

# Práctica 4

Juan Llinares Mauri  
74011239E  
jlm109@alu.ua.es

---

## 1. Paso 0

Este paso es simplemente cambiar los dos paquetes *pkg\_graficos* de nuestra anterior práctica con los nuevos.

## 2. Paso 1

Creamos un nuevo paquete llamado *pkg\_protegidos*. Aquí implementaremos todos los nuevos tipos que nos hagan falta para esta práctica.

El tipo protegido que hemos creado lo hemos encapsulado en el paquete llamado *pkg\_protegidos*.

```
1  protected tpye OP_Aereovias is
2      ...
3  end OP_Aereovias;
```

*El array que se nos pide entonces será implementado de la siguiente manera:*

```
1  aereovias : array(T_Rango_AereoVias) of OP_Aereovias;
```

*El otro array que debemos implementar en la práctica es el que contendrá la información para saber qué zonas están ocupadas por aviones. Lo haremos de la siguiente manera:*

```
1  zonas_ocupadas : T_Rejilla_Ocupacion := (others => False);
```

*Hemos definido un contador para limitar el número de aviones que entran en una aerovía de la siguiente manera:*

```
1  subtype contador is Integer range 0..MAX_AVIONES_AEROVIA;
```

*Este contador estará regulado por las siguientes funciones:*

```

1  - Funci n para revisar cu ntos aviones hay en el contador.
2  function cantidad return contador;
3  -- Incrementa el n mero de aviones que hay en el contador.
4  procedure incAviones;
5  -- Decrementa el n mero de aviones que hay en el contador.
6  procedure decAviones;
```

*Sus implementaciones son las siguientes:*

```

1  -- Funci n para revisar cu ntos aviones hay en el contador.
2  function cantidad is
3  begin
4      return cont;
5  end cantidad;
6
7  -- Incrementa el n mero de aviones que hay en el contador.
8  procedure incAviones is
9  begin
10     cont := cont + 1;
11 end incAviones;
12
13 -- Decrementa el n mero de aviones que hay en el contador.
14 procedure decAviones is
15 begin
16     cont := cont - 1;
17 end decAviones;
```

*Creamos la variable contador como privada:*

```

1  cont : contador := 0;
```

*Luego, los procedimientos que necesitamos para colocar aviones son los siguientes:*

```

1  -- Funci n para comprobar si una posici n est libre.
2  function estado(pos : T_Rango_Rejilla_X) return Boolean;
3  -- Procedimiento para ocupar una posici n en una aerov a.
4  procedure ocupar(pos : T_Rango_Rejilla_X);
5  -- Procedimiento para dejar libre una posici n en una aerov a.
6  procedure liberar(pos : T_Rango_Rejilla_X);
7  -- Procedimiento para cambiar las posiciones en una aerov a.
8  procedure cambiar(posBloquear : T_Rango_Rejilla_X; posLiberar :
    T_Rango_Rejilla_X);
```

*Su implementación la hemos codificado de la siguiente manera:*

```

1  -- Funci n para comprobar si una posici n est libre.
```

```

2  function estado(pos : T_Rango_Rejilla_X) return Boolean is
3  begin
4      zonas_ocupadas(pos);
5  end estado;
6
7  -- Procedimiento para ocupar una posición en una aeronave.
8  procedure ocupar(pos : T_Rango_Rejilla_X) is
9  begin
10     zonas_ocupadas(pos) := true;
11 end ocupar;
12
13 -- Procedimiento para dejar libre una posición en una aeronave.
14 procedure liberar(pos : T_Rango_Rejilla_X) is
15 begin
16     zonas_ocupadas(pos) := false;
17 end liberar;
18
19 -- Procedimiento para cambiar las posiciones en una aeronave.
20 procedure cambiar(posBloquear : T_Rango_Rejilla_X; posLiberar :
    T_Rango_Rejilla_X) is
21 begin
22     liberar(posLiberar);
23     ocupar(posBloquear);
24 end;

```

*Estos procedimientos y la función que acabamos de ver editan las zonas ocupadas para poder llevar un seguimiento de los aviones y sus posiciones durante la ejecución del simulador.*

### 3. Paso 2

*Tendremos que editar el paquete generaaviones para adaptarlo a la nueva versión del programa. Antes que nada, tendremos que implementar una barrera para limitar la aparición de aviones e intentar no provocar colisiones. Esta barrera será la siguiente entry que hemos creado en el paquete del tipo protegido:*

```

1  -- Barrera para poder entrar al espacio protegido.
2  entry barreraAereovia(pos : T_Rango_Rejilla_X);

```

*La implementación que le daremos a la barrera será la siguiente:*

```

1  entry barreraAereovia(pos : T_Rango_Rejilla_X) when not
    zonas_ocupadas(T_Rango_Rejilla_X'First) and not
    zonas_ocupadas(T_Rango_Rejilla_X'First + 1) and not
    zonas_ocupadas(T_Rango_Rejilla_X'Last - 1) and not
    zonas_ocupadas(T_Rango_Rejilla_X'Last) and cont <=
    MAX_AVIONES_AEROVIA is
2  begin
3      -- Ha pasado la barrera.

```

```

4   if pos = T_Rango_Rejilla_X'First then
5       zonas_ocupadas(T_Rango_Rejilla_X'First) := True;
6   else
7       zonas_ocupadas(T_Rango_Rejilla_X'Last) := True;
8   end if;
9   incAviones;
10  end barreraAereovia;

```

*Lo que hará esta barrera es activar una posición a true en el array de las zonas ocupadas sólo y sólo si se cumplen las condiciones mencionadas en el when. Luego, si se cumplen estas, se incrementa el número de aviones.*

*Una vez tengamos la barrera creada, declaramos la siguiente variable:*

```

1 siguiente : T_CoordenadaX;

```

*Esta variable nueva la usaremos para realizar un seguimiento de las posiciones que tomarán los aviones en el futuro.*

*El bucle que teníamos en la práctica anterior se queda obsoleto para nuestro objetivo. La nueva implementación es la siguiente:*

```

1  Escribir(cadena => "Generando el nuevo avi n...");
2  aereovias(avion.aereovia).barreraAereovia(
3      Posicion_ZonaEspacioAereo(avion.pos.X));
4  Escribir(cadena => "El nuevo avi n pas la barrera de
5      seguridad");
6  Aparece(avion);
7
8  loop
9      siguiente := Nueva_PosicionX(avion.pos.X, avion.velocidad.X);
10     if Posicion_ZonaEspacioAereo(avion.pos.X) /=
11         Posicion_ZonaEspacioAereo(siguiente) then
12         aereovias(avion.aereovia).cambiar(Posicion_ZonaEspacioAereo(
13             siguiente), Posicion_ZonaEspacioAereo(avion.pos.X));
14         Actualiza_Movimiento(avion);
15     else
16         Actualiza_Movimiento(avion);
17     end if;
18     delay(PKG_tipos.RETARDO_MOVIMIENTO);
19 end loop;

```

*Podemos observar que primero se imprime un mensaje de aviso a la hora de generar un nuevo avión. Este mensaje saldrá siempre. No obstante, después de él se da paso a la barrera, donde se comprueba la posición del avión. Si todo es correcto, al instante se mandará otro mensaje de aviso diciendo que el avión pasó la barrera de seguridad y se entrará al bucle. Si no, el simulador esperará hasta que sea el momento correcto para que aparezca el avión. En la figura 1 podemos ver el programa en ejecución.*

*Por último, dentro del loop, si las posiciones actuales y siguientes son diferentes llamamos al método cambiar visto en la sección 2*

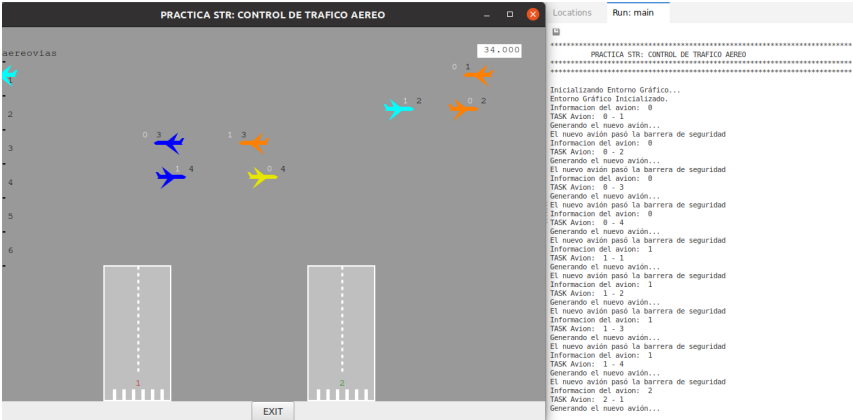


Figure 1. Simulación final con mensajes por consola.