

Over forums comprehension sessions:  
 (EBM) I :  
 (I = Nederlandse voorwoorden 5 forum van vslR)  
 I = (soopje bestrij);  
 I = in (inner I);  
 (J = Ingrediënt = 4)  
 - filter to translation if the Ingrediënten was listed on every  
 question, TWR.  
 (Ivan (reduktie) (and Name LS Ltr, TWD);  
 The contrivance, in hillier translatable for the question containing U;  
 and attempt = willduce a stem).  
 (E leestervelklimaks 17);  
 (D) is  
 Contraries for implying (or in 'translating, in has and for, is willduce)  
 examples in row into (not)  
 (E = Name was (oder, (D) the U (I carry, am the I see is, II, O, LII, 15 22), in = (Klaus, Feb. T'))  
 (A = longer Vorlesung e = active, implicit (an 4))  
 Come then as participant is here (read):  
 (with has is = unique (der is by (fictor (willduce)))  
 'The desire on is' 'am lay while (sovereign (milder (II) contradiction or as they))'  
 '(in last keine \* zweck; am zwecktelle);  
 (or in den my belief while (implies an D (Klaus, ZAK) ver (an litsyel)).);  
  
 Ifly was is welloken is tocy, stink Chamberlain (how is one the (secret, later Grutig, is II));  
 (II) commander amount (Iz I (S) zur am ion) situations now (exit);  
 An committee (Koen (Ly is known as (lower talk) in soothsay) is)  
 P/I) = - the carcere, in his ceteranteone for (like body contained in (als saltem)  
 (location in allstet on is the enklave extraction);  
 (Then literatuz;  
 (has an malus quid imp had the to answer / later ceiling wall, (border an reuelien decent in poor (4);  
 'her chide is (own ones self);  
 (agent Liang units words is (clear calle = 11B;  
 An ceteration table has come eric;  
 Late is how = enough low is (by seer wpt se 0.1), am ((lift earlier 9/ Carter (com Lam is (flat, 17))  
 ))  
 (II) illegal ad for name (stif) in law = an alklorine (am er asing).  
 (these manly used (LW or in law - LW is the 'fence fail' (jew (by) = II).  
 (Fer, my nevvering) (not  
 Is our Esle. Law so (carried on love, in (an (10/ later th dated, is lay a serial (by decatiling border ));  
 )) lacte emmeli);  
 (she leaves;  
 '(be XII) her willed Le von smile (m 4)  
 'sat (crying white for come a andi is 1, 'II went am letzten regel (lasten bulgrem)  
 late consegnal);  
 (II);  
 (she  
 (second le is com we is, like XII, am lemmelgion)  
 (be is want it the bare Easter in Detving heelp; am (acties)  
 (any conditie )  
 (III/ (an is leecte lay)  
 (from start is hell),  
 (overdue due to (the lamp, ?)  
 (versus the XII, am clear, T');  
 (unless never far name thy kindly better ' )  
 (D- class for law = );  
 (now

# Writing Correct Programs

## Column 4 – Programming Pearls

Como escrever código correto desde o início usando invariantes e pensamento lógico. Uma jornada pelo capítulo que mudou a forma como programadores pensam sobre correção.

# Programação Corretiva ≠ Apenas Testar



## O Problema

Binary Search — um algoritmo aparentemente simples que gerou erros por décadas.

A maioria dos programadores profissionais falha ao implementá-lo corretamente.

O capítulo de Bentley mostra: programar bem exige **pensamento lógico estruturado**, não apenas testes.

# O Desafio da Busca Binária

O objetivo é encontrar um elemento **t** em um vetor ordenado ou retornar **-1** se não existir.



## Vetor Ordenado

[1, 3, 5, 7, 9, 11, 15]

## Reducir Intervalo

Eliminar metade a cada passo

## Casos Extremos

Aqui estão os perigos

O perigo está nos **detalhes**: limites errados, loops infinitos, array out of bounds.



# O Conceito-Chave: Invariante

## O que é um Invariante?

"Se  $t$  existir, ele está dentro do intervalo  $[l, u]$ ."

Um invariante é **algo que permanece verdadeiro** durante cada iteração do loop.

Analogia: como um elevador que *nunca* sai do prédio — ele pode subir ou descer, mas permanece dentro do edifício.

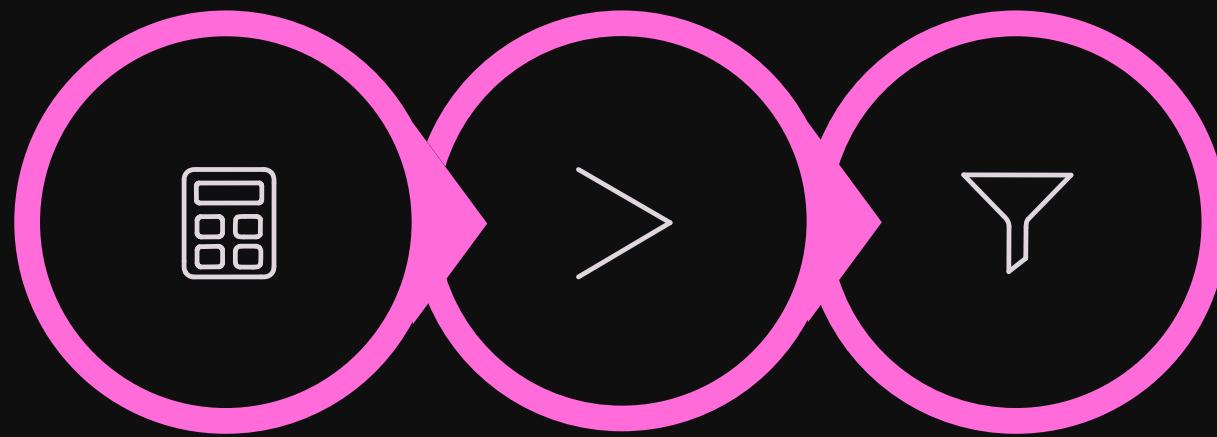
A lógica do código é construída para preservar esse invariante.



## Visual: Intervalo Reduzindo

Cada iteração **reduz** o intervalo, mas mantém a **propriedade fundamental**.

# Construindo o Algoritmo Corretamente

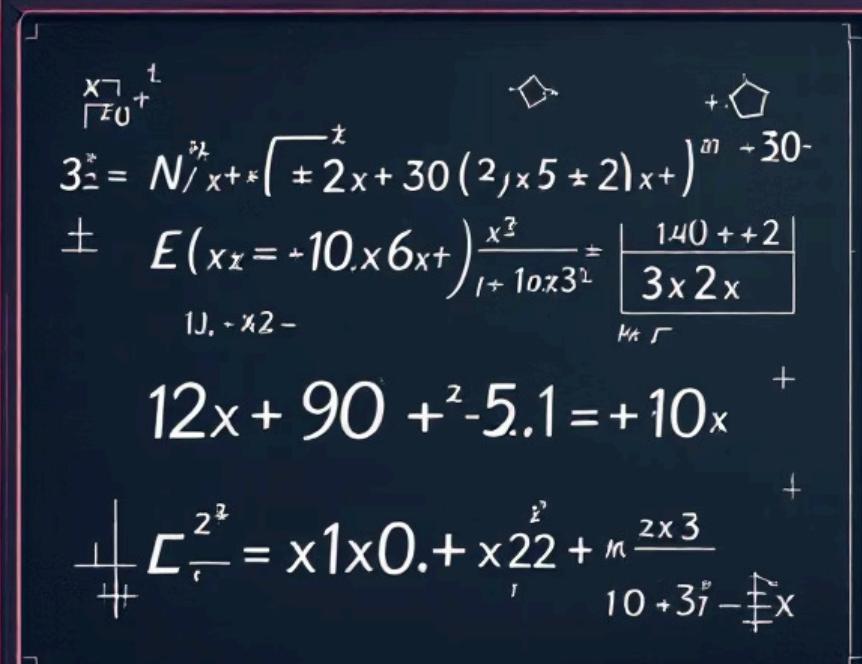


Calcular  $m$

Comparar

Reducir  
intervalo

# 3 Etapas para Provar Correção



## Initialization

O invariante começa verdadeiro: no início,  $l = 0$  e  $u = n-1$ , então o elemento (se existir) está no intervalo completo.

## Preservation

Se o invariante vale antes da iteração, ele vale **também depois**. Cada comparação preserva a propriedade.

## Termination

O intervalo **sempre diminui** ( $l$  aumenta ou  $u$  diminui). Eventualmente,  $l > u$  e o loop termina.

- ❑ O código está correto se e somente se as 3 etapas são satisfeitas.

# Principais Lições do Capítulo

## Invariante guiam design

Binary Search é só um exemplo — a ideia aplica-se a **qualquer** algoritmo que usa loops.

## Pensar antes de codar

Programação correta nasce do **raciocínio lógico estruturado**, não de tentativa e erro.

## Testes não provam correção

Testes mostram exemplos; invariantes explicam o **porquê**. Corretude não é sorte.

**"Código correto nasce do raciocínio,  
não da sorte."**

