Programación en Pascal. Selección

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

gsyc-profes (arroba) gsyc.urjc.es

Octubre de 2018

Problemas de selección

Hasta ahora hemos visto problemas de solución directa:

- En el tema 2, la solución al problema era una expresión, numérica o booleana, construida como una secuencia operandos y operadores, donde los operandos eran o constantes o funciones predefinidas
- En el tema 3, aprendimos a usar funciones, pero la solución seguía siendo una única expresión, con la novedad de que los operadores podían ser no solo constantes y funciones predefinidas, sino funciones definidas por nosotros
- En este tema, veremos las *sentencias de control*, que permiten como seleccionar una expresión u otra, o ejecutar una acción u otra, a partir de cierta condición booleana.
  En Pascal se hace mediante las setencias *if-then-else* y *case*

if then else

if *condición* then

    sentencia/s a ejecutar si la condición es cierta

else

    sentencia/s a ejecutar si la condición es falsa

- Es una única sentencia, nunca se pone ';' después de la condición

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}

program if_then_else;

const
    Limite_fiebre: real = 37.5;
    Temperatura: real = 37.8;

begin
    if Temperatura >= Limite_fiebre then
        writeln('Fiebre')
    else
        writeln('Temperatura normal')
end.
```

- En este curso, nuestro convenio será escribir la palabra reservada *if*, la condición y la palabra reservada *then* en una línea

Recuerda que el carácter ';' separa sentencias. If-Then-Else es una única sentencia, no puede haber un ';' en medio[1]

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program if_then_else_MAL;

const
    Limite_fiebre: real = 37.5;

var
    temperatura: real = 37.8;

begin
    if temperatura >= Limite_fiebre
    then
        writeln('Fiebre') ; // ¡¡MAL!! Sobra el ;
    else
        writeln('Temperatura normal')
end.
```

[1] Excepto dentro de un bloque begin-end, naturalmente

Es necesario que la tabulación sea consistente, pero también hay otros criterios posibles y habituales, p.e escribir *if*, *then* y *else* en la misma columna

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program if_then_else_v02;

// Otro convenio posible, aunque no lo seguiremos aquí

const
    Limite_fiebre: real = 37.5;
    Temperatura: real = 37.8;

begin
    if Temperatura >= Limite_fiebre
    then
        writeln('Fiebre')
    else
        writeln('Temperatura normal')
end.
```

Otra posibilidad (discrepante con el criterio de este curso) escribir bloques begin-end siempre, aunque no haga falta

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program if_then_else_v03;

// Otra posibilidad más: escribir siempre begin-end

const
    Limite_fiebre: real = 37.5;
    Temperatura: real = 37.8;

begin
    if Temperatura >= Limite_fiebre then begin
        writeln('Fiebre')
    end
    else begin
        writeln('Temperatura normal')
    end
end.
```

Si la rama del *then* o la rama del *else* tienen más de una sentencia, entonces sí será necesario el uso de *begin end*

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program if_then_else_v04;

const
    Limite_fiebre: real = 37.5;
    Temperatura: real = 37.8;

begin
    if Temperatura >= Limite_fiebre then begin
        writeln('Fiebre');
        write('La temperatura es ');
        write(Temperatura-Limite_fiebre:0:1);
        writeln(' grados superior a lo normal')
    end
    else
        writeln('Temperatura normal')
end.
```

Ejecución:

```
Fiebre
La temperatura es 0.3 grados superior a lo normal
```

- Nuestro convenio es que si escribimos begin, irá en la misma línea de *if condición then*
- Observa que las sentencias de la rama *then* aparecen con una tabulación adicional, igual que las de la rama *else*
- Observa que el ';' se coloca para separar sentencias. La última sentencia no lo necesita, aunque si se añade no generar errores

Problema: omisión del begin-end

Es muy importante usar begin-end cuando una de las ramas tiene más de una sentencia

- Si me olvido del begin-end en la rama *then*, no es un error demasiado serio porque resulta un error de sintaxis, el *else* queda descolocado y el compilador me avisará con un error `Fatal: Syntax error, ";" expected but "ELSE" found`

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program error_olvido_begin_end;
function comprueba_positivo(x:integer):boolean;
begin
    if x >= 0 then   //  ¡¡Mal!!
        writeln('Entra en rama then');
        result := True  ; // Olvidé el begin-end
                          // pero el compilador me avisa
    else
        result := False
end;

begin
    writeln(comprueba_positivo(3))
end.
```

Recuerda que las funciones no deben tener efectos laterales, excepto tal vez trazas

Pero si olvido el begin-end en la rama else, el error es más severo, porque la sintaxis es correcta

- Resultará un error lógico
- El compilador considerará que la primera sentencia pertenece a la rama else, pero que ahí acaba la rama
- El resto de sentencias aparentan ser correctas, se ejecutarán siempre, sin importar la condición del *if*
- La tabulación, que resultará incorrecta, hace que el error pueda ser más difícil de localizar

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program error_olvido_begin_end;

function comprueba_positivo(x:integer):boolean;
begin
    if x >= 0 then
        result := True
    else
        writeln('Entra en rama else, número negativo');
        result := False  // ¡¡Mal!! Se ejecuta siempre, está
                         // mal tabulado, olvidé el begin-end
end;

begin
    writeln(comprueba_positivo(3))   // Escribe FALSE
end.
```

Omisión del else

El else no es obligatorio, si nuestro algoritmo no ejecuta nada en caso de incumplimiento de la condición, lo omitimos

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program if_then;

const
    Limite_fiebre: real = 37.5;
    Temperatura: real = 37.8;

begin
    if Temperatura >= Limite_fiebre then
        writeln('Fiebre')
end.
```

If anidado

En la rama *then* puede haber otra sentencia *if-then-else*

Este ejemplo es correcto, aunque peligroso

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program anidacion_correcta;

const
    Mayoria_edad: integer = 18;
    Adolescencia: integer = 13;
    Edad: integer = 14;

begin
    if Edad < Mayoria_edad then
        if Edad < adolescencia then
            writeln(Edad, ': Niño')
        else
            writeln(Edad, ': Adolescente')

end.
```

El *else* se corresponde con el *then* más próximo
Ejecución:

```
14: Adolescente
```

El problema es que no queda del todo claro, si no conocemos bien el criterio de Pascal, o si nos descuidadmos, podemos pensar, erróneamente que el *else* se corresponde al primer *if*

El error anterior es aún más severo (más probable) si el programa está mal tabulado:

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}

program MALA_TABULACION;

const
    Mayoria_edad: integer = 18;
    Adolescencia: integer = 13;
    Edad: integer = 14;

begin
    if Edad < Mayoria_edad then
        if Edad < adolescencia then
            writeln(Edad, ': Niño')
    else    //  ¡¡Mal tabulado!!
        writeln(Edad, ': Adolescente')

end.
```

- En el ejemplo erróneo anterior, la tabulación nos haría pensar que se escribirá *adolescente* cuando $edad >= Mayoria\_edad$
- Como hemos dicho, lo que realmente hace el programa es escribir *adolescente* cuando $edad >= adolescencia$ (y también $edad > Mayoria\_edad$), puesto que estamos en la rama *then* del primer if-then-else

- Para evitar este problema, estableceremos como convenio que cuando en la rama *then* haya otra sentencia *if-then-else*, la *protegeremos* con un bloque *begin-end*

```
if condicion then begin
    if condicion then
        sentencia1
    else
        sentencia2
end
else
    sentencia
```

Observa que escribimos el *if*, el *end* y el *else* en la misma columna

Cuando el segundo *if-then-else* esté en la rama *else*

- Tendremos una estructura muy habitual, llamada *if encadenados*
- Entonces no añadiremos el bloque *begin-end* (a menos que sea necesario porque haya más de una sentencia)
- Ya no se da el problema potencial que hemos descrito del *else* ambiguo

if encadenados

Los *if* encadenados, esto es, una sucesión de *else if*, *else if*, *else if*, son una estructura muy habitual

- *Si se da cierta condición haz esto, y si no, haz lo otro, y si tampoco, esto otro, y si tampoco ...*

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program if_encadenado;

const
    Mayoria_edad: integer = 18;
    Adolescencia: integer = 13;
    Edad: integer = 10;

begin
    if Edad >= Mayoria_edad then
        writeln(Edad, ': Adulto')
    else if Edad > Adolescencia then
        writeln(Edad, ': Adolescente')
    else
        writeln(Edad, ': Niño')
end.
```

Resultado:

```
10: Niño
```

if anidado

En vez de encadenar

```
if then
else if
else if
```

tambien podríamos anidar, esto es, añadir un *begin-end* después del else y meter dentro el if-then-else

```
if then
else begin
   if ... then .. else
end
```

Pero siempre es preferible encadenar, la estructura *else if* es muy habitual, muy clara y añadiendo otro *begin end*, solo lo complicamos

Números mágicos

En un programa no debe haber constantes numérica literales (números *tal cual*).
A esto se le llama *número mágico*, es una mala práctica. Lo correcto es usar constantes. Pero hay excepciones. Por ejemplo

```
if n >=0  then
   result := 'positivo'
else
   result := 'negativo'
```

Escribir 0 como constante literal parece muy razonable en este caso

Varios if-then-else: resumen

Como resumen de todo lo anterior, recuerda:
Si una sentencia *if-then-else* tiene otro *if-then-else*...

- En la rama *then*, es potencialmente peligroso, así que siempre la anidaremos en un bloque *begin-end*
- En la rama *else*, es un if encadenado *else if, else if, else if*. No es peligroso, es muy común. No añadimos bloque *begin-end* (a menos, naturalmente, que sea imprescindible porque hay otras sentencias)

Anidamiento múltiple

¿Podemos escribir un if-then-else dentro de otro if-then-else que está dentro de un if-then-else que está un if-then-else que...?

- El lenguaje lo permite
- Pero si anidamos más de 2 o 3 niveles, muy probablemente estaremos escribiendo un programa de muy mala calidad, difícil de entender y propenso a errores
  - Esto es uno los defectos peores y más típicos de quienes no saben programar
- El anidamiento excesivo casi siempre indica que el programador no ha sabido descomponer su problema en subproblemas (escribiendo nuevas funciones)

Recuerda que el encadenamiento múltiple (else if, else if, else if) es un caso diferente, que, bien hecho, no tiene por qué dar problemas

Case

Otra sentencia de control disponible en Pascal es `case`. Permite ejecutar diferentes acciones a partir de un valor discreto

- Es similar a `if`, la diferencia es que `if` considera un booleano y `case` puede considerar, además, tipos más complejos como integer o char
  - Pero nunca real: ha de ser un valor discreto
- Se parece mucho a los if-then-else encadenados. De hecho, cualquier cosa que se haga con case, también se puede hacer con else-if, else-if
  - Hay lenguajes que no tienen nada parecido a `case`, sus diseñadores consideraron que `else-if` es suficiente
  - La ventaja es que `case` puede resultar más claro que `else-if`

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program ejemplo_case;

const
    Edad: integer = 14;
begin
    case Edad of
    0 :
        writeln('Bebé');
    1..12 :
        writeln('Niño');
    13..17:
        writeln('Adolescente');
    otherwise      // sin dos puntos
        writeln('Adulto');
    end;    // 'end' de case, único sin 'begin'
end.
```

- El compilador evalúa la expresión que escribamos a continuación de la palabra reservada *case*. En este ejemplo, la constante *edad*
- Separamos todos los posibles valores, pudiendo usar rangos. Indicamos las acciones a realizar para cada conjunto de casos
- Si el valor no está dentro de ninguno de los rangos descritos, se ejecutan las sentencias a continuación de *othewise*
- La rama *othewise* se puede omitir, pero es preferible ponerla siempre

Ejemplo de case en una función

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program case_en_funcion;
function clasificacion_edad(edad: integer): string;
begin
    case edad of
    0 :
        result := 'Bebé';
    1..12 :
        result := 'Niño';
    13..17:
        result := 'Adolescente';
    otherwise      // sin dos puntos
        result := 'Adulto';
    end;    // 'end' de case, único sin 'begin'
end;
const
    Edad_cliente: integer = 14;
begin
    writeln( clasificacion_edad(Edad_cliente))
end.
```

Resultado:

```
Adolescente
```

Ejemplo erróneo

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program case_erroneo;

function clasificacion_edad(edad: integer): string;
begin
    case edad of
    0 :
        result := 'Bebé';
    1..13 :
        result := 'Niño';
    13..17:    // ¡Mal! El caso de 13 años está duplicado
               // El compilador dará un error
        result := 'Adolescente';
    otherwise
        result := 'Adulto';
    end;
end;
const
    Edad_cliente: integer = 14;
begin
    writeln( clasificacion_edad(Edad_cliente))
end.
```

Otro ejemplo

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program case_hexa;

function valor_digito_hex(digito: char): integer;
begin
    case digito of
    '0':
        result := 0;
    '1'..'9':
        result := ord(digito)-ord('0');
    'A'..'F':
        result := 10+ord(digito)-ord('A');
    'a'..'f':
        result := 10+ord(digito)-ord('a');
    otherwise
        result := 0;
    end;
end;
const
    Digito:char = 'a' ;
begin
    writeln(digito, ':', valor_digito_hex(Digito)) // Escribe a:10
end.
```

Case con múltiples sentencias

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
program case_varias_sentencias;

const
    edad: integer = 14;
begin
    case edad of
    0..1 : begin
        write('Menor de edad, ');
        writeln('bebé')
    end;
    2..12 : begin
        write('Menor de edad, ');
        writeln('niño')
    end;
    13..17: begin
        write('Menor de edad, ');
        writeln('adolescente')
    end
    otherwise      // sin dos puntos
        writeln('Adulto')
    end;    // 'end' de case, único sin 'begin'
end.
```

Ejemplo: fecha válida

https://gsyc.urjc.es/~mortuno/fpi/fecha_valida_v01.pas
https://gsyc.urjc.es/~mortuno/fpi/fecha_valida_v02.pas