Cuadernillo de examen

ASIGNATURA:	**Fundamentos de Programación I**	CÓDIGO:	**106**
CONVOCATORIA:	**Febrero 2008**	PLAN DE ESTUDIOS:	**2000/2002**
CURSO:	**1º**	CURSO ACADÉMICO:	**2007/08**
TURNO:	**Mañana**	TITULACIÓN:	**Ingeniería Informática** **Ingeniería Técnica en Informática**
CARÁCTER:	**Cuatrimestral**	ESPECIALIDAD:	**Común**
DURACIÓN APROXIMADA:	**2 horas y media**		

SOLUCIONES PROPUESTAS

Preguntas teórico-prácticas

1. Enunciado del Teorema de la Programación Estructurada. Enumere y explique además las tres estructuras de control repetitivas básicas.

Apartados 2.3.4, 5.2, 5.4 y 5.5 del libro de texto

Aplicación

Codifique **una función** que calcule la suma de los dígitos de un número de 5 cifras que se pasará como **argumento**. Codifique **un procedimiento** permita leer por teclado una serie de caracteres. La lectura terminará cuando el carácter introducido sea '0' (cero). El procedimiento deberá devolver al programa que lo llamó el número de vocales que se introdujeron. Por ejemplo, si la serie de caracteres introducida es 'a', 'x', 'h', 'e', 'a', 'v', '0', el procedimiento deberá devolver el valor 3.

```
entero función SumaDígitos(valor entero : n)
var
   entero : suma
inicio
   suma ← 0
   mientras n > 10 hacer
      //n mod 10 es el último dígito del número
      suma ← suma + n mod 10
      //A n se le elimina el último dígito
      n ← n div 10
   fin_mientras
   devolver(suma + n)
fin_función

procedimiento ContarVocales(ref entero : n)
var
   carácter : c
inicio
   n ← 0
   leer(c)
   mientras c <> '0' hacer
      si posición('aeiou',c) <> 0 entonces
         n ← n +1
      fin_si
      leer(c)
   fin_mientras
fin_procedimiento
```

Puntuación: 1, 5 puntos

2. La estructura de datos array: definición y tipos de arrays. ¿Qué instrucciones del lenguaje algorítmico UPSAM se pueden utilizar con un array? ¿Y con los elementos de un array? Almacenamiento de arrays de una y de dos dimensiones en memoria

Aplicación

Se desea almacenar la información de 1000 de alumnos de una universidad. Por cada alumno se almacenará su número expediente y las 36 asignaturas de las que consta la carrera. Por cada asignatura se almacena el código de asignatura, la nota obtenida por el alumno en el último examen al que se ha presentado y fecha de dicho examen.

- ❑ Diseñe y codifique las estructuras de datos que permitan almacenar dicha información.
- ❑ Codifique un procedimiento que permita ordenar los alumnos en orden ascendente de expediente por el método de inserción directa.

```
tipos
   registro = asignatura
      cadena : código, fechaExamen
      real : nota
   fin_registro

   registro = alumno
      cadena : expediente
      array[1..36] de asignatura : nota
   fin_registro

   array[0..1000] de alumno = alumnos

procedimiento OrdenaciónInserciónDirecta(ref alumnos:v ;valor entero : n)
var
   entero : i,j
inicio
   desde i ← 2 hasta n hacer
      v[0] ← v[i]
      j ← i - 1
      mientras v[j].expediente > v[0].expediente hacer
          v[j+1] ← v[j]
          j ← j - 1
      fin_mientras
      v[j+1] ← v[0]
   fin_desde
fin_procedimiento
```

Puntuación: 2 puntos

3. Concepto de cadena. Tipos de cadenas según su almacenamiento en memoria. Operaciones con cadenas.

Aplicación

Codifique un subprograma que devuelva en un array de cadena cada una de las palabras de una frase que se pasará como argumento. La frase tendrá un máximo de 10 palabras y cada palabra estará separada de la siguiente por un espacio en blanco. Por ejemplo, si la frase es "Hoy es el examen de fundamentos", deberá devolver el array

Hoy	es	el	examen	de	fundamentos

```
cadena[] función PalabrasSeparadas(valor cadena : c)
var
   array[1..10] de cadena : palabras
   entero : i,p
inicio
   i ← 0
   //p es la posición del primer blanco,
   //la primera palabra serán todos los caracteres hasta ese primer blanco
   p ← posición(c,' ')
   //Mientras no existan más palabras,
   mientras p <> 0 hacer
      //Hay una palabra más,luego i(el contador de palabras)se incrementa en 1
      i ← i + 1
```

```
        //La primera palabra serán todos los caracteres desde el primero
        //hasta el carácter anterior al primer blanco
        palabras[i] ← subcadena(c,1,p-1)
        //Una vez sacada la palabra, la frase se queda con todos los
        //caracteres después del primer blanco (p+1)
        c ← subcadena(c,p+1)
        p ← posición(c,' ')
    fin_mientras
    //Una vez que ya no quedan blanco (o si la frase sólo tiene 1 palabra)
    //se añade al array la última palabra
    i ← i + 1
    palabras[i] ← c
    devolver(palabras)
fin_función
```

Puntuación: 1, 5 puntos

Preguntas prácticas

Un club deportivo alquila a sus socios las pistas de tenis de las que dispone desde las 9 de la mañana a las 9 de la noche. El club tiene 15 pistas y las alquila por periodos de una hora Diariamente se realiza una estadística de esos alquileres para lo cual almacena la información en una tabla de 15 filas (una por pista) y 13 columnas (una por cada periodo de una hora). En cada elemento de la tabla se guardará el número del socio que realizó el alquiler, si la pista no está alquilada en ese horario se guardará el valor 0. Además, en un array de registros almacena, por cada socio, el número de socio y número de alquileres que ha realizado hasta la fecha. En la actualidad el club dispone de 500 socios

Alquileres de pistas

Pistas \ Horarios	1	2	3	4	5	6	7	8	9	10	11	12	13
1													
2													
3													
4													
…													
15													

Socios

	Nº Socio	Nº Alquileres
1		
2		
3		
4		
5		
…		
500		

Se pide:

a) Defina y declare **todas** las estructuras de datos necesarias para realizar los distintos puntos que aparecen a continuación.

```
tipos
    array[1..15,1..13] de entero = alquileres

    registro = socio
        entero : nSocio, nAlquileres
    fin_registro
    array[1..500] de  socio = socios
```

Puntuación: 0, 5 puntos

b) Codifique un subprograma que permita alquilar una pista a un socio. El número de socio y la franja horaria (un número de 1 a 13) se pasarán como argumentos al procedimiento. El subprograma también deberá informar si no hay pistas libres en ese horario.

```
procedimiento Alquilar(ref alquileres : a; entero : socio, hora)
var
    entero : i
inicio
    i ← 1
    //Se recorre la columna hora hasta encontrar una pista (i) libre
    mientras (a[i,hora] <> 0) o (i < 15) hacer
        i ← i + 1
```

```
      fin_mientras
      //Si no se ha encontrado ninguna pista libre,
      //es que el último elemento de la tabla en esa hora es 0
      si a[i,hora] <> 0 entonces
         escribir('No hay pistas disponibles a esa hora')
      si_no
         //Si se ha encontrado una pista libre,
         //se mete en la posición i,hora al socio que quiere alquilar
         a[i,hora] ← socio
      fin_si
   fin_procedimiento
```

Puntuación: 1 punto

c) Codifique un subprograma que determine en qué franja horaria hay más pistas sin alquilar.

```
   entero función horaMásDesocupada(valor alquileres: a)
   var
      entero : i,máx,suma,horaMax
   inicio
      //Se inicializa el número mayor de pistas libres (máx) a 0
      máx ← 0
      desde j ← 1 hasta 13 hacer
         //Se acumula el número de pistas libres en la hora j
         suma ← 0
         desde i ← 1 hasta 15 hacer
            si a[i,j] = 0 entonces
               suma ← suma + 1
            fin_si
         fin_desde
         // Si el número de pistas libres de la hora j es mayor que el máximo...
         si suma >= máx entonces
            //La hora con más pistas libres es j
            horaMáx ← j
         fin_si
      fin_desde
      devolver(horaMáx)
   fin_función
```

Puntuación: 1 puntos

d) Codifique un subprograma que actualice el array de socios, incrementando el número de alquileres que ha realizado al valor que se almacenaba en el campo "Nº de Alquileres".

```
   procedimiento ActualizarSocios(valor alquileres: a; ref socios: s)
   var
      entero : i,j
   inicio
      //Se recorre toda la tabla de alquileres
      desde i ← 1 hasta 15 hacer
         desde j ← 1 hasta 13 hacer
            //Por cada pista ocupada...
            si a[i,j] <> 0 entonces
               //Se busca al socio...
               p ← Buscar(socios, v[i,j],500)
               //y, una vez encontrado, se incrementan sus alquileres
               socios[p].nAlquileres ← socios[p].nAlquileres + 1
            fin_si
         fin_desde
      fin_desde
   fin_procedimiento
```

```
entero función Buscar(valor socios:s;valor entero:el;valor entero:n)
var
   entero: i
inicio
   i ← 1
   mientras (el <> s[i].nSocio) y (i < n) hacer
      i ← i + 1
   fin_mientras
   si el = s[i].nSocio entonces
      devolver(i)
   si_no
      devolver(0)
   fin_si
fin_función
```

Puntuación: 1,5 puntos

e) Codifique un subprograma que permita ordenar el array de los socios de forma descendente por su número de alquileres.

```
procedimiento OrdenaciónShell(ref socios:s; valor entero : n)
var
   entero : i,j,incr
inicio
   incr ← n div 2
   mientras incr > 0 hacer
      desde i ← incr + 1 hasta n hacer
         j ← i - incr
         mientras j > 0 hacer
            si s[j].nAlquileres < v[j + incr].nAlquileres hacer
               intercambiar(s[j], s[j + incr])
               j ← j - incr
            si_no
               j ← 0
            fin_si
              fin_mientras
           fin_desde
           incr ← incr div 2
   fin_mientras
fin_procedimiento
```

Puntuación: 1 punto