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*.

```
protected tpye OP_Aereovias is
    ...
end OP_Aereovias;
```

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

```
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:

```
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:

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

Este contador estará regulado por las siguientes funciones:

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

Sus implementaciones son las siguientes:

```
-- Funci n para revisar cu ntos aviones hay en el contador.
function cantidad is
begin
  return cont;
end cantidad;

-- Incrementa el n mero de aviones que hay en el contador.
procedure incAviones is
begin
  cont := cont + 1;
end incAviones;

-- Decrementa el n mero de aviones que hay en el contador.
procedure decAviones is
begin
  cont := cont - 1;
end decAviones;
```

Creamos la variable contador como privada:

```
cont : contador := 0;
```

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

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

Su implementación la hemos codificado de la siguiente manera:

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

```
function estado(pos : T_Rango_Rejilla_X) return Boolean is
begin
  zonas_ocupadas(pos);
end estado;

-- Procedimiento para ocupar una posici n en una aerov a.
procedure ocupar(pos : T_Rango_Rejilla_X) is
begin
  zonas_ocupadas(pos) := true;
end ocupar;

-- Procedimiento para dejar libre una posici n en una aerov a.
procedure liberar(pos : T_Rango_Rejilla_X) is
begin
  zonas_ocupadas(pos) := false;
end liberar;

-- Procedimiento para cambiar las posiciones en una aerov a.
procedure cambiar(posBloquear : T_Rango_Rejilla_X; posLiberar :
   T_Rango_Rejilla_X) is
begin
 liberar(posLiberar);
 ocupar(posBloquear);
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:

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

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

```
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
begin
 -- Ha pasado la barrera.
```

```
 if pos = T_Rango_Rejilla_X'First then
    zonas_ocupadas(T_Rango_Rejilla_X'First) := True;
 else
    zonas_ocupadas(T_Rango_Rejilla_X'Last) := True;
 end if;
 incAviones;
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:

```
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:

```
Escribir(cadena => "Generando el nuevo avi n...");
aereovias(avion.aereovia).barreraAereovia(
   Posicion_ZonaEspacioAereo(avion.pos.X));
Escribir(cadena => "El nuevo avi n pas  la barrera de
   seguridad");
Aparece(avion);

loop
 siguiente := Nueva_PosicionX(avion.pos.X, avion.velocidad.X);
 if Posicion_ZonaEspacioAereo(avion.pos.X) /=
    Posicion_ZonaEspacioAereo(siguiente) then
    aereovias(avion.aereovia).cambiar(Posicion_ZonaEspacioAereo(
       siguiente),Posicion_ZonaEspacioAereo(avion.pos.X));
    Actualiza_Movimiento(avion);
 else
    Actualiza_Movimiento(avion);
 end if;
 delay(PKG_tipos.RETARDO_MOVIMIENTO);
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 para llevar a cabo el control correcto del pilotaje de los aviones en todas las aerolíneas.

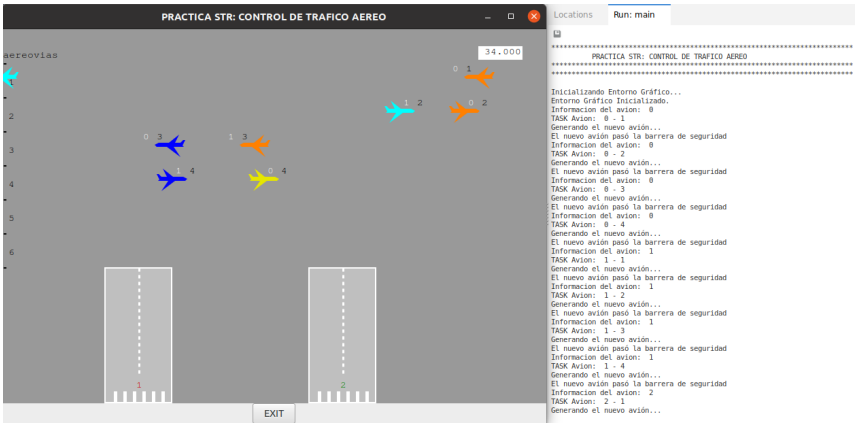

Figure 1. Simulación final con mensajes por consola.