

Estabilidad: un algoritmo es estable cuando no diverge del resultado real: recordemos los ordenadores tiene ciertos errores intrínsecos; 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.

Veremos 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:

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

\swarrow etiqueta de la 1ª componente (puede ser cualquier n°)
 \searrow etiqueta de los otros componentes

- los contadores, de una línea, empiezan por ! \rightarrow puede estar en contadores tras una orden, a la siguiente línea
- las variables (sus valores) deben empezar por una letra. Se asignan valores con \leftarrow \times \leftarrow 2
- las operaciones aritméticas serán habituales:

- +
 -
 - *
 - /
 - ^
 - √
 - o
 - no
 - <
 - >=
 - =
 - ≠
 - sqrt
 - sin
 - cos
 - ...
- } valores aritméticos

- Antin Tilly = todas partes salvo quizás a constantes
- leer var: asigna el input a la variable var. Así como escribir. Usar, constantes Textos.

- Estructura selectiva:

si (condición₁) entonces

```
    instrucción
  [ si no si (condición2) entonces
    instrucción
  si no
    instrucción ] → Operador
fin si
```

con condición₁ y condición₂ mutuamente excluyentes

- Estructura de casos:

seleccionar (variable)

caso (valor 1)

instrucción

caso (valor 2)

instrucción

caso (valor 3)

instrucción

[en otro caso]
instrucción

fin seleccionar

- Estructuras iterativas:

para i = valor_inicial hasta valor_final hacer

instrucción

fin para

y

hacer

instrucción] ejecuta esto una y otra vez hasta que se encuentre con un salir

fin hacer

Zimatek

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

Vejamos a las tres estructuras consideradas:

```

if (condicio 1) then
    instrucciones
else if (condicio 2) then
    instrucciones
else if (condicio 3) then
    instrucciones
else
    instrucciones
end if
    
```

} opcional

```

select case (selector)
    case (valor 1)
        instrucciones
    case (valor 2)
        instrucciones
    case (valor 3)
        instrucciones
    case default
        instrucciones
end select
    
```

} opcional

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

Variable entre los que se declara el rango

```

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

opcional opcional opcional

```

nombre: do
    instrucciones
end do
    
```

opcional opcional

Se sale con exit; o cycle (esto inicia anticipando una nueva iteración)
 Si usamos exit nombre; o cycle nombre; se refiere a cierto ciclo; y es útil para condiciones

En instrucciones print y read, se puede poner ciclo de implícito: $print \text{ * }, (a(i), b(i), \dots, i = inicio, fin)$
 lee/escribe $a(i), b(i), \dots (fin - inicio)$ i variado desde inicio hasta fin.

- Funciones y procedimientos.

funcion nombre_funcion (valor_estado_1, ..., valor_estado_n)

instrucciones

Operador

devolver salida → obligatorio lo que devuelve a función de procedimiento)

fin funcion nombre_funcion ⇒ OJO!!

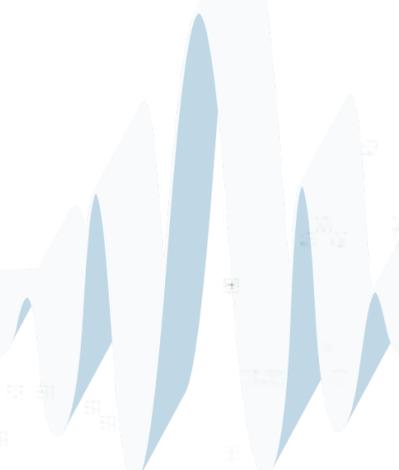
y

procedimiento nombre (valor_estado_1, ..., valor_estado_n)

instrucciones → puede devolver 1 valor, nil o ninguno

fin procedimiento nombre

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



Zimatek