Fundamentos de Programación
Relación de ejercicios 2

1. Ampliad el ejercicio 7 de la relación de 1 para que, una vez calculada la media y la desviación típica, el programa imprima, para cada uno de los valores introducidos previamente, si está por encima o por debajo de la media. Por ejemplo:

   ```
   33 es menor que su media
   48 es mayor o igual que su media
   .....
   ```

   Nota. Los valores introducidos son enteros, pero la media y la desviación son reales.

   Finalidad: Plantear un ejemplo básico con varias estructuras condicionales independientes. Dificultad Baja.

2. Cread un programa que incluya una variable `edad` en la que guardamos la edad de una persona (como una variable entera) y otra variable `ingresos`, en la que almacenamos sus ingresos (como un real). Subid el salario un 5% si es un jubilado con unos ingresos inferiores a 300 euros e imprimid el resultado por pantalla. En caso contrario imprimid el mensaje `"No es aplicable la subida"`. En ambos casos imprimid el salario resultante. Resolved el ejercicio de dos formas:

   - En primer lugar, mezclando E/S (entradas y salidas) con cómputos dentro de la misma estructura condicional.
   - En segundo lugar, separándolos en bloques distintos.

   Realizad el mismo ejercicio pero subiendo el salario un 4% si es mayor de 65 o menor de 35 años. Además, si también tiene un salario inferior a 300 euros, se le subirá otro 3%.

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

3. Realizar un programa en C++ 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 en C++ para 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 con una expresión lógica compuesta. Dificultad Baja.

5. 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`.

   (a) Se propone una primera solución como sigue:

```
char letra_convertida, letra_original;
const int DISTANCIA_MAY_MIN = 'a'-'A';

cout << "\nIntroduzca una letra --> ";
cin >> letra_original;

if ((letra_original >= 'A') && (letra_original <= 'Z')){
     letra_convertida = letra_original + DISTANCIA_MAY_MIN;
     cout << letra_convertida;
}
else{
     cout << letra_original << " no es una mayúscula";
}
```

El problema es que dentro de la misma sentencia condicional se realizan cómputos (`letra_convertida = letra_original + DISTANCIA_MAY_MIN`) y salidas de resultados en pantalla (`cout << letra_convertida;`). Para evitarlo, se propone el uso de una variable que nos indique lo sucedido en el bloque de cómputos.

(b) Usamos en primer lugar un `string`:

```
string tipo_letra;
.............

if ((letra_original >= 'A') && (letra_original <= 'Z'))
     tipo_letra = "es mayúscula";

if (tipo_letra == "es mayúscula")
     letra_convertida = letra_original + DISTANCIA_MAY_MIN;
else
     letra_convertida = letra_original;

cout << "\nEl carácter " << letra_original <<
      " una vez convertido es: " << letra_convertida;
```

Nota. Para poder usar el operador de comparación == entre dos `string`, hay que incluir la biblioteca `string`.

Si se opta por esta alternativa, el suspenso está garantizado. ¿Por qué?

(c) Mucho mejor si usamos una variable lógica llamada `es_mayuscula`, de la siguiente forma:

```
if ((letra_original >= 'A') && (letra_original <= 'Z'))
     es_mayuscula = true;

if (es_mayuscula)
     letra_convertida = letra_original + DISTANCIA_MAY_MIN;

cout << "\nEl carácter " << letra_original <<
      " una vez convertido es: " << letra_convertida;
```

Sin embargo, hay un error lógico ya que la variable `es_mayuscula` se podría quedar sin un valor asignado (en el caso de que la expresión lógica fuese `false`). El compilador lo detecta y da el aviso al inicio de la sentencia `if (es_mayuscula)`. Comprobadlo para familiarizarnos con este tipo de avisos y proponer una solución.

Moraleja: Hay que garantizar la asignación de las variables lógicas, en todos los casos posibles

Finalidad: Mostrar cómo se comunican los distintos bloques de un programa a través de variables que indican lo que ha sucedido en el bloque anterior y destacar la importancia de incluir un bloque `else`, para garantizar que siempre se ejecuta el código adecuado, en todas las situaciones posibles. Dificultad Baja.

6. Queremos modificar el ejercicio 5 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`. Para ello, añadimos una variable nueva `es_minuscula` para detectar el caso en el que la letra sea una minúscula. Si escribimos el siguiente código

   ```
   if (letra_original >= 'A') && (letra_original <= 'Z')
        es_mayuscula = true;
   else
        es_minuscula = true;
   ```

   estaremos cometiendo el tipo de error indicado en el ejercicio 5, ya que si le asignamos un valor a una de las variables lógicas, no se lo asignamos a la otra y se queda con un valor indeterminado. Para resolverlo, planteamos la siguiente solución:

   ```
   if ((letra_original >= 'A') && (letra_original <= 'Z')){
        es_mayuscula = true;
        es_minuscula = false;
   }
   else{
        es_mayuscula = false;
        es_minuscula = true;
   }
   ```

   o si se prefiere, de una forma más concisa:

   ```
   es_mayuscula = (letra_original >= 'A') && (letra_original <= 'Z');
   es_minuscula = !es_mayuscula;
   ```

   En este planteamiento hay un error lógico que se comete de forma bastante habitual, cuando se quiere detectar más de dos situaciones posibles. En la asignación `es_minuscula = !es_mayuscula;` estamos diciendo que una minúscula es todo aquello que no sea una mayúscula. Esto no es correcto, ya que un símbolo como `#` no es ni mayúscula ni minúscula. Deberíamos haber usado lo siguiente:

   ```
   es_mayuscula = (letra_original >= 'A') && (letra_original <= 'Z');
   es_minuscula = (letra_original >= 'a') && (letra_original <= 'z');
   ```

   Completad el ejercicio, asignándole el valor correcto a la variable `letra_convertida` e imprimid en pantalla el valor de `letra_convertida`.

   Finalidad: Plantear una estructura condicional anidada. Dificultad Baja.

7. Modificad la solución del ejercicio 2 (subida salarial) para que no se mezclen E/S (entradas y salidas) con cómputos dentro de la misma estructura condicional.

   Finalidad: Diseñar programas que separen Entradas/Salidas y cómputos. Dificultad Baja.

8. 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.

   Dificultad Baja.

9. 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 renta neta es la cantidad que le queda después de quitarle el porcentaje de retención fiscal, es decir:

   $$RentaNeta = RentaBruta - RentaBruta * Retencion/100.$$

   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)

   La modificación se hará de la siguiente forma:

   - Se baja 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 trabajadores se le sube un 2% la retención fiscal. Una vez hecha esta subida lineal del 2%, se le aplica (sobre el resultado anterior) las siguientes subidas adicionales, dependiendo de su estado civil y niveles de ingresos:
     - Se sube un 6% la retención fiscal si la renta bruta es menor de 20.000 euros.
     - Se sube un 8 % la retención fiscal a los casados con renta bruta superior a 20.000 euros.
     - Se sube un 10 % la retención fiscal a los solteros con renta bruta superior a 20.000 euros.

   Nota: Cuando se pide subir un $x\%$ la retención fiscal, significa que la nueva retención fiscal será la antigua más el resultado de realizar el $x\%$ sobre la antigua retención.

   $$NuevaRetencion = AntiguaRetencion + AntiguaRetencion * x/100$$

   De forma análoga, si se le baja la retención, habrá que restar en vez de sumar.

   Finalidad: Plantear una estructura condicional anidada. Dificultad Media.

10. Realizar un programa que lea desde teclado un entero `tope` 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. A continuación, mejorar el ejercicio obligando al usuario a introducir un entero positivo, usando un filtro con un bucle post test (`do while`).

    Finalidad: Plantear un ejemplo sencillo de bucle y de filtro de entrada de datos. Dificultad Baja.

11. Realizar un programa que lea enteros 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 enteros 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.

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

13. Modifiquemos el ejercicio 2 del capital y los intereses de la primera relación. Supongamos ahora que se quiere reinvertir todo el dinero obtenido (el original C más los intereses producidos) en otro plazo fijo a un año. Y así, sucesivamente. Construid un programa para que lea el capital, el interés y un número de años N, y calcule e imprima todo el dinero obtenido durante cada uno de los N años, suponiendo que todo lo ganado (incluido el capital original C) se reinvierte a plazo fijo durante el siguiente año. 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.

14. Sobre el mismo ejercicio del capital y los intereses, construid un programa para calcular cuantos años han de pasar hasta llegar a doblar, como mínimo, el capital inicial. Los datos que han de leerse desde teclado son el capital inicial y el interés anual.

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

15. Se pide 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. No pueden usarse las funciones `tolower` ni `toupper` de la biblioteca `cctype`. Si se quiere, se puede usar como base el proyecto que resolvió el ejercicio 15 de la relación 1.

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

16. 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 $(88+209)^2 = 88209$. Cread un programa que lea un entero n e indique si es o no desgarrable.

    Finalidad: Ejercitar los bucles. Dificultad Baja.

17. 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 $153 = 1^3 + 5^3 + 3^3$ (153 tiene 3 dígitos) y $8208 = 8^4 + 2^4 + 0^4 + 8^4$ (8208 tiene 4 dígitos). Construir un programa que, dado un número entero positivo, nos indique si el número es o no narcisista. Si se va a usar la función `pow`, no pueden ser ambos argumentos enteros, por lo que forzaremos a que la base (por ejemplo) sea `double`, multiplicando por 1.0.

    Finalidad: Ejercitar los bucles. Dificultad Media.

18. Calcular mediante un programa en C++ la función potencia x^n, y la función factorial $n!$ con n un valor entero y x un valor real. No pueden usarse las funciones de la biblioteca `cmath`. El factorial de un entero n se define de la forma siguiente:

    $0! = 1$

    $n! = 1 \times 2 \times 3 \cdots \times n$, para $n \geq 1$.

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

19. Calcular mediante un programa en C++ el combinatorio $\binom{n}{m}$ 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 como sigue:

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

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

20. Todo lo que se puede hacer con un bucle `while` se puede hacer con un `do while`. Lo mismo ocurre al revés. Sin embargo, cada bucle se usa de forma natural en ciertas situaciones. El no hacerlo, nos obligará a escribir más código y éste será más difícil de entender. Para comprobarlo, haced lo siguiente:

    (a) Modificad la solución del ejercicio 12 de forma que, el filtro de entrada usado para leer la variable `tope`, se haga con un bucle pre-test `while`.

    (b) Modificad la solución del ejercicio 15 sustituyendo el bucle `while` por un `do while`. Observad que debemos considerar el caso en el que el número de años leído fuese cero.

    Finalidad: Enfatizar la necesidad de saber elegir entre un bucle pre-test o un bucle post-test. Dificultad Media.

21. Una empresa que tiene tres sucursales decide llevar la contabilidad de las ventas de sus productos a lo largo de una semana. Para ello registra cada venta con tres números, el identificador de la sucursal (1,2 ó 3), el código del producto (1, 2 ó 3) y 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 serie de datos termina cuando la sucursal introducida vale -1. Por ejemplo, con la serie de datos

    1 2 10
    1 2 4
    1 1 1
    1 1 1
    1 3 2
    2 2 15
    2 2 6
    2 1 20
    3 3 40
    −1

    Se puede ver que la sucursal que más productos ha vendido es la número 2 con 41 unidades totales. La salida del programa deberá seguir exactamente el siguiente formato:

```
SUCURSAL: 2
VENTAS: 41
```

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

22. Una compañía aérea establece el precio del billete como sigue: en primer lugar se fija una tarifa base de 150 euros, la misma para todos los destinos. Si el destino está a menos de 200 kilómetros, el precio final es la tarifa inicial. Para destinos a más de 200 Km, se suman 10 céntimos por cada kilómetro de distancia al destino (a partir del Km 200). En una campaña de promoción se va a realizar una rebaja lineal de 15 euros a todos los viajes. Además, se pretenden añadir otras rebajas y se barajan las siguientes alternativas de políticas de descuento:

    (a) Una rebaja del 3% en el precio final, para destinos a más de 600Km.

    (b) Una rebaja del 4% en el precio final, para destinos a más de 1100Km. En este caso, no se aplica el anterior descuento.

    (c) Una rebaja del 5% si el comprador es cliente previo de la empresa.

    Cread un programa para que lea el número de kilómetros al destino y si el billete corresponde a un cliente previo de la empresa. Calcular el precio final del billete con las siguientes políticas de descuento:

    - Aplicando c) de forma adicional a los descuentos a) y b)
    - Aplicando c) de forma exclusiva con los anteriores, es decir, que si se aplica c), no se aplicaría ni a) ni b)

    Finalidad: Plantear una estructura condicional anidada. Dificultad Media.

23. Supongamos una serie numérica cuyo término general es:

$$a_i = a_1 r^i$$

    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:

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

    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 pedido usando esta fórmula. NO puede utilizarse la función `pow`.

    ¿Qué solución es preferible en términos de eficiencia (más rapidez)?

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

24. Reescribid la solución a los ejercicios 12 (divisores) y 15 (interés) usando un bucle `for`

    Finalidad: Familiarizarnos con la sintaxis de los bucles `for`. Dificultad Baja.

25. 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.

26. 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). Realizar un programa que lea una secuencia de números naturales terminada con un número negativo y la codifique mediante el método RLE. Por ejemplo,

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

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

27. Sobre la solución del ejercicio 15 de esta relación de problemas, se pide lo siguiente. Supondremos que sólo pueden introducirse intereses enteros (1, 2, 3, etc). Se pide 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 el usuario introduce un interés de 5, y un número de años igual a 3, hay que mostrar el capital ganado al término de cada uno de los tres años a un interés del 1%; a continuación, lo mismo para un interés del 2%, y así sucesivamente hasta llegar al 5%. El programa debe mostrar una salida del tipo:

```
Cálculos realizados al 1%:

Dinero obtenido en el año número 1 = 2020
Dinero obtenido en el año número 2 = 2040.2
Dinero obtenido en el año número 3 = 2060.6

Cálculos realizados al 2%:

Dinero obtenido en el año número 1 = 2040
Dinero obtenido en el año número 2 = 2080.8
Dinero obtenido en el año número 3 = 2122.42
.......
```

Finalidad: Empezar a trabajar con bucles anidados. Dificultad Baja.

28. Cread un programa que ofrezca en pantalla la siguiente salida:

```
1 2 3 4 5 6
2 3 4 5 6
3 4 5 6
4 5 6
5 6
6
```

Finalidad: Ejercitar los bucles anidados. Dificultad Baja.

29. Cread un programa que ofrezca en pantalla la siguiente salida:

```
1 2 3 4 5 6
2 3 4 5 6 7
3 4 5 6 7 8
4 5 6 7 8 9
5 6 7 8 9 10
6 7 8 9 10 11
```

    Finalidad: Ejercitar los bucles anidados. Dificultad Media.

30. Modificad los dos ejercicios anteriores para que se lea desde teclado el valor inicial y el número de filas a imprimir. En los ejemplos anteriores, el valor inicial era 1 y se imprimían un total de 6 filas.

    Finalidad: Ejercitar los bucles anidados. Dificultad Media.

31. 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

$$2^2 + 0^2 + 3^2 = 13 \rightarrow 1^2 + 3^2 = 10 \rightarrow 1^2 + 0^2 = 1.$$

    Se dice que un número es feliz de grado k si se ha podido demostrar que 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, 203 es un número feliz de grado 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.

32. ¿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.

33. 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.

    (a) En primer lugar, resolved el ejercicio mezclando entradas, cómputos y salidas de resultados

    (b) En segundo lugar, se pide resolver el ejercicio sin mezclar C/E,S. Para ello, se ofrecen varias alternativas. ¿Cual sería la mejor? Escoged una e implementar la solución.

        i. Utilizar un variable de tipo `string` de la forma siguiente:

```
string quien_divide;
.............
if (a%b==0)
     quien_divide = "b divide a a" ;
.............
if (quien_divide == "b divide a a")
     cout << b << " divide a " << a;
```

Nota. Para poder usar el operador de comparación == entre dos `string`, hay que incluir la biblioteca `string`.
Si se opta por esta alternativa, el suspenso está garantizado. ¿Por qué?

ii. Utilizar dos variables lógicas de la forma siguiente:

```
bool a_divide_b, b_divide_a;
.............
if (a%b==0)
     a_divide_b = true;
.............
if (a_divide_b)
     cout << a << "divide a " << b;
```

iii. Detectamos si se dividen o no y usamos otras dos variables que me indiquen quien es el dividendo y quien el divisor:

```
bool se_dividen;
int Divdo, Divsor;
.............
if (a%b==0){
     Divdo = a;
.............
if (se_dividen)
     cout << Dvsor << " divide a " << Dvdo;
```

(c) Completar la solución elegida para contemplar también el caso en el que alguno de los valores introducidos sea cero, en cuyo caso, ninguno divide al otro.

Dificultad Media.

34. 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.

35. En el ejercicio 6 se han usado dos variables lógicas (`es_mayuscula`, `es_minuscula`), que nos proporciona la distinción entre 4 casos distintos (VV, VF, FV, FF). Sin embargo, sólo queremos distinguir tres posibilidades, a saber, es mayúscula, es minúscula y no es un carácter alfabético. Para ello, volved a resolver el ejercicio 6 sustituyendo las dos variables lógicas por un tipo enumerado adecuado.

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

36. En el ejercicio 8 de la relación 1 se pedía escribir un programa que leyese un valor entero de tres dígitos e imprimiese los dígitos separados por un espacio en blanco. Haced lo mismo pero para un número entero arbitrario. 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 Media.

37. 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 intervalo [-3,3].

Dificultad Baja.

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

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

para todos los valores enteros de x e y con x en el intervalo $[-50, 50]$ e y en el intervalo $[-40, 40]$.

Dificultad Baja.

39. Diseñar un programa que presente una tabla de conversión de grados Celsius (C) a grados Fahrenheit (F), $F = \frac{9}{5}C + 32$, desde los 0 grados a los 300, con incremento de 20 en 20 grados.

Dificultad Baja.

40. Diseñar un programa que lea caracteres desde la entrada y los muestre en pantalla, hasta que se pulsa el '`.`' y diga cuantos separadores se han leído (espacios en blanco ' ', tabuladores '`\t`' y caracteres de nueva línea '`\n`').

Dificultad Baja.

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

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

para un valor n dado.

Dificultad Baja.

42. Se decide informatizar el acta de un partido de baloncesto para saber qué equipo es el ganador del partido. El acta contiene una serie de anotaciones formadas por una pareja de números cada una, con el dorsal del jugador y el número de puntos conseguidos teniendo en cuenta que la última anotación es un valor -1. Por ejemplo,

$124141236232525113 - 1$

El programa deberá indicar si ha ganado el equipo 1 (con los dorsales 1, 2 y 3), el equipo 2 (dorsales 4, 5 y 6) o han empatado. Por ejemplo, con la entrada anterior, gana el equipo 1.

Dificultad Baja.

43. 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.

Dificultad Baja.

44. Diseñar un programa para jugar a adivinar un número. El juego tiene que dar pistas de si el número introducido por el jugador está por encima o por debajo del número introducido. Como reglas de parada considerad a) que el jugador haya acertado o b) que el jugador se haya hartado y decida terminar (escoged cómo se quiere que se especifique esta opción). Realizar el mismo ejercicio pero permitiendo jugar tantas veces como lo desee el jugador.

    Dificultad Media.

45. Se dice que un número es triangular si se puede poner como la suma de los primeros m valores enteros, para algún valor de m. Por ejemplo, 6 es triangular ya que $6 = 1 + 2 + 3$. Una forma de obtener los números triangulares es a través de la fórmula $n(n+1)/2$ para todo $n \in \mathbb{N}$. Se pide construir un programa que obtenga todos los números triangulares que hay menores que un entero k introducido desde teclado, sin aplicar la fórmula anterior.

    Dificultad Baja.

46. Escribir un programa que lea una secuencia de números enteros en el rango de 0 a 100 terminada en un número mayor que 100 o menor que 0 y encuentre 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:

    ```
    23 25 7 40 45 45 73 73 71 4 9 101
    ```

    el programa nos debe indicar que la mayor subsecuencia empieza en la posición 3 (en el 7) y tiene longitud 6 (termina en la segunda aparición del 73).

    Dificultad Media.

47. El algoritmo de la multiplicación rusa es una forma distinta de calcular la multiplicación de dos números enteros n y 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 por 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

    Con lo que el resultado de multiplicar 37 por 12 sería la suma de los multiplicadores correspondientes a los multiplicandos impares (en negrita), es decir $37 * 12 = 12 + 48 + 384 = 444$. Cread un programa para leer dos enteros n y m y calcule su producto utilizando este algoritmo.

    Dificultad Media.

48. 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 la forma siguiente: 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 la aparición del 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.

- Una vez encontremos una letra en una palabra, ya no buscaremos más letras en dicha palabra.
- No nos planteamos una búsqueda barajando todas las posibilidades, en el sentido de que una vez encontrada una letra, no volveremos a buscarla de nuevo.

Por ejemplo, si el fichero es

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.

49. 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.

    Dificultad Media.

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

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

    Dificultad Media.

51. El número áureo 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

$$\phi = \frac{\sqrt{5} + 1}{2}$$

    Se pueden construir aproximaciones al número áureo mediante la fórmula

$$a_n = \frac{\text{fib}(n+1)}{\text{fib}(n)}$$

    siendo $\text{fib}(n)$ el término n-ésimo de la sucesión de fibonacci, que se define como:

$$\text{fib}(0) = 0,\ \text{fib}(1) = 1 \text{ y } \text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2) \text{ para } n \geq 2$$

    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 δ y la salida el valor de n. Por ejemplo, para un valor de $\delta = 0.1$, el valor de salida es $n = 3$.

    Dificultad Media.