Examen de Lenguajes de Alto Nivel

Febrero 2008

1) (2 puntos)

Implementar el procedimiento `Carga_Info` que obedece a la especificación indicada abajo, y sirve para leer del fichero de texto denominado `Fichero` la información correspondiente a una lista de libros de una biblioteca. El fichero contiene tres líneas por cada libro registrado en la biblioteca. La primera línea contiene el título del libro, la segunda el nombre del autor, y la tercera dos números separados por uno o varios espacios en blanco que indican el número de ejemplares del libro que tienen la biblioteca y el número de ellos que se encuentran prestados. Si se produce alguna excepción se debe dejar pasar después de mostrar un mensaje en pantalla indicando lo que ha sucedido.

```
subtype Ejemplares is Integer range 1..20;

type Libro is record
   Titulo : String(1..100);
   Long_Titulo : Integer range 0..100;
   Autor : String(1..60);
   Long_Autor : Integer range 0..60;
   Num_Ejemplares : Ejemplares;
   Num_Prestados : Ejemplares;
end record;

Max_Libros : constant Integer := 1000;

type Lista_Libros is array (1..Max_Libros) of Libro;

procedure Carga_Info (Fichero : String;
                      Lista : out Lista_Libros;
                      Num_Libros : out Positive);
```

2) (2 puntos)

Se dispone del tipo de dato `Articulo` definido en el siguiente paquete:

```
package Articulos is

   type Tipo_Codigo is new String (1..5);

   type Articulo is private;

   type Articulo_Ref is access Articulo;

   function Codigo (A : Articulo) return Tipo_Codigo;

   -- ... resto de las operaciones del paquete

private

   type Articulo is record
      Codigo      : Tipo_Codigo;
      Descripcion : String (1..200);
   end record;

end Articulos;
```

Tomando como base en este tipo, se define un almacén como una lista de punteros a artículos de la siguiente manera:

```
with Articulos, Listas_Simplemente_Enlazadas;
use Articulos;

package Almacen is new
   Listas_Simplemente_Enlazadas(Articulo_Ref,"=");
```

Por otra parte, se define el grupo de almacenes que posee una compañía como una lista de almacenes:

```
with Almacen, Listas_Simplemente_Enlazadas;
use Almacen;

package Grupo_De_Almacenes is new
   Listas_Simplemente_Enlazadas(Lista,"=");
```

El paquete `Listas_Simplemente_Enlazadas` corresponde al ADT que se ha visto en clase.

Crear una función compilada separadamente a la que se le pasa un grupo de almacenes (lista del paquete `Grupo_De_Almacenes`) y un código de artículo, y devuelve el número de almacenes (un `Positive`) en los que existe el artículo con ese código.

3) (3 puntos)

Se desea realizar una parte de una aplicación relativa a un sistema de control de tráfico aéreo militar. El objetivo es poder realizar el seguimiento de un avión enemigo previamente identificado y activar las alarmas oportunas en caso de un acercamiento peligroso a la base. Para ello, se dispone del paquete siguiente, que ya está implementado:

```
package Radar is

   type Grados is digits 6 range 0.0..360.0;
   type Altura is digits 6 range 0.0..65.000; -- en pies
   type Distancia is digits 6 range 0.0..50.000; -- en metros

   type Posicion is record
      Lat, Long : Grados;
   end record;

   Max_Aviones : constant := 100;

   type Id_Avion is range 1..Max_Aviones;

   procedure Posicion_Actual
     (Avión : in Id_Avion;
      Pos : out Posicion;
      Alt : out Altura);
   -- Devuelve en Pos y Alt, respectivamente, la posición y
   -- altura actuales del avión identificado por Avion.
   -- Si el radar no logra encontrar el avión, se eleva la
   -- excepción Desconocido.

   function Distancia_A_Base
     (Pos : Posicion;
      Alt : Altura) return Distancia;
   -- Devuelve la distancia en metros desde una posición
   -- cualquiera de un avión a la base del radar.
   -- Si la distancia resultande del cálculo no se encuentra en
   -- el rango definido eleva la excepción Fuera_De_Rango.
```

```
   Desconocido : exception;
   -- elevada por Posicion_Actual

   Fuera_De_Rango: exception;
   -- elevada por Distancia_A_Base;

end Radar;
```

Se pide realizar la implementación de la parte privada, así como del cuerpo del paquete cuya especificación es la siguiente:

```
with Radar;
package Seguimiento is

   type Objetivo is private;

   procedure Inicializa (Obj : in out Objetivo;
                         Avion :in Radar.Id_Avion);

   procedure Almacena_Posicion (Obj : in out Objetivo);

   Alarma : exception;
   No_Cabe : exception;
   -- elevadas por Almacena_Posicion

private
   ...
end Seguimiento;
```

Cada objetivo tiene asociado un identificador de avión y una lista capaz de almacenar hasta 500 posiciones.

El procedimiento `Inicializa` hace la lista de posiciones de un objetivo vacía y asigna un identificador de avión a seguir a dicho objetivo.

El procedimiento `Almacena_Posicion` para un objetivo, añade la posición actual dada por el radar a la lista de posiciones; además, comprueba que la distancia del avión asignado al objetivo a la base no es inferior a 10.000 metros. Si es inferior a esta distancia debe elevar la excepción `Alarma`. Debe elevar la excepción No_Cabe si la lista de posiciones se llena. En caso de que se pierda el rastro del avión sencillamente no se anotará la posición.

Se pide además realizar un segmento de programa capaz de realiza el seguimiento del avión con identificador 24. Desde la inicialización, el seguimiento consistirá en el almacenamiento de posiciones cada 12 segundos. Si sucede una alarma o se llena la lista de posiciones se deberá avisar a un sistema de emergencia mediante el procedimiento de librería siguiente:

```
with Radar;
procedure Emergencia (Avion :in Radar.Id_Avion);
```

En el diseño e implementación se deben utilizar los ADTs vistos en clase que se consideren más adecuados, justificando su uso.

4) (2 puntos)

Se dispone del siguiente paquete que no proporciona acceso mutuamente exclusivo al objeto que define:

```
package Operaciones is

   type Coordenadas is private;
```

```
   procedure Lee (C : Coordenadas; X,Y,Z : out Float);
   function Asigna (X,Y,Z : Float) return Coordenadas;

private

   type Coordenadas is record
      X : Float;
      Y : Float;
      Z : Float;
   end record;

end Operaciones;
```

Proponer las modificaciones necesarias a este paquete para que las operaciones `Lee` y `Asigna` proporcionen acceso mutuamente exclusivo a cualquier variable de tipo coordenadas.

Escribir además un procedimiento principal que declare una tarea. El procedimiento debe pedir al usuario en un lazo infinito que introduzca los valores de las coordenadas de un punto, para asignarlos a una variable en cuanto los tenga. La tarea debe presentar en pantalla cada 30 segundos los valores de las coordenadas actuales.

5) (1 punto)

Se dispone del siguiente tipo etiquetado definido en un paquete:

```
package Libros is

   type Tipo_Descripcion is private;

   type Libro is tagged private;

   type Libro_Ref is access Libro'Class;

   procedure Inicializa (V : Libro);
   -- Pide por teclado y pantalla todos los datos del libro
   -- para inicializarlo

   -- Más operaciones de la clase

private
   type Tipo_Descripcion is record
      Titulo : String(1..100);
      Long_Titulo : Integer range 0..100;
      Autor : String(1..60);
      Long_Autor : Integer range 0..60;
   end record;

   type Libro is tagged record
      Localizacion : String (1..10);
      Descripcion : Tipo_Descripcion;
   end record;

end Libros;
```

Se pide hacer otro paquete (definir la especificación) con un tipo etiquetado que sea extensión de `Libro`, el `Libro_Con_CD`, y que posea un campo más correspondiente a localización del CD que acompaña al libro. Escribir el código correspondiente a la operación `Inicializa` sobre este nuevo tipo para que pida al usuario la localización del CD, además de la información del libro que ya pedía.