PRÁCTICA 4 : *Memoria Compartida*

OBJETIVOS:

1. Comprender y saber resolver problemas de **comunicación y sincronización** de tareas concurrentes, mediante el uso de **memoria compartida** en Ada.
2. Emplear la **sincronización condicional** en tareas concurrentes que proporcionan los **objetos protegidos** en Ada.

PERIODO RECOMENDADO PARA SU REALIZACIÓN: 1 + ½ semanas

ENUNCIADO: Semáforo de coches y recogida de peatones.

Como continuación de la práctica 3, se debe implementar el control del paso de trenes que atraviesa la carretera utilizando el semáforo. Los trenes siempre tendrán preferencia de paso, y serán los coches y taxis los que deban pararse si el semáforo está rojo. También se debe implementar la recogida de peatones por parte de los taxis. En la siguiente figura, se muestra un ejemplo de la ejecución del sistema simulado en un instante determinado:

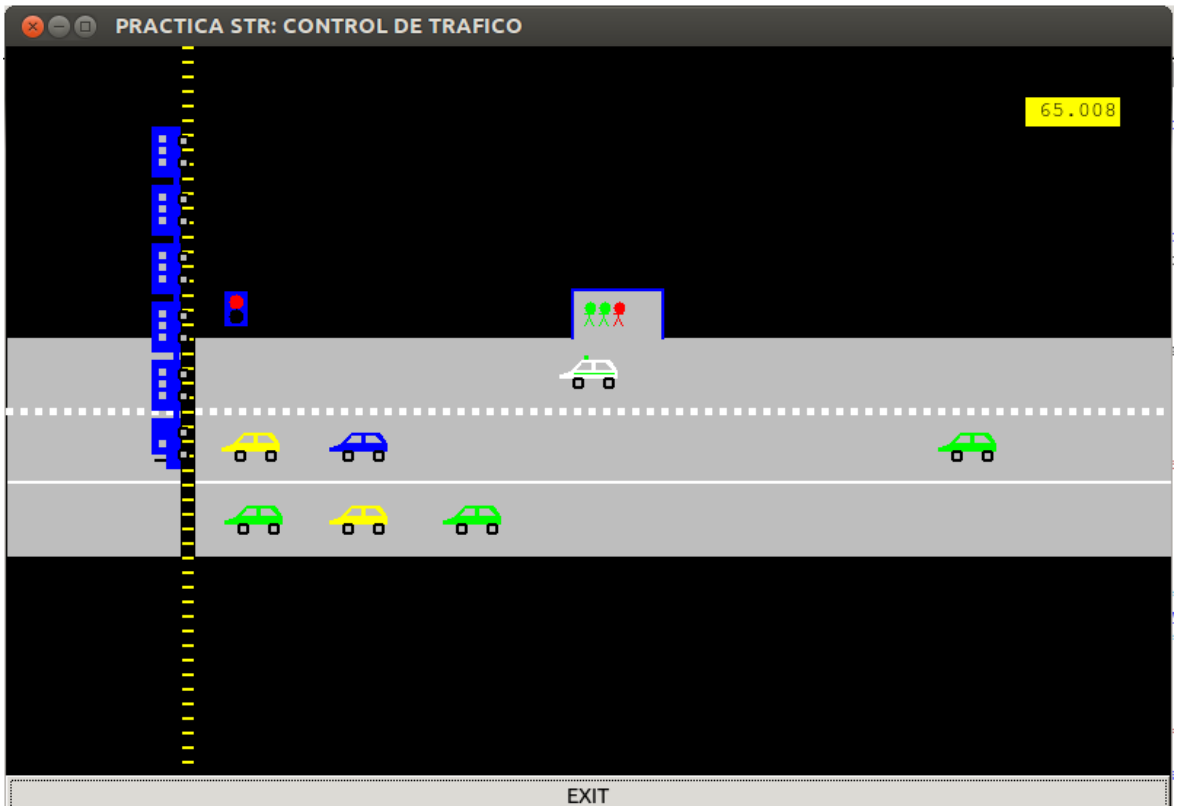

Figura 1. Ningún vehículo está cruzando la vía del tren al estar el semáforo en rojo. Sólo dos peatones (los visualizados en verde) podrán subir al taxi.

PASOS A SEGUIR:

En primer lugar debes añadir a tu proyecto el nuevo paquete proporcionado ***pkg_semaforo***.

PASO 1: Semáforo regulado por el tren

Añade en el body del tipo tarea T_Tarea_Tren lo siguiente:

La declaración de la variable local:
parado_trafico : Boolean := false;

Al principio del bucle de control, el siguiente if:

```
IF PKG_graficos.Dentro_Zona_Cruce(Tren) and not parado_trafico THEN
   PKG_Semaforo.OP_Semaforo.Cambia_Estado_Semaforo(Red);
   Parado_Trafico := True;
ELSIF NOT PKG_graficos.Dentro_Zona_Cruce(Tren) AND Parado_Trafico THEN
   PKG_Semaforo.OP_Semaforo.Cambia_Estado_Semaforo(Green);
   Parado_Trafico := False;
END IF;
```

El código anterior utiliza un objeto protegido declarado y definido en el paquete *pkg_semaforo* y que contiene la información del estado del semáforo (verde o rojo). El tren actualiza el estado del semáforo a rojo cuando se dispone a cruzar la carretera y a verde cuando abandona el cruce. El tren es la única tarea que modificará el estado del semáforo ¿Qué otras tareas deben compartir dicha información?

Si ejecutas ahora tu proyecto podrás observar además que también se cambia en la interfaz gráfica el color del semáforo.

PASO 2: Control de tráfico regulado por el semáforo

Modifica el comportamiento del tipo tarea T_Tarea_Taxi, de forma que cuando llegue a la posición de parada del semáforo y éste se encuentre en rojo, el taxi se pare (actualizando su velocidad a cero y suspendiendo la ejecución de la tarea). El taxi deberá reanudar su marcha cuando se ponga verde (actualizando su velocidad con el valor de la constante *pkg_tipos.**VELOCIDAD_TAXI***).

IMPORTANTE: mientras el vehículo espera a que el semáforo se ponga verde, la tarea debe estar suspendida, es decir, hay que evitar bucles de espera ocupada.

Después reutiliza esta misma modificación sobre la tarea de tipo coche, para que éstos tengan el mismo comportamiento cuando se encuentran con el semáforo en rojo. Para reanudar su velocidad, la actualizaremos con un valor constante *pkg_tipos.**VELOCIDAD_COCHE***, en lugar de la que originalmente llevaban.
Además, en el caso de los coches, ten en cuenta que cuando el semáforo está en rojo y hayamos detenido el primer coche que llega al mismo, el resto de coches se paran automáticamente para evitar choques, por tanto tendrás que añadir en el bucle de control el código necesario para que reanuden su marcha cuando el semáforo pase a verde. Para simplificar el código, suponemos que un coche sólo puede estar parado debido a que el semáforo se encuentre en rojo, por tanto, si el coche está parado, cuando el semáforo se pone en verde debemos hacer que el coche reanude su marcha.

Para implementar el nuevo comportamiento de los coches y taxis, utiliza las siguientes funcionalidades del paquete ***pkg_graficos***:

- Función ***Posicion_Stop_Semaforo***: devuelve true si el vehículo especificado está en las proximidades del semáforo
- Función **Coche_Parado() :** devuelve *true* si el coche especificado tiene velocidad cero.
- Procedure ***Actualiza_Atributo:*** se utiliza para actualizar la velocidad del vehículo especificado, y deberás hacer un *casting* del primer parámetro al tipo T_Rango_Velocidad. En el caso de los coches, es importante que inmediatamente después de actualizar su velocidad, añadas la invocación al procedimiento ***Actualiza_Movimiento***.

Ten en cuenta que cuando se modifica la velocidad de un coche con el procedimiento *Actualiza_Atributo* se está modificando una variable local de la tarea. Este cambio de velocidad sólo podrá ser conocido por otros coches (para evitar choques entre ellos) cuando se invoque al procedimiento Actualiza_Movimiento, que se encarga de actualizar un buffer de memoria común a todas las tareas y que está implementado en los paquetes de la interfaz gráfica.

Al finalizar el paso 2, como resultado de la ejecución de tu proyecto debes tener una circulación de coches y taxis en la que se eviten los choques con el tren mediante el semáforo regulado por éste último.

PASO 3: Recogida de peatones por el taxi

Debes modificar el comportamiento del tipo tarea T_Tarea_Taxi de forma que cumpla estas nuevas especificaciones:

1. El taxi debe circular con su correspondiente indicador de libre (verde) u ocupado (rojo). Para ello, utiliza el procedimiento ***pkg_graficos.Actualiza_Indicador_Taxi***, indicando en su primer parámetro *true* si está libre y *false* si está ocupado. Observa que por defecto los taxis siempre aparecen con su indicador a verde.
2. Cuando el taxi llega a la parada de peatones y después de transcurrir el tiempo de parada obligatorio, el taxi tiene un tiempo máximo adicional para recoger pasajeros (determinado por la constante ***pkg_tipos.TIEMPO_RECOGIDA_PASAJEROS***) durante el cual esperará a llenar su capacidad (determinada por la constante ***pkg_tipos.CAPACIDAD_TAXI***).
3. Cuando el taxi se haya llenado o termine su tiempo máximo de recogida de pasajeros, reanudará su marcha.

También debes modificar el comportamiento de la tarea de tipo peatón de forma que cumpla estas nuevas especificaciones:

1. Un peatón durante su tiempo de espera (determinado por la constante ***pkg_tipos.TIEMPO_ESPERA_PEATON***) puede subir a un taxi, siempre y cuando le queden plazas libres.
2. Después de que un peatón se haya subido a un taxi o haya terminado su plazo de espera, abandonará el sistema.
3. Antes de desaparecer del sistema, el peatón deberá visualizarse con un nuevo color durante 2,5 segundos (verde si se ha subido al taxi y negro si no lo ha conseguido). Para actualizar el color del peatón, utiliza el procedimiento *pkg_graficos.**Actualiza_Color_Peaton.***

Para implementar estos nuevos comportamientos para el taxi y los peatones, necesitarás utilizar un nuevo objeto protegido ¿Por qué?

Al finalizar el paso 3, se deberá haber añadido la siguiente funcionalidad a tu proyecto: los peatones acceden al taxi sin sobrepasar su capacidad máxima y el taxi tiene una espera de tiempo limitada para llenarse.