Pseudocódigo

- Acciones
- Estructuras de control
  - Secuencia
  - Selección
    - simple (**si** ... **entonces** ...)
    - doble (**si** ... **entonces** ... **si no** ...)
    - múltiple (**según el caso** ...)
  - Iteración
    - con condición al principio (**mientras** ... **hacer** ...)
    - con condición al final (**repetir** ... **hasta que** ...)
    - controlada por contador (**para** ...)
    - ...

<sentencia> ::= *<sentencia simple>* | *<sentencia compuesta>*

<sentencia simple> ::= *<sentencia de asignación>* |
read ... |
write ... |
<sentencia vacía>

<sentencia compuesta> ::= *<sentencia secuencial>* |
<sentencia alternativa> |
<sentencia iterativa>

Secuencia

<acción>
<acción>
...
<acción>

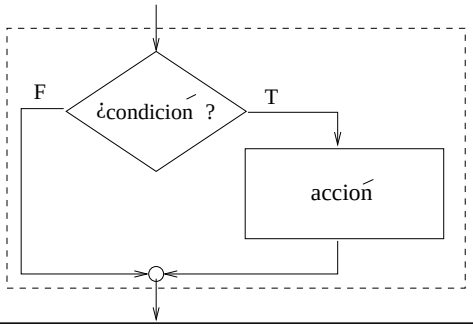
<sentencia secuencial> ::=
begin
<sentencia>
{;*<sentencia>* }
end

- Selección
  - simple (**si** ... **entonces** ...)
  - doble (**si** ... **entonces** ... **si no** ...)
  - múltiple (**según el caso** ...)

<sentencia alternativa> ::= *<alternativa simple>* |
<alternativa doble> |
<alternativa múltiple>

Alternativa simple

si *<condición>* **entonces**
<acciones>
fin si

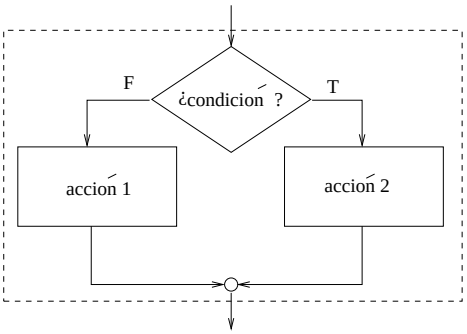


---

if *<expresión lógica>* **then** *<sentencia>*

Alternativa doble

si *<condición>* **entonces**
<accionesT>
si no
<accionesF>
fin si



---

if *<expresión lógica>* **then** *<sentenciaT>* **else** *<sentenciaF>*

```
if a<0 then
   a := a+1
else
   a := a-1
```

```
if a<0 then
   a := a+1 ;
if a>=0 then
   a := a-1
```

```
if a>= 0 then
   a := a-1;
if a<0 then
   a:= a+1
```

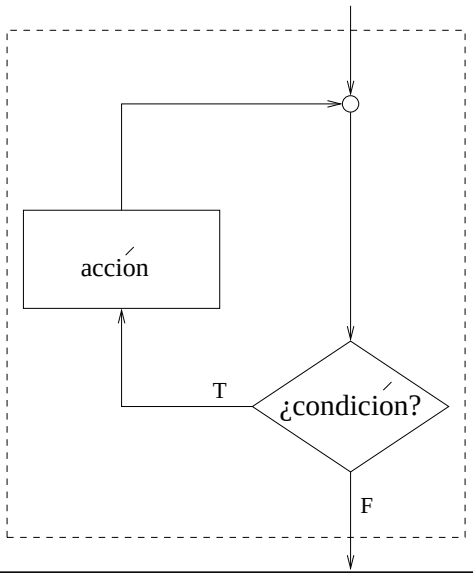

- Iteración
  - con condición al principio (**mientras** ... **hacer** ...)
  - con condición al final (**repetir** ... **hasta que** ...)
  - controlada por contador (**para** ...)
  - ...

<sentencia iterativa> ::= *<sentencia while>* |
<sentencia repeat> |
<sentencia for>

- Condición de permanencia/terminación del bucle
  - Número de veces que se realizan las acciones
  - Número de veces que se evalúa la condición
- Corrección parcial:
  Si el bucle termina, obtiene el resultado deseado
  - Terminación:
  El bucle termina

Iteración "mientras"

mientras <*condición*> **hacer**
<*acciones*>
fin mientras



---

while <*expresión lógica*> **do** <*sentencia*>

Acumulador: (aditivo, multiplicativo ...)

- Iniciación al elemento neutro de la operación. Una vez.
- Actualización: n veces:
  $<var>$:= $<var>$ $<op>$ $<expresión>$

Contador: (sobre tipo ordinal)

- Iniciación al valor inicial. Una vez.
- Actualización: n veces:
  $<var>$:= *succ* ($<var>$) ó $<var>$:= *pred* ($<var>$)

Sucesión explícita o calculada: $a_i = f(i) \quad \forall i \geq 0$

Sucesión implícita o recurrente:

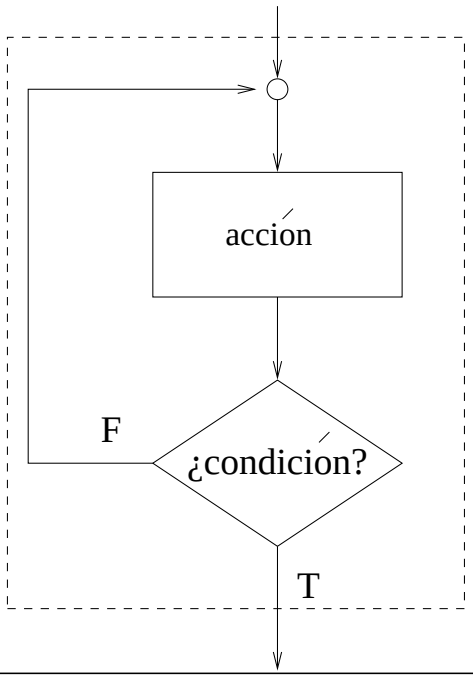
$$\begin{array}{lcl} a_0 & := & \text{valor base} \\ a_i & := & f(i, a_{i-1}) \quad \forall i \geq 1 \end{array}$$

Iteración "repeat"

repetir

$<acciones>$

hasta que $<condición>$



---

repeat $<sentencias>$ **until** $<expresión\ lógica>$

Control de bucles por centinela

obtener dato
mientras dato $\neq$ centinela **hacer**
    tratar dato
    obtener dato
fin mientras

Control de bucles por último dato

repetir
    obtener dato
    tratar dato
hasta que dato (antes del tto.) = último dato

Control de bucles por índice

iniciar índice con la expresión inicial
mientras índice $\leq$ expresión final **hacer**
    realizar acción (cuidado con lo que se modifica)
    actualizar el índice
fin mientras

Caso frecuente: contador

contador $\leftarrow$ expresión inicial
mientras contador $\leq$ expresión final **hacer**
    realizar acción (cuidado con lo que se modifica)
    incrementar el contador (contador $\leftarrow$ contador + expresión)
fin mientras

Control de bucles por bandera

1 o mas veces

repetir
    proceso
    mostrar '¿Quiere repetir?'
    leer respuesta
hasta que respuesta sea negativa

0 o mas veces

mostrar '¿Quiere realizar el proceso?'
leer respuesta
mientras respuesta sea afirmativa **hacer**
    proceso
    mostrar '¿Quiere seguir realizando el proceso?'
    leer respuesta
fin mientras

Control por fin de datos

En el archivo

mientras haya datos (`not eof`) **hacer**
    leer dato
    procesar dato
fin mientras

En la línea

mientras haya datos (`not eoln`) **hacer**
    leer dato
    procesar dato
fin mientras

mientras haya líneas (`not eof`) **hacer**
    (* procesar línea: *)
    proceso previo a la línea
    mientras haya datos (`not eoln`) **hacer**
        leer dato
        procesar dato
    fin mientras
    proceso de final de línea
fin mientras

CONDICIONAL MÚLTIPLE

case <*expresión ordinal*> **of**
<*elemento lista case*>
{; <*elemento de lista case*> }
<*parte else optativa (NO ESTÁNDAR)*>
end

<*elemento de lista case*> ::=
<*lista de rótulos de case*> : <*sentencia*> | <*vacía*>

<*lista de rótulos de case*> ::= <*rótulo case*> {,<*rótulo case*> }

<*rótulo de case*> ::= <*constante*> | <*constante*> .. <*constante*>

<*parte else optativa*> ::= ; **else** <*sentencia*> |


```
Program DiasDeMes (input, output);
type tnatural = 1 .. maxint ;
     tmeses = 1 .. 12 ;
var mes: tmeses; anyo : tnatural ;
BEGIN
   write (output, 'Mes y año:'); readln (input, mes, anyo);
   case mes of
      1,3,5,7..8,10,12 : writeln (output, '31');
      4,6,9,11         : writeln (output, '30');
      2                : if (anyo mod 4 = 0) and (anyo mod 100 <>0)
                            or (anyo mod 400 = 0)
                         then writeln (output, '29')
                         else writeln (output, '28')
   end (* case *)
END.
```

Iteración controlada por índice ordinal

<identVar> **:=** *<expInicial>;*
while *<identVar>* **<=** *<expFinal>* **do**
begin
<sentencia>;
<identVar> **:= succ(***<identVar>)*
end

---

for *<identVar>:=* *<expInicial>* **to** *<expFinal>* **do**
<sentencia>

Iteración controlada por índice ordinal

<identVar> := *<expInicial>* ;
while *<identVar>* <= *<expFinal>* **do**
begin
<sentencia> ;
<identVar> := **pred** (*<identVar>*)
end

---

for *<identVar>* := *<expInicial>* **downto** *<expFinal>* **do**
<sentencia>