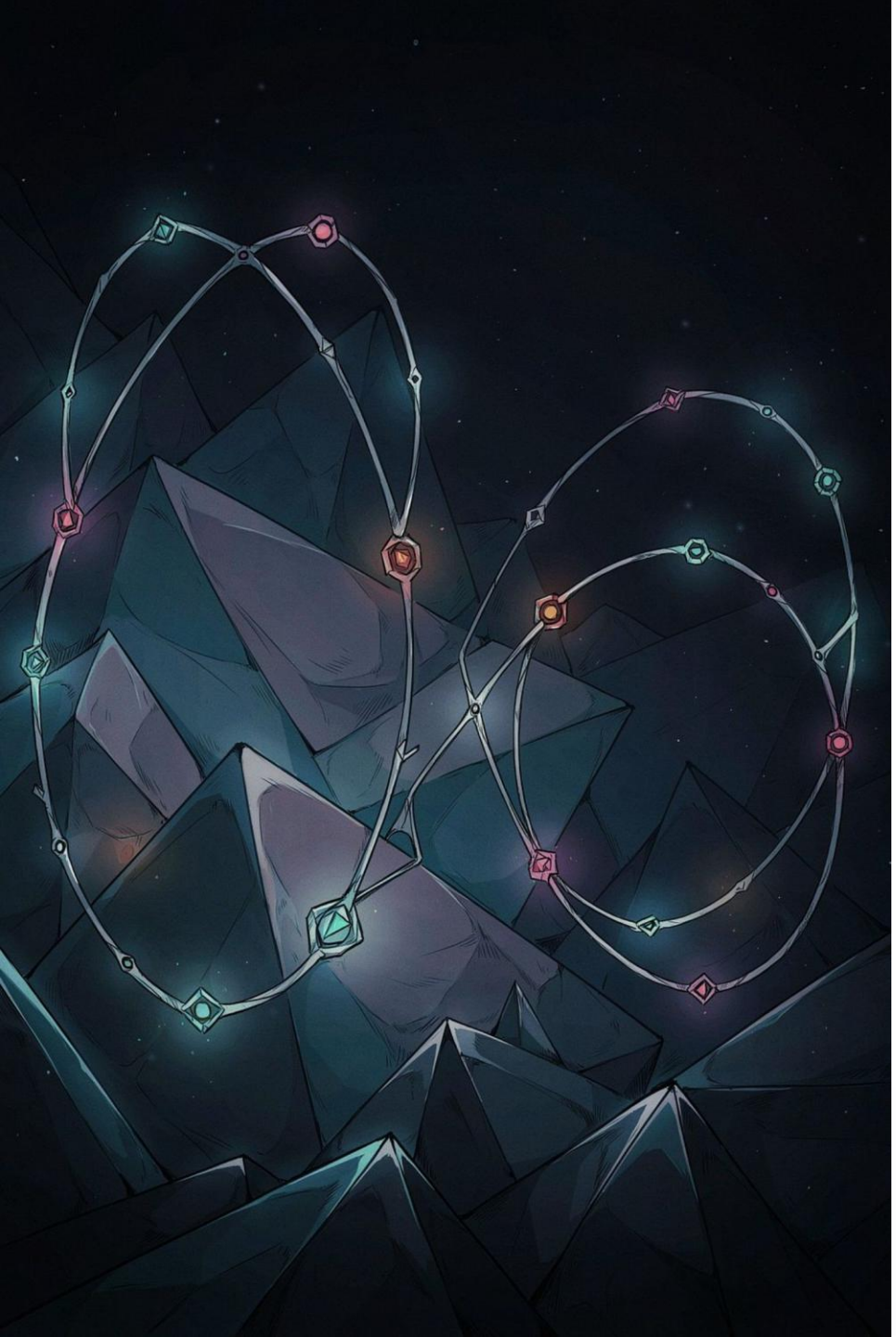


Verificación de Ciclos, Invariantes y Terminación

Fundamentos de Verificación

DESARROLLO FORMAL DE SOFTWARE

INGENIERÍA DE SISTEMAS · UNIVERSIDAD DEL PACÍFICO

by ykcaicedo

Conexión con clases anteriores

PRE y POST

Las precondiciones y postcondiciones describen el estado inicial y final esperado de un programa.

Cuerpo del ciclo

En un ciclo, el cuerpo puede ejecutarse múltiples veces; cada ejecución modifica el estado del programa.

Invariante

Una propiedad que se conserva a lo largo de todas las iteraciones, antes y después de cada una.

Terminación

No basta con que el ciclo cambie variables; hay que argumentar por qué la condición termina siendo falsa.



¿Qué se verifica en un ciclo?

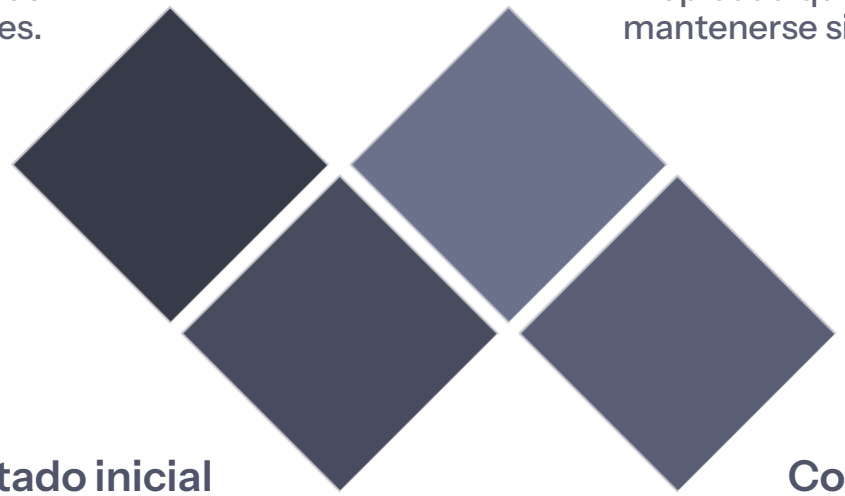
Verificar un ciclo implica responder cuatro preguntas fundamentales en orden:

Efecto por iteración

Describe cómo cambian las variables.

Invariante

Propiedad que debe mantenerse siempre.



Estado inicial

Define variables y condición inicial.

Condición de salida

Cuándo termina y cuál es el resultado.

Ejemplo Java: sumar del 1 al *n*

Un ciclo clásico que acumula la suma de los primeros *n* enteros positivos. Sirve de hilo conductor para ilustrar invariante y terminación.

```
int suma = 0;
int i = 1;

while (i <= n) {
 suma = suma + i;
 i++;
}
```

by ykcaicedo

Contrato del ejemplo

Precondición

`n >= 0`

Inicialización

`suma = 0`, `i = 1`

Condición de guarda

`i <= n`

Postcondición

`suma = 1 + 2 + … + n`

Pregunta clave: ¿qué representa `suma` tras exactamente *k* iteraciones? Responder esto conduce directamente al invariante.

by ykcaicedo

Traza para $n = 3$

Seguir la ejecución paso a paso confirma el comportamiento esperado y revela el patrón que el invariante debe capturar.

Paso a paso

Con n=3, suma toma los valores 1, 3 y 6; i termina en 4. Como 4<=3 es falso, se sale

by ykcaicedo



¿Qué es un invariante?

Propiedad persistente

Verdadera **antes de la primera iteración** y preservada tras cada una.

Puente hacia la postcondición

Debe ser lo suficientemente preciso para derivar la postcondición cuando la condición del `while` es falsa.

No es la condición del ciclo

La condición del `while` controla la ejecución; el invariante expresa una *relación entre variables*.

by ykcaicedo

Invariante para la suma

El invariante elegido es:

> `suma = 1 + 2 + … + (i − 1)` con `i ≥ 1`

- **Al inicio:** `i=1`, `suma=0`: la suma de cero términos es 0. ✓
- **Mantenimiento:** el cuerpo añade `i` a `suma` y luego incrementa `i`, preservando la relación.
- **Al salir:** `i = n+1`, luego `suma = 1+…+n`. ✓



by ykcaicedo

Las tres obligaciones de prueba

Para demostrar formalmente que un invariante es correcto deben satisfacerse tres condiciones:

Inicialización

El invariante se cumple **antes de la primera iteración**, es decir, tras ejecutar el código de inicialización.

Mantenimiento

Si el invariante es verdadero al entrar al cuerpo, sigue siendo verdadero **al terminar** esa iteración.

Salida

Invariante + condición falsa permiten derivar la **postcondición** deseada del ciclo.

by ykcaicedo

Terminación

¿Cuándo termina un ciclo?

Cuando su condición de guarda se vuelve **falsa**. Esto no ocurre automáticamente por el hecho de que el cuerpo ejecute instrucciones.

Razón de progreso

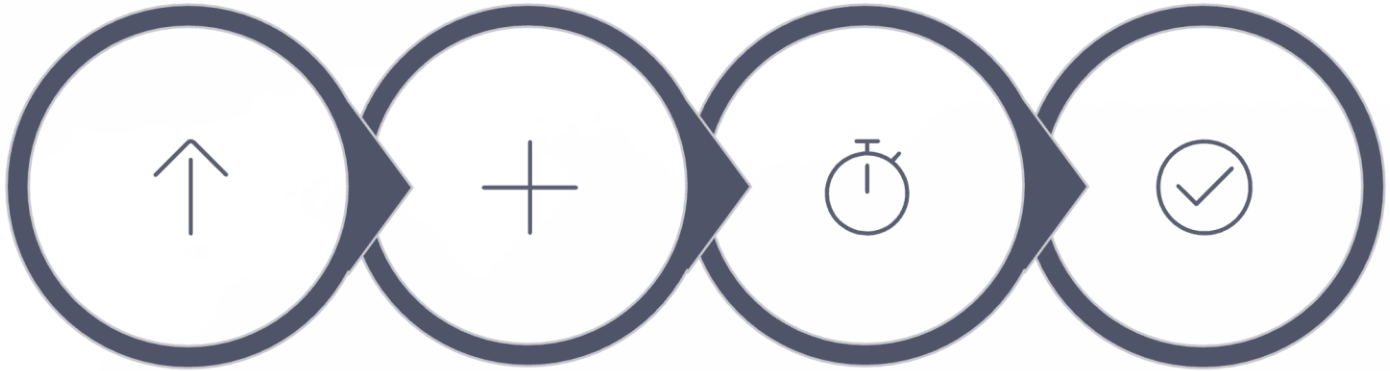
Debe existir una magnitud que **disminuya o aumente monótonamente** en cada iteración y esté acotada, garantizando que el ciclo finaliza.

Cambio $\neq$ terminación

Una variable que cambia en cada iteración **no garantiza por sí sola** que el ciclo se detiene. Es necesario demostrar la cota.



Argumento de terminación para el ejemplo



Progreso | Incremento | Límite | Fallo de condición

¿Qué ocurre sin `i++`?

Si se omite el incremento, `i` permanece en 1 para siempre. La condición `1 <= n` es permanentemente verdadera (para `n ≥ 1`) y el programa **no termina nunca**: ciclo infinito.

⊗ Omitir la actualización de la variable de control es el error más común que provoca no terminación.

by ykcaicedo

Errores frecuentes

Confundir condición e invariante

La condición del `while` controla cuándo ejecutar; el invariante expresa qué es verdad durante la ejecución.

Invariante falso al inicio

El invariante propuesto no se cumple antes de la primera iteración, invalidando toda la demostración.

No justificar mantenimiento

Afirmar que el cuerpo "preserva" el invariante sin demostrarlo paso a paso no constituye prueba.

Terminación sin argumento

Decir "el ciclo termina" sin identificar la variable de progreso y su cota es una justificación incompleta.

Salida desconectada de POST

No enlazar explícitamente invariante + condición falsa con la postcondición deja el argumento sin cerrar.

by ykcaicedo

Aplicación a sistemas de reservas



Invariante natural

Al procesar solicitudes de reserva en orden, el invariante puede enunciarse como: *"todas las solicitudes con índice menor que*

by ykcaicedo



Reto oral

1

Iteraciones

Para `i=1; i<=4; i++`, ¿cuántas iteraciones se ejecutan? Razona sin trazar el código.

2

Sin incremento

Si se elimina `i++`, ¿qué ocurre con el ciclo? Justifica usando el argumento de terminación.

3

¿Es invariante?

¿Es `i <= 4` el invariante del ciclo? Razona por qué sí o por qué no.

Las respuestas se dan oralmente o por chat durante la sesión. No se entrega ningún documento.

by ykcaicedo

Cierre

1

Invariante conservado

Demostrar que una propiedad se mantiene en cada iteración es el núcleo de la verificación de ciclos.

2

Argumento de progreso

La terminación exige identificar una magnitud acotada que avanza monotónamente hacia la condición de salida.

3

Próxima sesión

Estas ideas se formalizan en los conceptos de **correctitud parcial y total**: el siguiente paso en el desarrollo formal de software.