EXPLICADO VISUALMENTE

Algoritmos explicados con claridad

Un recorrido visual desde las estructuras de datos hasta la IA generativa



CHORAL SOFTWARE

Derechos de autor

Copyright © 2026 Choral Software. Todos los derechos reservados. Ninguna parte de esta publicación puede reproducirse, almacenarse, o transmitirse de ninguna forma ni por ningún medio sin permiso previo por escrito, excepto breves citas utilizadas en reseñas o trabajos académicos.

Aviso educativo

Este libro explica los algoritmos y la inteligencia artificial con fines de educación general. No ofrece asesoría médica, legal, financiera, de seguridad, ni ningún otro tipo de asesoría profesional. Los ejemplos están simplificados para facilitar la comprensión; los sistemas en producción requieren pruebas, experiencia en el dominio, salvaguardas de privacidad, revisión de seguridad, y monitoreo continuo.

Edición 1.1 • Julio de 2026

Nota sobre esta edición

Esta edición corrige el pseudocódigo ilustrativo en todo el libro. En la primera impresión, la mayoría de los bloques de pseudocódigo eran texto genérico en lugar de la lógica real del algoritmo nombrado. Cada bloque de pseudocódigo ha sido reescrito con lógica correcta, similar a código real, para su concepto (o, para un pequeño número de secciones puramente conceptuales, con un ejemplo concreto ilustrativo). Algunas consignas de reflexión de la parte V que mencionaban la «memoria» y el «crecimiento» de una manera que no encajaba con temas no algorítmicos también han sido reformuladas para ajustarse a su tema.

Cómo leer este libro

Cómo leer este libro. No necesita saber programación ni matemáticas avanzadas. Comience con las imágenes y las analogías. Lea la explicación en lenguaje sencillo. Luego siga el pseudocódigo como si fueran indicaciones escénicas. El pseudocódigo se parece deliberadamente a Python, pero está diseñado para leerse en lugar de ejecutarse. Cada concepto ocupa una doble página de aprendizaje. La primera página construye el modelo mental; la segunda convierte la idea en un laboratorio visual. El color es funcional: el azul marca la estructura, el turquesa marca la transformación, el dorado marca la elección o la prioridad, y el coral marca el riesgo o el error. Una regla de lectura útil: no se pregunte «¿Entiendo cada línea?». Pregúntese «¿Puedo explicar qué información conserva el método, qué acción repetida realiza, y por qué se detiene?». Una nota sobre la IA: la parte dedicada a la IA es deliberadamente extensa. Comienza con los datos y la evaluación, atraviesa el aprendizaje automático clásico y las redes neuronales, y luego desarrolla la visión, el lenguaje, los transformadores, la IA generativa, la recuperación, las recomendaciones, la voz, el aprendizaje por refuerzo, los agentes, y el despliegue responsable. Los sistemas cotidianos se tratan como flujos de procesamiento en lugar de modelos únicos misteriosos.

Índice

PARTE I: Pensar en estructuras

1. Los algoritmos sin intimidación
2. Datos: valores, registros y referencias
3. Arreglos y listas dinámicas
4. Listas enlazadas
5. Pilas, colas y deques
6. Tablas hash, conjuntos y mapas

PARTE II: Árboles, grafos, y la forma de las relaciones

7. Árboles y jerarquías
8. Árboles binarios de búsqueda y equilibrio
9. Montículos y colas de prioridad
10. Grafos: redes por todas partes
11. Recorriendo grafos

PARTE III: Algoritmos de propósito general

12. La búsqueda
13. La ordenación elemental
14. La ordenación rápida
15. Dividir, combinar, y reutilizar
16. Algoritmos voraces y planificación
17. Caminos más cortos y árboles de expansión

PARTE IV: Cómo aprenden las máquinas

18. IA, aprendizaje automático, y modelos
19. Preparación de datos y evaluación
20. Regresión: predecir cantidades
21. Clasificación: elegir una categoría
22. Árboles de decisión y conjuntos
23. Vecinos, agrupamientos, y similitud
24. Probabilidad, actualización bayesiana, e incertidumbre
25. Las redes neuronales por dentro
26. Entrenar redes profundas
27. La visión por computadora
28. El lenguaje como dato
29. La atención y los transformadores
30. Los grandes modelos de lenguaje y la IA generativa
31. Recuperación, búsqueda, y RAG
32. Sistemas de recomendación
33. IA de voz y audio
34. El aprendizaje por refuerzo
35. Agentes de IA, herramientas, y planificación

PARTE V: La IA en la vida cotidiana — y cómo pensar con claridad al respecto

36. Mapas, navegación, y entregas
37. Feeds, anuncios, y compras en línea
38. Spam, fraude, moderación, y seguridad
39. Equidad, privacidad, e IA responsable
40. Usar la IA con sabiduría

PARTE I

Pensar en estructuras

Antes de que un algoritmo pueda actuar, la información necesita una forma.

CAPÍTULO 1

Los algoritmos sin intimidación

En este capítulo

1.1 Una receta para decidir. Un algoritmo es una secuencia finita e inequívoca de decisiones y acciones.

1.2 Entradas, salidas y estado. Las entradas son lo que el procedimiento recibe; las salidas son lo que produce; el estado es lo que cambia en el camino.

1.3 La corrección antes que el ingenio. Un procedimiento útil debe resolver el problema prometido para toda entrada permitida, no solo para un ejemplo favorable.

1.4 La eficiencia como historia de crecimiento. La eficiencia pregunta cómo crece el trabajo a medida que crece la entrada, sin requerir matemáticas avanzadas.

Brújula del capítulo

Al leer Los algoritmos sin intimidación, compare las cuatro ideas de este capítulo: Una receta para decidir, Entradas, salidas y estado, La corrección antes que el ingenio, La eficiencia como historia de crecimiento. Trate cada bloque de pseudocódigo como un mapa de decisiones, no como una prueba de programación.

1.1 Una receta para decidir

Una receta para decidir

Un algoritmo es una secuencia finita e inequívoca de decisiones y acciones.



Figura 1.1. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Un punto de partida intuitivo. Empiece por seguir una receta de desayuno. En esa escena, note qué puede moverse, qué actúa como límite, y qué cuenta como éxito. Esas mismas preguntas revelan la idea central de una receta para decidir: Un algoritmo es una secuencia finita e inequívoca de decisiones y acciones.

El punto de vista de la máquina. Una computadora no recibe ni la metáfora ni su sentido común. Recibe una descripción estructurada elegida para esta tarea. En una receta para decidir, esa descripción debe sostener la operación implícita en la definición; en convertir una meta poco clara en pasos pequeños y verificables, también debe reflejar unidades, valores ausentes, y el significado de cada conexión.

El invariante que hay que vigilar. Durante una receta para decidir, algún hecho debe permanecer fiable incluso mientras el resto de la información cambia. Ese hecho puede ser una región ordenada, un conjunto de visitados, un mejor puntaje conocido, o una frontera alrededor del trabajo no resuelto. Nombrarlo hace que tanto el procedimiento como su condición de parada sean más fáciles de confiar.

POR DENTRO | Una receta para decidir

Dentro de un sistema real. Una receta para decidir contribuye a convertir una meta poco clara en pasos pequeños y verificables, pero es solo una etapa. Un uso desplegado de una receta para decidir también recopila y verifica entradas, aplica políticas, gestiona fallos, presenta resultados, y observa lo que ocurre después. Las etapas circundantes suelen determinar si un algoritmo correcto se convierte en una experiencia útil.

Qué merece sospecha. Un ejemplo ordenado puede ocultar costo, ambigüedad, o una suposición frágil. Para una receta para decidir, inspeccione la entrada de peor forma y cualquier empate o condición límite. En convertir una meta poco clara en pasos pequeños y verificables, pregúntese también qué errores son más consecuentes y qué camino de recuperación existe.

Traducción cotidiana

Piense en seguir una receta de desayuno. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
algorithm make_tea(water, tea_bag):
heat water until boiling
place tea_bag in cup
pour in water
wait 3 minutes
remove tea_bag
return cup
```

LABORATORIO VISUAL | Una receta para decidir

1. Traducir la analogía

Dibuje seguir una receta de desayuno. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de una receta para decidir. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Imagine el ejemplo de seguir una receta de desayuno creciendo hasta que ya no quepa en memoria. Decida qué podría procesarse por partes.

4. Cuestionar una suposición

Cree un contraejemplo que haga que un uso ingenuo de una receta para decidir sea lento, ambiguo, o engañoso.

5. Localizar las decisiones humanas

En un sistema que involucra convertir una meta poco clara en pasos pequeños y verificables, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Una receta para decidir así: Un algoritmo es una secuencia finita e inequívoca de decisiones y acciones.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace una receta para decidir?
2. ¿Qué detalle en «seguir una receta de desayuno» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

1.2 Entradas, salidas y estado

Entradas, salidas y estado

Las entradas son lo que el procedimiento recibe; las salidas son lo que produce; el estado es lo que cambia en el camino.

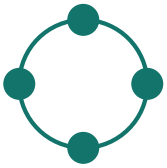


Figura 1.2. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Ver la regla antes que la notación. La imagen de una máquina expendedora que recuerda su crédito le da a entradas, salidas y estado una forma física. Las entradas son lo que el procedimiento recibe; las salidas son lo que produce; el estado es lo que cambia en el camino. Conservaremos la estructura útil de la comparación mientras descartamos los detalles que no pertenecen al algoritmo.

Elegir qué almacenar. Una implementación de entradas, salidas y estado trabaja con un modelo deliberado de la situación. El mejor modelo no es el más detallado; es la representación más simple que preserva las decisiones que el método debe tomar. Esto es especialmente importante en separar lo que entra, lo que se recuerda, y lo que sale, donde un detalle irrelevante puede añadir costo o ruido.

Seguir un recorrido completo. Parta de un estado inicial válido, aplique la acción que define entradas, salidas y estado, y registre lo que esa acción establece. El siguiente paso debe usar ese nuevo conocimiento en lugar de reiniciar a ciegas. El progreso se vuelve visible cuando la porción no resuelta se reduce, la frontera avanza, o la estimación mejora.

POR DENTRO | Entradas, salidas y estado

Por qué esta idea se traslada bien. La misma promesa estructural hace que entradas, salidas y estado sea útil en separar lo que entra, lo que se recuerda, y lo que sale. Los nombres de los objetos pueden cambiar, pero la operación sigue siendo reconocible. Esta transferencia —de una historia superficial a otra— es señal de que hemos aprendido el algoritmo en lugar de memorizar un ejemplo.

Compensaciones, no magia. Entradas, salidas y estado gana algo —velocidad, estructura, adaptabilidad, o una estimación útil— apoyándose en supuestos particulares. Cuando esos supuestos fallan, el resultado puede volverse lento o engañoso en separar lo que entra, lo que se recuerda, y lo que sale. Una implementación robusta detecta los casos no soportados en lugar de inventar confianza en silencio.

Traducción cotidiana

Piense en una máquina expendedora que recuerda su crédito. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function running_total(readings):
    total = 0
    for value in readings:
        total = total + value
    return total
# input: readings (what the procedure receives)
# state: total (what changes along the way)
# output: the final total (what it produces)
```

LABORATORIO VISUAL | Entradas, salidas y estado

1. Traducir la analogía

Dibuje una máquina expendedora que recuerda su crédito. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de entradas, salidas y estado. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Contraste un caso de entradas, salidas y estado casi resuelto con uno deliberadamente difícil. Explique por qué su comportamiento de ejecución puede diferir.

4. Cuestionar una suposición

Pruebe entradas, salidas y estado con un caso vacío, un caso único, empates, y un valor malformado. Escriba la respuesta más segura para cada uno.

5. Localizar las decisiones humanas

En un sistema que involucra separar lo que entra, lo que se recuerda, y lo que sale, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Entradas, salidas y estado así: Las entradas son lo que el procedimiento recibe; las salidas son lo que produce; el estado es lo que cambia en el camino.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace entradas, salidas y estado?
2. ¿Qué detalle en «una máquina expendedora que recuerda su crédito» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

1.3 La corrección antes que el ingenio

La corrección antes que el ingenio

Un procedimiento útil debe resolver el problema prometido para toda entrada permitida, no solo para un ejemplo favorable.

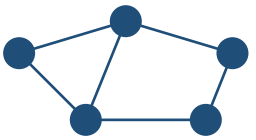


Figura 1.3. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Una vía concreta de entrada. Piense en probar indicaciones en una carretera cerrada. Ofrece una ruta memorable hacia esta definición: Un procedimiento útil debe resolver el problema prometido para toda entrada permitida, no solo para un ejemplo favorable. Una vez que esa promesa central está clara, el pseudocódigo se convierte en la descripción de acciones familiares en lugar de un muro de símbolos.

Representar este problema. Para la corrección antes que el ingenio, la entrada debe conservar las distinciones necesarias para aplicar la regla. Según la aplicación, eso puede significar posiciones ordenadas, vecinos enlazados, características medidas, etiquetas, o conexiones ponderadas. La representación usada para escribir ejemplos, casos límite, y una condición de parada debe omitir el detalle decorativo pero conservar cada hecho que pudiera cambiar la respuesta.

Progreso y parada. Cada paso de la corrección antes que el ingenio debe acercar el proceso a una condición verificable. Un rastro útil indica el estado actual, la decisión recién tomada, y el trabajo restante. Si nada de eso cambia, el procedimiento puede estar atascado, o su regla de parada puede ser incompleta.

POR DENTRO | La corrección antes que el ingenio

Del ejemplo de clase al producto. En escribir ejemplos, casos límite, y una condición de parada, los ingenieros combinan la corrección antes que el ingenio con almacenamiento, validación, monitoreo, y decisiones de interfaz. La escala puede exigir aproximación o procesamiento por lotes, mientras que las consecuencias humanas pueden exigir explicaciones, anulaciones, y umbrales prudentes.

Una prueba de calidad práctica. Varíe el tamaño, el orden, y la limpieza de la entrada. Compruebe no solo si la corrección antes que el ingenio devuelve una respuesta, sino si esa respuesta sigue siendo significativa para escribir ejemplos, casos límite, y una condición de parada. Mida la memoria, el retraso, y distintos tipos de error en lugar de reducir la calidad a un único conteo de éxito.

Traducción cotidiana

Piense en probar indicaciones en una carretera cerrada. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function find_maximum(numbers):
    if length(numbers) == 0:
        report an unsupported case
    best = numbers[0]
    for value in numbers[1:]:
        if value > best:
            best = value
    return best
# check: works for one item, ties, negative numbers, and rejects empty input
# instead of only the friendly example you first imagined
```

LABORATORIO VISUAL | La corrección antes que el ingenio

1. Traducir la analogía

Dibuje probar indicaciones en una carretera cerrada. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de la corrección antes que el ingenio. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Compare un ejemplo pequeño de la corrección antes que el ingenio con uno un millón de veces más grande. Marque qué acción se convierte en el cuello de botella.

4. Cuestionar una suposición

Elimine un supuesto de la definición de la corrección antes que el ingenio —orden, conectividad, costo no negativo, o datos representativos— y prediga el fallo.

5. Localizar las decisiones humanas

En un sistema que involucra escribir ejemplos, casos límite, y una condición de parada, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender La corrección antes que el ingenio así: Un procedimiento útil debe resolver el problema prometido para toda entrada permitida, no solo para un ejemplo favorable.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace la corrección antes que el ingenio?
2. ¿Qué detalle en «probar indicaciones en una carretera cerrada» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

1.4 La eficiencia como historia de crecimiento

La eficiencia como historia de crecimiento

La eficiencia pregunta cómo crece el trabajo a medida que crece la entrada, sin requerir matemáticas avanzadas.



Figura 1.4. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Construir la imagen mental. Imagine atender a diez invitados frente a diez mil. La eficiencia pregunta cómo crece el trabajo a medida que crece la entrada, sin requerir matemáticas avanzadas. La comparación resalta el papel del orden y del cambio sin pretender que el mecanismo real es literalmente la escena cotidiana.

Del mundo real a los datos. La escena «atender a diez invitados frente a diez mil» tiene más detalle del que un programa necesita. Un diseñador selecciona las partes relevantes para la eficiencia como historia de crecimiento y las codifica como valores y relaciones. Si una distinción crucial se pierde durante esa traducción, ningún ingenio posterior podrá reconstruirla de forma fiable.

Rastrear el mecanismo. Recorra un ejemplo pequeño de la eficiencia como historia de crecimiento y deténgase después de cada operación. Pregúntese qué promesa de la definición es ahora cierta y qué casos siguen siendo posibles. Este rastreo disciplinado es la forma más segura de descubrir casos omitidos, actualizaciones incorrectas, y bucles que nunca terminan.

POR DENTRO | La eficiencia como historia de crecimiento

Dónde podría encontrarlo. Las aplicaciones que involucran comparar barridos repetidos con divide y vencerás usan esta idea porque la información subyacente tiene la forma descrita arriba. Las versiones en producción también deben lidiar con el retraso, mediciones imperfectas, comportamiento cambiante, y objetivos que compiten entre sí.

Límites y modos de fallo. Pruebe la eficiencia como historia de crecimiento sin elementos, con un elemento, con elementos iguales, con una entrada malformada, y con un caso mucho más grande de lo esperado. Luego pregúntese si la representación misma es justa y fiel. Los pasos correctos no pueden compensar datos que describen el problema equivocado.

Traducción cotidiana

Piense en atender a diez invitados frente a diez mil. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function contains_slow(items, target):     # checks every item
    for value in items:
        if value == target:
            return true
    return false                            # work grows with length(items)


function contains_fast(sorted_items, target):  # binary search
    low, high = 0, length(sorted_items) - 1
    while low <= high:
        mid = (low + high) // 2
        if sorted_items[mid] == target:
            return true
        if sorted_items[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return false                            # work grows with log(length(items))
```

LABORATORIO VISUAL | La eficiencia como historia de crecimiento

1. Traducir la analogía

Dibuje atender a diez invitados frente a diez mil. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de la eficiencia como historia de crecimiento. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Duplique la cantidad de información usada por la eficiencia como historia de crecimiento. Prediga qué trabajo se duplica, cuál podría crecer más rápido, y cuál permanece fijo.

4. Cuestionar una suposición

Introduzca información faltante en el ejemplo de atender a diez invitados frente a diez mil. Decida si rechazarla, repararla, ignorarla, o representarla explícitamente.

5. Localizar las decisiones humanas

En un sistema que involucra comparar barridos repetidos con divide y vencerás, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender La eficiencia como historia de crecimiento así: La eficiencia pregunta cómo crece el trabajo a medida que crece la entrada, sin requerir matemáticas avanzadas.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace la eficiencia como historia de crecimiento?
2. ¿Qué detalle en «atender a diez invitados frente a diez mil» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

Capítulo 1 Repaso

Consignas de explicación

Enseñe las ideas centrales de los algoritmos sin intimidación a alguien que nunca haya programado. Use las analogías del capítulo, pero también nombre la información exacta almacenada por una receta para decidir y contrástela con la eficiencia como historia de crecimiento. Termine explicando cómo cada proceso sabe cuándo detenerse.

Mapa de conexiones

Concepto	Se conecta con	Pregúntese
Una receta para decidir	convertir una meta poco clara en pasos pequeños y verificables	¿Qué cambia a medida que crece la entrada?
Entradas, salidas y estado	separar lo que entra, lo que se recuerda, y lo que sale	¿Qué cambia a medida que crece la entrada?
La corrección antes que el ingenio	escribir ejemplos, casos límite, y una condición de parada	¿Qué cambia a medida que crece la entrada?
La eficiencia como historia de crecimiento	comparar barridos repetidos con divide y vencerás	¿Qué cambia a medida que crece la entrada?

Pruébelo en papel

Elija cinco objetos cotidianos que encajen con la idea de una receta para decidir. Dibuje la representación, realice una operación a mano, y anote solo la información que cambió. Luego explique por qué la información sin cambios sigue satisfaciendo la promesa central del capítulo.

CAPÍTULO 2

Datos: valores, registros y referencias

En este capítulo

2.1 Valores y tipos. Un tipo describe qué significa un valor y qué operaciones tienen sentido para él.

2.2 Registros. Un registro mantiene varios campos relacionados juntos para que viajen como una sola unidad significativa.

2.3 Referencias. Una referencia es una ruta hacia otro valor en lugar de una copia de ese valor.

2.4 Identidad e igualdad. Dos registros pueden contener información igual y aun así ser objetos distintos.

Brújula del capítulo

Al leer Datos: valores, registros y referencias, compare las cuatro ideas de este capítulo: Valores y tipos, Registros, Referencias, Identidad e igualdad. Trate cada bloque de pseudocódigo como un mapa de decisiones, no como una prueba de programación.

2.1 Valores y tipos

Valores y tipos

Un tipo describe qué significa un valor y qué operaciones tienen sentido para él.



Figura 2.1. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Un punto de partida intuitivo. Empiece por no sumar una dirección postal a una temperatura. En esa escena, note qué puede moverse, qué actúa como límite, y qué cuenta como éxito. Esas mismas preguntas revelan la idea central de valores y tipos: Un tipo describe qué significa un valor y qué operaciones tienen sentido para él.

El punto de vista de la máquina. Una computadora no recibe ni la metáfora ni su sentido común. Recibe una descripción estructurada elegida para esta tarea. En valores y tipos, esa descripción debe sostener la operación implícita en la definición; en distinguir etiquetas, medidas, valores de verdad, y colecciones, también debe reflejar unidades, valores ausentes, y el significado de cada conexión.

El invariante que hay que vigilar. Durante valores y tipos, algún hecho debe permanecer fiable incluso mientras el resto de la información cambia. Ese hecho puede ser una región ordenada, un conjunto de visitados, un mejor puntaje conocido, o una frontera alrededor del trabajo no resuelto. Nombrarlo hace que tanto el procedimiento como su condición de parada sean más fáciles de confiar.

POR DENTRO | Valores y tipos

Dentro de un sistema real. Valores y tipos contribuye a distinguir etiquetas, medidas, valores de verdad, y colecciones, pero es solo una etapa. Un uso desplegado de valores y tipos también recopila y verifica entradas, aplica políticas, gestiona fallos, presenta resultados, y observa lo que ocurre después. Las etapas circundantes suelen determinar si un algoritmo correcto se convierte en una experiencia útil.

Qué merece sospecha. Un ejemplo ordenado puede ocultar costo, ambigüedad, o una suposición frágil. Para valores y tipos, inspeccione la entrada de peor forma y cualquier empate o condición límite. En distinguir etiquetas, medidas, valores de verdad, y colecciones, pregúntese también qué errores son más consecuentes y qué camino de recuperación existe.

Traducción cotidiana

Piense en no sumar una dirección postal a una temperatura. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function safe_add(a, b):
    if type_of(a) != type_of(b):
        report an unsupported case      # e.g. adding a street address to a tempera
ture
    return a + b


is_number(37)     -> true
is_text("37")     -> true
is_boolean(true)  -> true
```

LABORATORIO VISUAL | Valores y tipos

1. Traducir la analogía

Dibuje no sumar una dirección postal a una temperatura. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de valores y tipos. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Imagine el ejemplo de no sumar una dirección postal a una temperatura creciendo hasta que ya no quepa en memoria. Decida qué podría procesarse por partes.

4. Cuestionar una suposición

Cree un contraejemplo que haga que un uso ingenuo de valores y tipos sea lento, ambiguo, o engañoso.

5. Localizar las decisiones humanas

En un sistema que involucra distinguir etiquetas, medidas, valores de verdad, y colecciones, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Valores y tipos así: Un tipo describe qué significa un valor y qué operaciones tienen sentido para él.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace valores y tipos?
2. ¿Qué detalle en «no sumar una dirección postal a una temperatura» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

2.2 Registros

Registros

Un registro mantiene varios campos relacionados juntos para que viajen como una sola unidad significativa.



Figura 2.2. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Ver la regla antes que la notación. La imagen de una ficha de catálogo de biblioteca le da a registros una forma física. Un registro mantiene varios campos relacionados juntos para que viajen como una sola unidad significativa. Conservaremos la estructura útil de la comparación mientras descartamos los detalles que no pertenecen al algoritmo.

Elegir qué almacenar. Una implementación de registros trabaja con un modelo deliberado de la situación. El mejor modelo no es el más detallado; es la representación más simple que preserva las decisiones que el método debe tomar. Esto es especialmente importante en modelar un contacto, un producto, o una cita, donde un detalle irrelevante puede añadir costo o ruido.

Seguir un recorrido completo. Parta de un estado inicial válido, aplique la acción que define registros, y registre lo que esa acción establece. El siguiente paso debe usar ese nuevo conocimiento en lugar de reiniciar a ciegas. El progreso se vuelve visible cuando la porción no resuelta se reduce, la frontera avanza, o la estimación mejora.

POR DENTRO | Registros

Por qué esta idea se traslada bien. La misma promesa estructural hace que registros sea útil en modelar un contacto, un producto, o una cita. Los nombres de los objetos pueden cambiar, pero la operación sigue siendo reconocible. Esta transferencia —de una historia superficial a otra— es señal de que hemos aprendido el algoritmo en lugar de memorizar un ejemplo.

Compensaciones, no magia. Registros gana algo —velocidad, estructura, adaptabilidad, o una estimación útil— apoyándose en supuestos particulares. Cuando esos supuestos fallan, el resultado puede volverse lento o engañoso en modelar un contacto, un producto, o una cita. Una implementación robusta detecta los casos no soportados en lugar de inventar confianza en silencio.

Traducción cotidiana

Piense en una ficha de catálogo de biblioteca. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
record Contact:
    name: text
    phone: text
    email: text


function make_contact(name, phone, email):
    return Contact(name=name, phone=phone, email=email)


contact = make_contact("Aisha", "555-0100", "aisha@example.com")
contact.phone      # field access, not a search
```

LABORATORIO VISUAL | Registros

1. Traducir la analogía

Dibuje una ficha de catálogo de biblioteca. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de registros. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Contraste un caso de registros casi resuelto con uno deliberadamente difícil. Explique por qué su comportamiento de ejecución puede diferir.

4. Cuestionar una suposición

Pruebe registros con un caso vacío, un caso único, empates, y un valor malformado. Escriba la respuesta más segura para cada uno.

5. Localizar las decisiones humanas

En un sistema que involucra modelar un contacto, un producto, o una cita, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Registros así: Un registro mantiene varios campos relacionados juntos para que viajen como una sola unidad significativa.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace registros?
2. ¿Qué detalle en «una ficha de catálogo de biblioteca» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

2.3 Referencias

Referencias

Una referencia es una ruta hacia otro valor en lugar de una copia de ese valor.

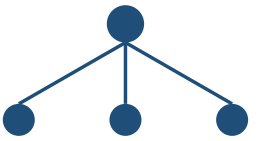


Figura 2.3. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Una vía concreta de entrada. Piense en un resguardo para reclamar un abrigo. Ofrece una ruta memorable hacia esta definición: Una referencia es una ruta hacia otro valor en lugar de una copia de ese valor. Una vez que esa promesa central está clara, el pseudocódigo se convierte en la descripción de acciones familiares en lugar de un muro de símbolos.

Representar este problema. Para referencias, la entrada debe conservar las distinciones necesarias para aplicar la regla. Según la aplicación, eso puede significar posiciones ordenadas, vecinos enlazados, características medidas, etiquetas, o conexiones ponderadas. La representación usada para compartir y enlazar información sin duplicarla debe omitir el detalle decorativo pero conservar cada hecho que pudiera cambiar la respuesta.

Progreso y parada. Cada paso de referencias debe acercar el proceso a una condición verificable. Un rastro útil indica el estado actual, la decisión recién tomada, y el trabajo restante. Si nada de eso cambia, el procedimiento puede estar atascado, o su regla de parada puede ser incompleta.

POR DENTRO | Referencias

Del ejemplo de clase al producto. En compartir y enlazar información sin duplicarla, los ingenieros combinan referencias con almacenamiento, validación, monitoreo, y decisiones de interfaz. La escala puede exigir aproximación o procesamiento por lotes, mientras que las consecuencias humanas pueden exigir explicaciones, anulaciones, y umbrales prudentes.

Una prueba de calidad práctica. Varíe el tamaño, el orden, y la limpieza de la entrada. Compruebe no solo si referencias devuelve una respuesta, sino si esa respuesta sigue siendo significativa para compartir y enlazar información sin duplicarla. Mida la memoria, el retraso, y distintos tipos de error en lugar de reducir la calidad a un único conteo de éxito.

Traducción cotidiana

Piense en un resguardo para reclamar un abrigo. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
locker_a = Contact("Aisha", "555-0100")
locker_b = locker_a          # locker_b now refers to the same record
locker_b.phone = "555-0199"


print(locker_a.phone)         # "555-0199" -- both names point to one record
# a copy would have left locker_a.phone unchanged
```

LABORATORIO VISUAL | Referencias

1. Traducir la analogía

Dibuje un resguardo para reclamar un abrigo. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de referencias. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Compare un ejemplo pequeño de referencias con uno un millón de veces más grande. Marque qué acción se convierte en el cuello de botella.

4. Cuestionar una suposición

Elimine un supuesto de la definición de referencias —orden, conectividad, costo no negativo, o datos representativos— y prediga el fallo.

5. Localizar las decisiones humanas

En un sistema que involucra compartir y enlazar información sin duplicarla, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Referencias así: Una referencia es una ruta hacia otro valor en lugar de una copia de ese valor.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace referencias?
2. ¿Qué detalle en «un resguardo para reclamar un abrigo» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

2.4 Identidad e igualdad

Identidad e igualdad

Dos registros pueden contener información igual y aun así ser objetos distintos.

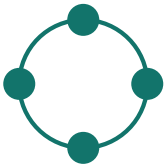


Figura 2.4. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Construir la imagen mental. Imagine dos llaves idénticas cortadas del mismo patrón. Dos registros pueden contener información igual y aun así ser objetos distintos. La comparación resalta el papel del orden y del cambio sin pretender que el mecanismo real es literalmente la escena cotidiana.

Del mundo real a los datos. La escena «dos llaves idénticas cortadas del mismo patrón» tiene más detalle del que un programa necesita. Un diseñador selecciona las partes relevantes para identidad e igualdad y las codifica como valores y relaciones. Si una distinción crucial se pierde durante esa traducción, ningún ingenio posterior podrá reconstruirla de forma fiable.

Rastrear el mecanismo. Recorra un ejemplo pequeño de identidad e igualdad y deténgase después de cada operación. Pregúntese qué promesa de la definición es ahora cierta y qué casos siguen siendo posibles. Este rastreo disciplinado es la forma más segura de descubrir casos omitidos, actualizaciones incorrectas, y bucles que nunca terminan.

POR DENTRO | Identidad e igualdad

Dónde podría encontrarlo. Las aplicaciones que involucran razonar sobre copias, alias, y cambios usan esta idea porque la información subyacente tiene la forma descrita arriba. Las versiones en producción también deben lidiar con el retraso, mediciones imperfectas, comportamiento cambiante, y objetivos que compiten entre sí.

Límites y modos de fallo. Pruebe identidad e igualdad sin elementos, con un elemento, con elementos iguales, con una entrada malformada, y con un caso mucho más grande de lo esperado. Luego pregúntese si la representación misma es justa y fiel. Los pasos correctos no pueden compensar datos que describen el problema equivocado.

Traducción cotidiana

Piense en dos llaves idénticas cortadas del mismo patrón. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
a = Contact("Aisha", "555-0100")
b = Contact("Aisha", "555-0100")
c = a


a == b        # true  -- equal contents
a is b        # false -- two distinct records
a is c        # true  -- the same record, reached two ways
```

LABORATORIO VISUAL | Identidad e igualdad

1. Traducir la analogía

Dibuje dos llaves idénticas cortadas del mismo patrón. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de identidad e igualdad. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Duplique la cantidad de información usada por identidad e igualdad. Prediga qué trabajo se duplica, cuál podría crecer más rápido, y cuál permanece fijo.

4. Cuestionar una suposición

Introduzca información faltante en el ejemplo de dos llaves idénticas cortadas del mismo patrón. Decida si rechazarla, repararla, ignorarla, o representarla explícitamente.

5. Localizar las decisiones humanas

En un sistema que involucra razonar sobre copias, alias, y cambios, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Identidad e igualdad así: Dos registros pueden contener información igual y aun así ser objetos distintos.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace identidad e igualdad?
2. ¿Qué detalle en «dos llaves idénticas cortadas del mismo patrón» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

Capítulo 2 Repaso

Consignas de explicación

Enseñe las ideas centrales de datos: valores, registros y referencias a alguien que nunca haya programado. Use las analogías del capítulo, pero también nombre la información exacta almacenada por valores y tipos y contrástela con identidad e igualdad. Termine explicando cómo cada proceso sabe cuándo detenerse.

Mapa de conexiones

Concepto	Se conecta con	Pregúntese
Valores y tipos	distinguir etiquetas, medidas, valores de verdad, y colecciones	¿Qué cambia a medida que crece la entrada?
Registros	modelar un contacto, un producto, o una cita	¿Qué cambia a medida que crece la entrada?
Referencias	compartir y enlazar información sin duplicarla	¿Qué cambia a medida que crece la entrada?
Identidad e igualdad	razonar sobre copias, alias, y cambios	¿Qué cambia a medida que crece la entrada?

Pruébelo en papel

Elija cinco objetos cotidianos que encajen con la idea de registros. Dibuje la representación, realice una operación a mano, y anote solo la información que cambió. Luego explique por qué la información sin cambios sigue satisfaciendo la promesa central del capítulo.

CAPÍTULO 3

Arreglos y listas dinámicas

En este capítulo

3.1 Una fila de casilleros numerados. Un arreglo almacena elementos en orden y da a cada posición un índice.

3.2 Inserción y eliminación. Hacer espacio en el medio puede requerir desplazar cada elemento posterior.

3.3 Crecer más allá de la capacidad. Un arreglo dinámico ocasionalmente se traslada a un espacio más grande para que la mayoría de las adiciones sigan siendo económicas.

3.4 Segmentos y vistas. Un segmento describe una región contigua; algunos sistemas lo copian mientras que otros comparten el almacenamiento original.

Brújula del capítulo

Al leer Arreglos y listas dinámicas, compare las cuatro ideas de este capítulo: Una fila de casilleros numerados, Inserción y eliminación, Crecer más allá de la capacidad, Segmentos y vistas. Trate cada bloque de pseudocódigo como un mapa de decisiones, no como una prueba de programación.

3.1 Una fila de casilleros numerados

Una fila de casilleros numerados

Un arreglo almacena elementos en orden y da a cada posición un índice.

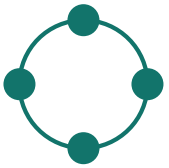


Figura 3.1. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Un punto de partida intuitivo. Empiece por una fila de casilleros numerados. En esa escena, note qué puede moverse, qué actúa como límite, y qué cuenta como éxito. Esas mismas preguntas revelan la idea central de una fila de casilleros numerados: Un arreglo almacena elementos en orden y da a cada posición un índice.

El punto de vista de la máquina. Una computadora no recibe ni la metáfora ni su sentido común. Recibe una descripción estructurada elegida para esta tarea. En una fila de casilleros numerados, esa descripción debe sostener la operación implícita en la definición; en acceso directo rápido y almacenamiento compacto, también debe reflejar unidades, valores ausentes, y el significado de cada conexión.

El invariante que hay que vigilar. Durante una fila de casilleros numerados, algún hecho debe permanecer fiable incluso mientras el resto de la información cambia. Ese hecho puede ser una región ordenada, un conjunto de visitados, un mejor puntaje conocido, o una frontera alrededor del trabajo no resuelto. Nombrarlo hace que tanto el procedimiento como su condición de parada sean más fáciles de confiar.

POR DENTRO | Una fila de casilleros numerados

Dentro de un sistema real. Una fila de casilleros numerados contribuye a acceso directo rápido y almacenamiento compacto, pero es solo una etapa. Un uso desplegado de una fila de casilleros numerados también recopila y verifica entradas, aplica políticas, gestiona fallos, presenta resultados, y observa lo que ocurre después. Las etapas circundantes suelen determinar si un algoritmo correcto se convierte en una experiencia útil.

Qué merece sospecha. Un ejemplo ordenado puede ocultar costo, ambigüedad, o una suposición frágil. Para una fila de casilleros numerados, inspeccione la entrada de peor forma y cualquier empate o condición límite. En acceso directo rápido y almacenamiento compacto, pregúntese también qué errores son más consecuentes y qué camino de recuperación existe.

Traducción cotidiana

Piense en una fila de casilleros numerados. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
locker = ["tea", "cups", "kettle", "napkins"]


locker[0]          # "tea"      -- direct access by index, O(1)
locker[2] = "pot"  # replace in place, O(1)
length(locker)     # 4
```

LABORATORIO VISUAL | Una fila de casilleros numerados

1. Traducir la analogía

Dibuje una fila de casilleros numerados. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de una fila de casilleros numerados. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Imagine el ejemplo de una fila de casilleros numerados creciendo hasta que ya no quepa en memoria. Decida qué podría procesarse por partes.

4. Cuestionar una suposición

Cree un contraejemplo que haga que un uso ingenuo de una fila de casilleros numerados sea lento, ambiguo, o engañoso.

5. Localizar las decisiones humanas

En un sistema que involucra acceso directo rápido y almacenamiento compacto, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Una fila de casilleros numerados así: Un arreglo almacena elementos en orden y da a cada posición un índice.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace una fila de casilleros numerados?
2. ¿Qué detalle en «una fila de casilleros numerados» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

3.2 Inserción y eliminación

Inserción y eliminación

Hacer espacio en el medio puede requerir desplazar cada elemento posterior.

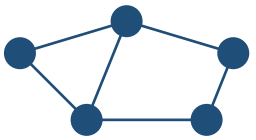


Figura 3.2. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Ver la regla antes que la notación. La imagen de añadir una persona a la mitad de una fila de teatro le da a inserción y eliminación una forma física. Hacer espacio en el medio puede requerir desplazar cada elemento posterior. Conservaremos la estructura útil de la comparación mientras descartamos los detalles que no pertenecen al algoritmo.

Elegir qué almacenar. Una implementación de inserción y eliminación trabaja con un modelo deliberado de la situación. El mejor modelo no es el más detallado; es la representación más simple que preserva las decisiones que el método debe tomar. Esto es especialmente importante en entender por qué los cambios de posición pueden ser costosos, donde un detalle irrelevante puede añadir costo o ruido.

Seguir un recorrido completo. Parta de un estado inicial válido, aplique la acción que define inserción y eliminación, y registre lo que esa acción establece. El siguiente paso debe usar ese nuevo conocimiento en lugar de reiniciar a ciegas. El progreso se vuelve visible cuando la porción no resuelta se reduce, la frontera avanza, o la estimación mejora.

POR DENTRO | Inserción y eliminación

Por qué esta idea se traslada bien. La misma promesa estructural hace que inserción y eliminación sea útil en entender por qué los cambios de posición pueden ser costosos. Los nombres de los objetos pueden cambiar, pero la operación sigue siendo reconocible. Esta transferencia —de una historia superficial a otra— es señal de que hemos aprendido el algoritmo en lugar de memorizar un ejemplo.

Compensaciones, no magia. Inserción y eliminación gana algo —velocidad, estructura, adaptabilidad, o una estimación útil— apoyándose en supuestos particulares. Cuando esos supuestos fallan, el resultado puede volverse lento o engañoso en entender por qué los cambios de posición pueden ser costosos. Una implementación robusta detecta los casos no soportados en lugar de inventar confianza en silencio.

Traducción cotidiana

Piense en añadir una persona a la mitad de una fila de teatro. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function insert_at(items, index, value):
    items.append(placeholder)
    for i from length(items) - 1 down to index + 1:
        items[i] = items[i - 1]      # shift right to make room
    items[index] = value
    return items                      # O(n) in the worst case

function delete_at(items, index):
    for i from index to length(items) - 2:
        items[i] = items[i + 1]      # shift left to close the gap
    items.remove_last()
    return items                      # O(n) in the worst case
```

LABORATORIO VISUAL | Inserción y eliminación

1. Traducir la analogía

Dibuje añadir una persona a la mitad de una fila de teatro. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de inserción y eliminación. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Contraste un caso de inserción y eliminación casi resuelto con uno deliberadamente difícil. Explique por qué su comportamiento de ejecución puede diferir.

4. Cuestionar una suposición

Pruebe inserción y eliminación con un caso vacío, un caso único, empates, y un valor malformado. Escriba la respuesta más segura para cada uno.

5. Localizar las decisiones humanas

En un sistema que involucra entender por qué los cambios de posición pueden ser costosos, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Inserción y eliminación así: Hacer espacio en el medio puede requerir desplazar cada elemento posterior.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace inserción y eliminación?
2. ¿Qué detalle en «añadir una persona a la mitad de una fila de teatro» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

3.3 Crecer más allá de la capacidad

Crecer más allá de la capacidad

Un arreglo dinámico ocasionalmente se traslada a un espacio más grande para que la mayoría de las adiciones sigan siendo económicas.



Figura 3.3. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Una vía concreta de entrada. Piense en mudarse a un estante más grande antes de que cada estante esté lleno. Ofrece una ruta memorable hacia esta definición: Un arreglo dinámico ocasionalmente se traslada a un espacio más grande para que la mayoría de las adiciones sigan siendo económicas. Una vez que esa promesa central está clara, el pseudocódigo se convierte en la descripción de acciones familiares en lugar de un muro de símbolos.

Representar este problema. Para crecer más allá de la capacidad, la entrada debe conservar las distinciones necesarias para aplicar la regla. Según la aplicación, eso puede significar posiciones ordenadas, vecinos enlazados, características medidas, etiquetas, o conexiones ponderadas. La representación usada para equilibrar el espacio de reserva frente al costo de copiar debe omitir el detalle decorativo pero conservar cada hecho que pudiera cambiar la respuesta.

Progreso y parada. Cada paso de crecer más allá de la capacidad debe acercar el proceso a una condición verificable. Un rastro útil indica el estado actual, la decisión recién tomada, y el trabajo restante. Si nada de eso cambia, el procedimiento puede estar atascado, o su regla de parada puede ser incompleta.

POR DENTRO | Crecer más allá de la capacidad

Del ejemplo de clase al producto. En equilibrar el espacio de reserva frente al costo de copiar, los ingenieros combinan crecer más allá de la capacidad con almacenamiento, validación, monitoreo, y decisiones de interfaz. La escala puede exigir aproximación o procesamiento por lotes, mientras que las consecuencias humanas pueden exigir explicaciones, anulaciones, y umbrales prudentes.

Una prueba de calidad práctica. Varíe el tamaño, el orden, y la limpieza de la entrada. Compruebe no solo si crecer más allá de la capacidad devuelve una respuesta, sino si esa respuesta sigue siendo significativa para equilibrar el espacio de reserva frente al costo de copiar. Mida la memoria, el retraso, y distintos tipos de error en lugar de reducir la calidad a un único conteo de éxito.

Traducción cotidiana

Piense en mudarse a un estante más grande antes de que cada estante esté lleno. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function append(dynamic_array, value):
    if dynamic_array.count == dynamic_array.capacity:
        new_capacity = dynamic_array.capacity * 2      # double, don't add one
        new_storage = allocate(new_capacity)
        copy dynamic_array.storage into new_storage
        dynamic_array.storage = new_storage
        dynamic_array.capacity = new_capacity
    dynamic_array.storage[dynamic_array.count] = value
    dynamic_array.count += 1
# occasional O(n) copies, but appends are O(1) on average
```

LABORATORIO VISUAL | Crecer más allá de la capacidad

1. Traducir la analogía

Dibuje mudarse a un estante más grande antes de que cada estante esté lleno. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de crecer más allá de la capacidad. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Compare un ejemplo pequeño de crecer más allá de la capacidad con uno un millón de veces más grande. Marque qué acción se convierte en el cuello de botella.

4. Cuestionar una suposición

Elimine un supuesto de la definición de crecer más allá de la capacidad —orden, conectividad, costo no negativo, o datos representativos— y prediga el fallo.

5. Localizar las decisiones humanas

En un sistema que involucra equilibrar el espacio de reserva frente al costo de copiar, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Crecer más allá de la capacidad así: Un arreglo dinámico ocasionalmente se traslada a un espacio más grande para que la mayoría de las adiciones sigan siendo económicas.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace crecer más allá de la capacidad?
2. ¿Qué detalle en «mudarse a un estante más grande antes de que cada estante esté lleno» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

3.4 Segmentos y vistas

Segmentos y vistas

Un segmento describe una región contigua; algunos sistemas lo copian mientras que otros comparten el almacenamiento original.



Figura 3.4. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Construir la imagen mental. Imagine mirar por una ventana una parte de un mural. Un segmento describe una región contigua; algunos sistemas lo copian mientras que otros comparten el almacenamiento original. La comparación resalta el papel del orden y del cambio sin pretender que el mecanismo real es literalmente la escena cotidiana.

Del mundo real a los datos. La escena «mirar por una ventana una parte de un mural» tiene más detalle del que un programa necesita. Un diseñador selecciona las partes relevantes para segmentos y vistas y las codifica como valores y relaciones. Si una distinción crucial se pierde durante esa traducción, ningún ingenio posterior podrá reconstruirla de forma fiable.

Rastrear el mecanismo. Recorra un ejemplo pequeño de segmentos y vistas y deténgase después de cada operación. Pregúntese qué promesa de la definición es ahora cierta y qué casos siguen siendo posibles. Este rastreo disciplinado es la forma más segura de descubrir casos omitidos, actualizaciones incorrectas, y bucles que nunca terminan.

POR DENTRO | Segmentos y vistas

Dónde podría encontrarlo. Las aplicaciones que involucran evitar cambios compartidos sorprendentes usan esta idea porque la información subyacente tiene la forma descrita arriba. Las versiones en producción también deben lidiar con el retraso, mediciones imperfectas, comportamiento cambiante, y objetivos que compiten entre sí.

Límites y modos de fallo. Pruebe segmentos y vistas sin elementos, con un elemento, con elementos iguales, con una entrada malformada, y con un caso mucho más grande de lo esperado. Luego pregúntese si la representación misma es justa y fiel. Los pasos correctos no pueden compensar datos que describen el problema equivocado.

Traducción cotidiana

Piense en mirar por una ventana una parte de un mural. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
items = [10, 20, 30, 40, 50]


items[1:4]        # [20, 30, 40] -- a new list, indices 1 up to (not including) 4
items[:2]         # [10, 20]
items[-1]         # 50 -- last element
# a view can reference the same storage instead of copying;
# check which behavior a language gives you before mutating one side
```

LABORATORIO VISUAL | Segmentos y vistas

1. Traducir la analogía

Dibuje mirar por una ventana una parte de un mural. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de segmentos y vistas. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Duplique la cantidad de información usada por segmentos y vistas. Prediga qué trabajo se duplica, cuál podría crecer más rápido, y cuál permanece fijo.

4. Cuestionar una suposición

Introduzca información faltante en el ejemplo de mirar por una ventana una parte de un mural. Decida si rechazarla, repararla, ignorarla, o representarla explícitamente.

5. Localizar las decisiones humanas

En un sistema que involucra evitar cambios compartidos sorprendentes, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Segmentos y vistas así: Un segmento describe una región contigua; algunos sistemas lo copian mientras que otros comparten el almacenamiento original.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace segmentos y vistas?
2. ¿Qué detalle en «mirar por una ventana una parte de un mural» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

Capítulo 3 Repaso

Consignas de explicación

Enseñe las ideas centrales de arreglos y listas dinámicas a alguien que nunca haya programado. Use las analogías del capítulo, pero también nombre la información exacta almacenada por una fila de casilleros numerados y contrástela con segmentos y vistas. Termine explicando cómo cada proceso sabe cuándo detenerse.

Mapa de conexiones

Concepto	Se conecta con	Pregúntese
Una fila de casilleros numerados	acceso directo rápido y almacenamiento compacto	¿Qué cambia a medida que crece la entrada?
Inserción y eliminación	entender por qué los cambios de posición pueden ser costosos	¿Qué cambia a medida que crece la entrada?
Crecer más allá de la capacidad	equilibrar el espacio de reserva frente al costo de copiar	¿Qué cambia a medida que crece la entrada?
Segmentos y vistas	evitar cambios compartidos sorprendentes	¿Qué cambia a medida que crece la entrada?

Pruébelo en papel

Elija cinco objetos cotidianos que encajen con la idea de crecer más allá de la capacidad. Dibuje la representación, realice una operación a mano, y anote solo la información que cambió. Luego explique por qué la información sin cambios sigue satisfaciendo la promesa central del capítulo.

CAPÍTULO 4

Listas enlazadas

En este capítulo

4.1 Seguir la flecha siguiente. Cada nodo almacena un valor y un enlace al nodo siguiente.

4.2 Enlazada simple o doblemente. Un segundo enlace hacia atrás facilita el recorrido inverso y la eliminación, a costa de espacio adicional.

4.3 Por qué el acceso aleatorio es lento. Para llegar al nodo número cien, una lista enlazada sigue los primeros noventa y nueve enlaces.

4.4 Centinelas. Un nodo límite ficticio elimina casos especiales incómodos al principio o al final.

Brújula del capítulo

Al leer Listas enlazadas, compare las cuatro ideas de este capítulo: Seguir la flecha siguiente, Enlazada simple o doblemente, Por qué el acceso aleatorio es lento, Centinelas. Trate cada bloque de pseudocódigo como un mapa de decisiones, no como una prueba de programación.

4.1 Seguir la flecha siguiente

Seguir la flecha siguiente

Cada nodo almacena un valor y un enlace al nodo siguiente.



Figura 4.1. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Un punto de partida intuitivo. Empiece por una búsqueda del tesoro hecha de pistas. En esa escena, note qué puede moverse, qué actúa como límite, y qué cuenta como éxito. Esas mismas preguntas revelan la idea central de seguir la flecha siguiente: Cada nodo almacena un valor y un enlace al nodo siguiente.

El punto de vista de la máquina. Una computadora no recibe ni la metáfora ni su sentido común. Recibe una descripción estructurada elegida para esta tarea. En seguir la flecha siguiente, esa descripción debe sostener la operación implícita en la definición; en una inserción económica cuando la ubicación ya se conoce, también debe reflejar unidades, valores ausentes, y el significado de cada conexión.

El invariante que hay que vigilar. Durante seguir la flecha siguiente, algún hecho debe permanecer fiable incluso mientras el resto de la información cambia. Ese hecho puede ser una región ordenada, un conjunto de visitados, un mejor puntaje conocido, o una frontera alrededor del trabajo no resuelto. Nombrarlo hace que tanto el procedimiento como su condición de parada sean más fáciles de confiar.

POR DENTRO | Seguir la flecha siguiente

Dentro de un sistema real. Seguir la flecha siguiente contribuye a una inserción económica cuando la ubicación ya se conoce, pero es solo una etapa. Un uso desplegado de seguir la flecha siguiente también recopila y verifica entradas, aplica políticas, gestiona fallos, presenta resultados, y observa lo que ocurre después. Las etapas circundantes suelen determinar si un algoritmo correcto se convierte en una experiencia útil.

Qué merece sospecha. Un ejemplo ordenado puede ocultar costo, ambigüedad, o una suposición frágil. Para seguir la flecha siguiente, inspeccione la entrada de peor forma y cualquier empate o condición límite. En una inserción económica cuando la ubicación ya se conoce, pregúntese también qué errores son más consecuentes y qué camino de recuperación existe.

Traducción cotidiana

Piense en una búsqueda del tesoro hecha de pistas. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function traverse(head):
    node = head
    while node is not null:
        visit(node.value)
        node = node.next        # follow the arrow, one hop at a time
# no index arithmetic -- just "is there a next node?"
```

LABORATORIO VISUAL | Seguir la flecha siguiente

1. Traducir la analogía

Dibuje una búsqueda del tesoro hecha de pistas. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de seguir la flecha siguiente. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Imagine el ejemplo de una búsqueda del tesoro hecha de pistas creciendo hasta que ya no quepa en memoria. Decida qué podría procesarse por partes.

4. Cuestionar una suposición

Cree un contraejemplo que haga que un uso ingenuo de seguir la flecha siguiente sea lento, ambiguo, o engañoso.

5. Localizar las decisiones humanas

En un sistema que involucra una inserción económica cuando la ubicación ya se conoce, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Seguir la flecha siguiente así: Cada nodo almacena un valor y un enlace al nodo siguiente.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace seguir la flecha siguiente?
2. ¿Qué detalle en «una búsqueda del tesoro hecha de pistas» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

4.2 Enlazada simple o doblemente

Enlazada simple o doblemente

Un segundo enlace hacia atrás facilita el recorrido inverso y la eliminación, a costa de espacio adicional.

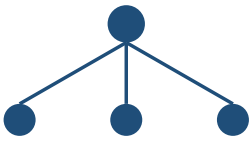


Figura 4.2. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Ver la regla antes que la notación. La imagen de vagones de tren acoplados en una o dos direcciones le da a enlazada simple o doblemente una forma física. Un segundo enlace hacia atrás facilita el recorrido inverso y la eliminación, a costa de espacio adicional. Conservaremos la estructura útil de la comparación mientras descartamos los detalles que no pertenecen al algoritmo.

Elegir qué almacenar. Una implementación de enlazada simple o doblemente trabaja con un modelo deliberado de la situación. El mejor modelo no es el más detallado; es la representación más simple que preserva las decisiones que el método debe tomar. Esto es especialmente importante en elegir la flexibilidad de navegación, donde un detalle irrelevante puede añadir costo o ruido.

Seguir un recorrido completo. Parta de un estado inicial válido, aplique la acción que define enlazada simple o doblemente, y registre lo que esa acción establece. El siguiente paso debe usar ese nuevo conocimiento en lugar de reiniciar a ciegas. El progreso se vuelve visible cuando la porción no resuelta se reduce, la frontera avanza, o la estimación mejora.

POR DENTRO | Enlazada simple o doblemente

Por qué esta idea se traslada bien. La misma promesa estructural hace que enlazada simple o doblemente sea útil en elegir la flexibilidad de navegación. Los nombres de los objetos pueden cambiar, pero la operación sigue siendo reconocible. Esta transferencia —de una historia superficial a otra— es señal de que hemos aprendido el algoritmo en lugar de memorizar un ejemplo.

Compensaciones, no magia. Enlazada simple o doblemente gana algo —velocidad, estructura, adaptabilidad, o una estimación útil— apoyándose en supuestos particulares. Cuando esos supuestos fallan, el resultado puede volverse lento o engañoso en elegir la flexibilidad de navegación. Una implementación robusta detecta los casos no soportados en lugar de inventar confianza en silencio.

Traducción cotidiana

Piense en vagones de tren acoplados en una o dos direcciones. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function insert_after(node, value):        # singly linked
    new_node = Node(value)
    new_node.next = node.next
    node.next = new_node                    # O(1) once you hold the node

function insert_after_doubly(node, value):  # doubly linked
    new_node = Node(value)
    new_node.next = node.next
    new_node.prev = node
    if node.next is not null:
        node.next.prev = new_node
    node.next = new_node                    # can also walk backward via .prev
```

LABORATORIO VISUAL | Enlazada simple o doblemente

1. Traducir la analogía

Dibuje vagones de tren acoplados en una o dos direcciones. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de enlazada simple o doblemente. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Contraste un caso de enlazada simple o doblemente casi resuelto con uno deliberadamente difícil. Explique por qué su comportamiento de ejecución puede diferir.

4. Cuestionar una suposición

Pruebe enlazada simple o doblemente con un caso vacío, un caso único, empates, y un valor malformado. Escriba la respuesta más segura para cada uno.

5. Localizar las decisiones humanas

En un sistema que involucra elegir la flexibilidad de navegación, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Enlazada simple o doblemente así: Un segundo enlace hacia atrás facilita el recorrido inverso y la eliminación, a costa de espacio adicional.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace enlazada simple o doblemente?
2. ¿Qué detalle en «vagones de tren acoplados en una o dos direcciones» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

4.3 Por qué el acceso aleatorio es lento

Por qué el acceso aleatorio es lento

Para llegar al nodo número cien, una lista enlazada sigue los primeros noventa y nueve enlaces.

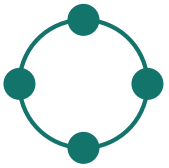


Figura 4.3. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Una vía concreta de entrada. Piense en caminar de puerta en puerta sin números de casa. Ofrece una ruta memorable hacia esta definición: Para llegar al nodo número cien, una lista enlazada sigue los primeros noventa y nueve enlaces. Una vez que esa promesa central está clara, el pseudocódigo se convierte en la descripción de acciones familiares en lugar de un muro de símbolos.

Representar este problema. Para por qué el acceso aleatorio es lento, la entrada debe conservar las distinciones necesarias para aplicar la regla. Según la aplicación, eso puede significar posiciones ordenadas, vecinos enlazados, características medidas, etiquetas, o conexiones ponderadas. La representación usada para reconocer el equilibrio entre el acceso y la edición debe omitir el detalle decorativo pero conservar cada hecho que pudiera cambiar la respuesta.

Progreso y parada. Cada paso de por qué el acceso aleatorio es lento debe acercar el proceso a una condición verificable. Un rastro útil indica el estado actual, la decisión recién tomada, y el trabajo restante. Si nada de eso cambia, el procedimiento puede estar atascado, o su regla de parada puede ser incompleta.

POR DENTRO | Por qué el acceso aleatorio es lento

Del ejemplo de clase al producto. En reconocer el equilibrio entre el acceso y la edición, los ingenieros combinan por qué el acceso aleatorio es lento con almacenamiento, validación, monitoreo, y decisiones de interfaz. La escala puede exigir aproximación o procesamiento por lotes, mientras que las consecuencias humanas pueden exigir explicaciones, anulaciones, y umbrales prudentes.

Una prueba de calidad práctica. Varíe el tamaño, el orden, y la limpieza de la entrada. Compruebe no solo si por qué el acceso aleatorio es lento devuelve una respuesta, sino si esa respuesta sigue siendo significativa para reconocer el equilibrio entre el acceso y la edición. Mida la memoria, el retraso, y distintos tipos de error en lugar de reducir la calidad a un único conteo de éxito.

Traducción cotidiana

Piense en caminar de puerta en puerta sin números de casa. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
function get_at(head, index):
    node = head
    steps = 0
    while steps < index:
        node = node.next        # must hop node by node
        steps += 1
    return node.value            # O(index) -- unlike array[index], which is O(1)
```

LABORATORIO VISUAL | Por qué el acceso aleatorio es lento

1. Traducir la analogía

Dibuje caminar de puerta en puerta sin números de casa. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de por qué el acceso aleatorio es lento. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Compare un ejemplo pequeño de por qué el acceso aleatorio es lento con uno un millón de veces más grande. Marque qué acción se convierte en el cuello de botella.

4. Cuestionar una suposición

Elimine un supuesto de la definición de por qué el acceso aleatorio es lento —orden, conectividad, costo no negativo, o datos representativos— y prediga el fallo.

5. Localizar las decisiones humanas

En un sistema que involucra reconocer el equilibrio entre el acceso y la edición, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Por qué el acceso aleatorio es lento así: Para llegar al nodo número cien, una lista enlazada sigue los primeros noventa y nueve enlaces.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace por qué el acceso aleatorio es lento?
2. ¿Qué detalle en «caminar de puerta en puerta sin números de casa» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

4.4 Centinelas

Centinelas

Un nodo límite ficticio elimina casos especiales incómodos al principio o al final.

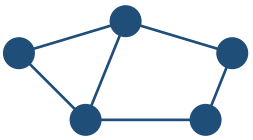


Figura 4.4. Un ritmo visual para la idea: entrada, relación, cambio, y resultado.

Construir la imagen mental. Imagine un sujetalibros que no es un libro. Un nodo límite ficticio elimina casos especiales incómodos al principio o al final. La comparación resalta el papel del orden y del cambio sin pretender que el mecanismo real es literalmente la escena cotidiana.

Del mundo real a los datos. La escena «un sujetalibros que no es un libro» tiene más detalle del que un programa necesita. Un diseñador selecciona las partes relevantes para centinelas y las codifica como valores y relaciones. Si una distinción crucial se pierde durante esa traducción, ningún ingenio posterior podrá reconstruirla de forma fiable.

Rastrear el mecanismo. Recorra un ejemplo pequeño de centinelas y deténgase después de cada operación. Pregúntese qué promesa de la definición es ahora cierta y qué casos siguen siendo posibles. Este rastreo disciplinado es la forma más segura de descubrir casos omitidos, actualizaciones incorrectas, y bucles que nunca terminan.

POR DENTRO | Centinelas

Dónde podría encontrarlo. Las aplicaciones que involucran simplificar un código que de otro modo sería frágil usan esta idea porque la información subyacente tiene la forma descrita arriba. Las versiones en producción también deben lidiar con el retraso, mediciones imperfectas, comportamiento cambiante, y objetivos que compiten entre sí.

Límites y modos de fallo. Pruebe centinelas sin elementos, con un elemento, con elementos iguales, con una entrada malformada, y con un caso mucho más grande de lo esperado. Luego pregúntese si la representación misma es justa y fiel. Los pasos correctos no pueden compensar datos que describen el problema equivocado.

Traducción cotidiana

Piense en un sujetalibros que no es un libro. Use esa imagen para recordar la regla, y luego vuelva a la definición exacta cuando los detalles importen.

Pseudocódigo al estilo Python

```
sentinel = Node(value=None)      # a dummy node that is never real data
sentinel.next = sentinel         # empty list points to itself


function is_empty(list):
    return list.sentinel.next is list.sentinel
# insert/delete code no longer needs a special case for "the list is empty"
```

LABORATORIO VISUAL | Centinelas

1. Traducir la analogía

Dibuje un sujetalibros que no es un libro. Identifique los objetos que representan los datos almacenados, las relaciones, el estado de trabajo, y el resultado final.

2. Rastrear un paso decisivo

Realice una operación significativa de centinelas. Anote qué se leyó, qué cambió, y qué promesa se volvió cierta.

3. Examinar el crecimiento

Duplique la cantidad de información usada por centinelas. Prediga qué trabajo se duplica, cuál podría crecer más rápido, y cuál permanece fijo.

4. Cuestionar una suposición

Introduzca información faltante en el ejemplo de un sujetalibros que no es un libro. Decida si rechazarla, repararla, ignorarla, o representarla explícitamente.

5. Localizar las decisiones humanas

En un sistema que involucra simplificar un código que de otro modo sería frágil, identifique una decisión automatizada, una elección de diseño o política, y un resultado que una persona debería poder cuestionar.

Explicarlo sin jerga

Se puede entender Centinelas así: Un nodo límite ficticio elimina casos especiales incómodos al principio o al final.

Verificación rápida

1. Con sus propias palabras, ¿qué promesa hace centinelas?
2. ¿Qué detalle en «un sujetalibros que no es un libro» corresponde al estado que cambia?
3. Dé una situación en la que este método encajaría mal.
4. ¿Qué evidencia lo convencería de que una implementación funciona?

Capítulo 4 Repaso

Consignas de explicación

Enseñe las ideas centrales de listas enlazadas a alguien que nunca haya programado. Use las analogías del capítulo, pero también nombre la información exacta almacenada por seguir la flecha siguiente y contrástela con centinelas. Termine explicando cómo cada proceso sabe cuándo detenerse.

Mapa de conexiones

Concepto	Se conecta con	Pregúntese
Seguir la flecha siguiente	una inserción económica cuando la ubicación ya se conoce	¿Qué cambia a medida que crece la entrada?
Enlazada simple o doblemente	elegir la flexibilidad de navegación	¿Qué cambia a medida que crece la entrada?
Por qué el acceso aleatorio es lento	reconocer el equilibrio entre el acceso y la edición	¿Qué cambia a medida que crece la entrada?
Centinelas	simplificar un código que de otro modo sería frágil	¿Qué cambia a medida que crece la entrada?

Pruébelo en papel

Elija cinco objetos cotidianos que encajen con la idea de centinelas. Dibuje la representación, realice una operación a mano, y anote solo la información que cambió. Luego explique por qué la información sin cambios sigue satisfaciendo la promesa central del capítulo.