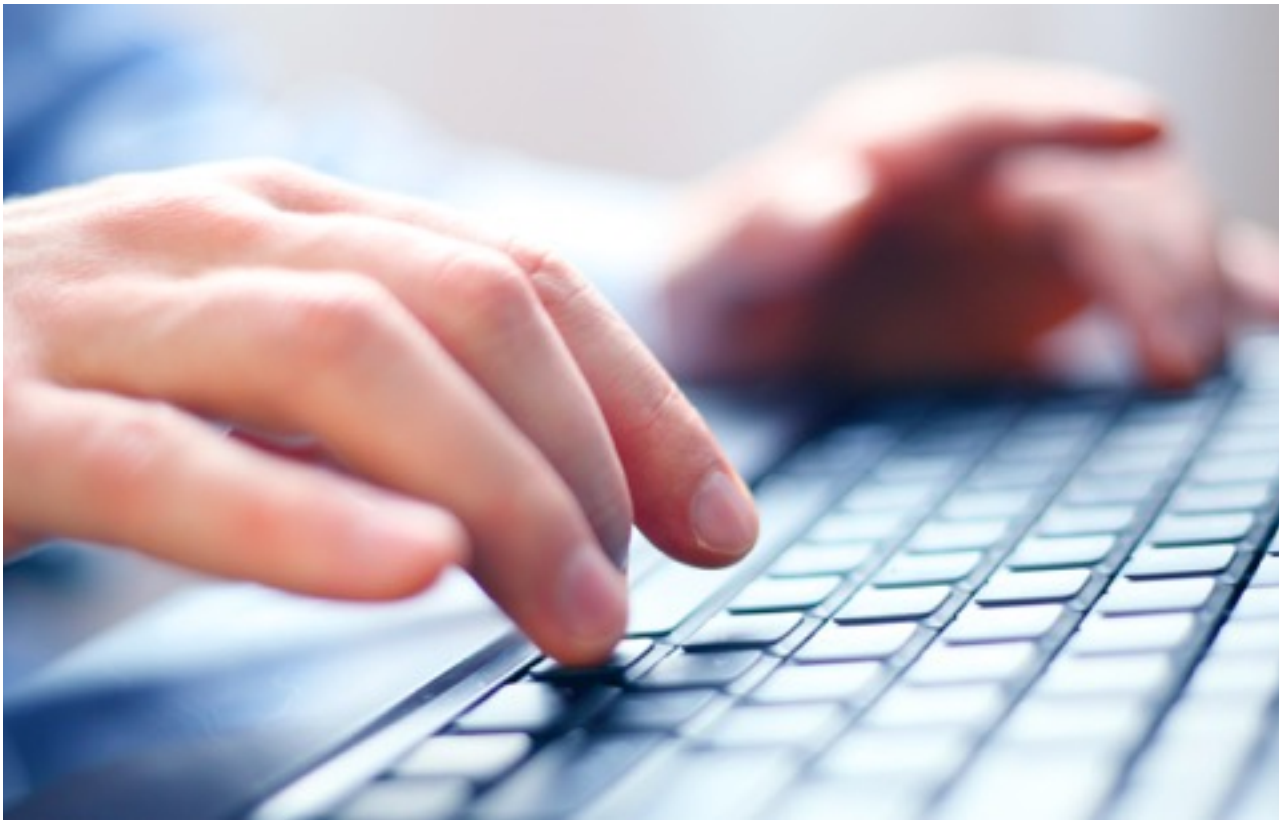
Resolución de Problemas

Introducción a las ciencias de la computación y al pensamiento computacional.



Curso de Programación en Java
Juan Francisco Maldonado León

Conocer el problema a resolver

Como primer paso a la hora de desarrollar un programa tenemos "conocer el problema a resolver". Necesitaremos un conocimiento profundo de todos los aspectos relacionados con el problema, lo cual implica saber responder las siguientes preguntas:

1. ¿Cuál es mi objetivo?
2. ¿Cuáles son los condicionantes que afectan al problema?
3. ¿Qué método o esquema de resolución voy a aplicar?
4. ¿Cuáles son los datos de partida?
5. ¿Qué resultado quiero obtener?

Sobre el objetivo

A la hora de plantear un objetivo trataremos de subdividir la extensión y complejidad del problema hasta niveles lo más fácilmente abarcables por una persona, según la conocida estrategia del "divide y vencerás".

Aunque será la experiencia la que mejor nos guíe a la hora de plantear objetivos podemos usar esta regla: "Sólo trataremos de programar aquello que mentalmente somos capaces de abarcar en método, extensión y condicionantes".

Sobre los condicionantes

Llamamos condicionantes a todos los factores que afectan a la resolución del problema. Después de tener un objetivo centrado y enfocado tendremos que valorar qué condicionantes nos afectan y si tenemos un conocimiento suficiente de ellos.

Podemos hacer la siguiente clasificación:

1. ***Condicionantes de cálculo***:

   Aquellos que afectan a la estrategia de resolución del problema.

   Ejemplo: En un programa para calcular las pérdidas de carga en tuberías, admitir o no la presencia de accesorios (codos, válvulas, etc).

2. ***Condicionantes tipo parámetro:***

   Son los referidos a materiales, dimensiones o formas geométricas, tiempos, etc.

   Ejemplo: viscosidad de un líquido, peso específico de un material.

3. ***Condicionantes bifurcadores:***

   Serían los que, en función de un valor introducido por el usuario o un valor resultado intermedio llevan a distintas vías de resolución.

Ejemplo: para el cálculo de pérdidas de carga en una tubería el condicionante: "¿se trata de régimen laminar?" supone la bifurcación:

Sí -> Aplicar hf = formula1

No -> Aplicar hf = formula2

4. ***Condicionantes tipo restricción:***

   4.1. Para admisión de datos: pueden resultar no admisibles ciertos valores para datos de entrada, bien por motivos técnicos, comerciales u operativos, o bien por imposibilidad física, matemática, etc.

Ejemplo: para ordenar una serie de números es viable aceptar números positivos o negativos. En cambio, si pedimos la altura de un pilar no tiene sentido admitir un valor negativo.

Si tratamos de realizar ciertos cálculos relacionados con una nave industrial, podemos acotar el rango de luz o distancias entre pilares a distancias técnicamente viables. Podemos preparar el programa para que detecte cualquier valor que consideremos anómalo e impida la continuación de procesos hasta tanto no se corrija.

4.2. Para la emisión de resultados: ciertos datos de entrada en principio viables pueden dar lugar a resultados no admisibles.

Ejemplo: se trata de determinar el número de farolas necesarias por kilómetro para la iluminación de una avenida, siendo la dimensión de la base de la farola 40x40 cm. Si por cualquier circunstancia el programa llegara a un valor de más farolas de las que física o razonablemente se pueden disponer, deberíamos abortar la presentación del resultado e instar a corregir los datos de partida como potencia, altura, etc., para poder obtener un resultado viable. Podemos preparar el programa para que detecte cualquier resultado que pudiéramos considerar anómalo, impidiendo su presentación y mostrando un mensaje de error o advertencia.

Sobre el método o esquema de resolución

Una vez determinado el objetivo y conocidos los condicionantes, valorar el método o esquema de resolución para el problema planteado será el siguiente paso en el proceso de conocer el problema. Será una etapa más a cumplir antes de empezar a trabajar en su programación.

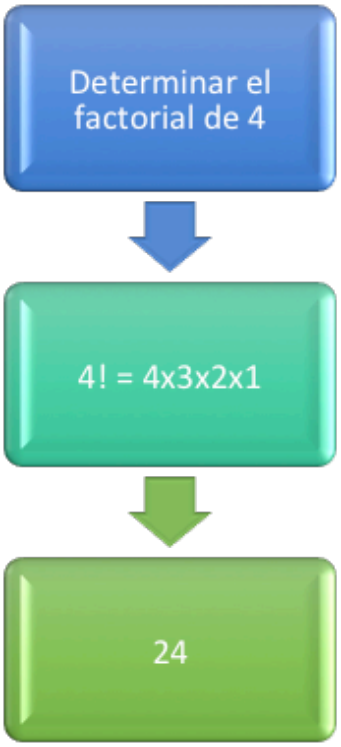
Los métodos aplicables son muy diversos por lo que es difícil hacer acotaciones teóricas respecto a los mismos. Haremos pues una clasificación práctica de distintas formas o tipos de resolución de problemas a los que nos enfrentaremos como programadores. Un programa extenso puede ser una mezcla de distintos tipos de problema.

Problema con resolución directa

Se trataría de todo tipo de problemas que solucionamos mentalmente, de forma sencilla, en uno o varios pasos. El esquema a que nos referimos sería del tipo:

Se trataría de programas o partes de programas en los que únicamente utilizamos razonamientos más o menos directos, operaciones básicas (sumas, restas, ...) reglas de tres, fórmulas poco complejas, etc.

Ejemplo:

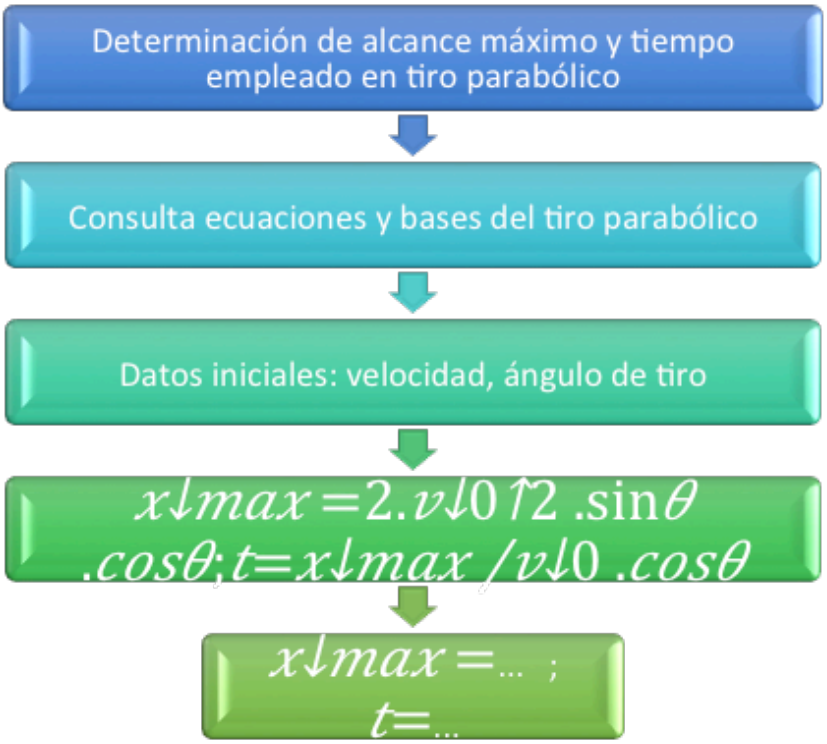


Puede darse el caso de problemas largos (muchos pasos a dar) pero operaciones sencillas. Más que la longitud, será la existencia de múltiples bifurcaciones lo que pueda complicar la programación.

Problema con resolución documentada

Se trataría de un problema con tipología similar al anterior pero que no resolvemos directamente sino mediante una consulta a prontuarios, manuales, libros o apuntes. El esquema sería del tipo:

Ejemplo:



Aparte de lo dicho para el caso anterior, convendrá tener cuidado con el uso de fórmulas matemáticas y su escritura.

Problema con iteración para búsqueda de soluciones

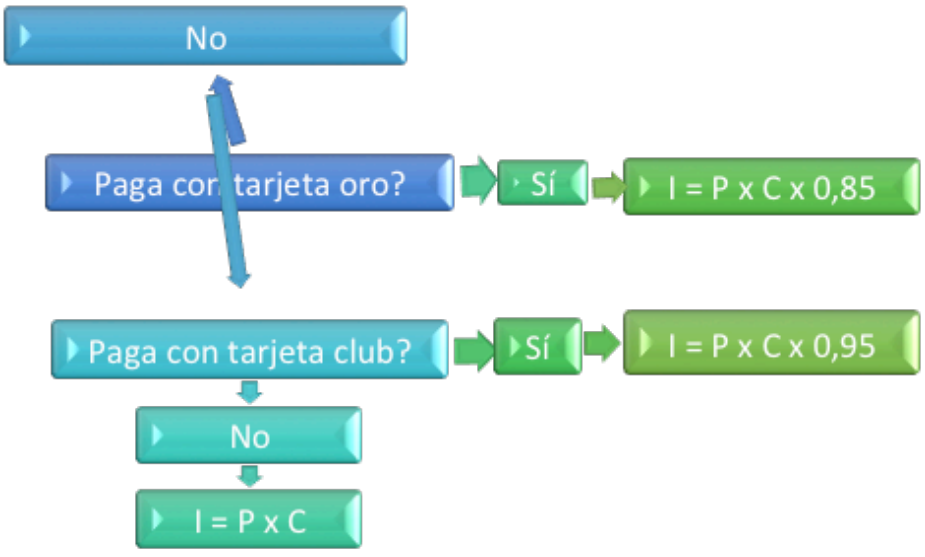
Problemas con tipología diversa en cuanto a métodos de resolución, complejidad, etc., pero que tienen en común adoptar esquemas reiterativos de forma que es la repetición n veces de un método de búsqueda o iteración la que da lugar a unos resultados. El esquema sería del tipo:

Problema a plantear con una tabla de decisión

A la hora de resolver problemas, serán habituales esquemas de tipo lineal o lineal con bifurcaciones, en los que una bifurcación se decide a través de una pregunta sencilla. Supongamos que trabajamos en unos grandes almacenes y queremos programar el importe a facturar a los clientes.

Importe = Precio unitario x Cantidad x Descuento

Supongamos a su vez dos formas de pago: con tarjeta oro (15% de descuento) y con tarjeta club (5% de descuento). El planteamiento sería:



Nuestro esquema de decisión es prácticamente inmediato por ser el número de condicionantes relativamente pequeño. En cambio, cuando las cosas se complican ya no será tan fácil usar un esquema de decisión de este tipo.

Ejemplo facturación grandes almacenes:

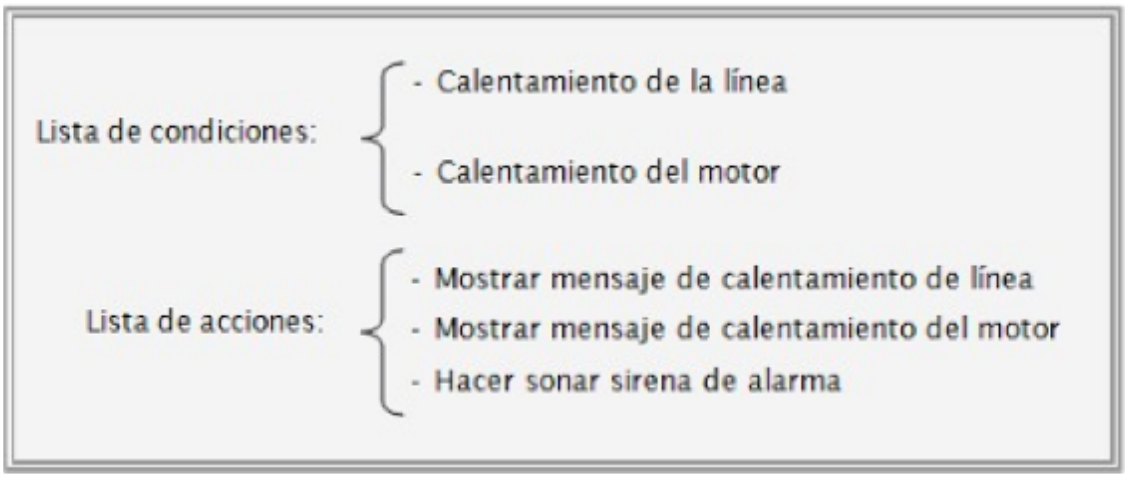
- Pago con tarjeta oro ---> 15% de descuento.
- Pago con tarjeta club ---> 5% de descuento.
- Modalidad joven de tarjeta ---> incrementa en 5% de descuento.
- Período de rebajas ---> incrementa 5% de descuento.
- Artículo en oferta ---> descuento del 10% sólo acumulable si la tarjeta es joven o es período de rebajas.

En este caso, con diferentes acciones a desarrollar en función de múltiples condicionantes, es de utilidad recurrir a una tabla de decisión, que no es otra cosa que una tabla donde organizamos los condicionantes y los resultados a los que dan lugar, con el fin de lograr una mejor estructuración del problema.

Las tablas de decisión, por su carácter de tablas donde se combinan condiciones y acciones, pueden trabajarse en base a reglas y criterios matemáticos. Para construir una tabla de decisión

partiremos de una tabla dividida en cuatro zonas: lista de condiciones, cumplimiento de condiciones, lista de acciones y acciones a realizar.

Supongamos que existe un ordenador que controla un motor y su línea de alimentación. Se quiere programar para que el calentamiento de línea o motor dé lugar a un mensaje de alarma en pantalla y que además, en caso de calentamiento del motor se haga sonar una sirena de alarma.



Construimos la tabla de decisión con las listas de condiciones y acciones y rellenando las cuadrículas de la zona 2 con Sí o No y las de la zona 4 con una cruz (indica hacer) o un espacio en blanco (no hacer).

Calentamiento línea	Sí	Sí	No	No
Calentamiento motor	Sí	No	Sí	No
Mensaje calentamiento línea	X	X		
Mensaje calentamiento motor	X		X	
Hacer sonar sirena de alarma	X		X	
Continuar el proceso				X

Cada conjunto de condiciones constituye un caso, por ejemplo el caso de que haya calentamiento de línea sin calentamiento de motor. El número de casos que tendremos es 2^n, siendo n el número de condiciones. En el ejemplo anterior tenemos dos condiciones y $2^n = 2^2 = 4$ casos, que son Sí-Sí, Sí-No, No-Sí y No-No. Para construir la zona 2 o de cumplimiento de condiciones seguiremos la siguiente estrategia:

- Primero la columna con todo Sí.

- Segundo todas las columnas posibles con un único No.
- Tercero todas las columnas posibles con dos No.
- Sucesivamente todas las columnas posibles con tres, cuatro, cinco, etc. No.
- Finalmente la columna de todo No.

Problema con resolución intuida pero método "paso a paso" a determinar

Se trataría, junto a lo que hemos llamado “problemas con iteración para búsqueda de soluciones”, de un tipo de problemas que invitan a pensar y a desarrollar nuestra creatividad como programadores.

En estos casos, no sólo tenemos que trasladar al ordenador instrucciones para que ejecute un proceso conocido, sino que, como paso previo, hemos nosotros de determinar cuál es el proceso “paso a paso”. Nos referimos, en general, a problemas que sabemos responden a premisas matemáticas, mecánicas, lógicas, ... muchas veces de apariencia trivial pero de los que se nos escapan cuáles son los procesos intermedios.

Vamos a relacionar múltiples problemas indicando cómo los clasificaríamos:

a) Cerrar una puerta --> resolución directa.
b) Clavar un clavo --> resolución con iteraciones.
c) Calcular volumen de un cilindro --> resolución directa.
d) Hablar --> resolución intuida.
e) Preparar una bechamel --> resolución documentada.
f) Pintar una casa --> resolución directa.
g) Ordenar una serie de números --> resolución intuida.

h) Facturar a un cliente en una agencia de viajes, existiendo múltiples condicionantes para determinar el precio --> resolución con tabla de decisión.

i) Realizar la declaración de la renta --> resolución documentada.

j) Elegir el color para pintar una casa --> subjetivo, no programable.

k) Conducir un coche --> resolución intuida.

l) Recoger a una persona que ha entrado a un cajero si llevamos un coche y nos vemos obligados a dar vueltas a la manzana --> resolución con iteración.

m) Calcular pérdidas de carga en tuberías --> resolución documentada.

n) Calcular nóminas de los empleados de una empresa, existiendo múltiples condicionantes (pluses, sanciones, niveles, etc.) --> resolución con tabla de decisión.

o) Determinar los números primos entre uno y mil --> resolución intuida.

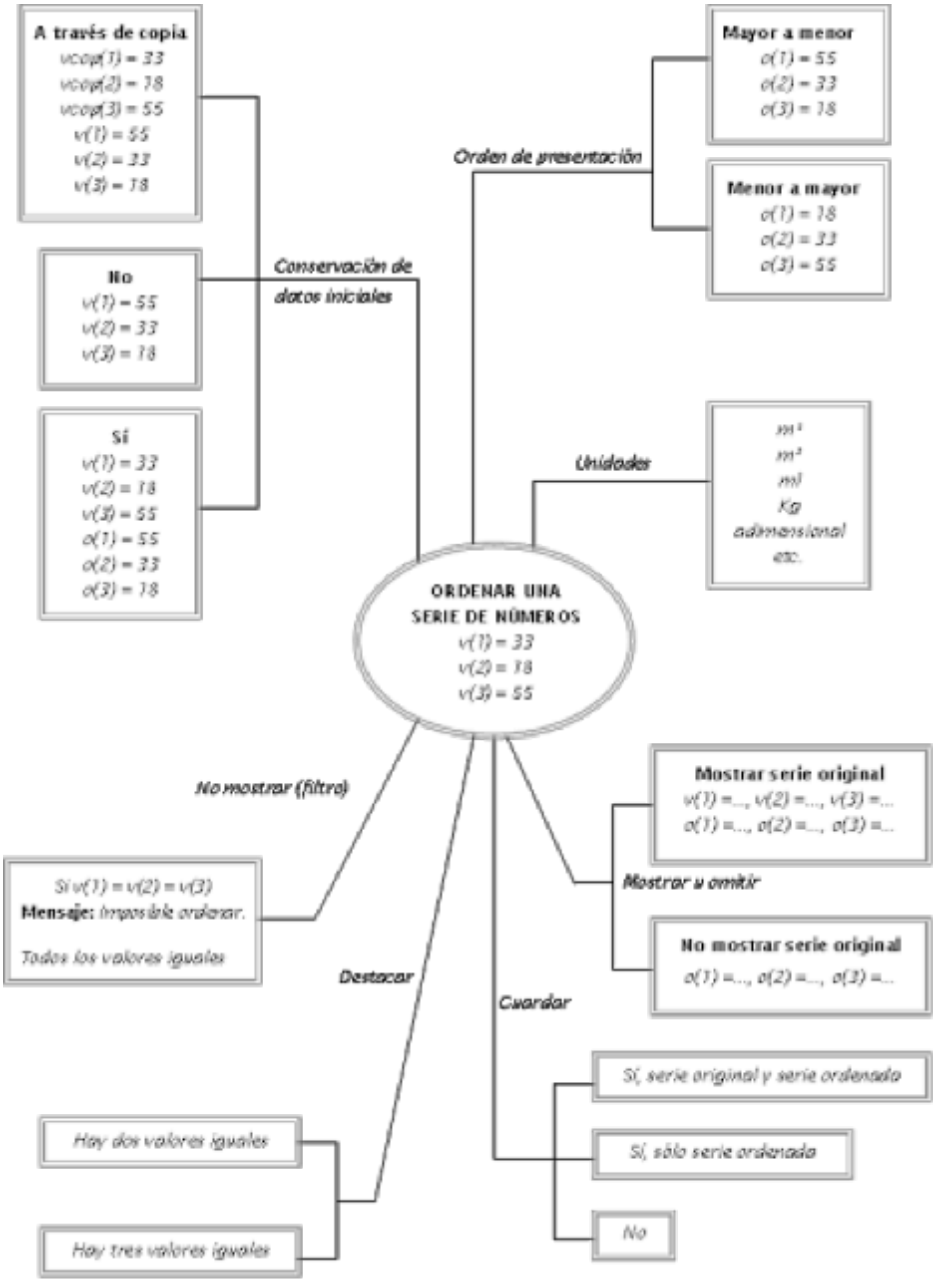
p) ¿Qué hacer si nos encontramos a 100 Km del lugar donde tenemos una reunión dentro de 30 minutos? --> subjetivo, no programable.

Sobre los resultados a obtener

Podemos pensar en:

- Conservar o no los datos iniciales del problema.
- En qué unidades expresamos los resultados.
- En qué orden mostramos los resultados.
- Resultados a mostrar y resultados a omitir (por ejemplo resultados intermedios).
- Resultados que se guardan y que no se guardan.
- Resultados a destacar en caso de que resulten valores extraños, especiales, sorprendentes, etc.
- Resultados a mostrar por no superar un filtro (por ejemplo sólo se aceptan valores enteros positivos).

La forma de presentar los resultados puede condicionar la estructura misma del programa, de ahí la necesidad de tener claro qué se quiere hacer. El esquema representado muestra, a modo de ejemplo, algunas reflexiones que se podrían hacer en torno a los resultados de un programa para “ordenar una serie de números”.


ORDENAR UNA SERIE DE NÚMEROS
v(1) = 33
v(2) = 18
v(3) = 55
Conservación de datos iniciales
A través de copia
vcop(1) = 33
vcop(2) = 18
vcop(3) = 55
v(1) = 55
v(2) = 33
v(3) = 18
No
v(1) = 55
v(2) = 33
v(3) = 18
Sí
v(1) = 33
v(2) = 18
v(3) = 55
o(1) = 55
o(2) = 33
o(3) = 18
Orden de presentación
Mayor a menor
o(1) = 55
o(2) = 33
o(3) = 18
Menor a mayor
o(1) = 18
o(2) = 33
o(3) = 55
Unidades
m³
m²
ml
Kg
adimensional
etc.
No mostrar (filtro)
Si v(1) = v(2) = v(3)
Mensaje: Imposible ordenar.
Todos los valores iguales
Mostrar u omitir
Mostrar serie original
v(1) =..., v(2) =..., v(3) =...
o(1) =..., o(2) =..., o(3) =...
No mostrar serie original
o(1) =..., o(2) =..., o(3) =...
Destacar
Hay dos valores iguales
Hay tres valores iguales
Guardar
Sí, serie original y serie ordenada
Sí, sólo serie ordenada
No