3.- Un programa concurrente está formado por hebras que representan coches y que son bucles infinitos (el número de hebras será una constante del programa). En algún momento los coches llegan a una zona de peaje con dos cabinas y se colocan en la cabina cuya cola tenga menos coches (si tienen el mismo número es indiferente la cola) y esperan a que les toque el turno de pagar (obviamente, el primer coche en llegar no espera). El "pago" del peaje y la parte de código después de pagar el peaje se simulan con retardos aleatorios. Implementar dicho programa (el archivo se deberá llamar "**ejercicio3.cpp**") con un monitor de acuerdo al esquema propuesto, en el que la función del monitor "llegada_peaje" devuelve la cabina en la que se ha colocado el coche. Incluir mensajes que permitan seguir la traza del programa.

```
Monitor Peaje

 var
    variables permanentes
    variables condición
```

```
Function llegada_peaje: integer;
var  X: integer;
begin
    · X:= cabina con  menos coches en su cola
    · actualizar estado (otro coche en cola cabina X)
    · si hay más coches en cabina X → esperar (bloqueo)
    · devolver X
end
```

```
Procedure pagado (var cab: integer)

begin
    · actualizar estado
    · liberar un coche
end
```

```
Hebra_coche;
 var cabina: integer;
  begin
     while (true) do
       begin
          cabina:= Peaje.llegada_peaje;
          {retardo aleatorio} (pago del peaje)
          Peaje.pagado(cabina);
          {retardo aleatorio}  (resto de código)
       end
  end
```