Slide 1

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** ...)
    - ...

Slide 2

- 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>

Slide 3

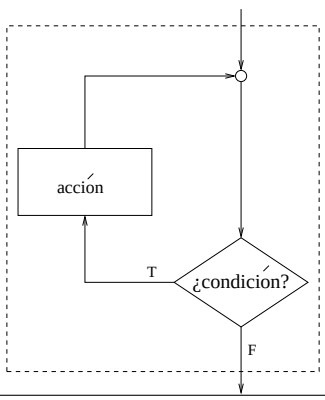
- Condición de permanencia/terminación del bucle
- Número de veces que se realizan las acciones
- Número de veces que se evalúa la condición

- Corrección parcial:
  Si el bucle termina, obtiene el resultado deseado
- Terminación:
  El bucle termina

Slide 4

ITERACIÓN "MIENTRAS"

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



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

Slide 5

```
program MCD (input, output);
(* Entrada: 2 números enteros positivos m y n
   Salida : el máximo común divisor de ambos
   Método : algoritmo de Euclides
   Suposiciones: los números de entrada son positivos y
                 m>=n.
                 Si no, no funciona.
*)

var
        m, n : integer;
        r    : integer;
BEGIN
        writeln (output, 'Escriba dos enteros positivos',
                         'primero >=  segundo');
        readln (input, m, n);
```

Slide 6

```
        r := m MOD n;

        while r <> 0 do
              begin

                   m := n;
                   n := r;

                   r := m MOD n

              end;

        writeln (output, 'El máximo común divisor es ', n)
END.
```

Slide 7

```
        (*V MCD (m0, n0) = MCD (m, n) *)
        r := m MOD n;
        (*V m=nq+r, 0<=r<n, MCD (m0, n0) = MCD (m, n) *)
        while r <> 0 do
              begin
                (*V m=nq+r, 0<r<n, MCD(m0,n0) = MCD(n,r) *)
                   m := n;
                   n := r;
                (*V          0<n    MCD(m0,n0) = MCD(m,n) *)
                   r := m MOD n
                (*V m=nq+r, 0<=r<n, MCD(m0,n0)=MCD(m,n) *)
              end;
        (*V m=nq+r, 0 =r<n, MCD(m0,n0) = MCD(m,n) *)
        writeln (output, 'El máximo común divisor es ', n)
END.
```

Slide 8

Ejemplo: Suma de dos enteros sin usar el signo +

(* $x = x_0$, $y = y_0$ x, y,: enteros, *suma*: entero *)

```
suma := y;
while x <> 0 do
      begin
          suma := succ(suma);
          x := pred(x) ;
      end
```

(* $x = 0$, $y = y_0$ $suma = x_0 + y_0$ *)

(* Termina sii $x_0 \geq 0$ *)

Slide 9

Ejemplo: cálculo de $\sum_{n=0}^{\infty} 1/2^n$, hasta un término "pequeño": no suma términos estrictamente menores que un cierto ϵ

(* *suma*, *term*, *epsilon*: reales, *expon*: entero *)

```
{leer epsilon}
suma := 0; expon := 0; term := 1 (* term := 1/2^0 *);
while term >= epsilon do
      begin
          suma := suma + term ;
          expon := expon + 1 ;
          {term := 1/2^expon}
      end;
writeln (output, 'Suma aproximada: ', suma)
```

Slide 10

Cálculo de $1/2^{expon}$

(* i: entero (contador de multiplicaciones) *)

```
term := 1; i := 1;
while i<= expon do
    begin
       term := term * 1/2;
       i := i+1
    end
```

o aprovechando el cálculo realizado en una pasada del bucle para la siguiente:

```
    term := term /2
```

Slide 11

Ejemplo: cálculo de la unidad de redondeo:

```
program UniRed (output);
(* Calcula la Unidad de Redondeo: mayor número de la
   forma  1/2^n tal que 1+1/2^n, para la máquina, es 1.
   Salida  : exponente n y el correspondiente 1/2^n *)
var     uno, u: real;
        n: integer;
BEGIN
   uno := 1.0; u := 1; n := 0;
   while uno+u > uno do
      begin
         n := n+1; u := u/2;
      end;
   (* uno + u = uno *)
   writeln ('Exponente: ', n, 'Unidad de redondeo: ', u)
END.
```

Slide 12

Ejemplo: Muestra del número asociado a cada letra minúscula:

```
program NumDeLetras (output);
var c : char;
BEGIN
  c := 'a';
  while c <= 'z' do
        begin
            writeln (output, c, ord(c));
            c := succ(c)
        end;
END.
```

Slide 13

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:

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

Slide 14

Ejemplo: Suma de los pares hasta uno dado

```
program SumaDePares (input, output);
(* Entrada : n entero positivo y par
   Salida : suma de todos los pares hasta n
   Sup : la entrada es válida *)
var   n, suma: integer;
      p : integer; (* contador *)
BEGIN
   write (output, 'escriba n. par '); readln (input, n);
   suma := 0; p := 2;
   while p <= n do
         begin
              suma := suma + p; p := p+2
         end;
   writeln (output, 'Suma pares hasta ', n , ':', suma)
END.
```

Slide 15

Ejemplo: cálculo del menor divisor de un entero mayor que 1.

Entrada: un número entero mayor que 1
Salida : su divisor más pequeño mayor que 1
Observaciones: Si es primo, la salida es él mismo
Suposiciones: La entrada es válida (entero mayor que 1)
Variables: n : entero de entrada
candidato : (a divisor) entero

```
Inicio
   leer n
   candidato ← 2
   mientras n mod candidato ≠ 0 hacer
            candidato ← +1
   fin mientras
   mostrar candidato
fin
```

Slide 16

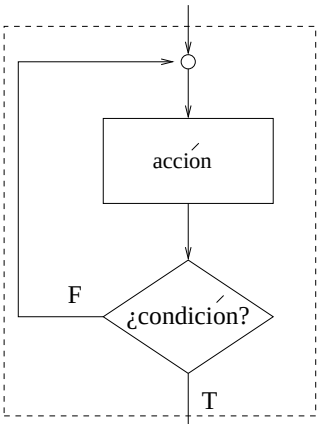
Mejora del algoritmo anterior: sólo se buscan candidatos hasta la raíz cuadrada del número:

```
Inicio
   leer n
   candidato ← 2
   mientras (n mod candidato >0) and(candidato ≤ √n) hacer
            candidato ← +1
   fin mientras
   (* candidato divide a n ó candidato demasiado grande *)
   si n mod candidato = 0 entonces
      mostrar candidato
   si no
      mostrar n
fin
```

Slide 17

ITERACIÓN "REPEAT"

repetir
<*acciones*>
hasta que <*condición*>

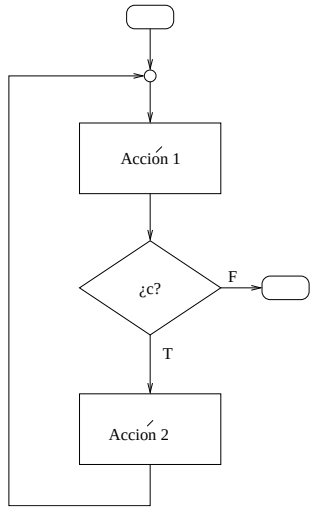


repeat <*sentencias*> **until** <*expresión lógica*>

Slide 18

Algoritmo: factorizar un entero >1
Entrada : entero $n > 1$
Salida : lista de factores primos
Subalgoritmos : PrimerDivisor
Método : Mostrar el primer divisor de n y reducir a n/divisor
Variables: n : entero de entrada
m : entero, primer divisor
Inicio
leer n
repetir
$m \leftarrow$ primer divisor de n
mostrar m
$n \leftarrow n$ div m
hasta que n=1
fin

Slide 19



Slide 20

Ejemplo: números de Fibonacci:

Dado un entero estrictamente positivo n, escribir los números de Fibonacci hasta alcanzar uno que supere estrictamente a n.

variables: n: entero de entrada
f1, f2, f3 : enteros de cálculo

Inicio
leer n
f1 ← 0; f2 ← 1
mostrar f1, f2
repetir
f3 ← f1 + f2
mostrar f3
f1 ← f2; f2 ← f3
hasta que f3>n
fin

Slide 21

Ejemplo: números de Fibonacci:

Versión con **mientras**

```
Inicio
    leer n
    f1 ← 0; f2 ← 1
    mostrar f1, f2
    f3 ← f1 + f2
    mientras f3 ≤ n hacer
        mostrar f3
        f1 ← f2; f2 ← f3
        f3 ← f1 + f2
    fin mientras
    (* f3 >n ∧ f3 es el primero que lo supera *)
    mostrar f3
fin
```

Slide 22

Control de bucles por centinela

```
obtener dato
mientras dato ≠ centinela hacer
    tratar dato
    obtener dato
fin mientras
```

Slide 23

Control de bucles por último dato

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

Slide 24

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

Slide 25

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

Slide 26

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

Slide 27

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

Slide 28

Ejemplo: Contar el número de zetas en un texto de teclado.

```
program cuentaZetas (input, output);
var nzetas : integer;   c : char;
BEGIN
    nzetas := 0;
    while not eof do begin
          while not eoln do begin
                read (c);
                if (c='z') or (c='Z')  then
                      nzetas := nzetas +1
          end;
          readln;
          writeln ('De momento: ', nzetas)
    end;
    writeln ('En total ', nzetas)
END.
```