

Estructuras de Control. Secuencia y Selección

InCo - FING

Section 1

Estructuras de control

Instrucciones simples

- Asignación
- Llamada a procedimiento

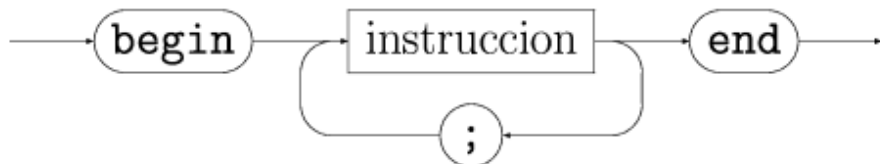
Instrucciones compuestas

- Secuencia.
- Selección (`if`, `case`)
- Repetición (`for`, `while`, `repeat`)

La secuencia

Diagrama

secuencia



La secuencia (cont)

BNF

secuencia = 'begin' instrucción { ';' instrucción } 'end'

Ejemplo:

```
begin
  read(v);
  v:= v * 4;
  writeln(v)
end
```

Section 2

Selección

La instrucción if-then-else

Ejemplo: Determinar si un número es par o impar.

La instrucción if-then-else

Ejemplo: Determinar si un número es par o impar.

```
program ParImpar;  
var  
    numero : integer;  
begin  
    (* Ingresar numero *)  
    write('Ingrese numero: ');  
    readln(numero);
```


La instrucción if-then-else

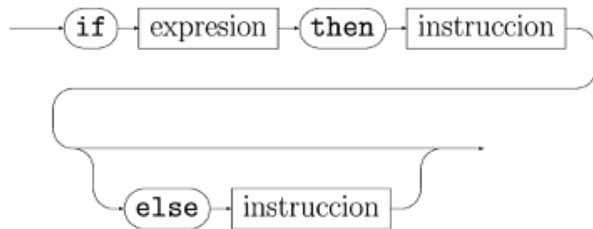
Ejemplo: Determinar si un número es par o impar.

```
program ParImpar;
var
    numero : integer;
begin
    (* Ingresar numero *)
    write('Ingrese numero: ');
    readln(numero);

    (* analizar el numero*)
    if numero mod 2 = 0 then
        writeln('El numero ingresado es par')
    else
        writeln('El numero ingresado es impar')
end.
```

Diagrama

if_then_else



Observaciones:

- *if*, *then*, *else* son *palabras reservadas*.
- *expresión* debe ser una *expresión booleana*.
- Observar que no lleva *;* en ninguna parte.

Semántica de la instrucción if-then-else

La ejecución de:

```
if expresión
  then instrucción1
  else instrucción2
```

involucra los siguientes pasos:

- 1 Se evalúa la expresión booleana; sea b su valor.
- 2 Si b es true se ejecuta *instrucción1*
- 3 Si b es false se ejecuta *instrucción2*.

La ejecución de:

```
if expresión
  then instrucción1
```

involucra los siguientes pasos:

- 1 Se evalúa la expresión booleana; sea b su valor.
- 2 Si b es true se ejecuta *instrucción1*
- 3 Si b es false no se ejecuta nada.

Anidamiento de instrucciones.

Las instrucciones dentro del `if` pueden ser simples o compuestas:

```
if i > 0 then
begin
    i:= i+1;
    writeln('Es positivo')
end
else
begin
    i:= i*4;
    writeln('Es negativo')
end
```

En este ejemplo se *anidan* secuencias dentro de una sentencia `if`.

Indentación de sentencias anidadas

Indentación: Es la cantidad de espacios que se dejan al principio de una línea de programa.

- El compilador no toma en cuenta la indentación.
- Sin embargo es muy importante para la legibilidad y mantenimiento de un programa.
- La indentación debe ayudar a entender la estructuración lógica del programa.

Indentación de if-then-else

Existen muchas maneras de indentar una instrucción if:

```
if expression then           (* estilo Wirth *)
begin
  ...
end else
begin
  ...
end
```

Indentación de if-then-else (cont)

```
if expression                               (* estilo Konvalina *)
then begin
    ...
end
else begin
    ...
end
```

```
if expression then begin (* estilo Kernighan-Ritchie *)
    ...
end else begin
    ...
end (*if*)
```


Sentencias if anidadas

Las instrucciones asociadas con el then y/o else pueden ser a su vez instrucciones if

```
if exp1 then (* 1 *)
  if exp2 then (* 2 *)
    inst1
  else (* cierra 2 *)
    inst2
else (* cierra 1 *)
  if exp3 then (* 3 *)
    inst3
  else (* cierra 3 *)
    inst4
```

Regla: Cada else se asocia con el último if que no esté cerrado.

Anidamiento if-then-else con if-then

```
if cond1 then
  if cond2 then instrucción-1
else
  instrucción-2
```

La instrucción anterior se interpreta así:

```
if cond1 then
begin
  if cond2 then
    instrucción-1
  else
    instrucción-2
end
```

Anidamiento if-then-else con if-then (cont)

Para que interprete que el else cierra el primer if

```
if cond1 then
begin
  if cond2 then
    instrucción-1
end
else
  instrucción-2
```

O mejor aún:

```
if not cond1 then
  instrucción-2
else if cond2 then
  instrucción-1
```

Sentencias if anidadas (ejemplo)

Determinar si un año es bisiesto.

Un año es bisiesto si cumple alguna de las siguientes condiciones:

- es múltiplo de 4 pero no es múltiplo de 100.
- es múltiplo de 400.

No son bisiestos: 1900, 1999, 2003

Sí son bisiestos: 1976, 2000, 2004

Ejemplo: Año bisiesto.

```
program bisiesto;
var anio : integer;
begin
  write('Ingrese un año: ');
  readln(anio);

  if anio < 0 then
    writeln('El año debe ser un numero positivo')
  else if anio mod 400 = 0 then
    writeln('Es bisiesto')
  else if anio mod 100 = 0 then
    writeln('No es bisiesto')
  else if anio mod 4 = 0 then
    writeln('Es bisiesto')
  else
    writeln('No es bisiesto')
end.
```

Operadores relacionales

Los operadores relacionales sirven para comparar valores de tipos simples:

operador	Explicación
=	igual
<>	distinto
<	menor
>	mayor
<=	menor o igual
>=	mayor o igual

Operadores lógicos

Los operadores lógicos sirven para escribir expresiones complejas:

A	B	A and B	A or B	not A
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Pascal **estándar**:

- Para evaluar (E1 and E2)
 - Se evalúa E1. Sea b1 su valor.
 - Se evalúa E2. Sea b2 su valor.
 - Se evalúa (b1 and b2).

Análogamente para el or

Booleanos con circuito corto

Free Pascal evalúa usando *circuito corto*:

- Para evaluar (E1 and E2)
 - Se evalúa E1. Sea b1 su valor.
 - Si b1 es false **no se evalúa E2** y el resultado es false.
 - Si b1 es true se evalúa E2. Sea b2 su valor.
El resultado es b2.

La mayoría de los lenguajes de programación evalúan por circuito corto.

Pregunta: ¿cómo funciona la evaluación por circuito corto para el or?

Precedencia y asociatividad de los operadores

Precedencia:

- 1 not, +, - (unarios)
- 2 *, /, div, mod, and (multiplicativos)
- 3 +, -, or (aditivos)
- 4 =, <>, <, <=, >, >=, IN (relacionales)

Asociatividad:

- Todos los operadores asocian a la izquierda.

Ejemplos.

① not p or q and r \iff

Ejemplos.

$$\textcircled{1} \text{ not } p \text{ or } q \text{ and } r \iff (\text{not } p) \text{ or } (q \text{ and } r)$$

$$\textcircled{2} \text{ not } p \text{ or } q \iff$$

Ejemplos.

$$\textcircled{1} \text{ not } p \text{ or } q \text{ and } r \iff (\text{not } p) \text{ or } (q \text{ and } r)$$

$$\textcircled{2} \text{ not } p \text{ or } q \iff (\text{not } p) \text{ or } q$$

$$\textcircled{3} p \text{ or } q \text{ and } r \iff$$

Ejemplos.

$$\textcircled{1} \text{ not } p \text{ or } q \text{ and } r \iff (\text{not } p) \text{ or } (q \text{ and } r)$$

$$\textcircled{2} \text{ not } p \text{ or } q \iff (\text{not } p) \text{ or } q$$

$$\textcircled{3} p \text{ or } q \text{ and } r \iff p \text{ or } (q \text{ and } r)$$

$$\textcircled{4} \text{ not } p \text{ and } q \text{ or } \text{not } r \iff$$

Ejemplos.

$$\textcircled{1} \text{ not } p \text{ or } q \text{ and } r \iff (\text{not } p) \text{ or } (q \text{ and } r)$$

$$\textcircled{2} \text{ not } p \text{ or } q \iff (\text{not } p) \text{ or } q$$

$$\textcircled{3} p \text{ or } q \text{ and } r \iff p \text{ or } (q \text{ and } r)$$

$$\textcircled{4} \text{ not } p \text{ and } q \text{ or } \text{not } r \iff ((\text{not } p) \text{ and } q) \text{ or } (\text{not } r)$$

$$\textcircled{5} \text{ not } 4 > 5 \iff$$

Ejemplos.

$$\textcircled{1} \text{ not } p \text{ or } q \text{ and } r \iff (\text{not } p) \text{ or } (q \text{ and } r)$$

$$\textcircled{2} \text{ not } p \text{ or } q \iff (\text{not } p) \text{ or } q$$

$$\textcircled{3} p \text{ or } q \text{ and } r \iff p \text{ or } (q \text{ and } r)$$

$$\textcircled{4} \text{ not } p \text{ and } q \text{ or } \text{not } r \iff ((\text{not } p) \text{ and } q) \text{ or } (\text{not } r)$$

$$\textcircled{5} \text{ not } 4 > 5 \iff \text{error}$$

$$\textcircled{6} i < \text{num} \text{ and } \text{num} < 10 \iff$$

Ejemplos.

$$\textcircled{1} \text{ not } p \text{ or } q \text{ and } r \iff (\text{not } p) \text{ or } (q \text{ and } r)$$

$$\textcircled{2} \text{ not } p \text{ or } q \iff (\text{not } p) \text{ or } q$$

$$\textcircled{3} p \text{ or } q \text{ and } r \iff p \text{ or } (q \text{ and } r)$$

$$\textcircled{4} \text{ not } p \text{ and } q \text{ or } \text{not } r \iff ((\text{not } p) \text{ and } q) \text{ or } (\text{not } r)$$

$$\textcircled{5} \text{ not } 4 > 5 \iff \text{error}$$

$$\textcircled{6} i < \text{num} \text{ and } \text{num} < 10 \iff \text{error}$$

$$\textcircled{7} 2 < 3 < 4 \iff$$

Ejemplos.

$$\textcircled{1} \text{ not } p \text{ or } q \text{ and } r \iff (\text{not } p) \text{ or } (q \text{ and } r)$$

$$\textcircled{2} \text{ not } p \text{ or } q \iff (\text{not } p) \text{ or } q$$

$$\textcircled{3} p \text{ or } q \text{ and } r \iff p \text{ or } (q \text{ and } r)$$

$$\textcircled{4} \text{ not } p \text{ and } q \text{ or } \text{not } r \iff ((\text{not } p) \text{ and } q) \text{ or } (\text{not } r)$$

$$\textcircled{5} \text{ not } 4 > 5 \iff \text{error}$$

$$\textcircled{6} i < \text{num} \text{ and } \text{num} < 10 \iff \text{error}$$

$$\textcircled{7} 2 < 3 < 4 \iff \text{error}$$

$$\textcircled{8} 2 > 3 < \text{true} \iff$$

Ejemplos.

$$\textcircled{1} \text{ not } p \text{ or } q \text{ and } r \iff (\text{not } p) \text{ or } (q \text{ and } r)$$

$$\textcircled{2} \text{ not } p \text{ or } q \iff (\text{not } p) \text{ or } q$$

$$\textcircled{3} p \text{ or } q \text{ and } r \iff p \text{ or } (q \text{ and } r)$$

$$\textcircled{4} \text{ not } p \text{ and } q \text{ or } \text{not } r \iff ((\text{not } p) \text{ and } q) \text{ or } (\text{not } r)$$

$$\textcircled{5} \text{ not } 4 > 5 \iff \text{error}$$

$$\textcircled{6} i < \text{num} \text{ and } \text{num} < 10 \iff \text{error}$$

$$\textcircled{7} 2 < 3 < 4 \iff \text{error}$$

$$\textcircled{8} 2 > 3 < \text{true} \iff (2 > 3) < \text{true}$$

Ejemplo (1)

```
program bisiesto;
var anio : integer;
begin
    write('Ingrese un año: ');
    readln(anio);

    if anio < 0 then
        writeln('El año debe ser un número positivo')
    else if (anio mod 400 = 0)
        or
        ((anio mod 100 <> 0) and (anio mod 4 = 0))
    then
        writeln('Es bisiesto')
    else
        writeln('No es bisiesto')
    end.
end.
```

Ejemplo (2)

Obtener una calificación a partir de un puntaje según la siguiente regla:

- 90-100: A, 80-89: B, 70-79: C, 60-69: D, 0-59: F

```
readln(puntaje);  
if (puntaje < 0) or (puntaje > 100) then  
    writeln('Puntaje Invalido: ',puntaje)  
else begin  
    if puntaje >= 90 then nota:= 'A'  
    else if puntaje >= 80 then nota:= 'B'  
    else if puntaje >= 70 then nota:= 'C'  
    else if puntaje >= 60 then nota:= 'D'  
    else (* puntaje < 59 *) nota:= 'F';  
    writeln('La calificación es: ', nota)  
end;
```

Section 3

Selección generalizada

La instrucción case

Permite seleccionar **una** instrucción de un conjunto de instrucciones, según el valor de una expresión:

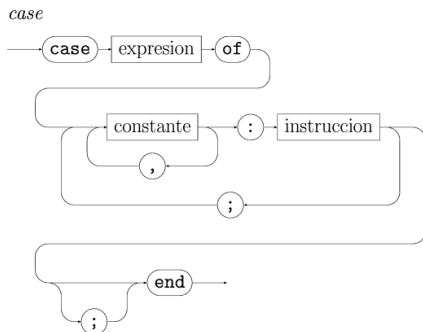
```
readln(numero);  
case numero of  
  0 : writeln('cer0');  
  1 : writeln('uno');  
  2 : writeln('dos');  
  3 : writeln('tres');  
  4 : writeln('cuatro');  
  5 : writeln('cinco');  
  6 : writeln('seis');  
  7 : writeln('siete');  
  8 : writeln('ocho');  
  9 : writeln('nueve');  
end; { case }
```

case (cont)

Cada instrucción puede estar rotulada con más de un valor.

```
var mes : integer;  
case mes of  
  1,3,5,  
  7,8,10,12 : writeln('Mes de 31 dias');  
  4,6,9,11  : writeln('Mes de 30 dias');  
  2         : writeln('Mes de 28 dias');  
end;
```


Sintaxis de case



- La expresión debe ser de tipo **ordinal**: integer, char, boolean, enumerados, subrangos
- Las constantes deben ser del mismo tipo que la expresión y no pueden aparecer repetidas.
- case y of son palabras reservadas.

Semántica de case

Para ejecutar la instrucción:

```
case e of
  constantes-1: instrucción-1;
  ...
  constantes-i: instrucción-i;

  constantes-n: instrucción-n;
end
```

se procede así:

- 1 Se evaluará e . Sea v su valor.
- 2 Se ejecuta *instrucción- k* tal que v pertenece a *constantes- k*
- 3 Si v no aparece en ninguna lista de constantes, **se produce un error**.

Instrucción case modificada (Free Pascal)

En Free Pascal la instrucción case tiene algunas variantes:

```
case exp of
    constantes-1: instrucción-1;
    ...
    constantes-n: instrucción-n;
else
    instrucción-alternativa
end
```

Si el valor de *exp* no coincide con ninguna de las constantes:

- Si **else** se especifica: se ejecuta *instrucción-alternativa*.
- Si no hay **else**, no se ejecuta nada (no hay error)

Observación: En este curso usaremos la instrucción case de Free Pascal.

Relación entre if y case

La instrucción:

```
if exp then inst1 else inst2
```

es equivalente a:

```
case exp of  
  true  : inst1;  
  false : inst2;  
end
```

Relación entre case e if

La instrucción:

```
case exp of
  a1,a2... : inst-a;
  b1,b2,...: inst-b
  ...
  z1,z2,...: inst-z;
end;
```

es equivalente a:

```
v:= exp;
if (v=a1) or (v=a2) or ... then
  inst-a
else if (v=b1) or (v=b2) or ... then
  inst-b
else if ...
  ...
else if (v=z1) or (v=z2) or ... then
  inst-z;
```

Cuando usar case

- La selección es entre más de 2 instrucciones.
- La selección se basa en el valor que adopta una cierta expresión simple (no puede ser real).
- Es frecuente usar case en situaciones donde hay que elegir una acción según una opción ingresada por el usuario (menú)

Ejemplo de menú

```
(* mostrar menu *)
writeln('A - Agregar');
writeln('B - Borrar');
writeln('M - Modificar');

(* pedir opcion *)
write('Ingrese opción(A,B,M): ');
readln(opcion);

(* procesamiento *)
case opcion of
  'A' : (* código para agregar *)
    ...
  'B' : (* código para borrar *)
    ...
  'M' : (* código para modificar *)
    ...
else writeln('Codigo incorrecto: ', opcion);
```