Programación en Pascal. Convenios de la asignatura

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universidad Rey Juan Carlos

gsyc-profes (arroba) gsyc.urjc.es

Noviembre de 2018

Convenios en programación

Pascal, como cualquier otro lenguaje de programación, permite hacer las cosas de muchas maneras diferentes
En un proyecto software es conveniente establecer una serie de convenios para

- Garantizar el uso de buenas prácticas
- Mantener la homogeneidad del estilo en el código

Enumeramos aquí los de esta asignatura

Ficheros. Mayúsculas

- Los ficheros con el código fuente tendrán la extensión `.pas`
- Usaremos minúsculas para todos los identificadores y palabras reservadas excepto:
  - Los nombres de constante, que empezarán en mayúscula

```
const
    Pi : real = 3.141592653589793;
```

  - Los nombres de tipo de datos, para los que usaremos *NotaciónCamello*, con el prefijo *Tipo*

```
type
    TipoSemaforo = ( rojo, amarillo, verde );
```

Punto y coma. Tabuladores o espacios

- El programador tiene libertad para usar el carácter ';' estrictamente como separador de sentencias o añadirlo al final de cada sentencia
- El programador tiene libertad para indentar con tabuladores, con 4 espacios o con 8 espacios. Con tal de que cada fichero mantenga el mismo criterio
- Se permite cambiar de criterio entre fichero y fichero
- Se considerará un fallo muy severo mezclar tabuladores y espacios en la indentación de un mismo fichero

if then else

- Escribiremos *if* y *then* en la misma línea
- Dedicaremos una línea entera al *else*

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

- Cuando sea necesario insertar un bloque `begin end`, estarán en la misma línea las palabras `if then begin`, reservando una linea entera para `end else`

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

El propósito del convenio es que en todos los casos, la sentencia o sentencias del bloque *then* y la sentencia o sentencias del bloque *else* tengan exactamente 1 nivel de indentado

Uso adicional de begin end

Como sabes, para una única sentencia no es necesario añadir un bloque `begin end`
Solo lo añadiremos en un caso:

- Cuando dentro de un `if then else` haya otro `if then else` en la rama `then`
  Si está en la rama `else`, (if encadenado), no lo añadimos

El propósito del convenio es que no haya dudas sobre a qué `then` corresponde el `else`

Case

Escribiremos cada uno de los posibles valores de la expresión con la misma indentación que la palabra reservada `case`, que será la misma que la palabra reservada `end`

```
begin
    case Edad of
    0 :
        writeln('Bebé');
    1..18 :
        writeln('Menor de edad');
    otherwise      // sin dos puntos
        writeln('Adulto');
    end;     // 'end' de case, único sin 'begin'
end.
```

El propósito es que las sentencias de cada bloque tengan exactamente 1 nivel de indentación

```
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'
```

Si hay `begin` y `end` el criterio es el mismo: exactamente 1 nivel de indentación en cada bloque de sentencias

Indentación de bucles

Añadiremos `begin end` solo cuando sea necesario.
Como en los casos anteriores, buscamos que el cuerpo de los bucles tenga exactamente 1 nivel de indentación
Una setencia:

```
while i <= 3 do
    i := i + 1;
```

Varias sentencias:

```
while i <= 3 do begin
    writeln( 'Probando bucles');
    i := i + 1
end;
```