PARTE DECIMAL A BINARIO

Código del programa

Primero de todo adjuntamos el código completo:

```
cat("\014")
rm(list=ls())
num<- as.numeric(readline(prompt = "Introduce el numero en base decimal "))
n_bit <- as.numeric(readline(prompt = "Introduce un número de bit"))
binario=c()
p_real=num-floor(num)
precision=1
while(p_real>0 && precision<=n_bit){
  n=p_real*2
  n_entera=floor(n)
binario=c(binario,n_entera)
p_real=n-floor(n)
precision=precision+1
}
print(binario)
```

Desglosado y explicación por partes

1. Limpiar la consola y el entorno de trabajo

```
cat("\014")
rm(list=ls())
```

cat("\014"): Este comando limpia la consola en R, es útil porque simplifica a la hora de poner nuevos "source" el no tener que ir limpiando todo el tiempo.

rm(list=ls()): Elimina todas las variables del campo de trabajo, asegurando que el entorno esté limpio antes de ejecutar el código y que no se genere confusión con variables anteriores.

2. Lectura de entrada

```
num<- as.numeric(readline(prompt = "Introduce el número en base decimal "))
n_bit <- as.numeric(readline(prompt = "Introduce un número de bit"))
```

num<- as.numeric(readline(prompt = "Introduce el número en base decimal ")): Muestra un mensaje pidiendo al usuario que introduzca un número decimal. Esto se almacena en la variable num.

n_bit <- as.numeric(readline(prompt = "Introduce un número de bit")): Solicita al usuario que introduzca el número de bits que quiere utilizar para representar la parte fraccionaria en binario.Esta variable lo que hace es determinar cuántos 1 y 0 tendrá nuestra cifra en binario,es decir actúa como agente delimitador del decimal.Si ponemos 5 bits,tendrá cinco cifras en binario= 5 bits,una para cada cifra. Este valor se almacena en la variable n_bit.

3. Variables

```
binario = c()
p_real = num - floor(num)
precision = 1
```

binario = c(): Se crea una lista vacía (vector) llamada binario, donde se irán guardando los dígitos de la parte fraccionaria binaria.Esta parte es clave porque si no no se almacenan las cifras en binario y no se puede obtener nuestro código binario.

p_real = num - floor(num):Se calcula la parte fraccionaria de num. floor(num) devuelve la parte entera del número, y al restarlo de num, obtienes la parte decimal o fraccionaria, que se almacena en p_real.Es una resta de una numero con decimal menos ese mismo número sin decimal,dando el decimal correspondiente.

precision = 1: Se inicializa una variable precision en 1. Esta variable sirve para contar cuántos bits de la fracción binaria se han calculado hasta ahora,perfectamente se podría empezar por 0 o 3 o 5,el problema es que aumenta mucho el número de bits que habría que poner.

4. Ciclo while de conversión a binario

```
while(p_real > 0 && precision <= n_bit) {
  n = p_real * 2
```

```
  n_entera = floor(n)
  binario = c(binario, n_entera)
  p_real = n - floor(n)
  precision = precision + 1
}
```

while(p_real > 0 && precision <= n_bit): Este ciclo while presenta dos condiciones a cumplir,separadas por un "&&" que literalmente significa "AND" lo que hace que se deban cumplir ambas.Entonces ejecuta el proceso de conversión fraccionaria mientras haya parte decimal por convertir (p_real > 0) y no se hayan alcanzado el número de bits deseados (precision <= n_bit).

n = p_real * 2 :Multiplica la parte fraccionaria (p_real) por 2. Esta es la base del proceso de conversión de fracción decimal a binario.Se hace así porque al multiplicar por 2 un decimal obtenemos la cifra binaria ya que sale "0,x" o "1,x" de ese decimal original(0 y 1).No es como con números sin decimales que se dividía entre 2 (posible error de concepto)

n_entera = floor(n):Se toma la parte entera de n. Si el valor es mayor o igual a 1, n_entera será 1; si es menor que 1, será 0. Este valor será el próximo bit de la fracción binaria.

binario = c(binario, n_entera):El bit (n_entera) se añade al vector binario, que está acumulando todos los bits de la representación binaria fraccionaria.

p_real = n - floor(n):Se actualiza la parte fraccionaria, restando la parte entera de n de sí misma, para así continuar el ciclo con la parte decimal restante.

precision = precision + 1:Se incrementa la variable precision para contar cuántos bits se han calculado.Recordamos que en bucle while se debe continuar hasta que se incumplan las condiciones establecidas.

5. Imprimir el resultado

print(binario)

print(binario):imprime el vector binario, que contiene la representación binaria de la parte fraccionaria del número decimal introducido con el número de bits comandado.

Tips y consejos

-Los números decimales ponerlos con puntos,no con comas.Si no da error.
-Recordar que podéis cambiar el nombre de las variables,de hecho si lo hacéis con otros nombres os ayudaría a entenderlo mejor ya que estaría ajustado a vuestro gusto.
-Aunque lo hayáis entendido,miraros la tabla de seguimiento porque es lo que sucede a tiempo real.

Tabla de seguimiento

Para rematar la explicación, dejamos un ejemplo(0.78 y 5 bits) para que veais que está pasando en cada ciclo numéricamente y como se van llenando las variables

ciclos	p_real	n	n_entera	binario	precision
Inicio	0.78	-	-	c()	1
1	0.78	1.56	1	c(1)	2
2	0.56	1.12	1	c(1, 1)	3
3	0.12	0.24	0	c(1, 1, 0)	4
4	0.24	0.48	0	c(1, 1, 0, 0)	5
5	0.48	0.96	0	c(1, 1, 0, 0, 0)	6
Fin	-	-	-	c(1, 1, 0, 0, 0)	6

Explicación de cada ciclo

1. Inicio:

-p_real: La parte decimal de 0.78, que es 0.78.
-precision: Inicialmente en 1.

2. Primer ciclo:

-n: $0.78 \times 2 = 1.56$
-n_entera: $\text{floor}(1.56) = 1$
-binario: Se añade 1, ahora c(1).
-p_real: $1.56 - 1 = 0.56$
-precision: Pasa a 2.

3. Segundo ciclo:

-p_real: Ahora es 0.56.
-n: $0.56 \times 2 = 1.12$
-n_entera: $\text{floor}(1.12) = 1$
-binario: Se añade otro 1, ahora c(1, 1).
-p_real: $1.12 - 1 = 0.12$
-precision: IPasa a 3.

4. Tercer ciclo:

-p_real: Ahora es 0.12.
-n: $0.12 \times 2 = 0.24$
-n_entera: $\text{floor}(0.24) = 0$
-binario: Se añade 0, ahora c(1, 1, 0).
-p_real: $0.24 - 0 = 0.24$

-precision: Pasa a 4.

5. Cuarto ciclo:

-p_real: Ahora es 0.24.

-n: 0.24×2=0.48

-n_entera: floor(0.48)=0

-binario: Se añade 0, ahora c(1, 1, 0, 0).

-p_real: 0.48−0=0.48

-precision: Pasa a 5.

6. Quinto ciclo:

-p_real: Ahora es 0.48.

-n: 0.48×2=0.96

-n_entera: floor(0.96)=0

-binario: Se añade 0, ahora c(1, 1, 0, 0, 0).

-p_real: 0.96−0=0.96

-precision: Pasa a 6.

7. Fin:

–Salida Final: El bucle se detiene porque precisión =6, que excede n_bit (5).

binario: La representación binaria es c(1, 1, 0, 0, 0).

Resultado

La representación binaria de 0.78 en 5 bits sería [1, 1, 0, 0, 0].

Esto representa 0.11000 en binario