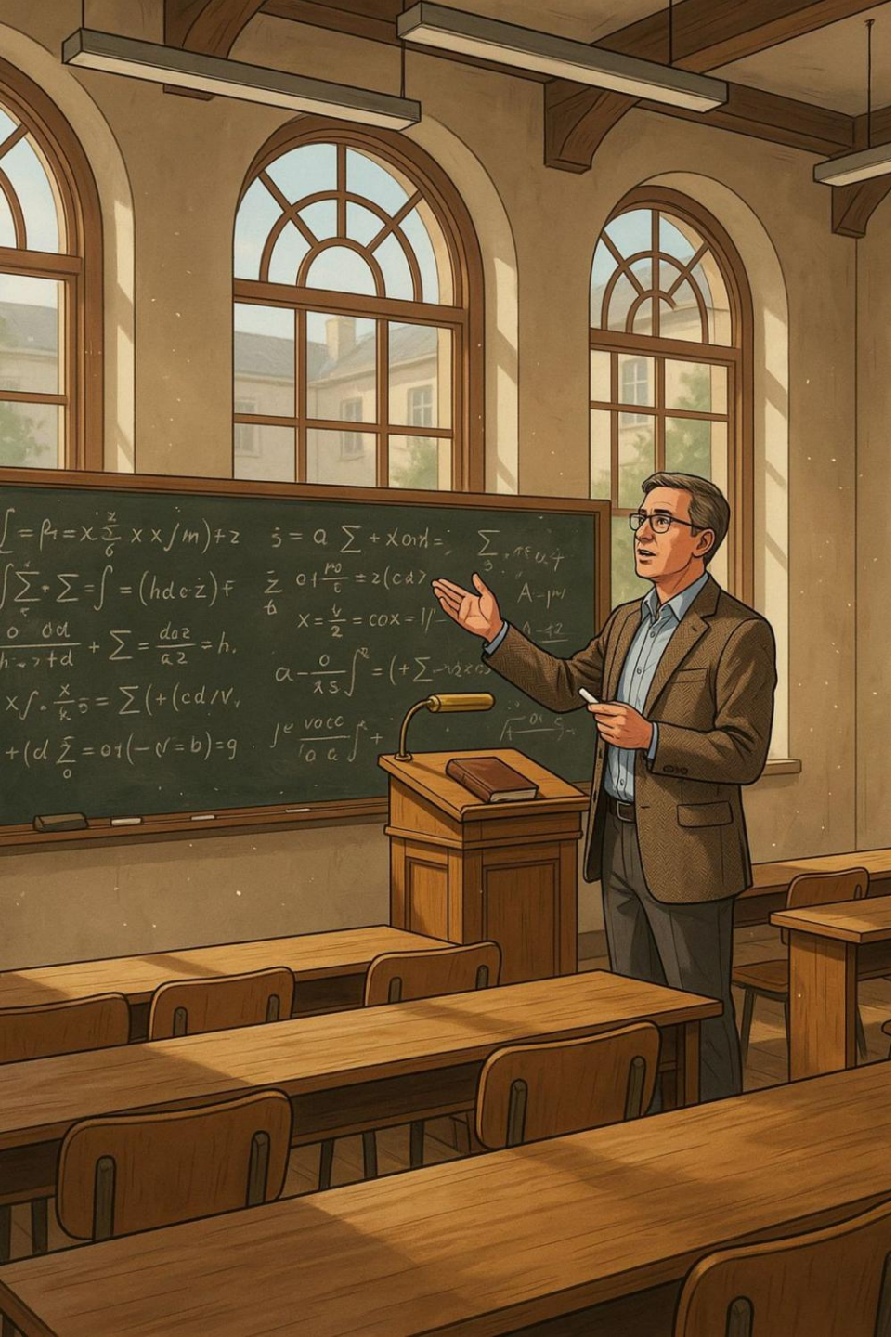


Correctitud Parcial y Correctitud Total

Fundamentos de Verificación

DESARROLLO FORMAL DE SOFTWARE

INGENIERÍA DE SISTEMAS · UNIVERSIDAD DEL PACÍFICO

by ykcaicedo

Conexión con lo aprendido

Especificación

La especificación establece condiciones previas y resultados esperados para un programa.

Terna de Hoare

La terna `{P} C {Q}` relaciona precondición, comando y postcondición.

Invariantes

Los invariantes de ciclo permiten razonar sobre la corrección de las iteraciones.

Nuevo enfoque

Ahora distinguimos entre **resultado correcto** y **terminación garantizada**.





DEFINICIÓN FORMAL

Correctitud parcial

Una tripla `{P} C {Q}` es **parcialmente correcta** si, cuando la precondición `P` se cumple *y* el programa termina, entonces la postcondición `Q` se cumple al finalizar.

> ⚠ La correctitud parcial **no garantiza** por sí sola que el programa termine. La terminación es una obligación adicional e independiente.

by ykcaicedo

Ejemplo sencillo

```
// Precondición: a y b son valores válidos
int resultado = a + b;
// Postcondición: resultado == a + b
```

Si la instrucción se ejecuta y termina como se espera, la postcondición se satisface de forma inmediata.

Precondición

`a` y `b` son valores enteros válidos.

Comando

`resultado = a + b`

Postcondición

`resultado == a + b`

by ykcaicedo

Ciclo que no termina

```
int i = 0;
while (i < 5) {
    System.out.println(i);
    // falta i++;
}
```

Este fragmento ilustra un caso de **no terminación**: el cuerpo del ciclo imprime el valor de `i`, pero nunca lo modifica. La condición `i < 5` permanece siempre verdadera.

⊗ Un programa que no termina **no puede satisfacer** la correctitud total, independientemente de que cada iteración individual sea correcta.

by ykcaicedo

Análisis del ciclo sin terminación

1 Sin modificación de la variable

El cuerpo imprime `i`, pero nunca ejecuta `i++`, por lo que la variable no cambia.

2 Condición siempre verdadera

Como `i` vale siempre 0, la guarda `i < 5` se evalúa como `true` en cada iteración.

3 Sin progreso hacia la salida

No existe una medida que disminuya. Podemos analizar cada iteración, pero **no podemos afirmar que el programa termina**.





DEFINICIÓN FORMAL

Correctitud total

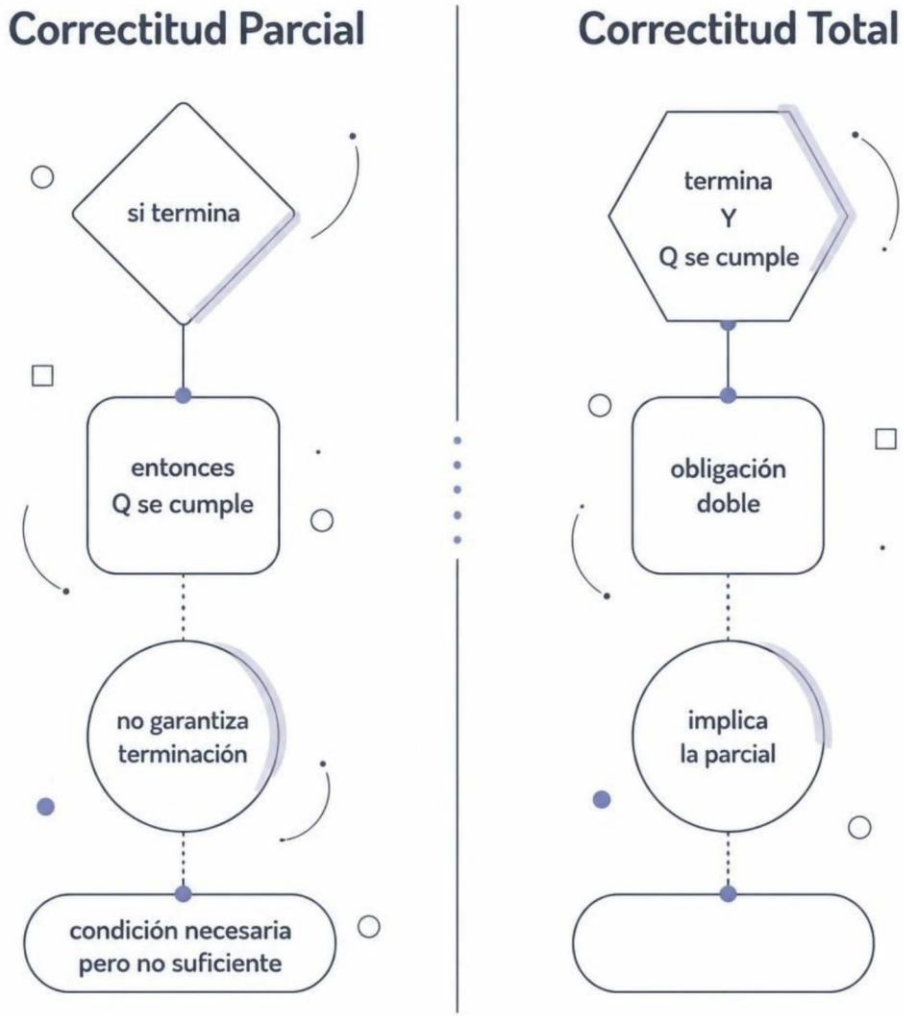
Una tripla `{P} C {Q}` es **totalmente correcta** si, para toda entrada que satisface `P`:

1. El programa **termina** en un número finito de pasos.
2. Al terminar, la postcondición `Q` se satisface.

La correctitud total integra la **correctitud parcial** y la **terminación** en una sola obligación.

by ykcaicedo

Comparación directa



Relación entre ambas

La correctitud total **implica** la correctitud parcial bajo las mismas premisas: todo programa totalmente correcto es también parcialmente correcto.

Sin embargo, la correctitud parcial **no implica** la total, porque no garantiza que el programa llegue a un final.

by ykcaicedo

Cómo justificar la terminación



Identificar la medida

Definir una expresión entera no negativa que represente el «trabajo pendiente».



Mostrar decremento

Demostrar que la medida **disminuye estrictamente** en cada iteración del ciclo.



Verificar la cota

Confirmar que la medida tiene una cota inferior (normalmente 0) y que el ciclo se detiene al alcanzarla.

No basta con decir que «una variable cambia». Debe demostrarse que la medida **siempre decrece** y que está **acotada inferiormente**.

Ejemplo: contar hacia abajo

```
int restante = n;

while (restante > 0) {
    restante--;
}
```

En cada iteración, `restante` se reduce en una unidad. Cuando alcanza 0, la guarda falla y el ciclo concluye.

1

Precondición

`n ≥ 0` y `restante = n` al inicio.

2

Medida de progreso

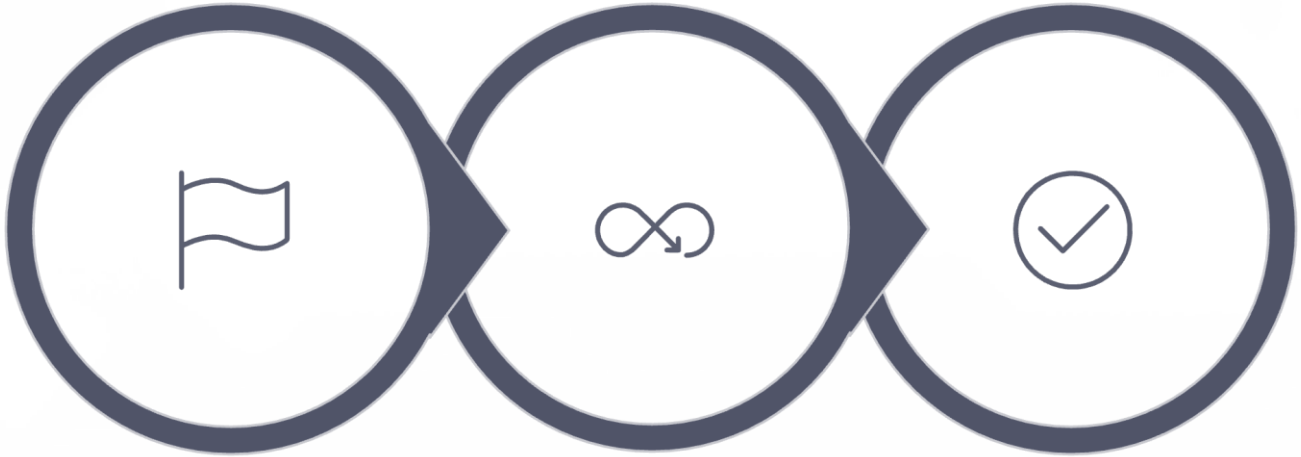
El valor de `restante`, que es entero no negativo.

3

Terminación

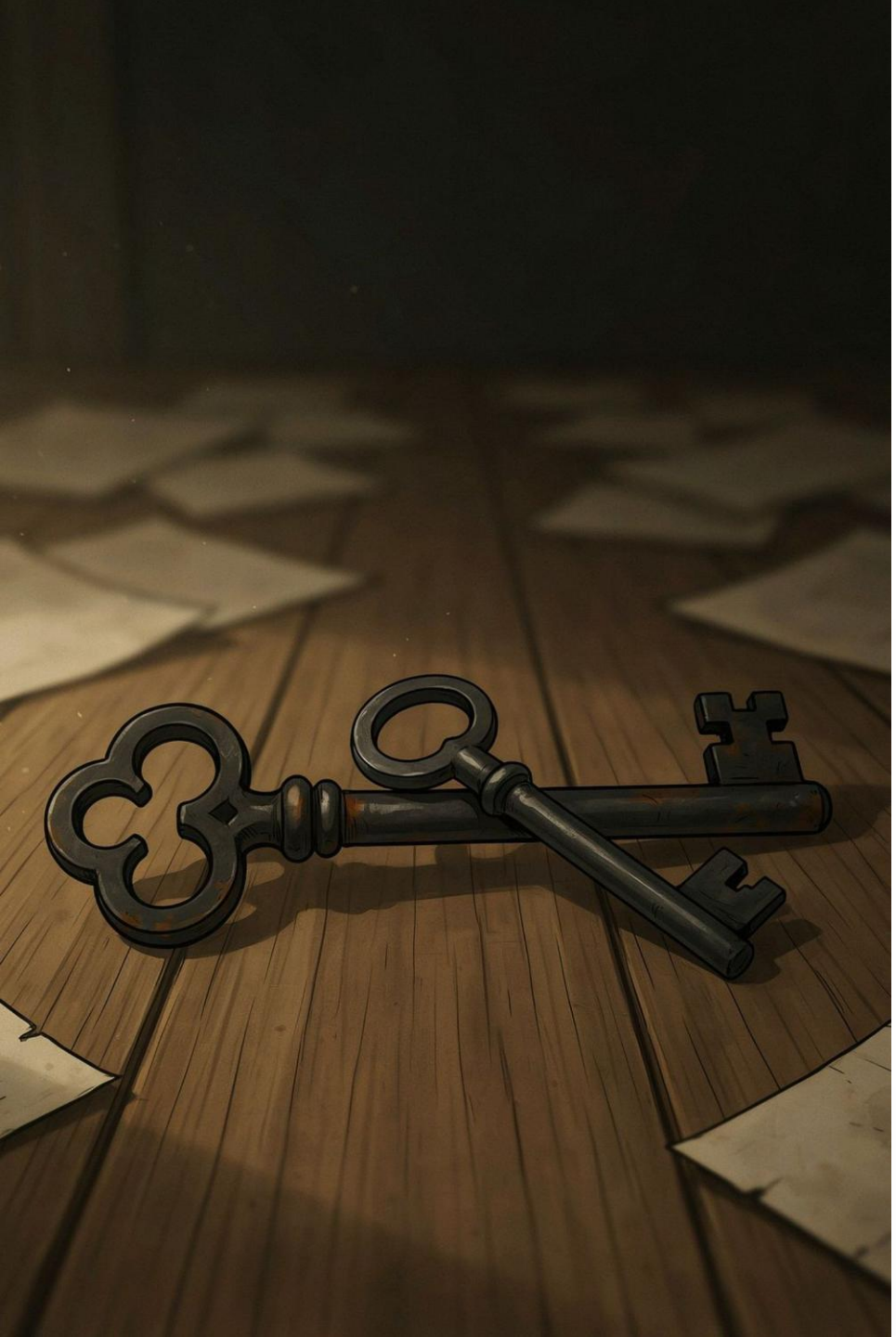
`restante` llega a 0 tras exactamente `n` iteraciones.

Análisis completo del ejemplo



Precondición **Iteración** **Término**

La medida `restante` es un entero no negativo que decrece estrictamente en cada iteración. Esto garantiza terminación en exactamente `n` pasos.



Invariante y terminación

Invariante de ciclo

Es una propiedad lógica que **se conserva** antes y después de cada iteración. Sirve para demostrar que el resultado será correcto si el ciclo termina.

Medida de progreso

Es una expresión numérica que **decrece** en cada iteración. Sirve para demostrar que el ciclo **terminará** en un número finito de pasos.

Son razonamientos relacionados pero distintos. La **correctitud total** exige ambos argumentos: el invariante justifica la corrección del resultado; la medida justifica la terminación.



Aplicación al proyecto

Correctitud parcial

Si el procesamiento de solicitudes termina, cada solicitud queda en un estado permitido por la especificación.

Correctitud total

Además, el procesamiento **termina** para toda entrada válida: ninguna solicitud puede dejar el sistema en espera indefinida.

Pregunta al grupo

¿Qué medida de progreso definirían para demostrar la terminación en vuestro sistema?

by ykcaicedo

Errores frecuentes

❌ Un caso exitoso no prueba la correctitud total

Que el programa funcione para una entrada concreta no implica que funcione para *todas* las entradas que cumplen la precondición.

❌ Correctitud parcial ≠ software incompleto

La correctitud parcial es una propiedad formal precisa, no un sinónimo de código inacabado o defectuoso.

❌ Tener condición de salida no garantiza terminación

Un ciclo puede tener guarda y aun así no terminar si el cuerpo no progresa hacia hacerla falsa.

❌ Ignorar precondiciones

Ambas nociones de correctitud aplican solo a entradas que satisfacen `P`. El comportamiento fuera de `P` no está especificado.

by ykcaicedo

Reto oral

Afirmación A

«Si termina, devuelve el total correcto.»

¿Se trata de correctitud **parcial** o **total**? ¿Qué queda sin garantizar?

Afirmación B

«Siempre termina y devuelve el total correcto para toda entrada válida.»

¿Se trata de correctitud **parcial** o **total**? ¿Qué prueba adicional se requirió?

Analicemos

¿Qué evidencia concreta falta para pasar de la afirmación **A** a la afirmación **B**?

Afirmación A

→ Correctitud **parcial**

Afirmación B

→ Correctitud **total**

Cierre

Correctitud

¿El resultado satisface la postcondición `Q` cuando el programa termina?

Terminación

¿El programa llega a un estado final en un número finito de pasos para toda entrada válida?

Correctitud total

Exige **ambas** obligaciones: resultado correcto *y* terminación garantizada bajo la precondición.

La distinción entre correctitud parcial y total es fundamental en el **Desarrollo Formal de Software**: ambas propiedades deben demostrarse, no asumirse.