1. Averigua **qué hace** este programa modular:

```
algoritmo ejercicio_1

variables
   operacion es carácter
   A, B, result son enteros

inicio
   repetir
   inicio
      escribir("Introduzca la operación (Q para terminar)")
      leer(operacion)
      si (operacion != 'Q') entonces
      inicio
         escribir("Introduzca los operandos")
         leer(A, B)
         si (operacion == '+') entonces
            result = sumar(A, B)
         si (operacion == '-') entonces
            result = restar(A, B)
         si (operacion == '+') o (operacion == '-') entonces
            escribir("El resultado es: ", result)
          si_no
            escribir("Error: operación no reconocida")
      fin
   fin
   mientras que (operacion != 'Q')
fin

entero función sumar (N1, N2 son enteros)
variables
   s es entero
inicio
   s = N1 + N2
   devolver (s)
fin

entero función restar (N1, N2 son enteros)
variables
   r es entero
inicio
   r = N1 - N2
   devolver (N1 - N2)
fin
```

2. Escribe un programa modular que pregunte al usuario si desea ver los **números pares o impares** y que, dependiendo de la respuesta, muestre en la pantalla los números pares o impares entre 1 y 1000.
3. Escribe una **función `salario()`** que calcule el dinero que debe cobrar un trabajador a la semana, pasándole como parámetros el número de horas semanales que ha trabajado y el precio que se le paga por cada hora. Si ha trabajado más de 40 horas, el salario de cada hora adicional es 1,5 veces el de las horas convencionales.

   Escribe **dos versiones** de la función: una con paso de parámetros por valor y otra por referencia.
4. **Modifica** el programa anterior para que calcule mediante subprogramas:
   - El **salario bruto mensual**, sabiendo que todas las horas que excedan de 40 semanales se consideran horas extra. Las primeras 5 horas extra se cobran a 1,5 veces la tarifa normal, y las demás al doble de la tarifa normal.
   - Los descuentos por **impuestos**: se le descontará un 10% si gana menos de 1000 € al mes, y un 15% si gana más de esa cantidad.
   - El **salario neto**, es decir, el dinero que cobrará después de descontar los impuestos
5. Cuando se trabaja con **polígonos regulares**, conociendo su número de lados, la longitud de cada lado y la apotema[2], se puede calcular el área y el perímetro según estas expresiones:

```
Área = nº lados x longitud del lado x apotema / 2
   Perímetro = nº de lados x longitud del lado
```

   Escribe un programa modular que pregunte esos tres valores y calcule el área y el perímetro de cualquier polígono regular, y además escriba su denominación (triángulo, rectángulo, pentágono, hexágono, etc, hasta polígonos de 12 lados)

6. Escribe un programa para **predecir el tiempo** que va a hacer mañana a partir de varios datos atmosféricos suministrados por el usuario. Estos datos son:

   a. La **presión atmosférica**: puede ser alta, media o baja.
   b. La **humedad relativa**: también puede ser alta, media o baja

   Tienes que hacer **tres subalgoritmos**. El primero se encargará de calcular la probabilidad de lluvia, el segundo la probabilidad de sol y el tercero la probabilidad de que haga frío, mediante estos cálculos:

   - **Para calcular la probabilidad de lluvia**:

Presión	Humedad	Probabilidad de lluvia
Baja	Alta	Muy alta
Baja	Media	Alta
Baja	Baja	Media
Media	Media	Media
En cualquier otro caso		Baja

   - **Para calcular la probabilidad de que haga sol**:

Presión	Humedad	Probabilidad de que haga sol
Baja	Alta	Baja
Baja	Media	Media
Baja	Alta	Media
Media	Media	Media
En cualquier otro caso		Alta

   - **Para calcular la probabilidad que haga frío**:

Presión	Humedad	Probabilidad de que haga frío
Baja	Alta	Alta
Baja	Media	Alta
Media	Alta	Alta
Media	Media	Media
En cualquier otro caso		Baja

7. Escribe un programa que simule el mecanismo de **devolución de monedas** de una máquina expendedora. El programa preguntará una cantidad el euros y luego calculará qué monedas debería devolver la máquina para completar esa cantidad. Por ejemplo, si la cantidad es 1,45 €, la respuesta del programa debe ser: una moneda de 1 €, dos de 20 céntimos y una de 5 céntimos.

   Utiliza programación modular. Puedes elegir entre estos dos enfoques:

   - Escribir **un único subalgoritmo** que se encargue del calcular todas las monedas que hacen falta.
   - Escribir **varios subalgoritmos**, uno para cada tipo de moneda, y que cada uno se encargue de determinar cuántas monedas de ese tipo hacen falta para realizar la devolución.

8. Escribe un programa que simule el funcionamiento de **un reloj**. El usuario introducirá la hora actual a través de teclado y, a partir de entonces, el programa incrementará esa hora cada segundo. Supondremos que existe una **función llamada `esperar()`**, que hace que el ordenador espere la cantidad de segundos indicada entre paréntesis antes de pasar a la siguiente instrucción.

   Por ejemplo, `esperar(5)` detiene momentáneamente la ejecución del programa durante 5 segundos. Del mismo modo, `esperar(1)` la detiene durante 1 segundo.

   Utiliza **un subalgoritmo para actualizar el tiempo** una vez por segundo. En este subalgoritmo puedes reutilizar el código que escribimos en el ejercicio del reloj del tema 2.

9. Vamos a suponer que existe una función de librería que escriba el carácter correspondiente a cierto **código ASCII**. La función la llamaremos `carácter_ascii()`. Por ejemplo, al escribir la siguiente asignación, la variable letra tomará el valor "A", que es el carácter correspondiente al código ASCII 65:

```
letra es carácter
letra = carácter_ascii(65)
```

   Escribe un programa que permita al usuario introducir cualquier número entre 0 y 255 y **muestre el carácter al que corresponde ese código**, con las siguientes salvedades:

   - Los caracteres **del 0 al 31** no son imprimibles, sino que son caracteres de control especiales (como, por ejemplo, "fin de línea"). Si el usuario introduce un código en este rango, el programa debe informarle de su error.

- Cualquier número **fuera del rango de 0 a 255** debe provocar que se escriba toda la tabla de caracteres ASCII desde el 32 hasta el 255.

10. Escribe un programa modular que permita **convertir una cantidad de información** expresada en cualquier unidad de medida (bit, byte, KB, MB y GB) a cualquier otra. El programa, por tanto, debe preguntar la cantidad y la unidad en la que está expresada, así como la unidad a la que se desea convertir.

    Por ejemplo, la siguiente podría ser una secuencia de acciones del programa:

```
Escriba una cantidad de información
> 2
¿En qué unidad está expresada esta cantidad de información (bit,
byte, KB, MB o GB)?
> MB
¿A qué unidad desea convertirla (bit, byte, KB, MB o GB)?
> KB
La solución es: 2048 KB
```

11. Escribe un programa modular que pregunte al usuario su **fecha de nacimiento** y la **fecha del dia de hoy**, y calcule **la edad del usuario en años**.

    Este programa se puede **mejorar** haciendo que calcule la edad en años, meses y días (¡incluso en horas, minutos y segundos!), pero es una labor por ahora solo apta para los/las más valientes.

12. Escribe un programa modular que puedan utilizar en una tienda para **calcular el descuento de los artículos**. En esta tienda aplican un descuento del **15%** en todos los artículos vendidos a los **mayores de 65 años**, y de un **10% a los menores de 25**.

    El programa debe preguntar, al inicio, la **fecha del dia de hoy**, y no volver a preguntarla más. Después irá preguntando por **el precio del artículo** que se vende y la **fecha de nacimiento** del cliente. Entonces calculará la **edad** y, a partir de ella, determinará el **descuento** que se debe aplicar al artículo.

    El proceso se repetirá hasta que se introduzca un precio de artículo negativo.

    Para calcular la edad de los clientes puedes retuilizar el subalgoritmo que escribiste en el ejercicio anterior (he aquí una de las grandes ventajas de utilizar subalgoritmos: el código se puede reutilizar fácilmente).

13. Escribe un programa que lea **tres números, A, B y C**, y que los muestre en la pantalla **ordenados** de menor a mayor, utilizando para ello un módulo que los ordene, intercambiando sus valores si fuera necesario. Usa el método de **paso de parámetros por referencia**.
14. Escribe un programa modular que lea un **número** de hasta 5 cifras por teclado y lo escriba **en forma de letra**. Por ejemplo, si se introduce el número 1980, la salida del programa debe ser "mil novecientos ochenta". Utiliza una función para cada posición del número (unidades, decenas, centenas, etc)
15. Escribe un programa modular que lea un **número binario** por teclado y lo convierta a **decimal**.
16. Estudia detenidamente el siguiente programa sobre un juego de dados y **averigua en qué consiste** exactamente, porque luego lo vas a tener que modificar.

```
algoritmo dados
variables
  puntos_ordenador, puntos_humano son enteros
  turno es cadena
  tirada es carácter
  dado1, dado2, dado3 son enteros
  premio es entero
inicio
  puntos_ordenador = 0
  puntos_humano = 0
  turno = "HUMANO"
  repetir
  inicio
    si (turno == "HUMANO") entonces
    inicio
      escribir ("Es tu turno. Pulsa Enter para tirar los dados o Q para terminar")
      leer (tirada)
    fin
    si_no
      escribir ("Es mi turno.")
    si (tirada != 'Q') entonces
```

```
    inicio
      dado1 = aleatorio(5) + 1
      dado2 = aleatorio(5) + 1
      dado3 = aleatorio(5) + 1
      escribir("La tirada ha sido: ", dado1, dado2, dado3)
      premio = 0
      si (dado1 == dado2) o (dado1 == dado3) entonces
         premio = 10
      si (dado1 == dado2) y (dado1 == dado3) entonces
      inicio
         si (dado1 == 6) entonces
            premio = 50
         si_no
            premio = 20
      fin
      si (turno == "HUMANO") entonces
      inicio
         puntos_humano = puntos_humano + premio
         turno = "ORDENADOR"
      fin
      si_no
      inicio
         puntos_ordenador = puntos_ordenador + premio
         turno = "HUMANO"
      fin
    fin
  fin
  mientras que (tirada != 'Q')
  escribir ("Tu puntuación final: ", puntos_humano)
  escribir ("Mi puntuación funal: ", puntos_ordenador)
  si (puntos_humano > puntos_ordenador) entonces
    escribir("¡Has ganado, enhorabuena!")
  si_no
    si (puntos_ordenador < puntos_humano) entonces
       escribir("Te he ganado. Inténtalo en otra ocasión.")
    si_no
       escribir("Hemos empatado. Intenta ganarme otro día.")
fin
```

Este es un ejemplo de **algoritmo demasiado complejo** y, por lo tanto, demasiado difícil de entender, de modo que lo vamos a descomponer en subproblemas más sencillos. Los subproblemas (módulos) serán:

- ***Módulo turno**: escribe en la pantalla de quién es el turno y lee la tirada en caso de que el turno sea del jugador humano*
- ***Módulo tirar_dados**: tira los tres dados y escribe el resultado en la pantalla*
- ***Módulo comprobar_dados**: compara el resultado de los dados y determina qué premio corresponde a la tirada*
- ***Módulo sumar_premio**: suma el premio de la tirada a la puntuación del jugador que tiene el turno*
- ***Módulo escribir_resultado**: muestra en la pantalla la puntuación final de cada jugador y un mensaje diciendo quién ha ganado.*

Reescribe el programa completo, añadiendo comentarios donde consideres necesario. Respeta el siguiente diagrama de descomposición modular y complétalo señalando el trasiego de información entre los módulos: