ALGORITMOS

· **Estabilidad**: un algoritmo es estable cuando no diverge del resultado real: recordemos que los ordenadores tienen cierto error intrínseco; y hay algoritmos que, pese a ser matemáticamente correctos, crean un error enorme.

Como regla de oro, cuantas menos operaciones se hagan, más estable será el algoritmo.

· Usamos *pseudocódigo*:

- Empieza por inicio y acaba por fin
- No hace falta declarar el tipo de variable. Las estructuras complejas, como vectores o matrices, se declaran al principio:

  Vector(1:6): las etiquetas son Vector(1), Vector(2), Vector(3),..., Vector(6)

  (1 → la etiqueta de la 1ª componente (puede ser cualquier nº)) (6 → la etiqueta de la última componente)

- Los comentarios, de una línea, empiezan por ! → Puede ser un comentario tras una orden, a la misma línea
- Las variables (sus valores) deben empezar por una letra. Se asignan valores con ← x ← 2
- Las operaciones aritméticas son las habituales:

  +
  -
  *
  /
  ^
  y
  o
  no
  <
  >=
  ==
  /=
  raiz
  seno
  coseno
  ...
  } nombres coloquiales

- Omitir tildes en todas partes salvo quizás en comentarios
- leer var: asigna el input a la variable var. Análogo con escribir var, comentar Texto.

- Estructura selectiva:

si (condición$_1$) entonces
instrucciones
[si no si (condición$_2$) entonces
instrucciones
si no
instrucciones] → Opcional
fin si

con condición$_1$ y condición$_2$ mutuamente excluyentes

- Estructura de casos:

seleccionar (variable)
caso (valor 1)
instrucciones
caso (valor 2)
instrucciones
caso (valor 3)
instrucciones
[en otro caso
instrucciones]
fin seleccionar

- Estructuras iterativas:

para i ← valor_inicio hasta valor_final hacer
instrucciones
fin para

y

hacer
instrucciones] ejecuta esto una y otra vez hasta que se cumpla con una salida
fin hacer

- Es un lenguaje que hay que compilar. En este proceso, se optimiza todo.

- Vayamos a las tres estructuras conocidas:

```
if (condicion1) then
    instrucciones
else if (condicion2) then
    instrucciones
else if (condicion3) then
    instrucciones
else
    instrucciones
end if
```

(else if, else: opcional)

```
select case (selector)
  case (valor1)
    instrucciones
  case (valor2)
    instrucciones
  case (valor3)
    instrucciones
  case default
    instrucciones
end select
```

(case default: opcional)

Los valores puede ser varios, separados por comas (2,3,4) o por intervalos (2:4)

(2:4 opcional)

```
nombre: do i = v_ini, v_fin, paso
    instrucciones
end do nombre
```

(nombre: opcional; paso opcional; nombre opcional)

→ Variable entera que hay que declarar al principio

```
nombre: do
    instrucciones
end do nombre
```

(nombre: opcional; nombre opcional)

Se sale con exit; o cycle (esto sirve automáticamente una nueva iteración)

Si usamos exit nombre; o cycle nombre; se refiere a cierto ciclo; es útil para anidados

En instrucciones print y read, se puede poner ciclos do implícitos: print *, (a(i), b(i), ..., i = inicio, fin)

lee/escribe a(i), b(i), ... (fin - inicio) → i variado desde inicio hasta fin.

- Funciones y procedimientos.

función nombre_función (valor_entrada$_1$, ..., valor_entrada$_n$)
(Opcional)
instrucciones
devolver salida → (obligatorio y es lo que distingue a función de procedimiento)
fin función nombre_función → ¡OJO!!

y

procedimiento nombre(valor_entrada$_1$, ..., valor_entrada$_n$)
instrucciones → puede devolver 1 valor, nul o ninguno
fin procedimiento nombre

los procedimientos pueden modificar el valor de entrada (en las funciones no se debe)