En la función _generar_ se hace uso de una función auxiliar que calcula el resto obtenido al realizar la división ‘entera’ de dos números reales. Como no existe ninguna función predefinida que lleve a cabo esta tarea, se construye su pseudocódigo y, posteriormente, se verifica de manera formal:

Función que calcula a mód b, siendo a y b números reales

{P ≡ a >= 0.0 ⋀ b > 0.0}

```
fun modulo_real(a: real, b: real) dev <cociente: real, resto: real>

    <cociente, resto> := <0.0, a>;

    {I ≡ 0.0 <= resto ⋀ a = b * cociente + resto ⋀ b > 0.0}

    {C ≡ resto}

    mientras resto >= b hacer

        <cociente, resto> := <cociente + 1.0, resto – b>

    fmientras

ffun
```

{Q ≡ 0.0 <= resto < b ⋀ a = b * cociente + resto}

Verificación formal de la función _modulo_real_

(1) {P} <cociente, resto> := <0.0, a> {I}

Calculamos pmd(<cociente, resto> : = <0.0, a>, I):

pmd(<cociente, resto> : = <0.0, a>, I) ⇔ def(0.0) ⋀ def(a) ⋀ $I^{0.0,a}_{cociente,resto}$ ⇔

⇔cierto ⋀ cierto ⋀ 0.0 <= a ⋀ a = b * 0.0 + a ⋀ b > 0.0 ⇔ 0.0 <= a ⋀ a = a ⋀ b > 0.0 ⇔

⇔ 0.0 <= a ⋀ cierto ⋀ b > 0.0 ⇔ 0.0 <= a ⋀ b > 0.0

Por último, demostramos que P ⇒ pmd(<cociente, resto> : = <0.0, a>, I):

P ≡ a >= 0.0 ⋀ b > 0.0 ⇔ pmd(<cociente, resto> : = <0.0, a>, I)

(2) {I ⋀ resto >= b} <cociente, resto> := <cociente + 1.0, resto – b> {I}

Calculamos pmd(<cociente, resto> := <cociente + 1.0, resto – b>, I):

pmd(<cociente, resto> := <cociente + 1.0, resto – b>, I) ⇔

⇔ def(cociente + 1.0) ⋀ def(resto - b) ⋀ $I^{cociente+1.0,resto-b}_{cociente,resto}$ ⇔

⇔ cierto ⋀ cierto ⋀ 0.0 <= resto – b ⋀ a = b * (cociente + 1.0) + (resto – b) ⋀ b > 0.0 ⇔

⇔ b <= resto ⋀ a = b * (cociente + 1.0 – 1.0) + resto ⋀ b > 0.0 ⇔

⇔ b <= resto ⋀ a = b * cociente + resto ⋀ b > 0.0

Por último, demostramos que (I ⋀ resto >= b) ⇒ pmd(<cociente, resto> := <cociente + 1.0, resto – b>, I):

I ⋀ resto >= b ≡ 0.0 <= resto ⋀ a = b * cociente + resto ⋀ b > 0.0 ⋀ resto >= b ⇒

⇒ a = b * cociente + resto ⋀ resto >= b ⋀ b > 0.0

(3) {I ⋀ resto < b} nada {Q}

Calculamos pmd(nada, Q):

pmd(nada, Q) ⇔ Q

Por último, demostramos que (I ⋀ resto < b) ⇒ Q:

I ⋀ resto < b ≡ 0.0 <= resto ⋀ a = b * cociente + resto ⋀ b > 0.0 ⋀ resto < b ⇒

⇒ 0.0 <= resto < b ⋀ a = b * cociente + resto ⇔ Q

(4) I ⋀ resto >= b ⇒ C >= 0.0

I ⋀ resto >= b ⇒ 0.0 <= resto ⇔ C >= 0.0

(5) {I ⋀ resto >= b ⋀ C = T} <cociente, resto> := <cociente + 1.0, resto – b> {C < T}

Calculamos pmd(<cociente, resto> := <cociente + 1.0, resto – b>, C < T):

pmd(<cociente, resto> := <cociente + 1.0, resto – b>, C < T) ⇔

⇔ def(cociente + 1.0) ⋀ def(resto - b) ⋀ $(\mathrm{C} < \mathrm{T})^{cociente+1.0, resto-b}_{cociente, resto}$ ⇔

⇔ cierto ⋀ cierto ⋀ resto – b < T ⇔ resto < T + b

Por último, demostramos que (I ⋀ resto >= b ⋀ C = T) ⇒ pmd(<cociente, resto> := <cociente + 1.0, resto – b>, C < T):

I ⋀ resto >= b ⋀ C = T ⇔

⇔ 0.0 <= resto ⋀ a = b * cociente + resto ⋀ b > 0.0 ⋀ resto >= b ⋀ C = T ⇒

⇒ C = T ⋀ b > 0.0 ⇔ resto = T ⋀ b > 0.0 ⇒ resto < T + b

Pseudocódigo de la función *generar*

```
{P ≡ 20 <= N <= 50 ⋀ 0.0 < cota_sup <= 30.0}

fun generar(N: nat, cota_sup: real) dev poblacion[0..N) de real

var contador: ent

var sumando, individuo, cociente, resto: real

    <contador, sumando, individuo> : = <0, 1.0, 0.0>;

    {I ≡ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ 0 <= contador <= N ⋀

        ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando > 0.0}

    {C ≡ N – contador}

    mientras contador < N hacer

        {I ⋀ contador < N}

        individuo := individuo + sumando;

        {I ⋀ contador < N}

        si individuo >= cota_sup entonces

            {I ⋀ contador < N ⋀ individuo >= cota_sup}

            <cociente, resto> := modulo_real(individuo, cota_sup);

            {I ⋀ contador < N ⋀ individuo >= cota_sup ⋀ 0.0 <= resto < cota_sup ⋀

                ⋀ individuo = cota_sup * cociente + resto}

            individuo := resto

            {I ⋀ contador < N ⋀ individuo < cota_sup}

        fsi

        {I ⋀ contador < N ⋀ individuo < cota_sup}

        poblacion[contador] := individuo;

        <sumando, contador> := <sumando + individuo, contador + 1>

        {I}

    fmientras

    {I ⋀ contador >= N}

ffun

{Q ≡ (∀ n : 0 <= n < N : 0.0 <= poblacion[n] < cota_sup)}
```

Verificación formal de la función *generar*

(1) {P} <contador, sumando, individuo> : = <0, 1.0, 0.0> {I}

Calculamos pmd(<contador, sumando, individuo> : = <0, 1.0, 0.0>, I):

pmd(<contador, sumando, individuo> : = <0, 1.0, 0.0>, I) ⇔

⇔ def(0) ∧ def(1.0) ∧ def(0.0) ∧ $I^{0,1.0,0.0}_{contador,sumando,individuo}$ ⇔

⇔ cierto ∧ cierto ∧ cierto ∧ (∀ n : $\underbrace{\text{0 <= n < 0}}_{\emptyset}$: 0.0 <= poblacion[n] < cota_sup) ∧

∧ 0 <= 0 <= N ∧ 0.0 >= 0.0 ∧ cota_sup > 0.0 ∧ 1.0 > 0.0 ⇔

⇔ cierto ∧ 0 <= N ∧ cierto ∧ cota_sup > 0.0 ∧ cierto ⇔ 0 <= N ∧ cota_sup > 0.0

Por último, demostramos que P ⇒ pmd(<contador, sumando, individuo> : = <0, 1.0, 0.0>, I):

P ≡ 20 <= N <= 50 ∧ 0.0 < cota_sup <= 30.0 ⇒ 0 <= N ∧ cota_sup > 0.0

(2) {I ∧ contador < N} individuo := individuo + sumando {I ∧ contador < N}

Calculamos pmd(individuo := individuo + sumando, I ∧ contador < N):

pmd(individuo := individuo + sumando, I ∧ contador < N)⇔

⇔ def(individuo + sumando) ∧ $(I \wedge \mathrm{contador} < \mathrm{N})^{individuo+sumando}_{individuo}$ ⇔

⇔ cierto ∧ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ∧ 0 <= contador <= N ∧

∧ individuo + sumando >= 0.0 ∧ cota_sup > 0.0 ∧ sumando > 0.0 ∧ contador < N ⇔

⇔ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ∧ 0 <= contador < N ∧

∧ individuo + sumando >= 0.0 ∧ cota_sup > 0.0 ∧ sumando > 0.0

Por último, demostramos que (I ∧ contador < N) ⇒ pmd(individuo : = individuo + sumando, I ∧ contador < N):

I ∧ contador < N ≡ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ∧ 0 <= contador <= N ∧ individuo >= 0.0 ∧ cota_sup > 0.0 ∧ sumando > 0.0 ∧ contador < N ⇒

⇒ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ∧ 0 <= contador < N ∧

∧ individuo + sumando >= 0.0 ∧ cota_sup > 0.0 ∧ sumando > 0.0

(3) I ∧ contador < N ⇒ def(individuo >= cota_sup)

def(individuo >= cota_sup) ⇔ cierto

I ∧ contador < N ⇒ cierto

(4) {I ⋀ contador < N ⋀ individuo >= cota_sup} <cociente, resto> := modulo_real(individuo, cota_sup) {I ⋀ contador < N ⋀ individuo >= cota_sup ⋀ 0.0 <= resto < cota_sup ⋀ ⋀ individuo = cota_sup * cociente + resto}

Demostrado en la corrección de la función ‘modulo_real’.

(5) {I ⋀ contador < N ⋀ individuo >= cota_sup ⋀ 0.0 <= resto < cota_sup ⋀ ⋀ individuo = cota_sup * cociente + resto} individuo := resto {I ⋀ contador < N ⋀ ⋀ individuo < cota_sup}

Calculamos pmd(individuo := resto, I ⋀ contador < N ⋀ individuo < cota_sup):

pmd(individuo := resto, I ⋀ contador < N ⋀ individuo < cota_sup)⬄

⬄ def(resto) ⋀ $(I \land \text{contador} < \text{N} \land \text{individuo} < \text{cota\_sup})^{resto}_{individuo}$ ⬄

⬄ cierto ⋀ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ 0 <= contador <= N ⋀

⋀ resto >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando > 0.0 ⋀ contador < N ⋀ resto < cota_sup ⬄

⬄ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ 0 <= contador < N ⋀

⋀ resto >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando > 0.0 ⋀ resto < cota_sup

Por último, demostramos que (I ⋀ contador < N ⋀ individuo >= cota_sup ⋀ ⋀ 0.0 <= resto < cota_sup ⋀ individuo = cota_sup * cociente + resto) ⇒ pmd(individuo : = resto, I ⋀ contador < N ⋀ individuo < cota_sup):

I ⋀ contador < N ⋀ individuo >= cota_sup ⋀ 0.0 <= resto < cota_sup ⋀ ⋀ individuo = cota_sup * cociente + resto ⬄

⬄ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ 0 <= contador < N ⋀ ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando > 0.0 ⋀ individuo >= cota_sup ⋀ ⋀ 0.0 <= resto < cota_sup ⋀ individuo = cota_sup * cociente + resto ⇒

⇒ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ 0 <= contador < N ⋀

⋀ resto >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando > 0.0 ⋀ resto < cota_sup

(6) {I ⋀ contador < N ⋀ individuo < cota_sup} poblacion[contador] := individuo; <sumando, contador> := <sumando + individuo, contador + 1> {I}

Denotamos:

A_1 ≡ poblacion[contador] := individuo

A_2 ≡ <sumando, contador> := <sumando + individuo, contador + 1>

Calculamos pmd(A_1; A_2, I):

pmd(A_1; A_2, I) ⬄ pmd (A_1, pmd(A_2, I))

Por tanto, primeramente debemos calcular pmd(A_2, I):

pmd(A_2, I) ⇔ pmd(<sumando, contador> := <sumando + individuo, contador + 1>, I) ⇔

⇔ def(sumando + individuo) ⋀ def(contador + 1) ⋀ $I_{sumando,contador}^{sumando+individuo,contador+1}$ ⇔

⇔ cierto ⋀ cierto ⋀ (∀ n : 0 <= n < contador + 1 : 0.0 <= poblacion[n] < cota_sup) ⋀
⋀ 0 <= contador + 1 <= N ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando + individuo > 0.0 ⇔

⇔ (∀ n : 0 <= n < contador + 1 : 0.0 <= poblacion[n] < cota_sup) ⋀ -1 <= contador < N ⋀
⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando + individuo > 0.0 ≡ P_1

pmd (A_1, pmd(A_2, I)) ⇔ pmd(poblacion[contador] := individuo, P_1) ⇔

⇔ pmd(poblacion := asig(poblacion, contador, individuo), P_1) ⇔

⇔ def(asig(poblacion, contador, individuo)) ⋀ ${P_1}_{poblacion}^{\text{asig(poblacion,contador,individuo)}}$ ⇔

⇔ 0 <= contador < N ⋀ -1 <= contador < N ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀

⋀ sumando + individuo > 0.0 ⋀

⋀ (∀ n : 0 <= n < contador + 1 : 0.0 <= asig(población, contador, individuo)[n] < cota_sup) ⇔

⇔ 0 <= contador < N ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando + individuo > 0.0 ⋀

⋀ (∀ n : 0 <= n < contador + 1 : (0.0 <= individuo < cota_sup ⋀ contador = n) ⋁

⋁ (0.0 <= poblacion[n] < cota_sup ⋀ contador ≠ n ⋀ 0 <= n < N))⇔

(*) Cálculos intermedios

⇔ 0 <= contador < N ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando + individuo > 0.0 ⋀

⋀ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ 0.0 <= individuo < cota_sup

(*)

0 <= contador < N ⋀ (∀ n: 0 <= n < contador + 1 : (0.0 <= individuo < cota_sup ⋀ contador = n) ⋁

⋁ (0.0 <= poblacion[n] < cota_sup ⋀ contador ≠ n ⋀ 0 <= n < N)) ⇔

⇔ 0 <= contador < N ⋀ (∀ n : 0 <= n < contador : (0.0 <= individuo < cota_sup ⋀ contador = n) ⋁

⋁ (0.0 <= poblacion[n] < cota_sup ⋀ contador ≠ n ⋀ 0 <= n < N)) ⋀

⋀ ((0.0 <= individuo < cota_sup ⋀ contador = contador) ⋁

⋁ (0.0 <= poblacion[contador] < cota_sup ⋀ contador ≠ contador ⋀ 0 <= contador < N)) ⇔

⇔ 0 <= contador < N ⋀ (∀ n : 0 <= n < contador : (0.0 <= individuo < cota_sup ⋀ falso) ⋁

⋁ (0.0 <= poblacion[n] < cota_sup ⋀ cierto ⋀ cierto)) ⋀

⋀ ((0.0 <= individuo < cota_sup ⋀ cierto) ⋁

⋁ (0.0 <= poblacion[contador] < cota_sup ⋀ falso ⋀ 0 <= contador < N)) ⇔

⇔ 0 <= contador < N ⋀ (∀ n : 0 <= n < contador : falso ⋁ (0.0 <= poblacion[n] < cota_sup)) ⋀

⋀ ((0.0 <= individuo < cota_sup) ⋁ falso) ⇔

⇔ 0 <= contador < N ⋀ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀

⋀ 0.0 <= individuo < cota_sup

Por último, demostramos que (I ⋀ contador < N ⋀ individuo < cota_sup) ⇒ pmd(A_1; A_2, I):

I ⋀ contador < N ⋀ individuo < cota_sup ⇔

⇔ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ 0 <= contador < N ⋀

⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando > 0.0 ⋀ individuo < cota_sup ⇒

⇒ 0 <= contador < N ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando + individuo > 0.0 ⋀

⋀ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ 0.0 <= individuo < cota_sup

(7) {I ⋀ contador >= N} nada {Q}

Calculamos pmd(nada, Q):

pmd(nada, Q) ⇔ Q

Por último, demostramos que (I ⋀ contador >= N) ⇒ Q:

I ⋀ contador >= N ≡ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀

⋀ 0 <= contador <= N ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando > 0.0 ⋀ contador >= N ⇔

⇔ (∀ n : 0 <= n < contador : 0.0 <= poblacion[n] < cota_sup) ⋀ contador = N ⋀

⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀ sumando > 0.0 ⇔

⇔ (∀ n : 0 <= n < N : 0.0 <= poblacion[n] < cota_sup) ⋀ individuo >= 0.0 ⋀ cota_sup > 0.0 ⋀

⋀ sumando > 0.0 ⇒

⇒ (∀ n : 0 <= n < N : 0.0 <= poblacion[n] < cota_sup) ≡ Q

(8) I ⋀ contador < N ⇒ C >= 0

I ⋀ contador < N ⇒ contador < N ⇒ N – contador > 0 ⇒ N – contador >= 0 ⇔ C >= 0

(9) {I ⋀ contador < N ⋀ C = T} individuo := individuo + sumando; si individuo >= cota_sup entonces <cociente, resto> := modulo_real(individuo, cota_sup); individuo := resto fsi poblacion[contador] := individuo; <sumando, contador> := <sumando + individuo, contador + 1> {C < T}

Denotamos:

$A_3 \equiv$ individuo := individuo + sumando

$A_4 \equiv$ si individuo >= cota_sup entonces <cociente, resto> := modulo_real(individuo, cota_sup); individuo := resto fsi

$A_5 \equiv$ poblacion[contador] := individuo

$A_6 \equiv$ <sumando, contador> := <sumando + individuo, contador + 1>

Calculamos pmd(A_3; A_4; A_5; A_6, $C < T$):

pmd(A_3; A_4; A_5; A_6, $C < T$) $\Leftrightarrow$ pmd(A_3; A_4; A_5, pmd(A_6, $C < T$))

Por tanto, en primer lugar debemos calcular pmd(A_6, $C < T$):

pmd(A_6, $C < T$) $\Leftrightarrow$ pmd(<sumando, contador> := <sumando + individuo, contador + 1>, $C < T$) $\Leftrightarrow$

$\Leftrightarrow$ def(sumando + individuo) $\wedge$ def(contador + 1) $\wedge$ $(C < T)^{sumando+individuo,contador+1}_{sumando,contador}$ $\Leftrightarrow$

$\Leftrightarrow$ cierto $\wedge$ cierto $\wedge$ (N – (contador + 1) < T) $\Leftrightarrow$ N – contador – 1 < T $\Leftrightarrow$ N – contador < T + 1

Como los predicados A_3, A_4 y A_5 no modifican ninguna de las variables presentes en pmd(A_6, $C < T$), podemos afirmar que pmd(A_3; A_4; A_5; A_6, $C < T$) $\Leftrightarrow$ pmd(A_6, $C < T$) $\wedge$ $\wedge$ 0 <= contador < N.

Por último, demostramos que (I $\wedge$ contador < N $\wedge$ C = T) $\Rightarrow$ pmd(A_6, $C < T$) $\wedge$ 0 <= contador < N:

I $\wedge$ contador < N $\wedge$ C = T $\Rightarrow$ 0 <= contador <= N $\wedge$ contador < N $\wedge$ C = T $\Leftrightarrow$

$\Leftrightarrow$ 0 <= contador < N $\wedge$ N – contador = T $\Rightarrow$ 0 <= contador < N $\wedge$ N – contador < T + 1