Los números reales en coma flotante se convierten 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 convierte a binario y por otra la parte fraccionaria, según el algoritmo que se explica en el vídeo https://www.youtube.com/watch?v=VMcypTxcbvY. Este algoritmo deberá ser el utilizado, **no permitiéndose** el uso de otros algoritmos.

En esta práctica, apoyándonos en la **anterior**, y siguiendo las especificaciones de la **Figura 1**, se realizarán los siguientes pasos:

1. Convertir un número real con signo a **binario**
2. Escribir el número binario en **notación científica**
3. Representar el número binario según el **standard IEEE754** para 32 bits

```
numero real en base decimal ? -134.3125
numero real convertido a binario:
  10000110.0101
numero binario en notacion cientifica:
  1.00001100101
exponente: 7
exponente+127 en binario:
  10000110
representacion en memoria:
  11000011000001100101000000000000
```

Figura 1. Ejemplo de ejecución del programa

Se deben usar los prototipos, y defines, indicados en el siguiente recuadro:

```
// defines
#define maximo_chars 64


// prototipos usados en la práctica anterior
void convertir_entero(int, char [maximo_chars]);
// un int se convierte a binario (se almacena en el array)

void convertir_fraccionario(float , char [maximo_chars]);
// un float se convierte a binario (se almacena en el array)

void resetear(char [maximo_chars]);  // se resetea el array

void colocar_posicion (char [maximo_chars],int , char );
// se coloca un char en la posicion int del array
```

```
int bits_blanco(char [maximo_chars]);
// chars en blanco en el array

void insertar_final(char [maximo_chars],char);
// se inserta un char al final del array, desplazando el resto a la izquierda

void printar_binario(char [maximo_chars]);
// se printa el array con los char del numero binario

// prototipos a definir en esta práctica

void mover_izda(char [maximo_chars]);
// mueve a la izda todos los bits del array

void scan_real(float * );
// scan del número real a convertir

int posicion_punto_decimal (char [maximo_chars]);
// posicion en el array de '.'

void notacion_cientifica(char [maximo_chars], int * );
// convierte el binario en notacion cientifica, transmitiendo el exponente

void copiar_mantisa(char [maximo_chars],char [32]);
// copia la mantisa a un array de 32 chars, en las últimos 23 posiciones del array

void copiar_exponente(char [maximo_chars],char r[32]);
// copia el exponente a un array de 32 chars, en las 8 siguientes posiciones a la posicion
0

void colocar_signo(char ,char [32]);
// coloca el signo en un array de 32 chars, en la posicion 0



int main(){
    float x, fraccionario = 0;
    int exponente, entero = 0;
    char arrayBinario[maximo_chars], aux[maximo_chars], signo;
    char arrayFinal[32];

    resetear(arrayBinario);
    resetear(aux);
    printf("Numero real en base decimal ?");
    scan_real(&x);
    if(x >= 0){
        signo = '0';
    } else {
        signo = '1';
    }
    entero = (int)x;
    fraccionario = x - entero;
```

```
    convertir_entero(entero, arrayBinario);
    convertir_fraccionario(fraccionario, arrayBinario);

    printf("Numero real convertido en binario: ");
    printar_binario(arrayBinario);
    printf("\nNumero binario en notacion cientifica: ");
    notacion_cientifica(arrayBinario, &exponente);
    printar_binario(arrayBinario);
    printf("\nexponente: %d", exponente);
    printf("\nexponente + 127 en binario: ");
    exponente += 127;
    convertir_entero(exponente, aux);
    printar_binario(aux);

    copiar_exponente(aux, arrayFinal);
    copiar_mantisa(arrayBinario, arrayFinal);
    colocar_signo(signo, arrayFinal);

    printf("\nrepresentacion en memoria: ");
    for(int i=0; i<32; i++){
        printf("%c", arrayFinal[i]);
    }
return 0;
}

// funciones a definir en esta práctica

void mover_izda(char array[maximo_chars]){
    for(int i=bits_blanco(array);i<maximo_chars;i++){
        array[i-1] = array[i];
    }
    array[maximo_chars-1] = ' ';
}

void scan_real(float *numero_real){
    scanf("%f", numero_real);
    /*do{
    printf("numero real en base decimal ?");
    scanf("%f", numero_real);
    } while(*numero_real < -2147483648 || *numero_real > 2147483647);*/
}

int posicion_punto_decimal(char array[maximo_chars]){
    int encontrado = 0;
    int i = maximo_chars - 1;
    while(!encontrado && i<maximo_chars){
        if(array[i] == '.'){
            encontrado = 1;
        } else{
            i--;
        }
    }
```

```
    return encontrado;
}

void notacion_cientifica(char array[maximo_chars], int *numero_real){
    int x = bits_blanco(array);
    int pos = posicion_punto_decimal(array);
    *numero_real = (pos-x)-1;
    int i;
    for(i = pos; i>=x; i--){
        array[i] = array[i-1];
    }
    colocar_posicion(array, x, '1');
    colocar_posicion(array, x+1, '.');
}

void copiar_mantisa(char array[maximo_chars],char arrayFinal[32]){
    int x = posicion_punto_decimal(array)+1;
    for(int i=9; i<32; i++){
        if(x<maximo_chars){
            arrayFinal[i]=array[x];
            x++;
        }else{
            arrayFinal[i]='0';
        }
    }
}
void copiar_exponente(char array[maximo_chars],char arrayFinal[32]){
    int j = bits_blanco(array);
    for(int i=1; i<=8; i++){
        arrayFinal[i]=array[j];
        j++;
    }
}

void colocar_signo(char signo, char array[32]){
    array[0] = signo;
}
```