**Trabajo práctico**

**Representación de números reales en formato punto flotante**

**Ejercicio 1**

Analice y compile el archivo ex\_01.c.

1. Inspeccione el código y determine el objetivo del programa.

2. Analice los valores de las variables f1, f2, d1, acum\_1 y acum\_2.

3. ¿Qué conclusión puede obtener a partir de estos valores?

**Ejercicio 2. Modos de redondeo**

Analice y compile el archivo ex\_02.c.

1. ¿Para qué sirven las funciones fegetround() y fesetround()?

2. ¿Cuál es el modo de redondeo por defecto con el que arranca el programa?

3. Ejecutar el programa para los modos de redondeo FE\_DOWNWARD, FE\_UPWARD y FE\_TOWARDZERO, y compárelos con el modo FE\_TONEAREST.

4. ¿Observa diferencias? ¿Estas diferencias son consistentes con los modos de redondeos?

**Ejercicio 3. Valores especiales**

Investigue cómo inicializar en C variables en punto flotante con los valores especiales NaN, Inf e -Inf. Además, escriba un programa en C que produzca como resultado estos valores especiales.

**Ejercicio 4. Excepciones**

Analice y compile el archivo ex\_04.c.

1. Analice los resultados impresos por consola ¿Son los resultados consistentes con las operaciones ejecutadas?

2. Explique qué hacen las funciones feclearexcept(), feraiseexcept() y fetestexcept().

**Ejercicio 5. Excepciones**

Con el código visto en el ejercicio 4, ex\_04.c. genere los 4 tipo de excepciones estudiadas al ejecutar diferentes operaciones matemáticas, en lugar de utilizar la función feraiseexcept().

**Ejercicio 6. Manejo de excepciones**

Analice y compile el archivo ex\_05.c.

3. Analice los resultados impresos por consola ¿Son los resultados consistentes con las operaciones ejecutadas?

4. Descomentar las líneas 44 y 45 y volver a compilar.

5. ¿Qué observa por consola? ¿Cuál es la función de feenableexcept()?

6. Descomentar la línea 47 y volver a compilar.

7. ¿Qué observa por consola? ¿Cuál es el objetivo de la función signal(SIGFPE, fpe\_handler)?

**Ejemplo de resolución**

**Ejercicio 1**

1. El objetivo es ver cuál es el error que se comete cuando se cambia el orden las operaciones en punto flotante, donde influye mucho con qué precisión se realiza cada una de las operaciones intermedias. En las operaciones de acumulación aparece un error debido a sumar un número muy chico a un número muy grande.
2. En f1, primero se multiplican números muy distintos; y se obtiene un error por exceso. En f2 se multiplican primero números parecidos (iguales); y se obtiene un error por defecto. En acum\_1 se suman números muy distintos, por ejemplo 0.01 + 1000000. En acum\_2 los números sumados no son tan distintos, por ejemplo 1.00 + 1000000.
3. Hay que tener cuidado al operar con números muy distintos en punto flotante.

**Ejercicio 2. Modos de redondeo**

1. Para conocer y cambiar el modo de redondeo.
2. Arranca en FE\_TONEAREST
3. Si no se compila con los modificadores indicados en el encabezado, el modo de redondeo no cambia efectivamente.
4. Con el modo FE\_TONEAREST, redondea al más cercano y por lo tanto exp(1.1) y -exp(1.1) tienen el mismo valor absoluto.  
   Con el modo FE\_DOWNWARD, redondea al inmediato inferior, por lo tanto tienen valores absolutos distintos, teniendo mayor valor absoluto -exp(1.1), -3.004166 contra +3.004165  
   Con el modo FE\_UPWARD, redondea al inmediato superior, por lo tanto exp(1.1) tiene mayor valor absoluto que - exp(1.1), +3.004166 contra -3.004165  
   Con el modo FE\_TOWARDZERO, redondea al inmediato más próximo a cero, por lo tanto tienen igual valor absoluto; pero al menos en este caso, el valor absoluto del error resultó mayor que con FE\_TONEAREST.

**Ejercicio 3. Valores especiales**

feclearexcept() borra las últimas excepciones tiradas

feraiseexcept() tira una excepción

fetestexcept() se fija si se ha tirado una determinada excepción

**Ejercicio 5. Excepciones**

Hecho en ex\_05\_res.c