28 de abril de 2019

Laura Aguilera Checa

Ignacio Aguilera Gómez

Grado en ingeniería informática

Universidad de Almería

Ejercicios 11 y 12

Sistemas de tiempo real

Contenido

[Autores 2](#_Toc8755841)

[Ejercicio 11. Implementación de un ejecutivo cíclico en ADA. El juego de las carreras de caballos 2](#_Toc8755842)

[1. Planificación con ejecutivo cíclico 3](#_Toc8755843)

[2. Implementación en ADA 4](#_Toc8755844)

[Ejercicio 12. Herencia de prioridad 4](#_Toc8755845)

[1. Herencia de prioridad 4](#_Toc8755846)

[2. Techo de prioridad 5](#_Toc8755847)

# Autores

|  |  |  |
| --- | --- | --- |
| Apellidos | Aguilera | Checa |
| Nombre | Laura |  |
| Titulación | Grado de Ingeniería Informática | |

|  |  |  |
| --- | --- | --- |
| Apellidos | Aguilera | Gómez |
| Nombre | Ignacio |  |
| Titulación | Grado de Ingeniería Informática | |

# Ejercicio 11. Implementación de un ejecutivo cíclico en ADA. El juego de las carreras de caballos

En la Figura 1 se muestra el diagrama del juego de las carreras de caballos. En dicho juego, N caballos compiten por su respectiva calle para llegar el primero a la meta.

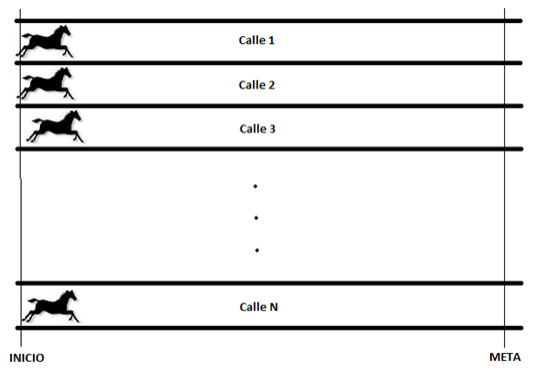


Figura 1. Juego de las carreras de caballos

El programa debe de poseer las siguientes características:

* El número de competidores (Nh) puede variar entre cada carrera (puede ser considerado como una variable constante o como un parámetro de entrada).
* La longitud del circuito (L) puede variar entre cada carrera y puede ser implementado como N.
* Cada caballo (H[i]) deberá de poseer un nombre.
* Para simular el movimiento de H[i], se utiliza un dado (D) de Nd caras.
* Cada caballo se moverá secuencialmente siguiendo una planificación estática del juego.
* Existe un comentarista de las carreras (C) que se encarga de anunciar la situación de la carrera después de cada ronda de tiradas (una tirada de cada caballo).
* Existen, por tanto, Nh + 1 tareas en ejecución para cada tarea (cada caballo Th[i] y el comentarista Tc).
* El tiempo de cómputo (Ch[i]) de cada tarea Th[i] tiene que ser igual para cada caballo (y elegido por el programador). Por tanto, el tiempo de cómputo de una ronda completa de tiradas puede ser calculado como: ∑Ch[i].
* El tiempo de cómputo (Cc) de la tarea Tc es elegido también por el programador y debe ser simulado utilizando retrasos. NOTA: es lógico asumir que el tiempo para la retransmisión de la carrera sea menor que el de una tirada (Cc < Ch[i]).
* El periodo (Ph[i]) de cada tarea Th[i] tiene que ser igual para cada caballo (y elegido por el programador) y debe de ser simulado utilizando retrasos. NOTA: este periodo no puede ser seleccionado arbitrariamente para poder garantizar los plazos de ejecución.
* El periodo (Pc) de la tarea Tc es elegido también por el programador. NOTA: este periodo también debe de satisfacer las respectivas condiciones para garantizar plazos de ejecución.

## 1. Planificación con ejecutivo cíclico

Siendo N el número de caballos que participan en la carrera (puede variar), cada caballo se referenciará como H. Como el periodo para todos los caballos es el mismo (TH), y dado que el comentarista debe esperar a que tiren todos los caballos para comentar las posiciones, el periodo para todas las tareas será TH y, por tanto, H = TH.

Por otra parte, establecemos el tiempo de cómputo para cada caballo a 10 ms y el de el comentarista a 5 ms, de forma que, para que de tiempo a que tiren todos los caballos en un ciclo, deberemos calcular TH en función al tiempo de cómputo y al número de caballos, siendo el valor de esta 10 x N + 5. De este modo, el tiempo de cada ciclo se usa al completo independientemente del número de caballos que participen. En la siguiente tabla se muestran el periodo y tiempo de cómputo para un caso general en el que se desconoce N.

|  |  |  |
| --- | --- | --- |
| Tarea | T | C |
| H[1] | TH | 10 |
| H[2] | TH | 10 |
| … | | |
| H[N] | TH | 10 |
| C | TH | 5 |
| TH = 10 x N + 5 | | |
| H = TH | | |

Para poder mostrar un ejemplo más práctico, supongamos que el N = 4:

|  |  |  |
| --- | --- | --- |
| Tarea | T | C |
| H[1] | 45 | 10 |
| H[2] | 45 | 10 |
| H[3] | 45 | 10 |
| H[4] | 45 | 10 |
| C | 45 | 5 |
| H = 45 | | |

La planificación de tareas para cada ciclo sería la siguiente:

## 2. Implementación en ADA

|  |
| --- |
| with Ada.Real\_Time; use Ada.Real\_Time; -- paquete para trabajar con tiempo (periodo) |
| with Ada.Text\_IO; use Ada.Text\_IO; |
| with Ada.Numerics; use Ada.Numerics; |
| with Ada.Numerics.Discrete\_Random; |
| with Ada.Strings.Unbounded; use Ada.Strings.Unbounded; |
| with Ada.Strings.Fixed; use Ada.Strings.Fixed; |
| with Ada.Strings; use Ada.Strings; |
| with Ada.Integer\_Text\_IO; use Ada.Integer\_Text\_IO; |
|  |
| procedure Ejercicio1 is |
| T : Time; |
| Nh : Integer; |
| L : Integer; |
| H : Time\_Span; |
| Ch : Time\_Span := Milliseconds (10); |
| Cc : Time\_Span := Milliseconds (5); |
|  |
| Dado : Integer; |
| newPos : Integer; |
|  |
| type listaNombres is array(Integer range <>) of Unbounded\_String; |
| type listaPosiciones is array(Integer range <>) of integer; |
|  |
| function Generate\_Number (MinValue : Integer; |
| MaxValue : Integer) return Integer |
| is |
| subtype Random\_Type is Integer range MinValue .. MaxValue; |
| package Random\_Pack is new Ada.Numerics.Discrete\_Random (Random\_Type); |
| G : Random\_Pack.Generator; |
| begin |
| Random\_Pack.Reset (G); |
| return Random\_Pack.Random (G); |
|  |
| end Generate\_Number; |
|  |
| task type Horse is |
| entry Tirada (t\_newPos : out Integer); |
| entry CargarPosicion (t\_pos : in Integer); |
| entry VolverAlEstablo; |
| end Horse; |
|  |
| task body Horse is |
| pos : integer := 0; |
| begin |
| loop |
| select |
| accept Tirada (t\_newPos : out Integer) do |
| Dado := Generate\_Number (1, 6); |
| --Put\_Line (Integer'Image (Dado)); |
| pos := pos + Dado; |
| t\_newPos := pos; |
| end Tirada; |
| or |
| accept CargarPosicion (t\_pos : in integer) do |
| --Put\_Line ("Cargar pos: " & Integer'Image (t\_pos)); |
| pos := t\_pos; |
| end CargarPosicion; |
| or |
| accept VolverAlEstablo; |
| exit; |
| end select; |
| end loop; |
| end Horse; |
|  |
| type Ref\_Horse is access Horse; |
|  |
| task Com is |
| entry Comentar (t\_nombre : Unbounded\_String; t\_pos : Integer); |
| entry Ganador (t\_nombre : Unbounded\_String); |
| entry VolverACasa; |
| end; |
| task body Com is |
| begin |
| loop |
| select |
| accept Comentar (t\_nombre : Unbounded\_String; t\_pos : Integer) do |
| Put\_Line ("El caballo " & To\_String(t\_nombre) & " va en posicion " & Integer'Image(t\_pos) & "!"); -- aquí irán las acciones de la tarea |
| end Comentar; |
| or |
| accept Ganador (t\_nombre : Unbounded\_String) do |
| Put\_Line ("Y el ganador es " & To\_String(t\_nombre) & "!!!!"); |
| end Ganador; |
| or |
| accept VolverACasa; |
| exit; |
| end select; |
| end loop; |
| end Com; |
|  |
| Caballito : Ref\_Horse; |
|  |
| begin |
| T := Clock; |
|  |
| Caballito := new Horse; |
|  |
| put\_Line ("Numero de caballos?"); |
| Ada.Integer\_Text\_IO.get(Nh); |
|  |
| put\_Line ("Longitud de la pista?"); |
| Ada.Integer\_Text\_IO.get(L); |
|  |
| H := Ch \* Nh + Cc; |
|  |
| declare |
| listaNom : listaNombres (0 .. Nh); |
| listaPos : listaPosiciones (0 .. Nh); |
|  |
| begin |
|  |
| -- Inicializar caballos |
| for i in Integer range 1 .. Nh loop |
| listaPos(i) := 0; |
| listaNom(i) := To\_Unbounded\_String ("Caballo\_" & Integer'Image (i)); |
| end loop; |
|  |
| -- loop de tareas |
| For\_Loop\_IteracionPrincipal : |
| loop |
| T := T + H; |
| delay until (T); |
|  |
| For\_Loop\_Tiradas : |
| for k in Integer range 1 .. Nh loop |
| T := T + Ch; |
| delay until (T); |
| --Put\_Line (To\_String(listaNom(k))); |
| --Put\_Line ("Posicion pre tirada: " & Integer'Image(listaPos (k))); |
|  |
| Caballito.CargarPosicion (listaPos (k)); |
|  |
| --Put\_Line("Datos cargados"); |
| Caballito.Tirada(newPos); |
| listaPos (k) := newPos; |
| --Put\_Line ("Dado: " & Integer'Image(newPos)); |
| --Put\_Line ("Posicion tras tirada: " & Integer'Image(listaPos (k))); |
|  |
| if newPos >= L then |
| Com.Ganador (listaNom (k)); |
| exit For\_Loop\_IteracionPrincipal; -- when newPos >= L; |
| end if; |
|  |
| end loop For\_Loop\_Tiradas; |
|  |
| T := T + Cc; |
| delay until (T); |
| for j in Integer range 1 .. Nh loop |
| Com.Comentar (listaNom (j), listaPos (j)); |
| end loop; |
| Put\_Line ("-------------"); |
| end loop For\_Loop\_IteracionPrincipal; |
| end; |
| --mandarAlEstablo := true; |
| Caballito.VolverAlEstablo; |
| Com.VolverACasa; |
|  |
| end Ejercicio1; |

# Ejercicio 12. Herencia de prioridad

Dado el diagrama de tareas expuesto en la Figura 1:

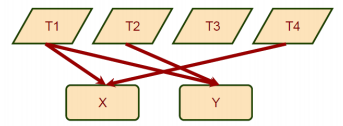
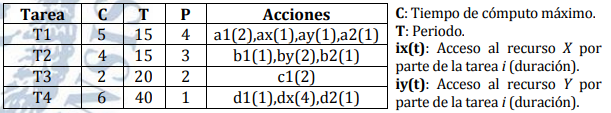


Figura 1. Diagrama de tareas para Ejercicio 12

* Realizar el cálculo de los tiempos de respuesta de las tareas a partir de la Tabla 1, suponiendo una estrategia de herencia de prioridad para la gestión de los recursos.
* Realizar el mismo ejercicio anterior suponiendo la gestión dinámica de prioridades mediante el protocolo de techo de prioridad.



## 1. Herencia de prioridad

B1 = by + dx = 6

B2 = dx = 4

B3 = dx = 4

B4 = 0

|  |  |
| --- | --- |
| T1 |  |
| T2 |  |
|  |
| T3 |  |
|  |
| T4 |  |
|  |

Para todas las tareas, el resultado final es menor que el periodo, por lo que se garantiza la ejecución de las tareas dentro de su plazo de ejecución.

## 2. Techo de prioridad

B1 = max(by, dx) = 4

B2 = dx = 4

B3 = dx = 4

B4 = 0

|  |  |
| --- | --- |
| T1 |  |
| T2 |  |
|  |
| T3 |  |
|  |
| T4 |  |
|  |

Una vez más, los resultados entran dentro del periodo, por lo que, al igual que en el apartado anterior, se garantiza la ejecución de todas las tareas dentro de su plazo de ejecución.