Trabajo práctico

Programación en C usando números en punto flotante

Ejercicio 1
Compile el siguiente código en C:

```
#include <stdio.h>
#include <float.h>
#include <math.h>
#include <fenv.h>

typedef float  fp32_t;
typedef double fp64_t;

int main(void)
{
    fp32_t a, b, c, r1, r2;
    fp64_t dr3;

    a = 1000000000.0;
    b =   20000000.0;
    c =   20000000.0;

    r1 = (a * b) * c;
    r2 = a * (b * c);

    dr3 = (double) (a) * ( (double) (b) * (double) (c) );

    printf("r1 = %f \n", r1 );
    printf("r2 = %f \n", r2 );
    printf("dr3 = %lf \n", dr3 );

    printf("error r1 = %10e \n", r1 - 4000000000000000000000000.0 );
    printf("error r2 = %10e \n", r2 - 4000000000000000000000000.0 );

    return 0;
}
```

1. Inspeccione el código y determine el objetivo del programa
2. Analice los valores de las variables `r1` y `r2`.
3. ¿Qué conclusión puede obtener a partir de estos valores?

Ejercicio 2
Compile el siguiente código en C:

```
#include <stdio.h>
#include <float.h>
#include <math.h>
#include <fenv.h>

typedef float  fp32_t;
typedef double fp64_t;

int main(void)
{
    fp32_t acc1, acc2;

    acc1 = 0.0;
    for (int i = 0; i < 10000; i++){ acc1 += 0.100; }

    acc2 = 0.0;
    for (int i = 0; i < 10000; i++){ acc2 += 0.125; }

    printf("acc1 = %.20lf \n", acc1 );
    printf("acc2 = %.20lf \n", acc2 );

    printf("acc1 error = %.20e \n", 1000 - acc1 );
    printf("acc2 error = %.20e \n", 1250 - acc2 );

    return 0;
}
```

1. Inspeccione el código y determine el objetivo del programa
2. Analice los valores de las variables `acc1` y `acc2`.
3. ¿Qué conclusión puede obtener a partir de estos valores?

Ejercicio 3
Encuentre los valores particulares de `b` y `c` en formato punto flotante precisión simple que producen que los números `a1` y `a2` no sean iguales. Recuerde que en C las variables especiales *Not a Number* y infinito se definen como `NAN` e `INFINITY`, respectivamente.

1. `a1 = b + b * c ,    a2 = b * (1.0 + c)`
2. `a1 = b / 10.0 ,     a2 = b * 0.1`
3. `a1 = b / b ,        a2 = 1.0`
4. `a1 = b - b ,        a2 = 0.0`
5. `a1 = b + 0.0 ,      a2 = b`

Ejercicio 4
Analice y compile el archivo `ex_04.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. Ejecute 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 5

Analice y compile el archivo `ex_05.c`.

1. Analice los resultados impresos por consola ¿Son los resultados consistentes con las operaciones ejecutadas?
2. Explique que hacen las funciones `feclearexcept()`, `feraiseexcept()` y `fetestexcept()`.

Ejercicio 6

Analice y compile el archivo `ex_06.c`.

3. Analice los resultados impresos por consola ¿Son los resultados consistentes con las operaciones ejecutadas?
4. Descomente las líneas 38 a 43 y vuelva a compilar.
5. ¿Qué observa por consola? ¿Cuál es la función de `feenableexcept()`?
6. Descomente la línea 36 y vuelva a compilar.
7. Qué observa por consola? ¿Cuál es la función de `signal(SIGFPE, fpe_handler)`?