Cuadernillo de examen

Soluciones propuestas

Pregunta teórico-práctica

Programación estructurada. Teorema de la programación estructurada. Estructuras de control selectivas.
Apartados 4.3, 4.3.3 y 4.5 del libro de texto

Se desea realizar una algoritmo que escriba por pantalla el factorial de un número entero mayor o igual que 0 introducido por teclado. Implemente Una versión del algoritmo por cada una de las estructuras repetitivas que conozca.

```
…
var
   entero : i,n,fact
…
//Con estructura mientras
leer(n)
fact ← 1
i ← 2
mientras i <= n hacer
   fact ← fact * i
   i ← i + 1
fin_mientras
escribir(fact)

//Con estructura repetir
leer(n)
fact ← 1
i ← 1
si n <> 0 entonces
   repetir
      fact ← fact * i
      i ← i + 1
   hasta_que fact > n
fin_si
escribir(fact)

//Con estructura desde
leer(n)
fact ← 1
desde i ← 2 hasta n hacer
   fact ← fact * i
fin_desde
escribir(fact)
```

Problema

La V Edición de la Vuelta Ciclista a los Campos Manchegos consta de 5 etapas en las que participan 120 ciclistas. Los resultados de cada etapa se almacenan en un array de 5 filas y 120 columnas en el que cada elemento contiene el tiempo en segundos que ha tardado el corredor en realizar la etapa. Si un corredor no ha finalizado una etapa aparecerá un 0 en el elemento correspondiente. Los corredores se almacenan en un array de 120 posiciones en el que se almacenará el dorsal del corredor y el tiempo total realizado.

Cuadernillo de examen

Array con los tiempos

Etapas \ Números de participante	1	2	3	4	...	120
1						
2						
3						
4						
5						

Array con los ciclistas

	Número	Tiempo
1		
2		
3		
4		
5		
6		
7		
...		
120		

Se desea:

- Definir las estructuras de datos necesarias para realizar la aplicación.
- Diseñar un módulo que obtenga, a partir de un número de etapa que se pasará como argumento, el dorsal del ganador de la etapa.
- Diseñar un procedimiento que obtenga la clasificación general individual al término de la carrera. La clasificación deberá estar ordenada de menor a mayor por el tiempo total de cada participante y deberá mostrar el número y el tiempo total obtenido. En la clasificación general sólo aparecerán los corredores que hayan finalizado todas las etapas.

Definición de las estructuras de dato

```
tipos
   registro = ciclista
      entero : numero, tiempo
   fin_registro
   array[0..120] de ciclista : ciclistas
   array[1..5,1..120] de entero : tabla
```

Procedimiento GanadorEtapa

```
entero : función GanadorEtapa(E tabla : t; E entero : n,etapa)
var
   entero : j,min
inicio
   min ← 1
   //Sólo se tienen en cuenta los corredores que hayan llegado a meta
   //Si un corredor no ha finalizado la etapa su tiempo será 0

   //Buscar el primer corredor que haya finalizado la etapa
   //min se inicializa con el primer corredor que haya terminado la etapa
   mientras t[etapa,min] <> 0 hacer
      min ← min + 1
   fin_mientras

   //Buscar el menor a partir de aquí
   desde j ← min + 1 hasta n hacer
      si t[etapa,j] < t[etapa, min] entonces
          min ← j
      fin_si
   fin_desde
   devolver(min)
fin_función
```

Procedimiento Clasificación

```
procedimiento Clasificación(E tabla : t; E/S ciclistas : v;
                            E/S entero : NumCiclistas;
                            E entero : m,n)
//El procedimiento devuelve en NumCiclistas con el número de ciclistas
//que han acabado la vuelta
var
   entero : i,j, suma //Suma de los tiempos de un ciclista
inicio
   //Cargar el array con los tiempo totales
   //Sólo se tienen en cuenta los corredores que hayan finalizado todas las etapas
   //Cuando un corredor finaliza una etapa se rellena
   //un elemento del array de corredores
   NumCiclistas ← 0
   desde j ← 1 hasta n hacer //Hasta número de corredores
      suma ← 0
      i ← 0
      //Sumar los tiempos de cada etapa
      repetir
         i ← i + 1
         //Si el ciclista j ha terminado la etapa i
         si t[i,j] <> 0 entonces
            suma ← suma + t[i,j]
         fin_si
      //Repetir hasta que se encuenta un ciclista que no haya acabado una etapa o
      //se haya llegado al final de la vuelta
      hasta_que (t[i,j] = 0) o (i = m)

      //Si ha acabado todas las etapas se incrementa el número de ciclistas que han
      //finalizado la vuelta (NumCiclistas), se rellena el número del ciclista (j)
      //y el tiempo efectuado (suma)
      si t[i,j] <> 0 entonces
         NumCiclistas ← NumCiclistas + 1
         v[NumCiclistas].numero ← j
         v[NumCiclistas].tiempo ← suma
      fin_si
   fin_desde

   //Ordenar el array por el método de inserción directa
   desde i ← 2 hasta NumCiclistas hacer
      v[0] ← v[i]
      j ← i - 1
      mientras v[i].tiempo > v[0].tiempo hacer
          v[j+1] ← v[j]
          j ← j - 1
      fin_mientras
      v[j] ← v[0]
   fin_desde
fin_procedimiento
```