Los números reales en coma flotante se convertían a binario en tres pasos:

1. Convertir al sistema binario
2. Escribir en notación científica
3. Seguir el standard IEEE754 para 32 bits

Por una parte la parte entera del número real se conviertía a binario y por otra la parte fraccionaria, según el algoritmo que se explicaba en el vídeo https://www.youtube.com/watch?v=VMcypTxcbvY. Este algoritmo debería haber sido el utilizado, no permitiéndose el uso de otros algoritmos.

En la práctica anterior, hemos terminado de convertir un número real en binario, según el standard IEEE754.

En esta práctica se generarán repetidamente, y de manera aleatoria, números reales en base decimal, que deberán ser convertidos a binario, recalculando finalmente el número real correspondiente al binario. La cantidad de números reales estará determinada por #define B, p.ej. #define B 4.

```
real random:
   14319.458008
numero real convertido a binario:
   11011111101111.0111010101
real recalculado:
   14319.458008

real random:
   9024.245117
numero real convertido a binario:
   10001101000000.0011111011
real recalculado:
   9024.245117

real random:
   11906.300781
numero real convertido a binario:
   10111010000010.01001101
real recalculado:
   11906.300781

real random:
   1761.237915
numero real convertido a binario:
   11011100001.0011110011101
real recalculado:
   1761.237915
```

Figura 1. Ejemplo de ejecución del programa

Los números **aleatorios** se generarán usando funciones predefinidas, de bibliotecas, de la siguiente manera:

```
#include "time.h"  // time()
#include "stdlib.h"  // srand(), rand()

//…
srand((int)time(NULL) );
aleatorio= rand();  // rand()%3 genera el aleatorio 0, 1 ó 2
//…
```

Se usará además, para el cálculo de la potencia, la función predefinida pow():

#include "math.h"
// float pow(float,float)

Se reutilizarán las funciones necesarias definidas en la práctica anterior, y además se definirán, y **se usarán**, **todas y cada una** de las funciones cuyos prototipos se dan en el siguiente recuadro.

Se añade también la restricción de que la función *longitud* se implementará de forma **recursiva**, no de forma iterativa.

```
// defines
#define maximo_chars 64
#define B 4


// prototipos de las funciones que se deben definir en esta práctica

int longitud(int );  // función recursiva
// longitud(543)=3

float random_real();
// devuelve un número real, calculado aleatoriamente

void real_binario(float real,char bits_binario[maximo_chars]);
// convierte un numero real en base decimal a base binaria, printando el binario

float numero_real(char b[maximo_chars]){
// devuelve en base 10 el numero real correspondiente a un binario con parte entera y fraccionaria


int main(){

    srand(time(NULL));
    float real;
    char cifra[maximo_chars];
    for(int cont=0; cont<B; cont++){
        resetear(cifra);
        real=random_real();
        printf("\nReal random: %f", real);
        printf("\nNumero real convertido a binario: ");
        real_binario(real, cifra);
        printf("\nReal recalculado: %f\n", numero_real(cifra));
    }

return 0;
}
```

```
// funciones definidas en prácticas anteriores y usadas en ésta

void binario_entera(int entera, char numero[maximo_chars]){
   float dec, ent=entera;
   for(int i=maximo_chars-1; ent>=1; i--){
      ent/=2;
      dec=ent-(int)ent;
      if(dec>=0.5)
         numero[i]='1';
      else
         numero[i]='0';
   }
}

void binario_fraccionaria(float decimal, char numero[maximo_chars]){
   do{
      decimal*=2;
      if(decimal<1)
         insertar_final(numero, '0');
      else{
         insertar_final(numero, '1');
         decimal-=1;
      }
   }while(decimal!=0);
}

void resetear(char numero[maximo_chars]){
   for(int i=0;i<maximo_chars;i++)
      numero[i]=' ';
}

void poner_posicion (char numero[maximo_chars],int n, char caracter){
   numero[n]=caracter;
}

int bits_blanco(char numero[maximo_chars]){
   int cont=0;
   for(int i=0; i<maximo_chars; i++)
      if(numero[i]==' ')
         cont++;
   return cont;
}

void insertar_final(char numero[maximo_chars],char c){
   for(int i=bits_blanco(numero)-1; i<maximo_chars-1; i++)
      poner_posicion(numero, i, numero[i+1]);
   poner_posicion(numero, maximo_chars-1, c);
}

void prn_binario(char numero[maximo_chars]){
   for(int i=0; i<maximo_chars; i++)
      printf("%c", numero[i]);
}
```

```
void mover_izda(char numero[maximo_chars]){
    for(int i=bits_blanco(numero), j=0; i<maximo_chars; i++, j++){
        poner_posicion(numero, j, numero[i]);
        poner_posicion(numero, i, ' ');
    }
}

int posicion_punto_decimal(char numero[maximo_chars]){
    int i;
    for(i=0; numero[i]!='.'; i++){}
    return i;
}

// funciones que se deben definir en esta práctica

int longitud(int num){
    int suma=1;
    if(num/10!=0)
        suma+=longitud(num/10);
    return suma;
}

float random_real(){
    float entera, fraccionaria, random;
    entera=rand();
    fraccionaria=rand();
    fraccionaria/=pow(10, longitud(fraccionaria));
    random=entera+fraccionaria;
    return random;
}

void real_binario(float num, char binario[maximo_chars]){
    binario_entera(num, binario);
    insertar_final(binario, '.');
    binario_fraccionaria(num-(int)num, binario);
    mover_izda(binario);
    prn_binario(binario);
}

float numero_real(char b[maximo_chars]){
    float real=0;
    int i, exp;
    for(i=posicion_punto_decimal(b)-1, exp=0; i>=0; i--, exp++)
        if(b[i]=='1')
            real+=pow(2,exp);
    for(i=posicion_punto_decimal(b)+1, exp=-1; b[i]!=' '; i++, exp--)
        if(b[i]=='1')
            real+=pow(2,exp);
    return real;
}
```