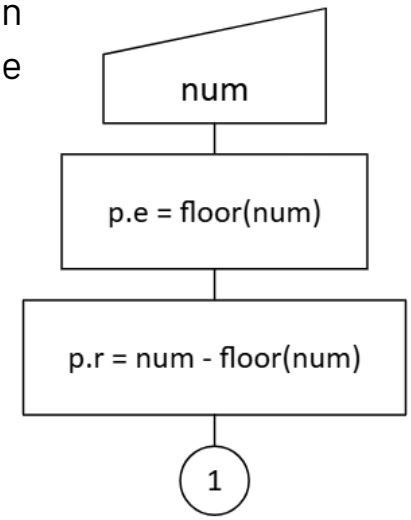
CODIFICACIÓN

La codificación es el proceso de convertir información en un formato legible por un ordenador para su almacenamiento y procesamiento. Aquí se trata de convertir números en código binario que utiliza tan solo los símbolos 0 y 1.

Habrá un programa para codificar la parte entera y otro para la parte real (es decir, después de la coma) de un número. Para separarlos se utiliza el comando **floor** que es una función que toma la parte entera de un número:

```
num <- as.numeric(readline(prompt =
"Introduce un número en base decimal: "))
p.e=floor(num)          # p.e = parte entera
p.r=num-floor(num)      # p.r = parte real
```



NOTA: los decimales se ponen con un punto, no con una coma

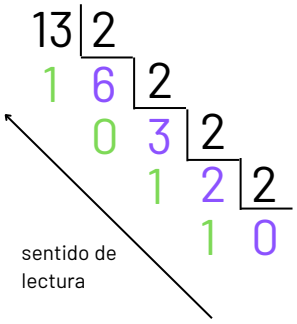
Para calcular la parte real de un número se le resta la parte entera.
Por ejemplo: 13.67 tiene como parte entera 13 y parte real 13.67-13=0.67.

IMPORTANTE: aunque el número introducido por el usuario sea entero, tenemos que crear una variable temporal (aquí p.e) porque no se puede aplicar la recursividad del programa en *num* dado que cuando queramos comunicar el resultado final al usuario ya no podremos utilizar *num* (el número introducido ya no estará dentro).

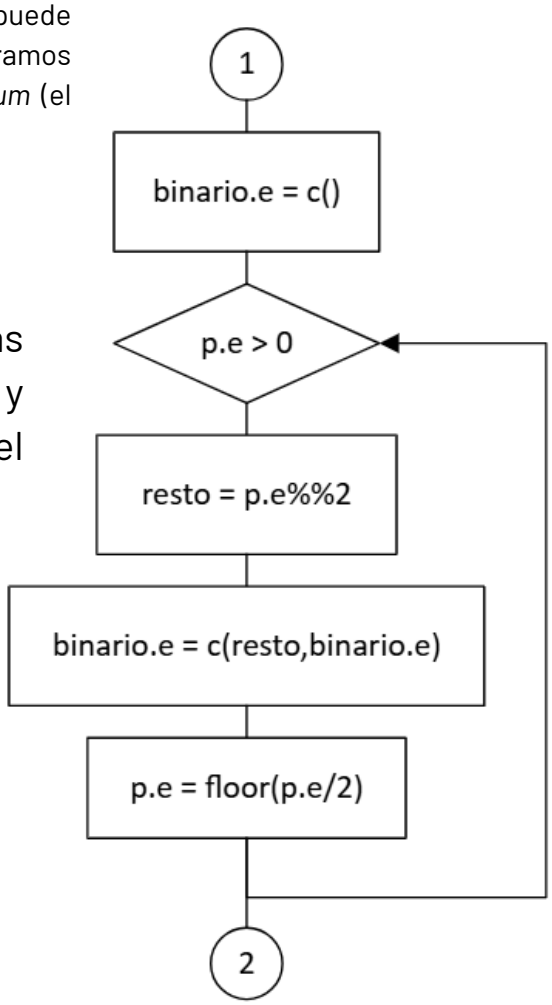
PARTE ENTERA

A mano para pasar de decimal a binario

Tenemos que hacer las divisiones sucesivas de las partes enteras resultantes de la division entre 2 y conservando los restos. Pararemos cuando el resto sea 0. Con el ejemplo anterior:



Partes enteras sucesivas.
Restos sucesivos, que corresponden al número en binario.
ATENCIÓN: el número binario se lee al revés
El código binario de 13 es 1101 y no 1011



Algoritmo para pasar de decimal a binario

Concatenar es un comando que une dos variables (¡no las suma!).

- Iniciálizamos la variable binario.e que en principio está vacía y no sabemos cuantos elementos concatenados tendrá (será una variable que contenga la secuencia binaria correspondiente a la parte entera).
- Es importante el orden de los elementos que contendrá esa variable que funciona como un vector.

Podremos por tanto concatenar a la izquierda: binario.e = c(resto, binario.e)
O bien a la derecha: binario.e = c(binario.e,resto). En este caso concatenamos a la izquierda porque el número en binario se lee al revés. "Los últimos serán los primeros".

Utilizamos un bucle **while** puesto que queremos efectuar las operaciones hasta que el resto sea nulo.

- resto = p.e%%2

#Almacenamos en la variable resto el resto de la división de la parte entera entre 2

- p.e = floor(p.e/2)

cogemos la parte entera resultante de esa división para poder usarla en la siguiente iteración

En R (continuación del programa anterior)

```
binario.e=c()

while (p.e>0){
    resto=p.e%%2
    binario.e=c(resto,binario.e)
    p.e=floor(p.e/2)
}
print(binario.e)
```

Tabla de seguimiento

num	binario.e	p.e	resto
Ej: 13,67	vacío	13	1
	1	6	0
	01	3	1
	101	1	1
	1101	0	

PARTE REAL

A mano para pasar de decimal a binario

En este caso tenemos que hacer las multiplicaciones por 2 sucesivas. La parte entera pasa a formar parte del binario y la parte decimal se vuelve a multiplicar. Paramos cuando el resultado sea 1 o se alcance un número determinado de bits. Con el ejemplo anterior: 13.67-13 = 0,67

0.67 * 2 = 1.34 → entero = 1
0.34 * 2 = 0.68 → entero = 0
0.68 * 2 = 1.36 → entero = 1
0,36 * 2 = 0,72 → entero = 0

sentido de lectura ↓

ATENCIÓN: el código binario no se lee al revés en este caso: 1010

Algoritmo para pasar de decimal a binario

- Creamos una variable precision que va a contabilizar el número de bits que se van añadiendo al código
- Se entra en el bucle while mientras se cumplan las dos condiciones: mientras la parte real sea superior a 0 y la precisión sea inferior al número de bits introducido (no olvidar que 1 bit = 1 número binario)

Este while simularía un for de un cierto modo ya que le impondríamos un número de iteraciónes máximo con el n_bit pero en este caso se deja la libertad de acabar antes si se llega antes a una parte real nula.

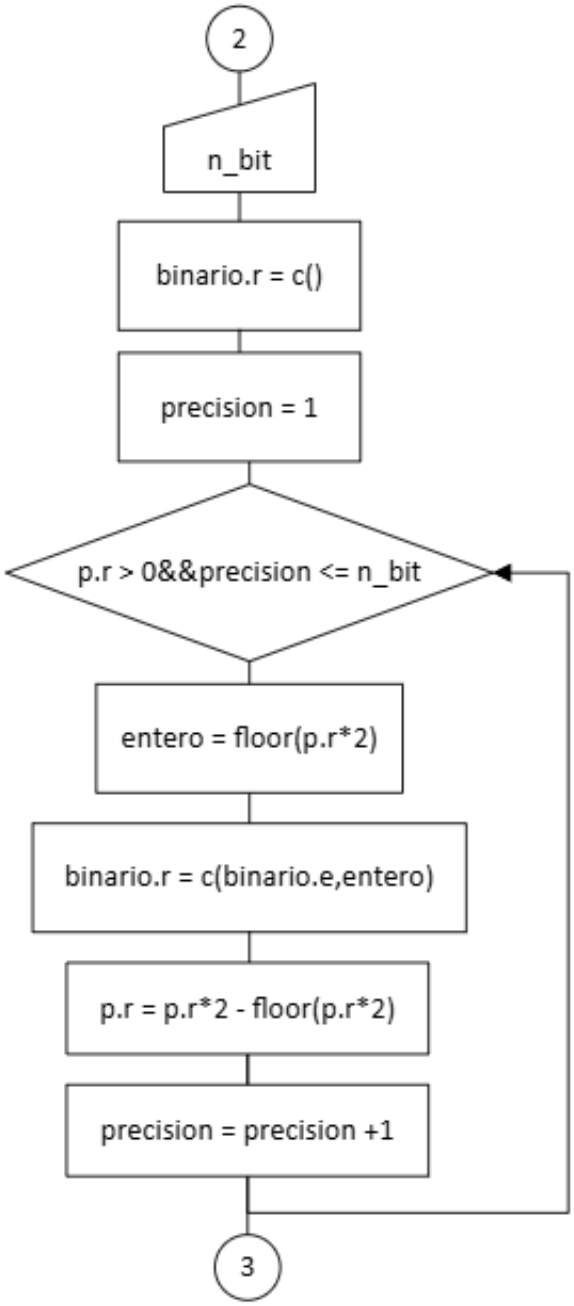


Tabla de seguimiento

num	n_bit	p.r	binario.r	precision	entero
Ej: 13,67	Ej: 4	0.67	vacío	vacío	1
		0,34	1	1	0
		0,68	10	2	1
		0,36	101	3	0
			1010	4	

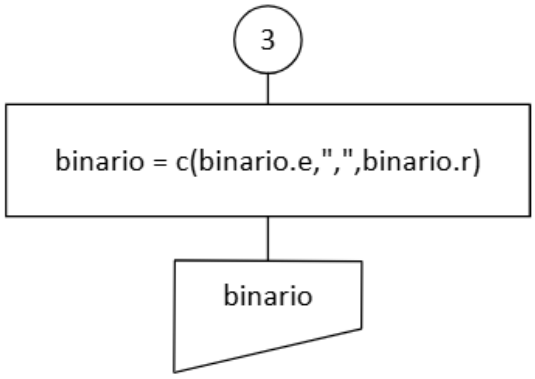
En R (continuación del programa anterior)

```
n_bit <- as.numeric(readline(prompt = "Introduce un número de bit: "))
binario.r=c()
precision=1
while(p.r>0 &&precision<=n_bit){
  entero=floor(p.r*2)
  binario.r=c(binario.r,entero)
  p.r=p.r*2-floor(p.r*2)
  precision=precision+1
}
print(binario.r)
```

COMBINACIÓN DE PARTE ENTERA Y PARTE REAL

Por último, debemos unir las dos variables: binario.e y binario.r (es decir, los códigos binarios de la parte entera y la parte real respectivamente) en una sola variable binario.

Para eso, concatenamos en la nueva variable binario.e a la izquierda y binario.r a la derecha (con una coma entre ambas, la cual se pone entre comillas, para diferenciar las dos partes)



En R (final del programa anterior)

```
binario=c(binario.e,",",binario.r)
print(binario)
```