Examen de Lenguajes de Alto Nivel

Septiembre 2004

Cuestiones (4 cuestiones, 4 puntos en total; contestar brevemente de forma razonada)

1) Se dispone del siguiente paquete con un procedimiento `P` que puede elevar la excepción `Error`:

```
package Paq is
   procedure P (I : Integer);

   Error : exception;
end Paq;
```

Se pide hacer otro procedimiento, compilado aparte, que llame a `P` con el valor de `I` que se indique en un parámetro. Si se elevase `Error`, se debe llamar de nuevo a `P` con un valor de `I` que sea uno más que el anterior. Así sucesivamente, hasta que la llamada tenga éxito.

2) Escribir una tarea que repita continuamente lo siguiente: espera a otra tarea en un punto de entrada (**`entry`**) sin parámetros; luego, espera durante 200 ms y pone un mensaje en pantalla.

3) Se dispone del siguiente tipo etiquetado definido en un paquete:

```
package Computadores is
   type Byte is range 0..255;
   type Direccion is array (1..4) of Byte;
   type Computador is tagged record
       Nombre : String(1..20);
       IP : Direccion;
   end record;
   procedure Muestra (C : Computador);
end Computadores;
```

Se pide hacer otro paquete (especificación y cuerpo) con un tipo etiquetado que sea extensión de `Computador`. El nuevo tipo debe tener un campo más, que es el `Responsable` (String de 20 caracteres). Además, debe redefinir `Muestra` para que llame al procedimiento `Muestra` de `Computadores`, y luego muestre en pantalla el campo `Responsable` con el formato de este ejemplo:

```
Responsable: María Gómez
```

4) Se dispone del siguiente tipo de datos declarado en un paquete:

```
package Libros is
   type Libro is private;
   procedure Lee_Libro
      (Nombre: in String; Casilla: in Positive; L: out Libro);
   procedure Escribe_Libro
      (Nombre: in String; Casilla: in Positive; L: in Libro);
private
```

```
      type Libro is record
         Autor : String(1..20);
         Titulo : String (1..40);
         N_Autor : Integer range 0..20;
         N_Titulo : Integer range 0..40;
      end record;
   end Libros;
```

Se pide escribir el cuerpo de este paquete con el par de procedimientos indicados para, respectivamente, leer o escribir un libro de un fichero binario cuyo nombre se indica, con entrada/salida directa. La casilla del fichero a leer o escribir es el parámetro `Casilla`. El fichero está cerrado al principio y debe dejarse cerrado al final. No es preciso tratar excepciones.

Examen de Lenguajes de Alto Nivel

Septiembre 2004

Problema (6 puntos)

Se desea escribir una aplicación que gestiona el funcionamiento de un robot de venta de productos que están en una estantería.

Se dispone de dos paquetes ya realizados que corresponden al robot y al teclado o panel de control a través del cual el usuario selecciona el producto que desea:

```
package Robot is

   -- Representa un robot capaz de desplazarse delante de una
   -- estanteria y recoger un objeto del estante que tiene delante

   function Coord_X_Es_Alcanzable (Coord : Float) return Boolean;
   function Coord_Y_Es_Alcanzable (Coord : Float) return Boolean;
   -- Estas operaciones indican si la coordenada X (horizontal) o Y
   -- (altura) indicada es alcanzable por el robot
   -- Las coordenadas van en metros

   procedure Mueve (Coord_X, Coord_Y : Float);
   -- Ordena al robot iniciar el movimiento hacia el punto indicado
   -- Eleva No_Alcanzable si la posición no es alcanzable

   function Destino_Alcanzado return Boolean;
   -- Indica si se ha alcanzado o no el punto destino

   procedure Recoge_Objeto;
   -- Recoge un objeto de la estanteria y lo pone en la bandeja

   No_Alcanzable : exception;

end Robot;



with Estanteria;
package Teclado is

   procedure Lee_Tecla
     (Pulsada : out Boolean;
      F : out Estanteria.Fila;
      C : out Estanteria.Columna);
   -- Consulta el teclado. Si no hay tecla pulsada retorna Pulsada
   -- a False. Si hay tecla pulsada y el pago se ha realizado
   -- retorna Pulsada igual a True y la fila y columna
   -- de la tecla, respectivamente en F y C.

   procedure Indica_Error;
   procedure Indica_Preparado;
   procedure Indica_En_Operacion;
   procedure Indica_Averiado;
   -- Estas operaciones encienden respectivamente la luz de error,
   -- preparado, en operacion, o averiado, apagando las otras tres

end Teclado;
```

La estantería tiene varias filas y columnas de tamaños diferentes. El número de columnas es fijo, pero cada una tiene un número distinto de filas o estantes. Cada estante tiene varias unidades de un producto. La estantería está representada en el siguiente paquete que contiene una máquina de estados abstracta donde se guardan todos los datos relativos a la misma.

```
package Estanteria is

   type Fila is range 1..10;
   type Columna is range 1..30;

   function Max_Filas(C : Columna) return Fila;
   procedure Cambia_Max_Filas (C : Columna; Nuevo_Max : Fila);
   -- Estas operaciones retornan o cambian el maximo numero de
   -- filas de una columna de la estantería

   function Posicion (C : Columna) return Float; -- metros
   procedure Cambia_Posicion(C : Columna; Nueva_Pos : Float);
   -- Estas operaciones retornan o cambian la posicion central de
   -- la columna C
   -- Cambia_Posicion eleva Robot.No_Alcanzable si la posicion no
   -- es alcanzable por el robot

   function Altura (F: Fila; C : Columna) return Float; -- metros
   procedure Cambia_Altura
      (F : Fila; C : Columna; Nueva_Altura : Float);
   -- Estas operaciones retornan o cambian la altura del estante
   -- de la columna C y fila F
   -- Elevan Fila_No_Existe si en la columna C no existe la fila F
   -- Cambia_Altura eleva Robot.No_Alcanzable si la altura no
   -- es alcanzable por el robot

   function Num_Unidades(F : Fila; C : Columna) return Natural;
   procedure Rellena_Producto
        (F : Fila; C : Columna; Nueva_Cantidad : Natural);
   -- Estas operaciones retornan o cambian el numero de unidades de
   -- producto que hay en el estante de la columna C y fila F
   -- Elevan Fila_No_Existe si en la columna C no existe la fila F

   procedure Quita_Unidad(F : Fila; C : Columna);
   -- Esta operacion quita una unidad al estante de la columna C
   -- y fila F
   -- Eleva Fila_No_Existe si en la columna C no existe la fila F
   -- Eleva No_Hay si el numero de unidades es cero

   Fila_No_Existe : exception;
   No_Hay : exception;

end Estanteria;
```

Se pide en primer lugar escribir parte de este paquete:

- la estructura de datos completa para almacenar: las filas de cada columna, las posiciones de cada columna, la altura de cada estante, y el número de unidades de cada estante
- las operaciones `Altura` y `Cambia_Altura`

También se desea escribir el programa principal, de modo que haga lo siguiente. Primero llama al procedimiento `Configura_Estanteria`, escrito aparte y ya realizado, que configura las posiciones, filas, alturas y unidades de la estantería. Luego entra en un lazo infinito en el que hace lo siguiente:

- Mueve el robot al punto de coordenadas X a Y (0.0, 0.0) (no hace falta esperar a que llegue).
- Enciende en el teclado la luz de "preparado" llamando a `Indica_Preparado`.
- Lee una tecla. Si no está pulsada vuelve al inicio del lazo. Si está pulsada, sigue con lo indicado a continuación.
- Comprueba el número de unidades del estante seleccionado con la tecla. Si es cero, indica el error en el teclado, espera un segundo, y luego va al principio del lazo. Si no hay error, continúa.
- Mueve el robot hasta el estante seleccionado (obteniendo la posicion y altura del estante del paquete `Estanteria`)
- Mientras el robot no alcance el destino se enciende la indicación "en operacion" en el teclado. Esto se hace mediante un lazo.
- Al alcanzar el destino se le manda al robot recoger un objeto de la estantería, y se quita una unidad a ese estante con `Quita_Unidad`.
- Por último, hay que esperar en un lazo a que el robot alcance de nuevo su destino.
- Si en cualquier momento se elevase `Estanteria.Fila_No_Existe` (normalmente porque el usuario ha elegido mal el botón), se debe indicar el error en el teclado, esperar un segundo, y luego ir al principio del lazo.

Si en algún momento se elevase `Robot.No_Alcanzable`, esto indica que la configuración está mal y no se puede hacer nada mas. Por ello, en este caso se llamará a `Teclado.Indica_Averiado` y se finalizará el programa.