# Problemas con enteros

## Problema 1.º - Secuencia de Collatz

La siguiente función escribe en la pantalla la secuencia de Collatz que comienza en el valor de su parámetro n. ¿Qué debería escribir en pantalla cuando se invoca con n = 13? ¿Y con n = 6?

```
/*
 * Pre:  n > 0
 * Post: Ha escrito en la pantalla la secuencia
 *       de Collatz que comienza en «n».
 */
void escribirSecuenciaCollatz(int n) {
    cout << n;
    while (n > 1) {
        if (n % 2 == 0) {
            n = n / 2;
        }
        else {
            n = 3 * n + 1;
        }
        cout << ", " << n;
    }
    cout << endl;
}
```

Lo escrito en pantalla con la invocación `escribirSecuenciaCollatz(13)`:

Lo escrito en pantalla con la invocación `escribirSecuenciaCollatz(6)`:

## Problema 2.º - Divisores

Diseña un programa que solicite al operador un número entero positivo y escriba en la pantalla un listado de sus divisores, a razón de uno por línea. Se muestra a continuación un ejemplo de ejecución del programa solicitado cuando el usuario introduce el número 28:

```
Escriba un número entero positivo: 28

DIVISORES DE 28:
1
2
4
7
14
28
```

## Problema 3.º - Números perfectos

Parte del control de prácticas del 24-6-2004

Diseña un programa que solicite al usuario un número entero positivo (asegurándose de que es positivo, volviendo a pedir un nuevo número si es preciso) y que le indique si se trata de un número perfecto o no. Un número perfecto es aquel que es igual a la suma de sus divisores propios (es decir, sus divisores positivos menores que él mismo). Así, 6 es un número perfecto, porque sus divisores propios son 1, 2 y 3; y 6 = 1 + 2 + 3. Los siguientes números perfectos son 28, 496 y 8128.

A continuación se muestran ejemplos de ejecución del programa solicitado:

```
Escriba un número entero positivo: 496
496 es un número perfecto.
```

```
Escriba un número entero positivo: 32
32 no es un número perfecto.
```

```
Escriba un número entero positivo: -8
El número debe ser entero positivo: -125
El número debe ser entero positivo: 8128
8128 es un número perfecto.
```

## Problema 4.º - Múltiplos de 3 y 5

Problema 1 de Project Euler[1]

Si se listan todos los números naturales inferiores a 10 que son múltiplos de 3 o 5, se obtienen los números 3, 5, 6 y 9. La suma de estos números es 23. Halla la suma de todos los múltiplos de 3 o 5 menores que 1000.

## Problema 5.º - Números capicúas

Añade al módulo `calculos` (capítulo 7 de los apuntes de la asignatura[2]) una función denominada `esCapicua` que devuelva el valor booleano *cierto* si y solo si el valor de su parámetro de tipo entero es un número capicúa. No olvides especificarla antes con su precondición y postcondición.

Evidentemente, puedes usar todas las funciones del módulo `calculos` que necesites.

## Problema 6.º - Números repitunos

Adaptación a C++ del problema 1.º del examen del 3-9-2014 (1 punto)

En matemáticas recreativas, un *número repituno* es un número formado exclusivamente con el dígito 1. Así, 1, 11, 111 y 1111 son ejemplos de números repitunos.

Se define el *orden* de un número repituno como el número de cifras que lo componen. Así, el orden de 11 es 2 y el orden de 111111 es 6.

Se debe escribir el código de la función `ordenRepituno` cuya especificación se muestra a continuación:

```
/*
 * Pre:  n > 0
 * Post: Si «n» es un número repituno, ha devuelto el orden del mismo.
 *       En caso contrario, ha devuelto -1.
 */
int ordenRepituno(int n);
```

[1] https://projecteuler.net/problem=1
[2] http://webdiis.unizar.es/asignaturas/PROG1/doc/programacionCPP/capitulo7/

## Problema 7.º - Eliminar cifras

Diseña la siguiente función:

```
/*
 * Pre:  n ≥ 0 y 0 ≤ c ≤ 9
 * Post: Ha devuelto un entero que, escrito en base 10, equivale al resultado de
 *       suprimir todas las ocurrencias de la cifra «c» en el entero «n» cuando se
 *       escribe también en base 10.
 *       Ejemplos:
 *           quitarCifra(902037122, 0) = 9237122
 *           quitarCifra(902037122, 1) = 90203722
 *           quitarCifra(902037122, 2) = 900371
 *           quitarCifra(902037122, 3) = 90207122
 *           quitarCifra(902037122, 4) = 902037122
 *           quitarCifra(902037122, 9) = 2037122
 */
int quitarCifra (int n, int c);
```

## Problema 8.º - Permutar cifras

Diseña la siguiente función:

```
/*
 * Pre:  n >= 0
 * Post: Ha devuelto un entero que, escrito en base 10, equivale al resultado de
 *       permutar cada cifra significativa de «n» que ocupa una posición impar cuando
 *       se escribe en base 10 con la cifra de posición par situada a su izquierda
 *       (se permutan unidades con decenas, centenas con millares y, así
 *       sucesivamente, las cifras significativas de «n»).
 *       Ejemplos:
 *           permutarCifras(12345678) = 21436587
 *           permutarCifras(123456789) = 132547698
 *           permutarCifras(10407) = 14070
 *           permutarCifras(104073) = 10437
 */
int permutarCifras(int n);
```

## Otros problemas

Capítulo 3 de la colección de problemas de la asignatura (sección Material docente común, *Problemas de Programación 1*[3]).

## Otros problemas de Project Euler

Problema 2: Números de Fibonacci pares
Problema 5: Menor múltiplo
Problema 6: Suma de cuadrados vs. cuadrado de la suma
Problema 7: Diezmilésimo primer número primo
Problema 12: Números triangulares altamente divisibles

[3] http://webdiis.unizar.es/asignaturas/PROG1/doc/materiales/problemasProg1.pdf