Intrucciones para Modificar la Propuesta de Solución de Robot en Laberinto para que el Código sea Concurrente

Por Juan C. Giraldo

A la fecha Usted debe estar en capacidad de Concebir, Diseñar e Implementar en Lenguaje C cualquier algoritmo mediante una Máquina de Estados Finitos.

> Un **Máquina de Estados Finitos** es un modelo de comportamiento de un sistema con entradas y salidas en donde las salidas dependen del estado y las entradas actuales.

El proceso fundamental en el desarrollo de *software* para sistemas de computador "embebido" y de sistemas digitales consiste en el diseño de *FSMs* en diagramas de estados (*FSM* son las siglas en inglés de *Finite State Machine*). En *software*, luego de realizar las pruebas de escritorio respectivas, se realiza la implementación de dicha *FSM* en Lenguaje C en una función. En la primera parte de la función SE ESTABLECE LAS CONDICIONES INICIALES DE OPERACIÓN DE LA MÁQUINA. A continuación LA TRANSICIÓN ENTRE ESTADOS se implementa mediante una ÚNICA secuencia de repetición con una variable centinela que vigila la ejecución de la máquina mediante un "while()". A continuación se escribe una secuencia de selección de ESTADOS mediante "switch-case" y en cada ESTADO se implementa mediante una secuencia de selección priorizada "if() … else if() … else …" la selección de UN EVENTO, externo o interno, que "dispara o nó" la transición al siguiente estado de la máquina.

En cada CICLO de la secuencia de repetición SE EVALÚA LA OCURRENCIA DE UN SOLO EVENTO. En caso de ser VERDADERO se ejecuta una SECUENCIA LINEAL DE INSTRUCCIONES que incluye como instrucción final, la determinación del SIGUIENTE ESTADO. En caso de ser FALSO se continua con la evaluación de las siguientes transiciones de menor prioridad.

CONSTRUYA SU MÁQUINA DENTRO DE UNA FUNCIÓN INDEPENDIENTE:

Cambien el "main()" por el nombre de la función de su robot. Llame a la función "username_robot()", en donde el prefijo "username" es el nombre de usuario de su correo electrónico de la Javeriana. Si "username" tiene un punto "." cámbielo por la secuencia "_dot_" y si tiene un guión "-" cámbielo por la secuencia "_dash_".

jcgiraldo@javeriana.edu.co quedará "jcgiraldo_robot()"
andrade.h@javeriana.edu.co quedará "andrade_dot_h_robot()"
brayan-lozano@javeriana.edu.co quedará "brayan_dash_lozano_robot()"

Cambie la definición

```
int
main()
```

por

```
void
username_robot()
```

EXTRACCIÓN DE PROCESOS PARA PREPARAR LA MÁQUINA:

Antes del "while()" puede existir procesos para preparar la máquina. Estos se eliminan de la función para re-ubicarlos fuera de la misma. En un sistema de computador "embebido" estos procesos corresponden generalmente a la CONFIGURACIÓN INICIAL DE PERIFÉRICOS (puertos paralelos, temporizadores, puertos de comunicación serial, entre otros).

Borre las líneas

```
Testing_maze();
```

y

```
Challenge_Was_Finished();
```

DETERMINACIÓN DEL ESTADO INICIAL E INICIACIÓN DE VARIABLES:

Para hacer que el CONTEXTO o conjunto de todas las variables locales (junto con la variable estado) usadas en la función perduren y permanezcan inalteradas incluso una vez la función termina, se declaran las variables de tipo "static", poniendo esta palabra reservada de Lenguaje C justo antes de la declaración de cada variable dentro de la función. ES MUY IMPORTANTE que la asignación del estado inicial lo haga en la misma línea en donde declara la variable. Finalmente haga también PRIVADO A LA FUNCIÓN la declaración de los tipos de enumeración para identificar cada uno de los estados de su diseño.

Declare como "static" TODAS LAS VARIABLES usadas de manera local a la función. Si requiere otorgar un valor inicial a una variable, hágalo en la misma línea donde la declara. El tipo de almacenamiento "static" permite que las variables dentro de la función sean privadas a la function y MÁS IMPORTANTE AÚN es que sus valores permanezcan entre dos llamados distintos a la misma función.

```
void
username_robot()
{
typedef enum {
    WALKING_FORWARD,        /* Walking forward until a wall in front is found */
    FOLLOWING_LATERAL_WALL, /* Walking with a wall just in a lateral side     */
    TURNING_AROUND          /* Going around the end of the wall in one side   */
} STATE_T;

static STATE_T state = WALKING_FORWARD;
...
} /* username_robot */
```

MODIFIQUE LA FUNCIÓN PARA QUE HAGA UNA SOLA PASADA:

Como ya ha creado un mecanismo para PRESERVAR EL CONTEXTO (mediante la declaración de tipo "static"), ahora debe modificar la función para que haga UNA SOLA PASADA y SALGA DE LA FUNCIÓN SIN PÉRDIDA DE LA INFORMACIÓN de todas las variables locales a la misma.

Cambie el

```
while( Running() )
```

por un

```
if( TRUE )
```

CAMBIO DE LA LLAMADA A LAS FUNCIONES:

Como su *FSM* manipula uno de varios robots en el laberinto, hemos creado una forma para identificar a cada robot usando su "USERNAME" en mayúsculas como su propio identificador.

Reemplace los de la columna izquierda por los de la derecha:

`Is_There_Wall()`	`Is_There_Wall( USERNAME )`
`Forward();`	`Forward( USERNAME );`
`Robot( COUNTER_CW );`	`Robot( USERNAME, COUNTER_CW );`
`Sonar(CLOCK_WISE );`	`Sonar( USERNAME, CLOCK_WISE );`

SALVANDO SU *FSM* EN UN ARCHIVO INDEPENDIENTE:

Luego de los cambios propuestos, se salvan los códigos como si fueran "archivos de bibliotecas" independientes.

Salve el archivo con letras minúsculas usando como nombre su "username" y poniéndole la extension ".h".

EXTRACCIÓN DEL CICLO DE REPETICIÓN:

Con el cambio previo, CADA VEZ QUE LLAMA A LA FUNCIÓN se realiza una sola pasada para evaluar una ÚNICA transición entre estados. Para compensar la etapa previa DEBE RE-UBICAR el ciclo de repetición y hacerlo POR FUERA DE LA FUNCIÓN.

Generalmente el ciclo "while()" extraido DEBE SER EL ÚNICO CICLO EXISTENTE en el *software* de un sistema de computador "embebido", el cual ES COMPARTIDO por todas las máquinas que operan de manera interdependiente en su sistema.

Sus monitores incluirán la función de la Máquina que Usted preparó para operar de manera concurrente junto con las Máquinas de otros concursantes de la siguiente manera (*lo siguiente NO TIENE QUE HACERLO USTED*):

```
int
main()
{
Testing_Maze();

while( Running() ) {
    username1_robot();
    username2_robot();
    username3_robot();
    username4_robot();
}

Challenge_Was_Finished();

} /* main */
```

OBSERVACIONES:

- Entregue su código antes de la hora que especifique el grupo de monitores.
- En la eliminatorias, tanto el orden en que se ejecutan las *FSM* como las posiciones serán adjudicadas por sorteo.
- Su Robot no puede pasar "por encima" de los otros, es decir, para cada robot, TODOS los demás competidores serán considerados otro obstáculo más. La nueva función: `Is_There_Wall( USERNAME )` se encargará de ésto.
- El concurso termina cuando EL PRIMER ROBOT salga del laberinto.