

Curso de Programación 1

Plan 97

Clase 4

(Selección)

Introducción

Muchos problemas requieren que diferentes acciones sean realizadas dependiendo del estado de alguna condición.

Ejemplo:

Caminar en dirección noroeste 5 kms
y entonces

Si el río esta bajo cruzarlo

En cambio, si el río está crecido caminar 500 m
hasta el puente y entonces cruzarlo.

El plan anterior ilustra la estructura característica de elección en un programa.

Lista todas las circunstancias previstas y describe una acción a seguir para cada caso.

En este plan hay dos circunstancias previstas: el río puede estar **bajo** o **crecido**.

Sentencia **if-then-else**

La mayoría de los lenguajes de programación proveen una sentencia de elección binaria que permite a un programa elegir entre dos acciones posibles, dependiendo de una condición booleana.

```
if <condición booleana>  
  then <sentencia>  
  else <sentencia>
```

Sentencia `if-then-else`

Ejemplo:

Sea `num` : Integer.

```
if num mod 2 = 0
  then writeln('El número es par')
  else writeln('El número es impar');
```

En caso de ser necesario escribir mas de una sentencia en el `then` o en el `else`, las mismas deben ser escritas como un bloque, esto es, entre un `begin` y un `end`.

```
if <condición booleana>
  then begin
    <sentencia>;
    ...
    <sentencia>
  end
  else begin
    <sentencia>;
    ...
    <sentencia>
  end;
```

Ejemplo

El siguiente procedimiento imprime la descomposición de un entero en decenas y unidades. Si el número es menor que 10 imprime sólo las unidades.

```
Procedure impresion(n: integer);
begin
  if n > 10
    then begin
      writeln('Decenas: ',n div 10:4);
      writeln('Unidades: ',n mod 10:1)
    end
    else writeln('Unidades: ',n:1)
  end;
```

Programación 1. Plan 97. InCo. Fac. de Ingeniería

7

Sentencia if-then

En algunos casos deseamos ejecutar ciertas acciones sólo si una condición se satisface.

```
if <condición booleana>
  then <sentencia>
```

Ejemplo:

```
if saldo < 0
then writeln('Su cuenta está sobregirada');
```

Programación 1. Plan 97. InCo. Fac. de Ingeniería

8

Sentencias `if` anidadas

Sentencias `if` pueden ocurrir dentro de otras sentencias `if`.

Ejemplo:

```
if x >= y
  then if x = y
        then writeln('Son iguales')
        else writeln('El mayor es',x)
  else writeln('El mayor es',y);
```

Ejemplo: Modifiquemos el procedimiento `impresion` visto anteriormente para que controle que el número es positivo.

```
Procedure impresion(n: integer);
begin
  if n < 0 then writeln('Es negativo')
  else
    if n > 10
    then
      begin
        writeln('Decenas: ',n div 10:4);
        writeln('Unidades: ',n mod 10:1)
      end
    else writeln('Unidades: ',n:1)
  end;
end;
```

Sentencias `if` anidadas (cont.)

Una sentencia `if-then` puede ocurrir dentro de una sentencia `if-then-else` si la encerramos entre un `begin` y un `end`.

Ejemplo:

```
if a >= b
  then begin
    if a > c
      then max := a
    end
  else max := 0;
```

Elección completa e incompleta

El programa para un robot ensamblador de vehículos seguramente incluye una elección entre acciones: si hay más de un tipo de vehículo obviamente habrá que elegir entre varias acciones a efectuar.

Supongamos que la elección está determinada por una etiqueta magnética que puede ser leída por el robot.

Si la etiqueta es de tipo 1:
ejecute la acción de ensamblado 1

Si la etiqueta es de tipo 2:
ejecute la acción de ensamblado 2

La descripción anterior no es satisfactoria, ya que no especifica qué hacer en toda posible circunstancia.

Por ejemplo, que pasa si:

- la etiqueta leída es de tipo 3 ?
- la etiqueta está tan dañada que no puede ser leída ?

En cualquiera de estas circunstancias el programa no es aplicable.

Un robot programado de esta forma es incapaz de hacer una elección adecuada y tendrá que parar su trabajo.

Una solución a este problema sería, por ejemplo, modificar el código de esta forma:

si la etiqueta es de tipo 1:

ejecute la acción de ensamblado 1

caso contrario

si la etiqueta es de tipo 2

ejecute la acción de ensamblado 2

caso contrario

ejecute una acción de excepción

En resumen: Debemos programar nuestros programas de forma tal que puedan hacer frente a cualquier posible situación. Esto es, que puedan reaccionar ante cualquier posible valor de los datos de entrada.

Ejemplo: El siguiente procedimiento determina si un número es divisible entre otro.

```
Procedure divisible(n,m: integer;  
                  var esdiv: boolean);  
begin  
  if n mod m = 0  
    then esdiv := true  
    else esdiv := false  
end;
```

El problema con este procedimiento es que no es completo, ya que no tiene en cuenta una alternativa: el número m puede ser cero!

Tenemos esencialmente dos maneras de actuar frente al caso de división por cero.

Una de ellas consiste en mantener el procedimiento en la forma en que está escrito, pero documentar que tiene la siguiente pre-condición: $m \neq 0$.

El control de dicha pre-condición debe ser realizado previo a la invocación del procedimiento y es responsabilidad del usuario. Por ejemplo,

```
if y <> 0  
  then divisible(x,y,esdiv)  
  else writeln('El entero', y, 'es cero');
```

Otra alternativa es controlar dentro del mismo procedimiento que el segundo argumento es distinto de cero. (Informamos el no cumplimiento de dicha condición mediante un valor booleano `err`.)

```
Procedure divisible(n,m: integer;
                  var esdiv,err: boolean);
begin
  if m = 0
  then err := true
  else begin
    esdiv := n mod m = 0;
    err := false
  end
end;
```

Es ahora el procedimiento quien nos informa si el segundo argumento es o no distinto de cero. Por ejemplo,

```
divisible(x,y,esdiv,error);
if not error
  then if esdiv
        then <sentencias>
        else <sentencias>
  else <caso anormal>;
```

La elección de cuál de las dos alternativas adoptar está en manos del programador y es lo que comunmente se conoce por decisión de diseño.

Expresiones booleanas

Fórmulas aritméticas denotan valores numéricos.
Expresiones booleanas denotan uno de los dos valores booleanos posibles, **true** o **false**.

Comparaciones, como **a = b**, **a < b**, **a >= b**, son expresiones booleanas simples que se construyen a partir de los operadores relacionales.

Expresiones más complejas pueden obtenerse combinando expresiones booleanas usando los conectivos **and**, **or** y **not** en una forma similar a como + y * son usados para combinar expresiones aritméticas.

Operadores relacionales

Los operadores relacionales son los siguientes:

=	igualdad
<>	desigualdad
<=	menor o igual
>=	mayor o igual
<	menor
>	mayor

Para los tipos **integer** y **real** estos operadores realizan las comparaciones numéricas habituales.

Operadores relacionales

Los valores de tipo `char` son comparados usando sus códigos. Esto significa que el orden entre los caracteres (conocido como **orden lexicográfico**) depende de la implementación.

De todas formas, la codificación siempre respeta que:

`'0' < '1' < ... < '9'`

`'a' < 'b' < ... < 'z'`

`'A' < 'B' < ... < 'Z'`

Los valores booleanos están ordenados de esta forma:

`false < true`

Operadores Lógicos

Si `P` y `Q` son expresiones booleanas, entonces

`P and Q`

`P or Q`

`not P`

lo son también.

`P and Q` se hace verdadero cuando `P` y `Q` son ambas verdaderas y se hace falso cuando al menos una de las dos expresiones es falsa.

`P or Q` se hace verdadero cuando al menos una de las dos expresiones es verdadera y se hace falso cuando `P` y `Q` son ambas falsas.

Precedencia de los Operadores

1. `not`
2. `*` `/` `div` `mod` `and`
3. `+` `-` `or`
4. `=` `<>` `<` `<=` `>` `>=`

Ejemplos

<code>1 < x and x < 10</code>	es incorrecta
<code>(1 < x) and (x < 10)</code>	es correcta
<code>not 3 < 4</code>	es incorrecta
<code>not P or Q</code>	<code>(not P) or Q</code>
<code>not P and Q or R</code>	<code>((not P) and Q) or R</code>

Sentencia case

La sentencia case permite seleccionar entre varias alternativas

Su formato es el siguiente:

```
case <selector> of
  etiqueta1 : <sentencia> ;
  ...
  etiquetan : <sentencia>
end
```

Sentencia case

- Las *etiquetas* deben ser constantes, distintas entre sí y todas de un mismo tipo de dato escalar (**integer**, **char**, **boolean**).
- El *selector* es una expresión de ese mismo tipo. El valor del selector determina la alternativa a escoger, y por lo tanto, la sentencia a ejecutar.
- Luego de ejecutar la sentencia en la alternativa seleccionada, el control continúa con la sentencia que sigue al case.

Relación entre `if` y `case`

Toda sentencia if-then-else

```
if b then s1 else s2
```

puede ser escrita como un case:

```
case b of
  true  : s1;
  false : s2
end
```

Ejemplo

```
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;
```

Varias etiquetas pueden ser agrupadas en cada opción.

Ejemplo

```
var num : integer;  
  
if (num >= 1) and (num <= 10)  
  then case num of  
    2,4,6,8,10: writeln('Numero par');  
    1,3,5,7,9 : writeln('Numero impar')  
  end;  
end;
```

Ejemplo

```
var vocal : char;  
  
case vocal of  
  'A' : writeln('Primera vocal');  
  'E' : writeln('Segunda vocal');  
  'I' : writeln('Tercera vocal');  
  'O' : writeln('Cuarta vocal');  
  'U' : writeln('Quinta vocal')  
end;
```