Fundamentos de Programación (MADO-17, versión 05)*. Universidad