PRÁCTICA 2 : Tareas *en Ada*

OBJETIVOS:

1. Conocer la representación de procesos concurrentes en el lenguaje Ada, implementando tareas para representar distintos elementos de un sistema simulado.
2. Detectar y comprender algunos de los problemas que surgen en la ejecución de un sistema concurrente.

PERIODO RECOMENDADO PARA SU REALIZACIÓN: 1 + ½ semanas

IMPORTANTE: Repositorio en GitHub

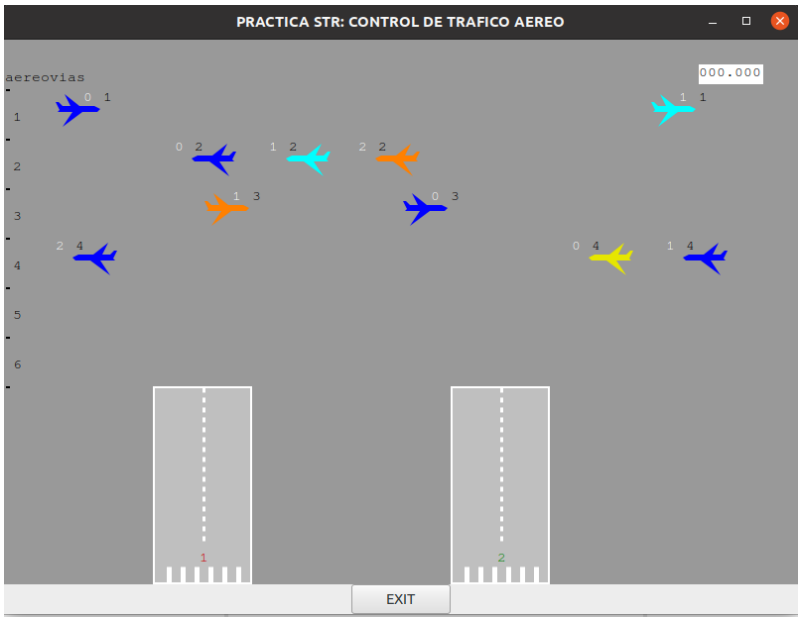
Para que el trabajo práctico que realices en la asignatura sea revisado, debes crear un repositorio en GitHub y compartirlo en modo lectura con el profesor.

El nombre del repositorio debe seguir el siguiente formato (todo en minúsculas y sin acentos):
str-2023-apellido1-apellido2-nombre

Debes dar permiso de lectura a tu repositorio de la asignatura al usuario abotia@gcloud.ua.es

ENUNCIADO: Vuelo de aviones

A partir del siguiente entorno gráfico se debe implementar el vuelo "simultáneo" de aviones. La siguiente figura muestra un ejemplo de la ejecución del sistema simulado completo en un instante determinado.



Tenemos 6 aereovías por las que vuelan distintos aviones a distintas altitudes. La dirección de vuelo es hacia la derecha en las aereovías impares y hacia la izquierda en las pares. Los aviones aparecen por el margen correspondiente de la aereovía y cuando llegan al final vuelven al principio de la misma (simulamos un vuelo circular en cada aereovía). De momento los aviones no pueden cambiar de aereovías, con el fin de ir descendiendo hasta aterrizar en las pistas. Esto lo implementaremos en futuras prácticas.

En la parte superior derecha aparece un cronómetro que no funciona. Lo implementaremos en la siguiente práctica.

En esta práctica debes implementar la aparición de 3 aviones en cada una de las 4 primeras aereovías, que irán volando por su aereovía hasta que el usuario finalice la ejecución del programa al pulsar el botón EXIT.

Cada avión que aparece tiene que ser representado por una tarea distinta.

PASOS A SEGUIR:

En primer lugar, se detalla una breve descripción de los paquetes proporcionados:

- **Paquetes** que se deben usar como **ayuda** para implementar el proyecto y que en los enunciados de las prácticas, se irá indicando en cada caso qué usaremos de cada uno:
  - ***pkg_tipos*** : declaración de constantes y tipos de datos (**sólo el fichero .ads**)
  - ***pkg_graficos*** : procedimientos y funciones para la representación gráfica

- ***pkg_debug*** **: paquete** para imprimir en un fichero de texto y por pantalla mensajes para facilitar la **depuración** del programa si se considera necesario:
  - sólo necesitarás usar el procedimiento ***Escribir***
  - ¿Cómo se llama el fichero de texto donde se escriben los mensajes de salida?
  - ¿Por qué no tienes que invocar al procedimiento ***Crear_Fichero***?
- ***pkg_buffer_generico***, ***double_buffer*** : otros **paquetes** que sólo se necesitarán para la **compilación** del proyecto:

IMPORTANTE! **NO debes modificar** los paquetes proporcionados por el profesor. **Debes crear tus propios ficheros** en los que se definan paquetes que incluyan el código fuente propio que implementes.

PASO 1: Visualizar el entorno estático simulado

Crea un proyecto denominado **practica2** utilizando la plantilla "*Basic → Simple Ada Project*" y copia en su carpeta /src todos los paquetes proporcionados.
El cuerpo del programa principal que debes implementar debe contener lo siguiente:

```
begin
   pkg_graficos.Simular_Sistema;
end main;
```

Aclaración: el procedimiento *pkg_graficos.Simular_Sistema* se encarga de la animación gráfica. Implementa un bucle infinito que redibuja y captura eventos de la interfaz gráfica. Se interrumpe al pulsar el botón **Exit** de la ventana principal, acabando automáticamente la ejecución del programa principal y de todas las tareas en ejecución (si las hubiere).

Antes de compilar el proyecto, hay que añadir como dependencia del mismo la librería gráfica gtkAda. Para ello, desde el menú contextual del proyecto utiliza la opción *Project → Properties → Dependencies* y pulsando el botón '+' selecciona el fichero gtkada.gpr que se encuentra en la siguiente ruta de la máquina virtual: /usr/gnat/lib/gnat/
El campo "Limited with" no hay que activarlo

Al finalizar el paso 1, el resultado de la ejecución de tu proyecto deberá ser simplemente la visualización de la ventana principal del entorno simulado. Tu programa finalizará su ejecución cuando pulses el botón Exit.

PASO 2: Creación de aviones

Utiliza el body de la tarea ***TareaGeneraAviones*** y del tipo tarea ***T_Tarea_Avion*** del ejercicio 8 de la práctica 1, para crear un nuevo paquete con las funcionalidades necesarias para la creación de aviones en el sistema simulado, dadas por los siguientes requerimientos:

1. Debe aparecer un nuevo avión en una aereovía con una frecuencia de aparición aleatoria. Para ello genera un valor aleatorio de retardo de espera y escribe Delay(Duration(*retardo*)); como última sentencia del bucle, donde *retardo* representa el valor aleatorio generado de tipo *T_RetardoAparicionAviones* y se realiza un casting al tipo *Duration* (expresado en segundos) para que la tarea suspenda su ejecución el tiempo especificado.
2. Se deben crear un número constante de aviones en las 4 primeras aereovías (ya lo tienes implementado en el código proporcionado de la práctica anterior).
3. Debes inicializar la información restante de cada avión, ya que el nuevo tipo record proporcionado contiene más campos que deberás actualizar. En concreto, la aereovía inicial, la pista asignada, un color aleatorio, la dirección de vuelo (velocidad en eje X positiva si el número de aereovía es impar y negativa si es par), y por último la posición de inicio en la que aparecerá el avión.
4. En el cuerpo del tipo tarea T_Tarea_Avion, añade la información restante del avión para que se imprima en el log
5. Incorpora en el cuerpo de las tareas el siguiente manejador de excepciones que imprimirá en el log de nuestro programa una posible terminación anormal de la tarea a causa de una excepción no manejada:

```
exception
   when event: others =>
      PKG_debug.Escribir("ERROR en TASK ….: " & Exception_Name(Exception_Identity(event)));
```

Del paquete pkg_tipos, debes utilizar lo siguiente:

- el tipo ***T_RecordAvion***, para la información de un avión
- el tipo ***Ptr_T_RecordAvion***, para pasar como parámetro discriminante la información del avión a la tarea
- el subtipo ***T_RetardoAparicionAviones***, que define el rango de segundos para obtener valores aleatorios de la frecuencia de aparición
- el subtipo ***T_ColorAparicionAvion***, que define el rango de colores para obtener el color aleatorio de un avión
- el tipo ***T_IdAvion***, que define el rango de número de identificación de un avión en una aereovía
- el tipo ***T_Rango_Aereovia***, que define el rango números de aereovías
- valor ***SIN_PISTA*** del tipo T_PistaAterrizaje, para inicializar el campo correspondiente
- constante ***VELOCIDAD_VUELO*** (velocidad constante en el eje X de un avión)

Del paquete pkg_graficos, utiliza lo siguiente:

- *Function* ***Pos_Inicio***, que devuelve las coordenadas en las que debe aparecer el avión en la aereovía indicada

Al finalizar el paso 2, se debe haber añadido la siguiente funcionalidad a tu proyecto: la creación con una frecuencia aleatoria de un nuevo avión con un color aleatorio en una aereovía. Cada avión solo mostrará en el log su información, pero de momento no se hará visible en el sistema

PASO 3: Vuelo de aviones

Modifica el body del tipo tarea T_Tarea_Avion con las funcionalidades necesarias para el vuelo de un avión en una aereovía, que vienen dadas por los siguientes requerimientos:

1. El avión aparecerá en su aereovía asignada.
2. El avión se moverá con una velocidad constante por la aereovía.

Para simular el movimiento de los aviones, se debe invocar al procedimiento pgk_graficos.Actualiza_Movimiento cada 0.1 segundos, usando la sentencia delay(*pkg_tipos.RETARDO_MOVIMIENTO*). Dependiendo de la velocidad del avión, el sistema calcula automáticamente su nueva posición.

Del paquete pkg_tipos, debes utilizar lo siguiente:

- constante ***RETARDO_MOVIMIENTO***

Del paquete pkg_graficos, utiliza lo siguiente:

- Procedure ***Aparece***, que hace visible el avión indicado
- Procedure ***Actualiza_Movimiento***, que actualiza la posición del avión según su velocidad
- Procedure ***Desaparece***, que deja de visualizar el avión indicado

Ten en cuenta que cuando aparece un nuevo avión al principio de su aereovía, puede haber otro avión cercano a esa posición. En ese caso, el sistema automáticamente generará la excepción *pkg_tipos.DETECTADA_COLISION* al invocar al procedimiento *Actualiza_Movimiento*. Debes tratar esta excepción de forma que se muestre en el log el identificador del avión accidentado y hacerlo desaparecer del sistema, terminando así la ejecución de esa tarea.

Al finalizar el paso 3, se deberá haber añadido la siguiente funcionalidad a tu proyecto: la visualización del vuelo de 3 aviones (*pkg_tipos.NUM_INICIAL_AVIONES_AEREOVIA*) en las 4 primeras aereovías, a menos que se produzca alguna colisión, en cuyo caso aparecerá en el log el identificador del avión accidentado. Los aviones volarán ininterrumpidamente por su aereovía, hasta que el usuario pulse la tecla EXIT.