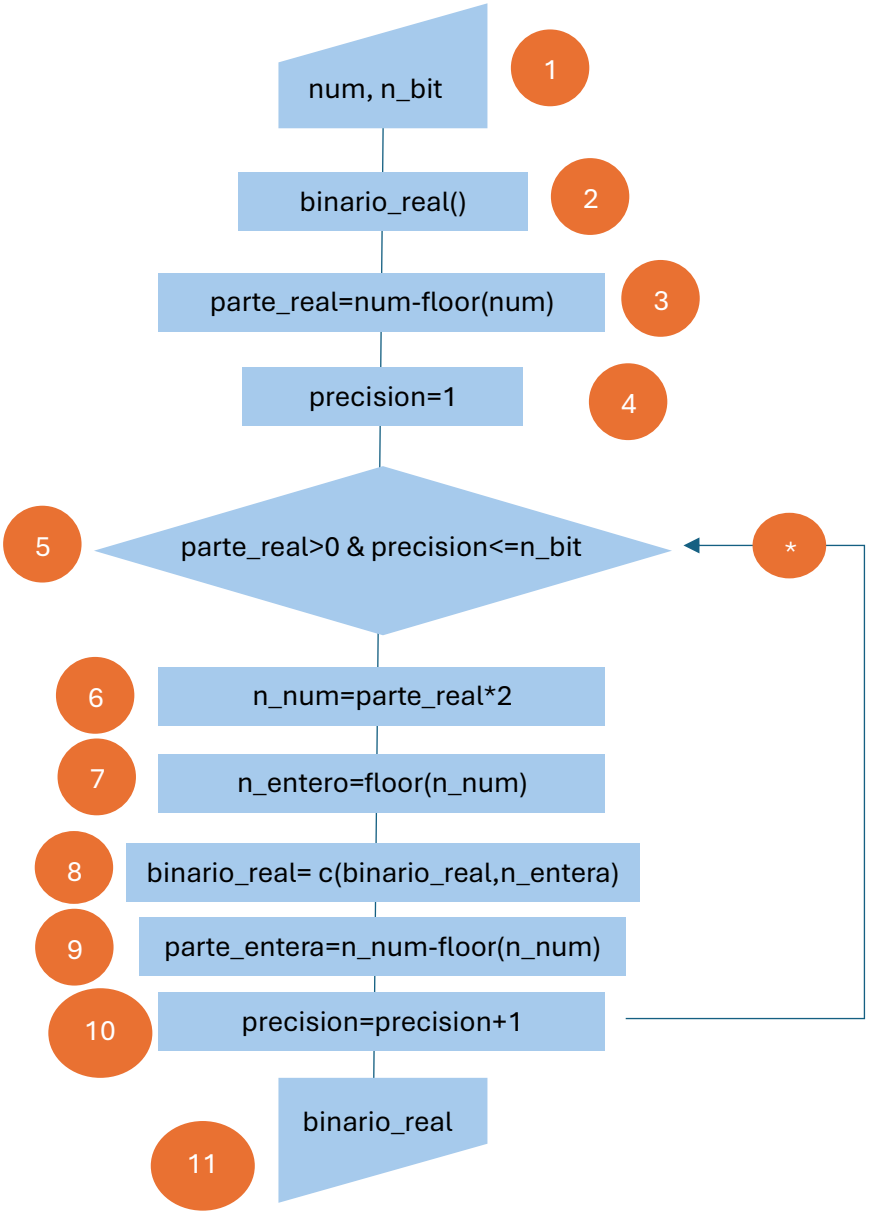
Recurso T6: Algoritmo (decimal a binario)

Solución al problema propuesto en clase el 6 de octubre de 2024: **Algoritmo para representar la parte real de un numero decimal en binario.**



EXPLICACIÓN PASO A PASO

1. Pedimos al usuario que introduzca los valores de las variables *num* y *n_bit*,

   num: número del que se extrae la parte decimal para transformarla en un número binario

   n_bit: número máximo de dígitos que tendrá el número binario

2. Establecemos el número binario *binario_real* como un vector, para poder completar los elementos de dicho vector con los valores del número binario

3. Para trabajar únicamente con la parte decimal del número (*num*), se establece una nueva variable*, parte_real*

   El valor de esta variable será el resultado de restar al número escogido (*num*) la parte entera, a través del comando *floor(num)*.

   parte*_real=num-floor(num)*

4. Damos a la variable precisión el valor de 1.

   Esta variable inicia un contador para seguir el número de bits que se están generando y nos asegura que el número de vueltas que va a dar el bucle while no sobrepasa el número máximo de bits que se haya establecido.

5. BUCLE WHILE:

   Mientras que se cumplan las dos condiciones establecidas, se ejecuta el cuerpo del bucle.

   - La primera condición (*parte_real*>0) consiste en comprobar que la parte real es mayor que 0.
   - La segunda condición (*precision* <= *n_bit*) controla que nuestro número binario no exceda los dígitos establecidos por la variable *n_bit*.

6. La primera instrucción del interior del bucle while define una nueva variable (*n_num*) que guarda la parte real multiplicada por 2:

   n_num = *parte_real* * 2

7. Se define la variable *n_entera*, la cual se queda con la parte entera del número decimal *n_num* calculado en el paso anterior.

8. Introducimos en la última posición (a la derecha) del vector *binario_real* el dígito guardado en la variable *n_entera*:

   binario_real = c(*binario_real*, *n_entera*)

9. Realizamos de nuevo el paso 3, esta vez aplicado a la variable *n_num* para quedarnos con su parte real.

10. Sumamos uno a la variable *precision,* que es la que lleva el contador para que no superemos el número de bits, ya que estamos sumando un bit por vuelta dada.

* Cuando el bucle termina de ejecutarse este revisa que las condiciones vuelven a cumplirse. En caso de que no, el bucle finaliza. El programa puede salir del bucle por dos razones:

1. Si *parte_real* == 0. En dicho caso ya tendremos todos los dígitos de la codificación en binario de la parte real del número introducido.

2. Si *precision* > *n_bit*, en cuyo caso habremos llegado al límite de dígitos que podíamos tener en nuestro vector *binario_real*.

11. Pedimos que el programa nos muestre el valor de la variable *binario_real*.

PROGRAMA REPRESENTATIVO EN R

```
1  cat("\014")
2  rm(list=ls())
3  num <- as.numeric(readline(prompt = "Introduce un número"))
4  n_bit <- as.numeric(readline(prompt = "Introduce un número de bit"))
5  binario_real=c()
6  parte_real=num-floor(num)
7  precision=1
8  while(parte_real>0 && precision<=n_bit){
9    n_num=parte_real*2
10   n_entera=floor(n_num)
11   binario_real=c(binario_real,n_entera)
12   parte_real=n_num-floor(n_num)
13   precision=precision+1
14 }
15 print(binario_real)
```

EJEMPLO DE EJECUCIÓN

Entrada:

- num = 0.625
- n_bit = 4

Proceso:

- parte_real = 0.625 – 0= 0.625

Bucle while:

- 0.625 * 2 = 1.25 → Nos quedamos con la parte entera (1)

- 0.25 * 2 = 0.5 → Nos quedamos con la parte entera (0)

- 0.5 * 2 = 1.0 → Nos quedamos con la parte entera (1)

- 0.0 * 2 = 0 → Como la parte real ya ha llegado a cero, esta parte entera no nos la quedamos y el bucle acaba aquí. Si no hubiese llegado a cero, sí que la contaríamos ya que n_bit = 4 y este sería el último valor.

Salida:

- binario_real = c (1,0,1)

Es decir, el número 0.625 se representa como 0.101 en binario.