EJERCICIO 1

El objetivo del presente ejercicio es desarrollar un programa que permita averiguar cual es el numero mas pequeño que puede ser sumado con uno sin que se pierda información.

Comprobaremos además las diferencias en este aspecto de los tipos de datos REAL y DOBLE.

Los resultados obtenidos son los siguientes:

TIPO REAL

Iteración 40:

El valor de la x = 9.09494701772928E−13

El numero mas pequeño que aporta información a uno es:

9.09494701772928E−13

TIPO DOBLE

Iteración 52:

El valor de la x = 2.22044604925031E−16

El numero mas pequeño que aporta información a uno es:

2.22044604925031E−16

En el programa realizado sumaremos a 1 el valor de una variable, la cual vamos dividiendo por 2 en cada interacción, con lo que provocamos un desplazamiento a la derecha de la mantisa, por lo que los valores obtenidos realmente no son los valores mínimos que aportan información a uno, sino los que podemos representan en los formatos REAL y DOBLE.

LISTADO DEL PROGRAMA

```
Program ejer1;

uses

wincrt;

var

x,y:real;

j,k:double;
```

```
i:integer;
−−>[Author:(null)]begin
x:=1;
y:=0;
i:=0;
while y<>1 do
begin
i:=i+1;
x:=x/2;
y:=1+x;
writeln('Numero de iteración: ',i,' El valor de la x es: ',x,' El valor de la y es: ',y);
end;
writeln;
writeln('El minimo numero que aporta información a 1 para tipo real es: ',x*2,' en la iteracion: ',i−1);
readln;
j:=1;
k:=0;
i:=0;
while k<>1 do
begin
i:=i+1;
j:=j/2;
k:=1+j;
writeln('Numero de iteración: ',i,' El valor de la j es: ',j,' El valor de la k es: ',k);
end;
writeln;
```

```
writeln('El minimo numero que aporta información a 1 para tipo double es: ',j*2,' en la iteracion: ',i−1);

end.
```

MODIFICACIÓN

Si modificamos el programa para que en cada iteracion el numero almacenado en x se divida por 10 en vez de por 2, los resultados obtenidos son los siguientes:

TIPO REAL:

Iteración: 13

El número mas pequeño que aporta información: 9.99999999999273E−14

TIPO DOUBLE:

Iteración: 16

El numero mas pequeño que aporta información: 1.00000000000000E−16

Como podemos observar el numero de iteraciones obtenidas al realizar la modificación del programa es mucho menor que en el programa original, sin embargo, la información obtenida en esta modificación es menos fiable ya que no son desplazamientos exactos de la mantisa, cosa que si ocurría en el programa original.

LISTADO DEL PROGRAMA

```
Program ejer1;

uses

wincrt;

var

x,y:real;

j,k:double;

i:integer;

begin

x:=1;

y:=0;
```

```
i:=0;
while y<>1 do
begin
i:=i+1;
x:=x/10;
y:=1+x;
writeln('Numero de iteración: ',i,' El valor de la x es: ',x,' El valor de la y es: ',y);
end;
readln;
writeln;
writeln('El minimo numero que aporta información a 1 para tipo real es: ',x*10,' en la iteracion: ',i−1);
j:=1;
k:=0;
i:=0;
while k<>1 do
begin
i:=i+1;
j:=j/10;
k:=1+j;
writeln('Numero de iteración: ',i,' El valor de la j es: ',j,' El valor de la k es: ',k);
end;
writeln;
writeln('El minimo numero que aporta información a 1 para tipo double es: ',j*10,' en la iteracion: ',i−1);
end.
```

EJERCICIO 2

En el presente ejercicio deberemos obtener el mínimo número que se puede representar tanto en formato REAL, como en formato DOUBLE.

Una vez ejecutado el programa los resultados obtenido han sido los siguientes.

TIPO REAL:

Iteración 126: x=1.17549435082229E−38

Iteración 127: x=5.87747175411144E−38

Iteración 128: x=2.93873587705572E−39

El minimo numero que se puede representar en este formato es:

2.93873587705572E−39

TIPO DOUBLE:

Iteracion 1072: x=1.97626258336499E−323

Iteracion 1073: x=9.88131291682493E−324

Iteracion 1074: x=4.94065645841247E−324

El minimo numero que se puede representar en este formato es:

4.94065645841247E−324

LISTADO DEL PROGRAMA

```
Program ejer2;

uses

wincrt;

var

x,y:real;

j,k:double;

i:integer;

begin

x:=1;

y:=0;
```

```
i:=0;
repeat
i:=i+1;
y:=x;
x:=x/2;
writeln('Iteracion numero ',i,'x:= ',x);
until x=0;
writeln('El minimo numero que se puede representar en formato real es: ',y);
readln;
j:=1;
i:=0;
k:=0;
repeat
i:=i+1;
k:=j;
j:=j/2;
writeln('Iteracion numero ',i,'x:= ',j);
until j=0;
writeln;
writeln('El minimo numero que se puede representar en formato double es:',k);
end.
```

EJERCICIO 3

Escribe un programa que resuelva utilizando la regla de Crámer el siguiente sistema de ecuaciones:

10000000000x+y=10000000001

10000000001x+y=10000000002

Aplícalo al mismo sistema pero con un cero menos para los coeficientes y términos indepedientes. Comenta los resultados obtenido.

```
program ej3pr1;

uses wincrt,windos;

const

x1=1000000.0;

y1=1.0;

t1=1000001.0;

x2=1000001.0;

y2=1.0;

t2=1000002.0;

var

temp:real;

x,y:real;

begin

clrscr;

writeln ('EJEMPLO DE PROBLEMA MAL CONDICIONADO.');

writeln;

writeln (' ',x1,'x+',y1,'y = ',t1);

writeln (' ',x2,'x+',y2,'y = ',t2);

writeln;

writeln (' Solución: ');

temp:=(x1*y2)−(x2*y1);

x:=((t1*y2)−(t2*y1))/temp;

y:=((x1*t2)−(x2*t1))/temp;

writeln (' x=',x);
```

```
writeln (' y=',y);

repeat until keypressed;

end.
```

Una vez ejecutado el programa los resultados obtenidos han sido los siguientes

x = 1;

y = 0;

A continuación hemos ejecutado el mismo programa pero realizando una pequeña variación en el sistema de ecuaciones a resolver:

1000000000x + y = 10000000001

1000000001x + y = 10000000002

Los resultados obtenidos han sido los que mostramos a continuación:

x= 1;

y= 1;

Esta variación en los resultados se debe a que en el primer caso, al realizar la división se ha excedido el rango del tipo de datos REAL, por lo que se redondea el valor y da un resultado incorrecto. En el segundo caso, al realizar la operación con un cero menos no se produce ester error y el resultado es correcto.

EJERCICIO 4

Escribe un algoritmo que calcule el producto de dos matrices. Calcula la complejidad temporal del algoritmo a priori y realiza unas pruebas de tiempo a posteriori.

Los conceptos de coste y complejidad temporal de un algoritmo se han estudiado en un problema matemático como es la multiplicación de dos matrices. Dicha complejidad se puede estudiar a priori, analizando el numero de pasos, funciones y procedimientos recursivos del programa, y a posteriori, analizando el tiempo que tarda el programa en resolver varios casos distintos.

Estudiando el algoritmo empleado a priori se han obtenido los siguientes resultados:

El algoritmo incluye tres bucles de repetición enlazados uno dentro de otro, todos ellos desde 1 hasta n, luego el coste de multiplicar cualquier matriz será:

bucle 1: 1n

bucle 2: 1n

bucle 3: 1n

coste(mult) *= n*n*n = n3* ***O(n3)***

Por lo tanto a medida que aumente el orden de la matriz de manera lineal, el coste de su multiplicación aumentará de forma cúbica.

Analogamente, estudiando los tiempos obtenidos con el programa, para los casos de matrices cuadradas de orden 5, 10, 15, 20, 25, 30, 35, 40, 45 y 50, representados en una gráfica, se ha visto que efectivamente se cumplen los resultados obtenidos a priori, es decir, a medida que aumenta el rango de las matrices multiplicadas, el coste temporal del programa aumenta de manera cúbica.

Rango de la matriz	Tiempos
10	0.5
15	2.2
20	5.6
25	10
30	17.6
35	27.4
40	40.6
45	58.2
50	80.2