RELACIÓN DE PROBLEMAS II. Estructuras de Control

Ejercicios sobre condicionales

1. Ampliad el ejercicio 13 de la *Relación de Problemas I* de manera que los dos instantes puedan pertenecer a dos días consecutivos.

   Finalidad: Trabajar con condicionales simples. Dificultad Baja.

2. Cread un programa que lea el valor de la edad (dato de tipo entero) y salario (dato de tipo real) de una persona. Subid el salario un 5 % si éste es menor de 750 euros y la persona es mayor de 55 años.

   ¿Es mejor incluir otra variable nueva `salario_final` o es mejor modificar la variable que teníamos?

   Imprimid el resultado por pantalla. Si no procede la subida imprimid el mensaje `"No es aplicable la subida"`. En ambos casos imprimid el salario final.

   Finalidad: Plantear una estructura condicional con una expresión lógica compuesta. Dificultad Baja.

3. Realizar un programa que lea dos valores enteros desde teclado y diga si cualquiera de ellos divide o no (de forma entera) al otro. En este problema no hace falta decir quién divide a quién. Supondremos que los valores leídos desde teclado son ambos distintos de cero.

   Finalidad: Plantear una estructura condicional doble con una expresión lógica compuesta. Dificultad Baja.

4. Escribid un programa que lea tres enteros desde teclado y nos diga si están ordenados (da igual si es de forma ascendente o descendente) o no lo están.

   Finalidad: Plantear una estructura condicional doble con una expresión lógica compuesta. Dificultad Baja.

5. Cread un programa que lea el número de un año e indique si es bisiesto o no. Un año es bisiesto si es múltiplo de cuatro, pero no de cien. Excepción a la regla anterior son los múltiplos de cuatrocientos que siempre son bisiestos. Por ejemplo, son bisiestos: 1600,1996, 2000, 2004. No son bisiestos: 1700, 1800, 1900, 1998, 2002.

   Finalidad: Plantear una estructura condicional con una expresión lógica compuesta. Dificultad Baja.

6. La tabla para el cálculo del precio a pagar en los parkings de Madrid para el 2015 es la siguiente:

   Desde el minuto 0 al 30: 0.0412 euros cada minuto
   Desde el minuto 31 al 90: 0.0370 euros cada minuto
   Desde el minuto 91 al 120: 0.0311 euros cada minuto
   Desde el minuto 121 al 660: 0.0305 euros cada minuto
   Desde el minuto 661 hasta máximo 24 horas: 24.00 euros

   Dado un tiempo de entrada y un tiempo de salida, construya un programa que calcule la tarifa final en euros a cobrar. Ejemplo: si el tiempo de permanencia es de 32 minutos, los primeros 30 minutos se facturan a 0.0412 el minuto y los 2 restantes a 0.0370.

   Finalidad: Utilización del condicional simple. Dificultad Baja.

7. Se quiere leer un carácter `letra_original` desde teclado, y comprobar con una estructura condicional si es una letra mayúscula. En dicho caso, hay que calcular la minúscula correspondiente en una variable llamada `letra_convertida`. En cualquier otro caso, le asignaremos a `letra_convertida` el valor que tenga `letra_original`. Finalmente, imprimiremos en pantalla el valor de `letra_convertida`. No pueden usarse las funciones `tolower` ni `toupper` de la biblioteca `cctype`.

   Finalidad: Plantear una estructura condicional con una expresión lógica compuesta. Dificultad Baja.

8. Queremos modificar el ejercicio 7 para leer un carácter `letra_original` desde teclado y hacer lo siguiente:

   - Si es una letra mayúscula, almacenaremos en la variable `letra_convertida` la correspondiente letra minúscula.
   - Si es una letra minúscula, almacenaremos en la variable `letra_convertida` la correspondiente letra mayúscula.
   - Si es un carácter no alfabético, almacenaremos el mismo carácter en la variable `letra_convertida`

   Finalmente, imprimiremos en pantalla alguno de los siguientes mensajes:

   - `La letra era una mayúscula. Una vez convertida es ...`
   - `La letra era una minúscula. Una vez convertida es ...`
   - `El carácter no era una letra.`

   Finalidad: Plantear una estructura condicional anidada. Diseñar programas que separen entradas/salidas y cálculos. Dificultad Baja.

9. Construya un programa para calcular el importe total a facturar de un pedido. El programa leerá el número de unidades vendidas y el precio de venta de cada unidad. Si la cantidad vendida es mayor de 100 unidades, se le aplica un descuento del 3 %. Por otra parte, si el precio final de la venta es mayor de 700 euros, se aplica un descuento del 2 %. Ambos descuentos son acumulables. Obtenga el importe final e imprímalo en pantalla.

   Vamos a cambiar el criterio de los descuentos. Supondremos que sólo se aplicará el descuento del 2 % (por una venta mayor de 700 euros) cuando se hayan vendido más de 100 unidades, es decir, para ventas de menos de 100 unidades no se aplica el descuento del 2 % aunque el importe sea mayor de 700 euros.

   Cambiar el programa visto en clase para incorporar este nuevo criterio.

   Finalidad: Plantear una estructura condicional anidada. Dificultad Baja.

10. Cread un programa que lea el valor de la edad (dato de tipo entero) y salario (dato de tipo real) de una persona. Subid el salario un 4 % si es mayor de 65 o menor de 35 años. Si además de cumplir la anterior condición, también tiene un salario inferior a 300 euros, se le subirá otro 3 %.

    Imprimid el resultado por pantalla.

    Finalidad: Plantear una estructura condicional anidada. Dificultad Baja.

11. Cread un programa que lea los datos fiscales de una persona, reajuste su renta bruta según el criterio que se indica posteriormente e imprima su renta neta final.

    - La renta bruta es la cantidad de dinero íntegra que el trabajador gana.
    - La retención fiscal es el tanto por ciento que el gobierno *se queda*.
    - La renta neta es la cantidad que le queda al trabajador después de quitarle el porcentaje de retención fiscal, es decir:
      Renta_neta = Renta_bruta - Renta_bruta * Retención / 100

    Los datos a leer son:

    - Si la persona es un trabajador autónomo o no
    - Si es pensionista o no
    - Estado civil
    - Renta bruta (total de ingresos obtenidos)
    - Retención inicial

    La modificación se hará de la siguiente forma:

    - Bajar un 3 % la retención fiscal a los autónomos
    - Para los no autónomos:

  - Se sube un 1 % la retención fiscal a todos los pensionistas
  - Al resto de los trabajadores se les aplica una subida lineal del 2 %.
    Una vez hecha esta subida, se le aplica (sobre el resultado anterior) las siguientes subidas adicionales:
    - Subir otro 2 % la retención fiscal si la renta bruta es menor de 20.000 €
    - Subir otro 2.5 % la retención fiscal a los casados con renta bruta superior a 20.000 €
    - Subir otro 3 % la retención fiscal a los solteros con renta bruta superior a 20.000 €

  Una vez calculada la retención final, habrá que aplicarla sobre la renta bruta para así obtener la renta final del trabajador.

  Finalidad: Plantear una estructura condicional anidada. *Dificultad Media.*

12. Vamos a modificar el ejercicio 3 de la siguiente forma. Queremos leer dos valores enteros desde teclado y, en el caso de que uno cualquiera de ellos divida al otro, el programa nos debe decir quién divide a quién. Debemos contemplar también el caso en el que alguno de los valores introducidos sea cero, en cuyo caso, ninguno divide al otro.

    Tened especial cuidado en no mezclar entradas/salidas con cálculos.

    Dificultad Media.

13. Modificad el ejercicio 4 para que el programa nos diga si los tres valores leídos están ordenados de forma ascendente, ordenados de forma descendente o no están ordenados.

    Para resolver este problema, se recomienda usar una variable de tipo *enumerado*.

    Finalidad: Usar el tipo enumerado para detectar cuándo se produce una situación determinada. *Dificultad Baja.*

Ejercicios sobre bucles

14. Leer un carácter desde teclado, obligando al usuario a que sea una letra mayúscula. Para ello, habrá que usar una estructura repetitiva `do-while`, de forma que si el usuario introduce un carácter que no sea una letra mayúscula, se le volverá a pedir otro carácter. Calculad la minúscula correspondiente e imprimidla en pantalla. Escribid dos versiones: una en la que que lo se usen las funciones `tolower` ni `toupper` de la biblioteca `cctype` y otra con estas funciones.

    Finalidad: Trabajar con bucles con condiciones compuestas. Trabajar con filtros. Dificultad Baja.

15. Ampliad el ejercicio 1 de manera que en esta versión filtre adecuadamente los valores de las horas, minutos y segundos.

    El programa repetirá la lectura, cálculo y presentación de resultados un número indeterminado de veces. La condición de parada será que el usuario introduzca un -1 cuando se solicite la hora del instante inicial.

    Finalidad: Trabajar con filtros y ciclos de lectura adelantada. Dificultad Baja.

16. Realizar un programa que lea desde teclado un entero positivo e imprima en pantalla todos sus **divisores propios**. Para obtener los divisores, basta recorrer todos los enteros menores que el valor introducido y comprobar si lo dividen.

    Finalidad: Bucle sencillo y filtro de entrada de datos. Dificultad Baja.

17. Ampliad el ejercicio 17 de la *Relación de Problemas I* de manera que en esta nueva versión se trabaje con un número entero arbitrario (positivo o negativo), con cualquier número de dígitos.

    Por ejemplo, si el número es 3519, la salida sería `3 5 1 9`

    En este ejercicio se pueden mezclar entradas y salidas con cómputos.

    Dificultad Baja.

18. Modifiquemos el ejercicio 4 de la *Relación de Problemas I*. Supongamos ahora que se quiere reinvertir todo el dinero obtenido (capital más intereses) en otro plazo fijo a un año. Y así, sucesivamente.

    Construid un programa para que lea el capital C, el interés I y un número de años N, y calcule e imprima todo el dinero obtenido durante cada uno de los N años, suponiendo que se reinvierte todo.

Filtrar adecuadamente los valores leidos de manera que cumplan las condiciones:

- Sobre el capital inicial, $C > 0$,
- Sobre el número de años, $1 <= N <= 20$
- Sobre el interés, $0 < I <= 10$

El programa debe mostrar una salida del tipo:

```
Total en el año número 1 = 240
Total en el año número 2 = 288
Total en el año número 3 = 345.6
.......................
```

Finalidad: Usar una variable acumuladora dentro del cuerpo de un bucle (aparecerá a la izquierda y a la derecha de una asignación). Dificultad Baja.

19. Modifiquemos nuevamente el ejercicio 4 de la *Relación de Problemas I*. Ahora se trata de construir un programa para calcular cuantos años han de pasar hasta llegar a doblar, como mínimo, el capital inicial.

    Aplicar los mismos filtros que se indican en el ejercicio 18.

    Finalidad: Usar la variable acumuladora en la misma condición del bucle. Dificultad Baja.

20. Usando como base el programa que soluciona el ejercicio 6 escribid un programa que lea la hora actual (minutos y segundos, por separado) y una cantidad de dinero (en euros) y nos indique hasta qué hora podemos tener el coche aparcado si vamos a pagar con la cantidad especificada.

    Implementar los filtros adecuados al problema.

    Si dispone de 5 euros y 25 céntimos deberá indicar el valor `5.25` cuando se le pida el dinero disponible.

    Dificultad Baja.

21. Realizar un programa que lea reales desde teclado y calcule cuántos se han introducido y cual es el mínimo de dichos valores (pueden ser positivos o negativos). Se dejará de leer datos cuando el usuario introduzca el valor 0. Realizad la lectura de los reales dentro de un bucle sobre una única variable llamada `dato`. Es importante controlar los casos extremos, como por ejemplo, que el primer valor leído fuese ya el terminador de entrada (en este caso, el cero).

    Finalidad: Destacar la importancia de las inicializaciones antes de entrar al bucle. Ejemplo de lectura anticipada. Dificultad Baja.

22. Realizar un programa que lea dos secuencias de enteros y nos diga si todos los valores de la primera secuencia son mayores que todos los de la segunda.

    Realizad la lectura de los enteros dentro de sendos bucles sobre una única variable llamada `dato`. El final de cada secuencia viene marcado cuando se lee el 0.

    Finalidad: Ejercitar el uso de bucles. Dificultad Baja.

23. Ampliad el ejercicio 5. El programa pedirá los valores de dos años y mostrará todos los años bisiestos comprendidos entre los dos valores dados.

    Finalidad: Practicar con filtros y ciclos básicos. Practicar con algoritmos más elaborados y eficientes. Reutilizar código ya escrito y verificado. Dificultad Media.

24. Se dice que un número n es **triangular** si se expresa como la suma de los primeros k valores enteros. Por ejemplo, $n = 6$ es triangular ya que $6 = 1 + 2 + 3$ ($k = 3$).

    Una forma de obtener los números triangulares es a través de la fórmula $k(k+1)/2 \ \forall k \in \mathbb{N}$.

    Escriba un programa que, sin aplicar directamente la fórmula anterior, muestre todos los números triangulares que hay menores que un entero n introducido desde teclado.

    Finalidad: Practicar con filtros y ciclos básicos. Dificultad Baja.

25. Una empresa que tiene tres sucursales decide llevar la contabilidad de las ventas de sus productos a lo largo de una semana. Registra cada venta con tres datos: 1) el identificador de la sucursal (1,2 ó 3), 2) el código del producto (`a`, `b` ó `c`) y 3) el número de unidades vendidas.

    Diseñar un programa que lea desde el teclado una serie de registros compuestos por `sucursal, producto, unidades` y diga cuál es la sucursal que más **productos** ha vendido. La lectura termina cuando la sucursal introducida vale -1.
    Por ejemplo, con la serie de datos

    ```
    2 a 20
    1 b 10
    1 b 4
    3 c 40
    1 a 1
    2 b 15
    1 a 1
    1 c 2
    2 b 6
    -1
    ```

    la sucursal que más productos ha vendido es la número 2 con 41 unidades totales.

    Para comprobar que el programa funciona correctamente, cread un fichero de texto y redirigid la entrada desde dicho fichero.

    Finalidad: Ver un bucle en el que se leen varios datos en cada iteración, pero sólo uno de ellos se usa como terminador de la entrada. Dificultad Media.

26. Se decide informatizar el acta de un partido de baloncesto para saber qué equipo es el ganador del partido. El acta de anotaciones está formada por una serie de *tríos* de números, y cada uno contiene: el número de equipo (1 ó 2), el dorsal del jugador y el número de puntos conseguidos. La última anotación es el valor terminador -1.

    Por ejemplo, con la entrada

    1 2 4 1 4 1 2 3 6 2 3 2 5 2 5 1 1 3 -1

    El programa muestra el equipo ganador (o si hay empate) y la puntuación.

    Por ejemplo, con la entrada anterior, gana el equipo 1, y la puntuación es 10-7.

    Finalidad: Ver un bucle en el que se leen varios datos en cada iteración, pero sólo uno de ellos se usa como terminador de la entrada. *Dificultad Media.*

27. La Unión Europea ha decidido premiar al país que más toneladas de hortalizas exporte a lo largo del año. Se dispone de un registro de transacciones comerciales en el que aparecen tres valores en cada apunte. El primer valor es el indicativo del país (`E`: España, `F`: Francia y `A`: Alemania), el segundo valor es un indicativo de la hortaliza que se ha vendido en una transacción (`T`: Tomate, `P`: Patata, `E`: Espinaca) y el tercer valor indica las toneladas que se han vendido en esa transacción. Diseñar un programa que lea desde el teclado este registro, el cual termina siempre al leer un país con indicativo `@`, y que diga qué país es el que más hortalizas exporta y las toneladas que exporta.

    Por ejemplo, con la entrada

    E T 10 E T 4 E P 1 E P 1 E E 2 F T 15 F T 6 F P 20 A E 40 @

    el país que más vende es *Francia* con un total de 41 toneladas.

    Finalidad: Ver un bucle en el que se leen varios datos en cada iteración, pero sólo uno de ellos se usa como terminador de la entrada. *Dificultad Media.*

28. Recupere la solución del ejercicio 16 (función gaussiana) de la relación de problemas .

    Construir un programa para imprimir el resultado de aplicar dicha función a varios valores de abscisas. En primer lugar, se leerán los parámetros que definen la función, es decir, la *esperanza* y la *desviación típica*. La esperanza puede ser cualquier valor, pero la desviación típica debe ser mayor o igual que cero.

    A continuación el programa pedirá un valor *mínimo*, un valor *máximo* y un *incremento*. Filtrar adecuadamente estos valores.

    El programa mostrará el valor de la función gaussiana en todos los valores de x (la abscisa) entre el mínimo y el máximo considerando entre dos puntos consecutivos un salto igual al valor dado en el incremento.

    Finalidad: usar cógido verificado en bucles sencillos. *Dificultad Baja.*

29. Construya un programa que calcule cuándo se produjo la mayor secuencia de días consecutivos con temperaturas crecientes. El programa leerá una secuencia de reales representando temperaturas, hasta llegar al `-1` y debe calcular la subsecuencia de

números ordenada, de menor a mayor, de mayor longitud. El programa nos debe decir la posición donde comienza la subsecuencia y su longitud.

Por ejemplo, ante la entrada siguiente:

```
17.2 17.3 16.2 16.4 17.1 19.2 18.9 -1
```

el programa nos debe indicar que la mayor subsecuencia empieza en la posición 3 (en el `16.2`) y tiene longitud 4 (termina en `19.2`)

Puede suponer que siempre se introducirá al menos un valor de temperatura.

Finalidad: Trabajar con bucles en los que se compara un valor leído con otro leído en la iteración anterior. *Dificultad Media.*

30. El método RLE (Run Length Encoding) codifica una secuencia de datos formada por series de valores idénticos consecutivos como una secuencia de parejas de números (valor de la secuencia y número de veces que se repite). Esta codificación es un mecanismo de compresión de datos (zip) sin pérdidas. Se aplica, por ejemplo, para comprimir los ficheros de imágenes en las que hay zonas con los mismos datos (fondo blanco, por ejemplo).

    Realizar un programa que lea una secuencia de números naturales terminada con un número negativo y la codifique mediante el método RLE. Leed los datos desde un fichero externo, tal y como se explica en la página 22.

Entrada:	1 1 1 2 2 2 2 2 3 3 3 3 3 3 5 -1
	(tres veces 1, cinco veces 2, seis veces 3, una vez 5)
Salida:	3 1 5 2 6 3 1 5

    Finalidad: Controlar en una iteración lo que ha pasado en la anterior. *Dificultad Media.*

31. El algoritmo de la multiplicación rusa es una forma distinta de calcular la multiplicación de dos números enteros $n * m$. Para ello este algoritmo va multiplicando por 2 el multiplicador m y dividiendo (sin decimales) por dos el multiplicando n hasta que n tome el valor 1 y suma todos aquellos multiplicadores cuyos multiplicandos sean impares. Por ejemplo, para multiplicar 37 y 12 se harían las siguientes iteraciones

Iteración	Multiplicando	Multiplicador
1	**37**	**12**
2	18	24
3	**9**	**48**
4	4	96
5	2	192
6	**1**	**384**

    El resultado de multiplicar 37 y 12 sería la suma de los multiplicadores correspondientes a los multiplicandos impares (en negrita), es decir 12+48+384=444

Cread un programa que lea dos enteros y calcule su producto con este algoritmo.

Dificultad Media.

32. Diseñar un programa para jugar a adivinar un número. En cada jugada el jugador introducirá un valor y el el juego indicará si el número introducido por el jugador está por encima o por debajo del número introducido.

    Como reglas de parada considerad: a) que haya acertado, o b) se no quiera seguir jugando a adivinar ese número (escoged cómo se quiere implementar esta opción).

    Una vez terminado un juego, podrá volverse a jugar de nuevo (escoged cómo se quiere implementar esta opción).

    Dificultad Media.

    Para poder generar números aleatorios en un rango determinado será necesario incluir las siguientes instrucciones:

```
...
#include<ctime>
#include<cstdlib>
...

// Queremos generar valores aleatorios entre MIN y MAX
const int MIN = 1, MAX = 100;
const NUM_VALORES = MAX-MIN + 1; // Número de valores posibles


time_t t;
srand(time(&t)); // Inicializa el generador de núms. aleatorios
                 // con el reloj del sistema (hora actual)
int incognita; // número aleatoria a generar

// Generación de incognita (MIN <= incognita <= MAX)
incognita = (rand() % NUM_VALORES) + MIN;
```

    La orden `srand(time(&t))` debe ejecutarse una única vez al principio del programa y sirve para inicializar la secuencia de números aleatorios. Posteriormente, cada vez que se ejecute:

    `incognita = (rand() % NUM_VALORES) + MIN`

    la instrucción `rand` devolverá valor (pseudo)aleatorio que se modificará para que quede comprendido entre `MIN` y `MAX`, y finalmente asignado a la variable `incognita`.

33. Un número entero n se dice que es **desgarrable** (*torn*) si al dividirlo en dos partes `izda` y `dcha`, el cuadrado de la suma de ambas partes es igual a n.

    Por ejemplo, 88209 es desgarrable ya que $(88 + 209)^2 = 88209$. El número 81 también es desgarrable ya que $(8 + 1)^2 = 81$.

    Cread un programa que lea un entero n e indique si es o no desgarrable.

    Finalidad: Ejercitar los bucles. Dificultad Baja.

34. Realizar un programa para calcular los valores de la función:

$$f(x) = \sqrt{\frac{3x + x^2}{1 - x^2}}$$

    para valores de x enteros en el rango [-3..3].

    Finalidad: Trabajar con bucles controlados por contador. Dificultad Baja.

35. Realizar un programa para calcular la suma de los términos de la serie

$$1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{6} + \frac{1}{8} - \frac{1}{10} + ... - \frac{1}{2n-1} + \frac{1}{2n}$$

    para un valor n dado ($n > 0$).

    Finalidad: Trabajar con bucles controlados por contador. Dificultad Baja.

36. Realizar un programa que presente una tabla de grados Celsius a grados Fahrenheit desde los 0 grados a los 300, con saltos de 20 grados.

    Finalidad: Trabajar con bucles controlados por contador. Dificultad Baja.

37. Calcular mediante un programa la función **potencia** x^n con n un valor entero y x un valor real. No puede usarse la funciones de la biblioteca `cmath`.

    Finalidad: Trabajar con bucles controlados por contador. Dificultad Baja.

38. Calcular mediante un programa la función **factorial**.

    Se dice que $n!$ es el factorial de un entero n y se define de la forma siguiente:

$$0! = 1$$
$$n! = 1 \times 2 \times 3 \times \cdots n, \ \ \forall n \geq 1$$

    Finalidad: Trabajar con bucles controlados por contador. Dificultad Baja.

39. Calcular mediante un programa en C++ el combinatorio $\begin{pmatrix} n \\ m \end{pmatrix}$ con n, m valores enteros. No pueden usarse las funciones de la biblioteca `cmath`.

    El combinatorio de n sobre m (con $n \geq m$) es un número entero que se define:

    $$\begin{pmatrix} n \\ m \end{pmatrix} = \frac{n!}{m!\,(n-m)!}$$

    Finalidad: Trabajar con bucles controlados por contador. Dificultad Media.

40. En matemáticas, la **sucesión de Fibonacci** (a veces mal llamada *serie* de Fibonacci) es la siguiente sucesión infinita de números naturales:

    $$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

    La sucesión comienza con los números 1 y 1, y a partir de éstos, cada término puede calcularse como la suma de los dos anteriores. A los elementos de esta sucesión se les llama *números de Fibonacci*.

    El número de Fibonacci de orden n, al que llamaremos f_n se puede definir mediante la siguiente relación de recurrencia:

    - $f_n = f_{n-1} + f_{n-2}$ para $n > 2$
    - $f_1 = f_2 = 1$

    Esta sucesión fue descrita en Europa por Leonardo de Pisa, matemático italiano del siglo XIII también conocido como Fibonacci. Tiene numerosas aplicaciones en ciencias de la computación, matemáticas y teoría de juegos. También aparece en diversas configuraciones biológicas.

    Escribir un programa que calcule el número de Fibonacci de orden n, donde n es un valor introducido por el usuario. A continuación, el programa solicitará un nuevo valor, k, y mostrará todos los números de Fibonacci $f_1, f_2, f_3, \ldots, f_k$.

    Finalidad: Trabajar con bucles controlados por contador. Dificultad Media.

41. El **número aúreo** se conoce desde la Antigüedad griega y aparece en muchos temas de la geometría clásica. La forma más sencilla de definirlo es como el único número positivo ϕ que cumple que $\phi^2 - \phi = 1$ y por consiguiente su valor es $\phi = \dfrac{1+\sqrt{5}}{2}$.

    Se pueden construir aproximaciones al número aúreo mediante la fórmula $a_n = \dfrac{f_{n+1}}{f_n}$ siendo f_n el número de Fibonacci de orden n (ver problema 40). La sucesión de valores así calculada proporciona, alternativamente, valores superiores e inferiores a ϕ, siendo cada vez más cercanos a éste, y por lo tanto la diferencia entre a_n y ϕ es cada vez más pequeña conforme n se hace mayor.

Escribir un programa que calcule el menor valor de n que hace que la aproximación dada por a_n difiera en menos de δ del número ϕ, sabiendo que $n \geq 1$. La entrada del programa será el valor de δ y la salida el valor de n. Por ejemplo, para un valor de $\delta = 0{,}1$ el valor de salida es $n = 4$

Finalidad: Trabajar con bucles. Reutilizar código ya validado. Dificultad Media.

42. Un número entero de n dígitos se dice que es **narcisista** si se puede obtener como la suma de las potencias n-ésimas de cada uno de sus dígitos. Por ejemplo 153 y 8208 son números narcisistas porque $153 = 1^3 + 5^3 + 3^3$ y $8208 = 8^4 + 2^4 + 0^4 + 8^4$. Construir un programa que nos indique si un entero positivo es narcisista.

    Finalidad: Ejercitar los bucles. Dificultad Media.

43. Realizar un programa para calcular los valores de la función:

$$f(x, y) = \frac{\sqrt{x}}{y^2 - 1}$$

    para los valores de (x, y) con $x = -50, -48, \ldots, 50$ $y = -40, -39, \ldots, 40$, es decir queremos mostrar en pantalla los valores de la función en los puntos

$$(-50, 40), (-50, -39), \cdots (-50, 40), (-48, 40), (-48, -39), \cdots (50, 40)$$

    Finalidad: Practicar con bucles anidados. Dificultad Baja.

44. Sobre la solución del ejercicio 18 de esta relación de problemas, se pide lo siguiente. Supondremos que sólo pueden introducirse intereses enteros (1, 2, 3, ...). Calcular el capital obtenido al término de cada año, pero realizando los cálculos para todos los tipos de interés enteros menores o iguales que el introducido (en pasos de 1).

    Por ejemplo, si interés es 5, y un número de años es 3, hay que mostrar el capital ganado al término de cada uno de los tres años a un interés del 1 %; a continuación para un interés del 2 %, y así hasta llegar al 5 %. El programa debe mostrar:

```
Cálculos realizados al 1%:
  Dinero obtenido en el año número 1 = 2020.00
  Dinero obtenido en el año número 2 = 2040.20
  Dinero obtenido en el año número 3 = 2060.60

Cálculos realizados al 2%:
  Dinero obtenido en el año número 1 = 2040.00
  Dinero obtenido en el año número 2 = 2080.80
  Dinero obtenido en el año número 3 = 2122.42
.......
```

    Finalidad: Practicar con bucles anidados. Dificultad Baja.

45. Escribid un programa que lea cuatro valores de tipo `char` (`min_izda`, `max_dcha`, `min_dcha`, `max_dcha`) e imprima las parejas que pueden formarse con un elemento del conjunto {`min_izda ... max_izda`} y otro de {`min_dcha ... max_dcha`}. Por ejemplo, si `min_izda = b`, `max_izda = d`, `min_dcha = j`, `max_dcha = m`, el programa debe imprimir las parejas que pueden formarse con un elemento del conjunto {`b, c, d`} y otro de {`j, k, l, m`}, es decir:

    ```
    bj bk bl bm
    cj ck cl cm
    dj dk dl dm
    ```

    Finalidad: Practicar con bucles anidados. Dificultad Baja.

46. *(Examen Septiembre 2014)* ¿Cuántas veces aparece el dígito 9 en todos números que hay entre el 1 y el 100? Por ejemplo, el 9 aparece una vez en los números 19 y 92 mientras que aparece dos veces en el 99.

    Pretendemos diseñar un algoritmo que responda a esta sencilla pregunta, pero de forma suficientemente generalizada. Para ello, se pide construir un programa que lea tres enteros: `cifra` (entre 1 y 9), `min` y `max`, y calcule el número de apariciones del dígito `cifra` en los números contenidos en el intervalo cerrado `[min, max]`.

    Finalidad: Ejercitar los bucles anidados. Dificultad Baja.

47. Construya un programa que lea un valor T y calcule la siguiente sumatoria:

$$\sum_{i=1}^{i=T} i! = \sum_{i=1}^{i=T} \left(\prod_{j=1}^{j=i} j \right)$$

    Por ejemplo, para $T = 4$, la operación a realizar es:

$$1! + 2! + 3! + 4!$$

    es decir:

$$1 + (1 * 2) + (1 * 2 * 3) + (1 * 2 * 3 * 4)$$

    Finalidad: Practicar con bucles `for` anidados. Dificultad Media.

48. Un número **perfecto** es aquel que es igual a la suma de todos sus divisores positivos excepto él mismo. El primer número perfecto es el 6 ya que sus divisores son 1, 2 y 3 y 6=1+2+3. Escribir un programa que muestre el mayor número perfecto que sea menor a un número dado por el usuario.

    Finalidad: Practicar con bucles anidados. Dificultad Media.

49. Escribir un programa que encuentre dos enteros n y m mayores que 1 que verifiquen lo siguiente:

$$\sum_{i=1}^{m} i^2 = n^2$$

Finalidad: Practicar con bucles anidados. Dificultad Media.

50. Supongamos una serie numérica cuyo término general es:

$$a_i \;=\; a_1 r^{i-1} \;=\; a_1, a_1 r, a_1 r^2, a_1 r^3, \ldots$$

Se pide crear un programa que lea desde teclado r, el primer elemento a_1 y el tope k y calcule la suma de los primeros k valores de la serie, es decir:

$$\sum_{i=1}^{i=k} a_i$$

Se proponen dos alternativas:

a) Realizad la suma de la serie usando la función `pow` para el cómputo de cada término a_i. Los argumentos de `pow` no pueden ser ambos enteros, por lo que forzaremos a que la base (por ejemplo) sea `double`, multiplicando por `1.0`.

b) Si analizamos la expresión algebraica de la serie numérica, nos damos cuenta que es una *progresión geométrica* ya que cada término de la serie queda definido por la siguiente expresión:

$$a_{i+1} = a_i \, r$$

Es decir, una progresión geométrica es una secuencia de elementos en la que cada uno de ellos se obtiene multiplicando el anterior por una constante denominada razón o factor de la progresión.

Cread el programa usando esta fórmula. NO puede utilizarse la función `pow`.

¿Qué solución es preferible en términos de eficiencia?

Finalidad: Trabajar con bucles que aprovechan cómputos realizados en la iteración anterior. Dificultad Baja.

51. Diseñar un programa para calcular la suma de los 100 primeros términos de la sucesión siguiente:

$$a_i = \frac{(-1)^i (i^2 - 1)}{2i}$$

No puede usarse la función `pow`. Hacedlo calculando explícitamente, en cada iteración, el valor $(-1)^i$ (usad un bucle `for`). Posteriormente, resolvedlo calculando dicho valor a partir del calculado en la iteración anterior, es decir, $(-1)^{i-1}$.

Finalidad: Enfatizar la conveniencia de aprovechar cómputos realizados en la iteración anterior. Dificultad Media.

52. Se dice que un número natural es **feliz** si cumple que si sumamos los cuadrados de sus dígitos y seguimos el proceso con los resultados obtenidos, finalmente obtenemos uno (1) como resultado. Por ejemplo, el número 203 es un número feliz ya que $\mathbf{2^2 + 0^2 + 3^2 = 13 \rightarrow 1^2 + 3^2 = 10 \rightarrow 1^2 + 0^2 = 1}$.
    Se dice que un número es feliz de grado k si es feliz en un máximo de k iteraciones. Se entiende que una iteración se produce cada vez que se elevan al cuadrado los dígitos del valor actual y se suman. En el ejemplo anterior, $\mathbf{203}$ es un número feliz de grado $\mathbf{3}$ (además, es feliz de cualquier grado mayor o igual que 3)
    Escribir un programa que diga si un número natural n es feliz para un grado k dado de antemano. Tanto n como k son valores introducidos por el usuario.

    Finalidad: Ejercitar los bucles anidados. *Dificultad Media.*

53. Diremos que un número entero positivo es **secuenciable** si se puede generar como suma de números consecutivos. Por ejemplo, $\mathbf{6 = 1 + 2 + 3}$, $\mathbf{15 = 7 + 8}$. Esta descomposición no tiene por qué ser única. Por ejemplo, $\mathbf{15 = 7 + 8 = 4 + 5 + 6 = 1 + 2 + 3 + 4 + 5}$.

    Escribir un programa que lea un entero n y nos diga cuántas descomposiciones posibles tiene. Por ejemplo:

    ```
    15 -> 3 descomposiciones
    94 -> 1 descomposición
    108 -> 3 descomposiciones
    ```

    Curiosidad: los únicos números con 0 descomposiciones son las potencias de 2.

    Dificultad Media.

54. Se pide leer dos enteros sabiendo que el primero no tiene un tamaño fijo y que el segundo siempre es un entero de dos dígitos. Se pide comprobar si el segundo está contenido en el primero.

    Entendemos que está contenido si los dos dígitos del segundo entero están en el primer entero de forma consecutiva y en el mismo orden. Por ejemplo, 89 está contenido en 7890, en 7789 y en 8977 pero no en 7980.

    Dificultad Media.

55. Recupere la solución del ejercicio 28. Escribir un programa en el que se presente un menú principal para que el usuario pueda elegir las siguientes opciones:

    - Introducir parámetros de la función (esperanza y desviación)
    - Salir del programa

    Si el usuario elige la opción de salir, el programa terminará; si elige la opción de introducir los parámetros, el programa leerá los dos parámetros (esperanza y desviación).

    A continuación, el programa presentará un menú con las siguientes opciones:

- Introducir rango de valores de abscisas
- Volver al menú anterior (el menú principal)

Si el usuario elige volver al menú anterior, el programa debe presentar el primer menú; si el usuario elige introducir los valores de abscisas, el programa le pedirá los extremos del intervalo y el incremento entre dos valores consecutivos de la abscisa, y mostrará los valores de la función gaussiana en todos los valores posibles de la abscisa (de manera similar a como se hizo en el ejercicio 28).

Después de mostrar los valores de la función, el programa volverá al menú de introducción del rango de valores de abscisas.

Finalidad: Ejercitar los bucles anidados. *Dificultad Media.*

56. Una **sucesión alícuota** es una sucesión iterativa en la que cada término es la suma de los divisores propios del término anterior.

    La sucesión alícuota que comienza con el entero positivo k puede ser definida formalmente mediante la función divisor σ_1 de la siguiente manera:

$$\begin{aligned} s_0 &= k \\ s_n &= \sigma_1(s_{n-1}) - s_{n-1} \end{aligned}$$

    Por ejemplo, la sucesión alícuota de 10 es $10, 8, 7, 1, 0$ porque:

$$\begin{aligned} \sigma_1(10) - 10 &= 5 + 2 + 1 = 8 \\ \sigma_1(8) - 8 &= 4 + 2 + 1 = 7 \\ \sigma_1(7) - 7 &= 1 \\ \sigma_1(1) - 1 &= 0 \end{aligned}$$

    Aunque muchas sucesiones alícuotas terminan en cero, otras pueden no terminar y producir una sucesión alícuota períodica de período 1, 2 o más. Está demostrado que si en una sucesión alícuota aparece un *número perfecto* (como el 6) se produce una sucesión infinita de período 1. Un *número amigable* produce una sucesión infinita de período 2 (como el 220 ó 284).

    Escribir un programa que lea un número natural menor que 1000 y muestre su sucesión alícuota.

    Hay que tener en cuenta que en ocasiones se pueden producir sucesiones infinitas, por lo que en estos casos habrá que detectarlas e imprimir puntos suspensivos cuando el período se repita. Solo hay que considerar períodos infinitos de longitud 2 como máximo.

Por ejemplo; para el número 6, se imprimiría: 6, 6, . . . y para el número 220, se imprimiría: 220, 284, 220, 284, . . .

Dificultad Media.

57. Construid un programa para comprobar si las letras de una palabra se encuentran dentro de otro conjunto de palabras. Los datos se leen desde un fichero de esta manera: el fichero contiene, en primer lugar un total de 3 letras que forman la palabra a buscar, por ejemplo `f e o`. Siempre habrá, exactamente, tres letras. A continuación, el fichero contiene el conjunto de palabras en el que vamos a buscar. El final de cada palabra viene determinado por el carácter `'@'`, y el final del fichero por el carácter `'#'`. La búsqueda tendrá las siguientes restricciones:

    - Deben encontrarse las tres letras
    - Debe respetarse el orden de aparición. Es decir, si por ejemplo encontramos la `'f'` en la segunda palabra, la siguiente letra a buscar `'e'` debe estar en una palabra posterior a la segunda.
    - Si encontramos una letra en una palabra, ya no buscaremos más letras en ella.

Entrada:	`f e o`
	`h o l a @`
	`m o f e t a @  <- f`
	`c o f i a @`
	`c e r r o @  <- e`
	`p e r a @`
	`c o s a @  <- o`
	`h o y @`
	`#`

En este caso, sí se encuentra.

Dificultad Media.