

# VALIDACIÓN DE DATOS



Introducción a la Programación (11071)  
Departamento de Ciencias Básicas  
Universidad Nacional de Luján



# MATERIAL RELACIONADO

## TEORIA



**Apunte**

**Cap.**

**6.3**

-

Validaciones.

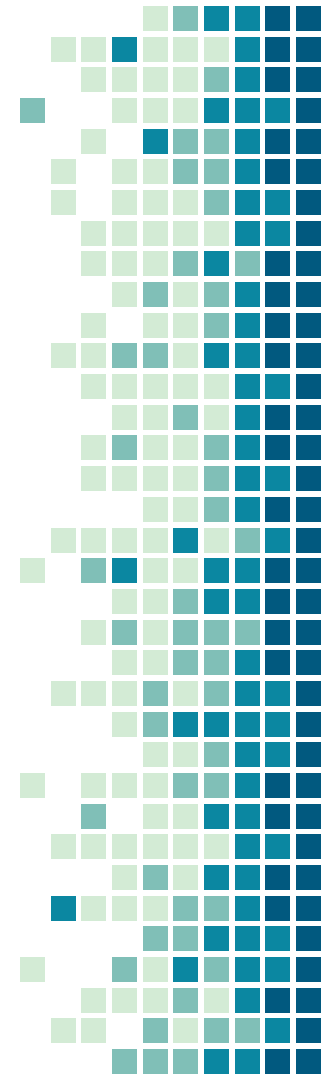
## PRÁCTICA



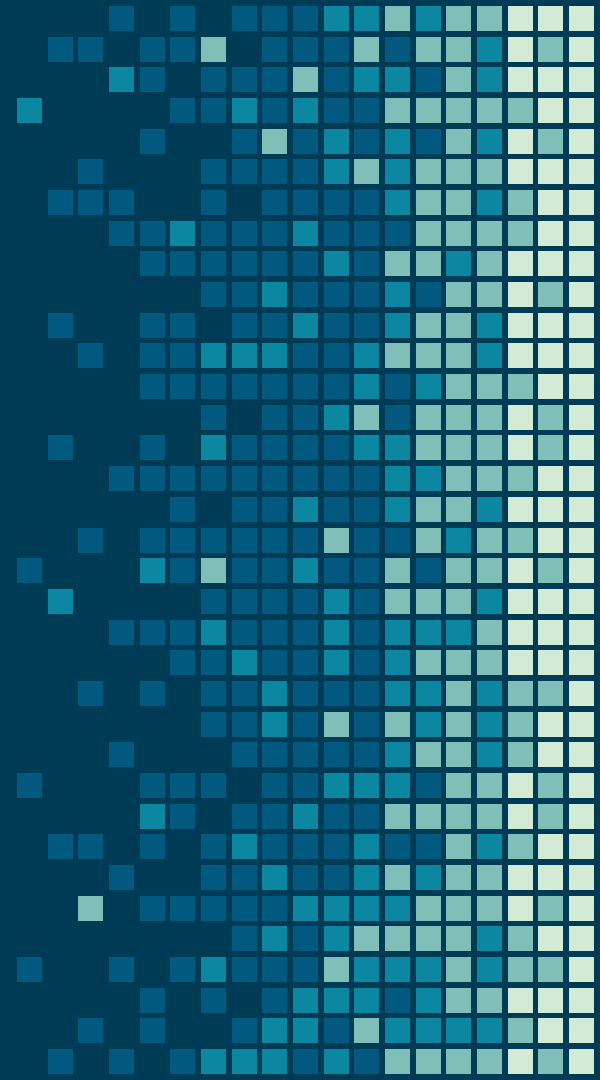
**TP IX** - Validación de entrada.

## ENTREGABLES

*No posee.*



¿QUÉ?



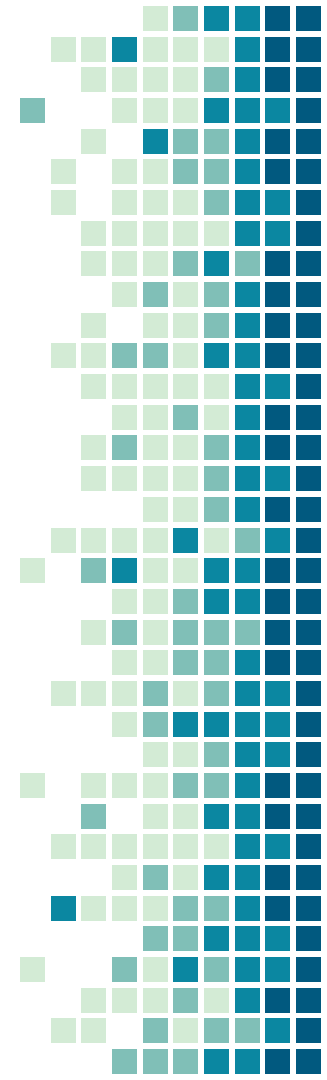
# CONCEPTO

## VALIDACIÓN DE DATOS

Proceso mediante el cual se asegura que los datos manejados en un programa son admisibles para su utilización.

## TIPOS DE VALIDACIÓN

- **Chequeo de tipos:** verificar que el dato es del tipo esperado.
- **Chequeo de rango/restricción:** verificar que el dato es un valor admitido dentro del dominio del problema.
- **Chequeo por referencia cruzada:** verificar que el dato es un valor admitido mediante una referencia o código externo (CP, código de país, etcétera).
- **Chequeo de consistencia:** verificar que el dato es consistente con lo que representa (por ejemplo, una fecha de entrega no puede ser anterior a la fecha de despacho).
- **Otros** (chequeo estructurado, mediante ejemplos, etcétera).



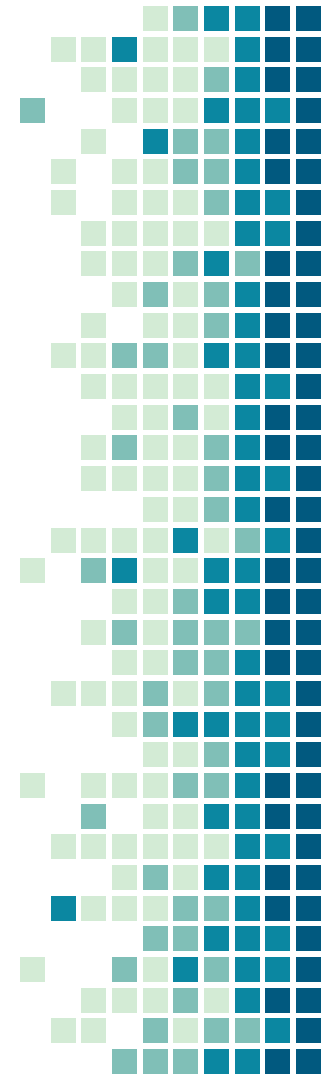
# CONCEPTO

## VALIDACIÓN DE DATOS

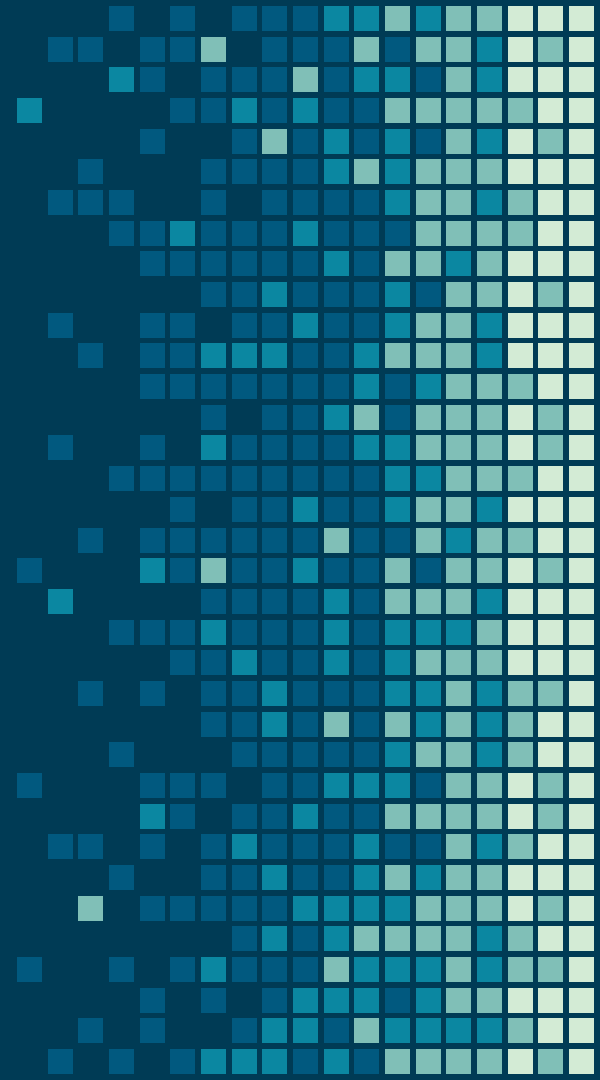
Proceso mediante el cual se asegura de que los datos a utilizar por un programa son válidos y útiles.

## TIPOS DE VALIDACIÓN

- **Chequeo de tipos:** verificar que el dato es del tipo esperado.
- **Chequeo de rango/restricción:** verificar que el dato es un valor admitido dentro del dominio del problema.
- **Chequeo por referencia cruzada:** verificar que el dato es un valor admitido mediante una referencia o código externo (CP, código de país, etcétera).
- **Chequeo de consistencia:** verificar que el dato es consistente con lo que representa (por ejemplo, una fecha de entrega no puede ser anterior a la fecha de despacho).
- **Otros** (chequeo estructurado, mediante ejemplos, etcétera).



¿POR QUÉ?



# LA IMPORTANCIA DE VALIDAR

## CORRECCIÓN

Para que el programa funcione como se espera.

## ROBUSTEZ

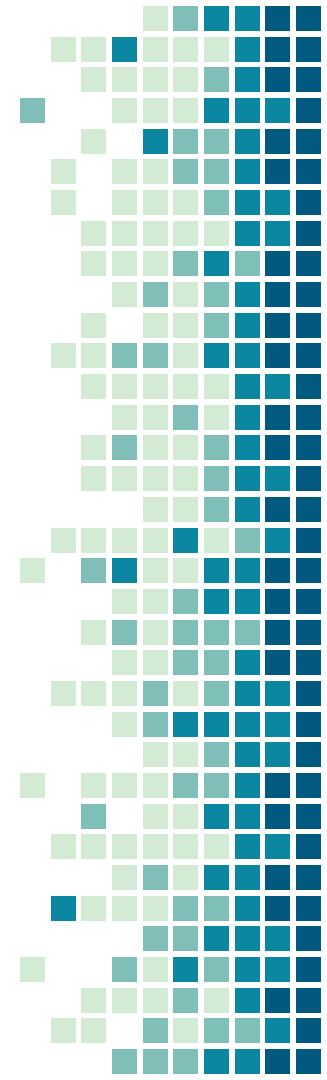
Para que el programa no colapse ante casos inesperados.

## INTEGRIDAD DE LOS DATOS

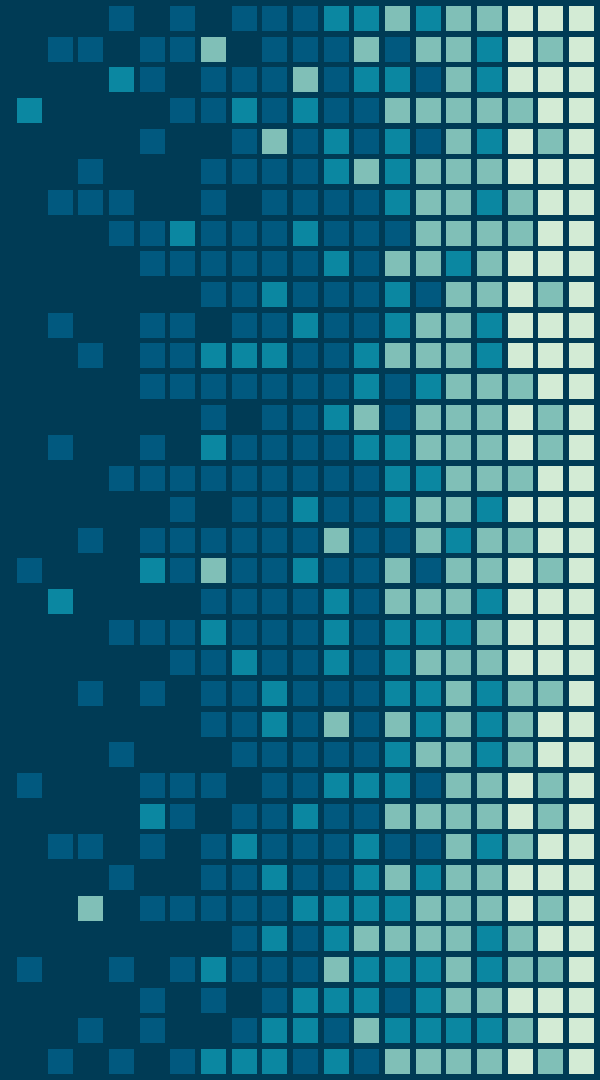
Para que los datos almacenados por el programa sean confiables.

## SEGURIDAD

Para evitar vulnerabilidades relacionadas a la validación de entrada (*buffer overflow*, *inyección*, etcétera).



¿CÓMO?





# CÓMO VALIDAR

## CÓDIGO ESPECÍFICO

Determinadas partes de nuestro programa se encargarán específicamente de realizar validaciones.

## DÓNDE CODIFICAR LA VALIDACIÓN

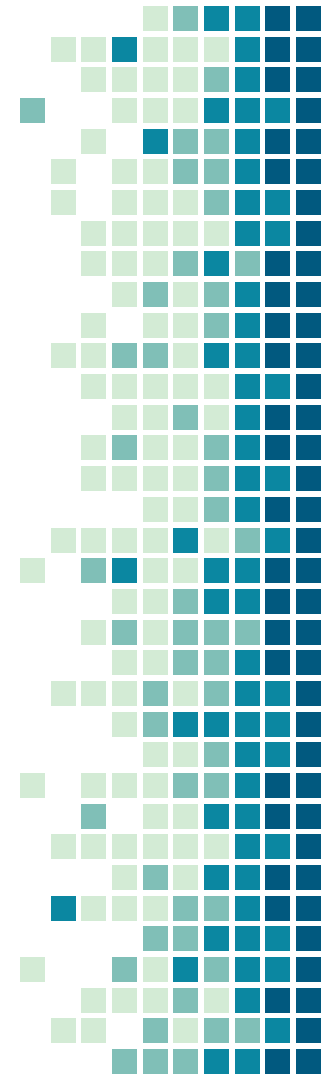
Depende de las necesidades del problema, pero en general podemos:

- **Validar in-line**, cuando es una condición simple. Ejemplo:

```
if edad > 110:  
    print('Edad inválida')
```

- **Utilizar una función de validación**, para modularizar el código. Ejemplo:

```
def edad_valida(edad):  
    return edad <= 110
```



# VALIDACIÓN DE ENTRADA



# VALIDACIÓN DE ENTRADA DE USUARIO

## DATA-ENTRY (INGRESO DE DATOS)

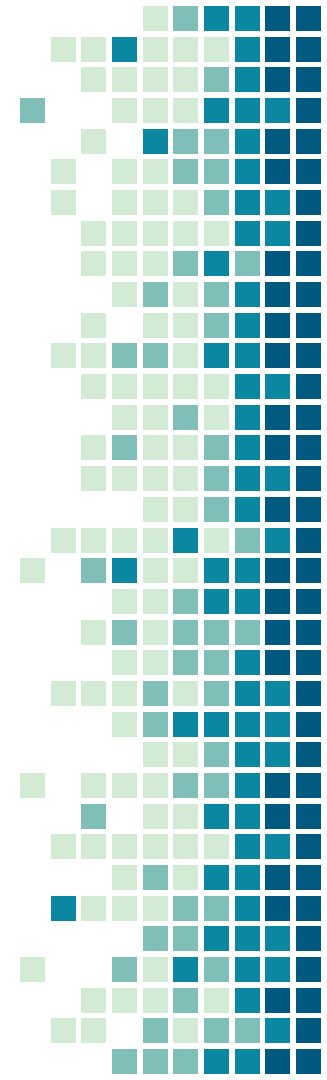
Cuando el programa toma datos desde una fuente humana (formularios, lecturas desde teclado, etcétera).

## VALIDACIÓN DE ENTRADA

Todo dato obtenido desde la entrada de usuario debe ser validado, **no importa cuán trivial sea**.

*Es imposible hacer todo “a prueba de tontos”, porque los tontos son muy ingeniosos.*

Ley de Murphy.



# CONSIDERACIONES DE LA ENTRADA

## NO CONFIAR EN EL USUARIO

Si hay alguna forma en que el usuario puede ingresar mal un dato (por error o malicia), entonces debe ser contemplado por nuestro código, no importa cuán trivial parezca.

## RESTRINGIR LA POSIBILIDAD DE “HACERLO MAL”

Podemos minimizar el error humano restringiendo la forma en que el usuario ingresa datos en nuestro programa (*difícil en programas de terminal de comandos como los que hacemos en la asignatura*).

## EJEMPLO


## TÍPICO

Date

DD / MM / YYYY

 /  / 

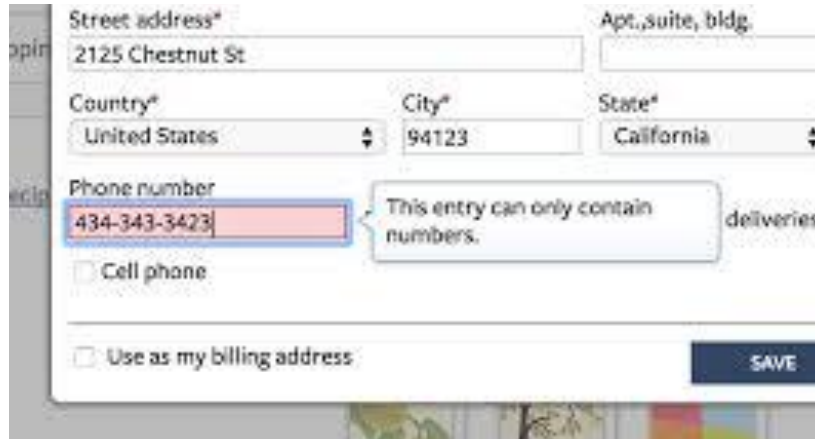
Date

# CONSIDERACIONES DE LA ENTRADA

## INFORMAR AL USUARIO

Cuando se requiere tomar un dato desde la entrada, siempre se debe ser explícito con el formato o valores esperados. Cuando se detecta un error, se le debe indicar al usuario **qué** dato ha ingresado mal, y **por qué**.



The image shows a web form for address and phone information. The form includes fields for Street address\*, Apt., suite, bldg., Country\*, City\*, State\*, Phone number, and a checkbox for Cell phone. A validation error message is displayed next to the Phone number field, stating "This entry can only contain numbers." The phone number entered is "434-343-3423". The form also has a checkbox for "Use as my billing address" and a "SAVE" button.

Street address*	Apt., suite, bldg.	
2125 Chestnut St		
Country*	City*	State*
United States	94123	California
Phone number	This entry can only contain numbers.	
434-343-3423	deliveries	
<input type="checkbox"/> Cell phone		
<input type="checkbox"/> Use as my billing address	SAVE	

# CONSIDERACIONES DE LA ENTRADA

## REITERACIÓN DE ERRORES

Se debe asumir que el usuario puede ingresar un dato inválido más de una vez.

```
1. numero = int(input('Ingrese un número entre 1 y 10:'))
2.
3. if numero < 1 or numero > 10:
4.     numero = int(input('El valor ingresado es inválido. Por favor, ingrese un número entre 1 y 10:'))
5.
6. # ... procesamiento del dato leído
```

# CONSIDERACIONES DE LA ENTRADA

## REITERACIÓN DE ERRORES

Se debe asumir que el usuario puede ingresar un dato inválido más de una vez.

```
1. numero = int(input('Ingrese un número entre 1 y 10:'))
2.
3. if numero < 1 or numero > 10:
4.     numero = int(input('El valor ingresado es inválido. Por favor, ingrese un número entre 1 y 10:'))
5.
6. # ... procesamiento del dato leído
```

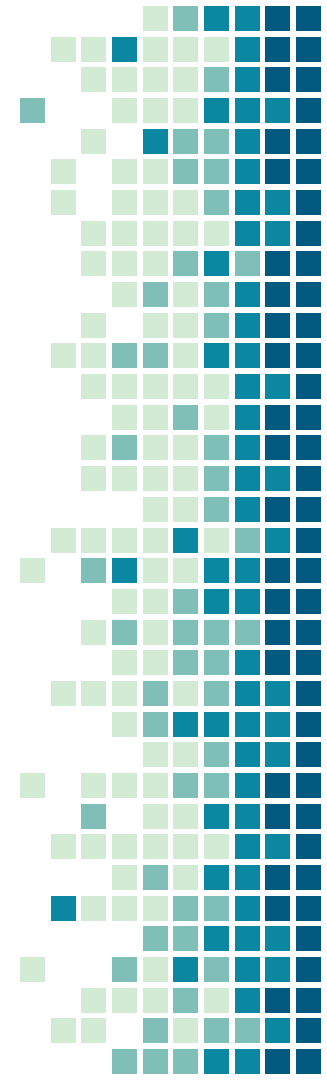
*En el código de arriba, si el usuario vuelve a ingresar un dato inválido en el segundo input (línea 4), el programa no lo detectará, y pasará a procesarlo de todas formas (consecuencias de corrección, robustez, integridad de datos, seguridad, etcétera).*

# CONSIDERACIONES DE LA ENTRADA

## REITERACIÓN DE ERRORES

Se debe asumir que el usuario puede ingresar un dato inválido más de una vez.  
Como consecuencia:

- La estructura condicional simple (*if*) no sirve como método de validación de entrada.
- Se debe utilizar una estructura iterativa condicional (*while*), utilizando como condición al estado de validación del dato en cuestión.





# CONSIDERACIONES DE LA ENTRADA

## REITERACIÓN DE ERRORES

Se debe asumir que el usuario puede ingresar un dato inválido más de una vez.

```
numero = int(input('Ingrese un número entre 1 y 10:'))

while numero < 1 or numero > 10:
    numero = int(input('El valor ingresado es inválido. Por favor, ingrese un número entre 1 y 10:'))

# ... procesamiento del dato leído
```

# CONSIDERACIONES DE LA ENTRADA

## REITERACIÓN DE ERRORES

Se debe asumir que el usuario puede ingresar un dato inválido más de una vez.

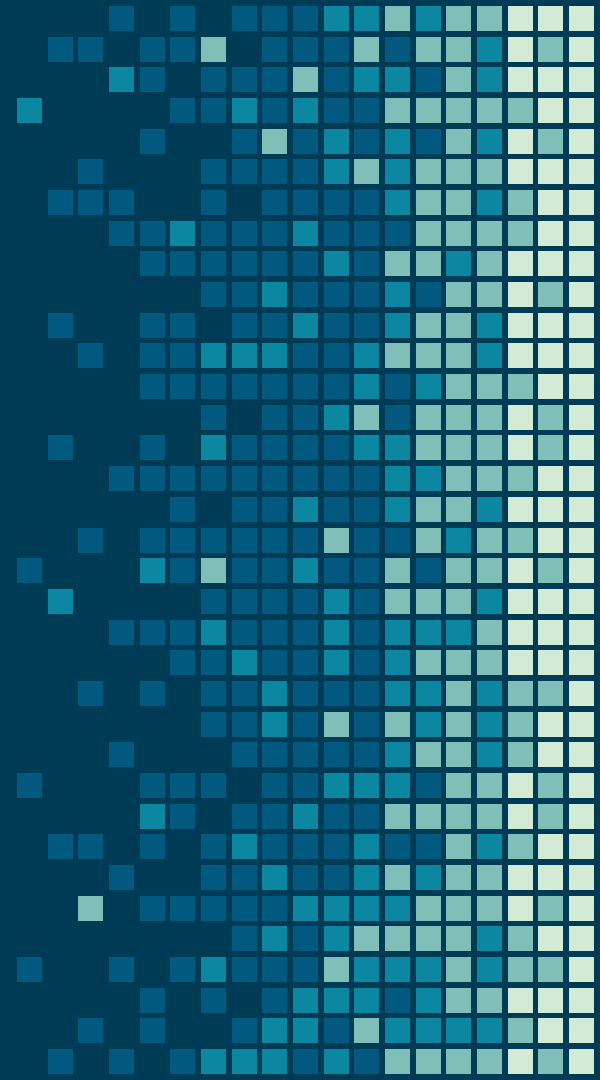
```
numero = int(input('Ingrese un número entre 1 y 10:'))

while numero < 1 or numero > 10:
    numero = int(input('El valor ingresado es inválido. Por favor, ingrese un número entre 1 y 10:'))

# ... procesamiento del dato leído
```

*Al utilizar una estructura repetitiva condicional teniendo al resultado de la validación como condición, el programa le pedirá al usuario reingresar el valor incorrecto cuantas veces sea necesario, hasta que el dato en cuestión sea válido.*

# RESUMEN



# RECORDAR

- **Validar todo dato susceptible a errores**, sea el parámetro de una función o un dato ingresado por el usuario.
- **Validar tipo y rango/restricción**; se pueden hacer más validaciones, pero quedan por fuera del alcance de esta asignatura.

Específicamente, cuando se trata de un dato ingresado por el usuario:

- **Contemplar errores reiterados en el ingreso**, no asumir que el usuario ingresará correctamente el dato en el segundo intento (el if no sirve en esta situación).
- **Informar al usuario** el formato/rango/valores esperados, y ser claro cuando ha cometido un error y debe reingresar un dato.

