Replacing a constant with a variable 5%

Question

Given an array f[0..N) of integers, where {N ≥ 0}. Write a specification and hence derive a solution that calculates the frequency of even and the frequency of odd values in f

1. Program specification

|[ const N : int {N ≥ 0}

f : array{0..N} of int;

var freqEven , freqOdd : int;

S

freqEven = (#j : 0 ≤ j < N: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < N: f.j mod 2 ≠ 0)

1. Invariants P0 and P1

Replacing N with n gives

freqEven = (#j : 0 ≤ j < n: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < n: f.j mod 2 ≠ 0) ∧0 ≤ n ≤ N

P0 = freqEven = (#j : 0 ≤ j < n: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < n: f.j mod 2 ≠ 0)

P1 = 0 ≤ n ≤ N

1. Outline solution

S0:

{P}

do n ≠ N →

{P ∧ n ≠ N}

S1;

{P}

od

{P ∧ n = N}

freqEven = (#j : 0 ≤ j < N: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < N: f.j mod 2 ≠ 0)

1. Derive S0; - Initialisation

Proof

(freqEven = (#j : 0 ≤ j < n: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < n f.j mod 2 ≠ 0) ∧ 0 ≤ n ≤ N)

(freqEven ,FreqOdd ,n := 0 , 0 , 0)

{Substitution}

0 = (#j : 0 ≤ j < 0: f.j mod 2 = 0 ) ∧ 0 = ( #j : 0 ≤ j < 0 : f.j mod 2 ≠ 0) ∧ 0 ≤ 0 ≤ N

{Empty range}

0 = (#j: false: f.j mod 2 = 0 ) ∧ 0 = ( #j :false : f.j mod 2 ≠ 0) ∧ 0 ≤ 0 ≤ N

{# over empty range , constants}

0 = 0 ∧ 0 ∧ true ∧ 0 ≤ N

{constants}

true ∧ true ∧ 0 ≤ N

{constants}

{0 ≤ N}

Given {N ≥ 0}.

1. Derive S1; - Loop body

Suggest n := n + 1

Proof

freqEven = (#j : 0 ≤ j < n: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < n: f.j mod 2 ≠ 0) ( n: n + 1)

{Substitution}

freqEven = (#j : 0 ≤ j < n + 1: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < n + 1: f.j mod 2 ≠ 0)

{Split off j = n}

freqEven = (#j : 0 ≤ j < n: f.j mod 2 = 0) ∧ f.n mod 2 = 0 ∧ freqOdd = ( #j : 0 ≤ j < n : f.j mod 2 ≠ 0) + # (f.n mod 2 = 0)

f.n mod 2 = 0 ∧ freqOdd = freqOdd +#(f.n mod 2 ≠ 0)

{Case analysis}

freqOdd = freqOdd + 1 , if ( f.n mod 2 = 0)

freqOdd = freqOdd , if (f.n mod 2 ≠ 0)

if(f.n mod 2 = 0) →

freqEven := freqEven +1;

[] (f.n mod 2 ≠ 0) →

freqOdd := freqOdd +1;

fi

n: n +1

1. Bound function

The bound function is N – n

1. Prove termination

Initial

( N – n ≥ 0 ) ( n := 1 )

{Substitution}

N – 1 ≥ 0

{Arithmetic}

N ≥ 1

{Given N ≥ 0}

Decreasing

(N – n) ( n: n + 1)

{Substitution}

N – ( n + 1)

{Arithmetic}

N – n – 1

<

N – n

1. Write down a complete solution

|[ const N : int {N ≥ 0}

f : array{0..N} of int;

var freqEven , freqOdd : int;

n: int

freqEven = (#j : 0 ≤ j < n: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < n: f.j mod 2 ≠ 0) ∧0 ≤ n ≤ N

do n < N →

{P ∧ n ≠ N}

if(f.n mod 2 = 0) →

freqEven := freqEven +1;

[] (f.n mod 2 ≠ 0) →

freqOdd := freqOdd +1;

fi

n: n + 1;

od

{P ∧ n = N}

freqEven = (#j : 0 ≤ j < N: f.j mod 2 = 0 )∧ freqOdd = ( #j : 0 ≤ j < N: f.j mod 2 ≠ 0)

1. Write a Java translation of solution
2. **public** **class** Frequency {
3. **public** **static** **void** main (String [] args)
4. {
5. **int** N = 10;
6. **int** f [] = {1,2,3,4,5,6,7,8,9,10};

9. **int** Freqeven = 0;
10. **int** FreqOdd = 0;
11. **int** n = 0;

14. **while** (n < N)
15. {
16. **if**(n % 2 == 0)
17. {
18. Freqeven = Freqeven +1;
19. System.***out***.println("The number is even \n"+ Freqeven);
20. }**else** **if** (n % 2 != 0)
21. {
22. FreqOdd = FreqOdd + 1;
23. System.***out***.println("The number is odd \n" + FreqOdd);
24. }
25. n = n + 1;
26. }
27. }
28. }