Cuadernillo de examen

Solución propuesta

Pregunta teórico-práctica

Programación modular. Concepto de módulo. Distintos tipos de módulos. Intercambio de información (paso de parámetros) entre los módulos y el programa principal.

Apartados 5.1, 5.2, 5.3 y 5.5 del libro de texto

Se desea realizar un módulo que devuelva el valor de la serie $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$. El número de terminos de la serie será un entero que se pasará como argumento al módulo. Implemente dos versiones del módulo, una como un procedimiento y otra como una función.

Como procedimiento

```
procedimiento serie(E entero : x, n ; E/S real : resultado)
var
   entero : i, factorial
   lógico : sumar
inicio
   sumar ← falso
   factorial ← 1
   resultado ← x
   desde i ← 3 hasta n incremento 2 hacer
       //Cuando i vale  se factorial será 2 * 3
       //cuando i vale 5 factorial será factorial * 5 * 4
       factorial ← factorial * i * i-1
       si sumar entonces
           resultado ← resultado + (x^1) / factorial
       si_no
           resultado ← resultado - (x^1) / factorial
       fin_si
       sumar ← no sumar
   fin_desde
fin_procedimiento
```

Como función

```
función serie(E entero : x, n)
var
   entero : i, factorial
   lógico : sumar
   real : resultado
inicio
   sumar ← falso
   factorial ← 1
   resultado ← x
   desde i ← 3 hasta n incremento 2 hacer
       //Cuando i vale  se factorial será 2 * 3
       //cuando i vale 5 factorial será factorial * 5 * 4
       factorial ← factorial * i * i-1
       si sumar entonces
           resultado ← resultado + (x^1) / factorial
       si_no
           resultado ← resultado - (x^1) / factorial
```

```
            fin_si
            sumar ← no sumar
        fin_desde
        devolver(resultado)
fin_función
```

Problema

La edición 2002 del Rally Arras-Madrid-Dakar mantiene los tiempos efecturados por los participantes en cada etapa en un array de dos dimensiones. En dicho array las filas corresponden a cada una de las etapas y las columnas a cada uno de los corredores, de forma que, por ejemplo, en el elemento 5,10 del array se almacenaría en segundos el tiempo que ha tardado el participante número 10 en la etapa 5.

La prueba tiene quince etapas y el número de participantes es de 120.

Los datos de los participantes se almacenan en un array de registros en el que consta el **número** del participante, el **nombre** y el **equipo** al que pertenece. Cada registro, además, cuenta con un campo numérico vacío (**tiempo**) deberá rellenar la aplicación.

Array con los tiempos

Etapas \ Números de participante	1	2	3	4	...	120
1						
2						
3						
...						
15						

Array con los participantes

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

Se desea:

- Definir las estructuras de datos necesarias para realizar la aplicación.
- Diseñar un procedimiento que obtenga la clasificación general individual al término del Rally. La clasificación deberá estar ordenada de menor a mayor por el tiempo total de cada participante y deberá mostrar el número, el nombre, el equipo y el tiempo total obtenido.
- Diseñar un procedimiento que obtenga, a partir de un número de etapa que se pasará como argumento, la clasificación individual de dicha etapa. La clasificación deberá estar ordenada de menor a mayor por el tiempo total de cada participante y deberá mostrar el número, el nombre, el equipo y el tiempo obtenido por el participante en la etapa..

Definición de las estructuras de dato

```
tipos
   registro = participante
      entero : número, tiempo
      cadena : nombre, equipo
   fin_registro
   array[0..120] de participante = participantes
   array[1..15,1..120] de entero : tabla
```

Clasificación individual

```
procedimiento ClasificaciónIndividual(E tabla : tiempos;
                                      E/S participantes : v; E entero : m,n)
var
   entero : i,j
```

```
inicio
   //Obtener la suma de tiempos de cada participante
   desde j ← 1 hasta n hacer //Hasta el número de participantes
      v[j].tiempo ← 0
      desde i ← 1 hasta m hacer //Hasta el número de etapas
          v[j] ← v[j] + tiempos[i,j]
      fin_desde
   fin_desde

   //Ordenar la tabla por inserción directa
   desde i ← 2 hasta n 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
```

Clasificación de una etapa

```
procedimiento ClasificaciónEtapa(E tabla : tiempos;
                                 E participantes : v;
                                 E/S participantes : Clasif E entero : m,n, etapa)
var
   entero : i,j, pos, min
inicio
   //Se copia todo el array de participantes en el array Clasif
   //para tener los datos de todos los participantes
   Clasif ← v

   //Rellenar el campo tiempo del array Clasif por el tiempo obtenido
   //en la etapa etapa
   desde j ← 1 hasta n hacer //Hasta el número de participantes
      //Buscar la posición del corredor número j en el array Clasif
      pos ← Buscar(Clasif,n,j)

      Clasif[pos].tiempo ← tiempos[etapa,j]
   fin_desde


   //Ordenar la tabla por el método de selección directa
   desde i ← 1 hasta n-1 hacer
      min ← 1
      desde j ← i hasta n hacer
         si Clasif[j].tiempo < Clasif[min].tiempo entonces
           min ← j
         fin_si
      fin_desde
      intercambio(Clasif[i],Clasif[min])
   fin_desde
fin_procedimiento


//Función de búsqueda con centinela
entero: función Buscar(E participantes:v;E entero:n,elemento)
//El vector v tiene índices entre 0 y MaxEl
var
   entero: i
inicio
```

```
   i ← n
   v[0] ← elemento
   mientras elelemento <> v[i].numero hacer
      i ← i - 1
   fin_mientras
fin_función


//Procedimiento de intercambio
procedimiento Intercambio(E/S participante : a,b)
var
   participante : aux
incio
   aux ← a
   a ← b
   b ← aux
fin_procedimiento
```