Cuadernillo de examen

ASIGNATURA:	**Fundamentos de Programación I**		
CONVOCATORIA:	**Febrero 2012**	PLAN DE ESTUDIOS:	**2010**
CURSO:	**1º**	CURSO ACADÉMICO:	**2011/2012**
TURNO:		TITULACIÓN:	**Grado en Ingeniería Informática**
CARÁCTER:	**Obligatoria**	ESPECIALIDAD:	**Común**
DURACIÓN APROXIMADA:	**2 horas y media**		

Soluciones propuestas al examen

Preguntas teórico-prácticas

1. Programación estructurada. Explicar sus características fundamentales. Describir, utilizando diagramas de flujo, los distintos tipos de estructuras de control que conozca.

Documentación del tema 3. Diapositivas 3-5 y 7-55.

Aplicación

Diseñe los siguientes algoritmos, utilizando la estructura de control que considere más adecuada:

- Calcular la retención del IRPF a partir de la base imponible que se leerá por teclado según el siguiente criterio:
  - De 0 a 17.707 €, la retención es del 24,75%.
  - De 17.707 a 33.007€, la retención es del 30%.
  - De 33.007 hasta 53.407€, la retención es del 40%.
  - De 53.407 hasta 120.000€, la retención es del 47%.
  - De 120.000 hasta 175.000€, la retención es del 49%.
  - De 175.000 a 300.000€, la retención es del 52%.
  - Más de 300.000€, la retención es del 52%.

```
algoritmo Retención
var
   real: base, retención
inicio
   leer(base)
   si base < 17707 entonces
      retención ← 0.2475
   si_no
      si base <= 33007 entonces
         retención ← base * 0.3
      si_no
         si base <= 53407 entonces
            retención ← base * 0.4
         si_no
            si base <= 120000 entonces
               retención ← base * 0.47
            si_no
               si base <= 175000 entonces
                  retención ← base * 0.49
               si_no
                  si base <= 300000 entonces
                     retención ← base *0.52
                  si_no
                     retención ← base * 0,52
                  fin_si
               fin_si
            fin_si
```

```
            fin_si
         fin_si
      fin_si
      escribir(retención)
   fin
```

- Obtener el nombre una provincia a partir de su código que se introducirá por teclado. Los códigos de provincias posibles serían: 3 para Almería, 8 para Barcelona, 13 para Ciudad Real, 28 para Madrid, 26 para La Rioja y 50 para Zaragoza

```
algoritmo CódigoProvincia
var
   entero: código
   cadena: provincia
inicio
   leer(código)
   según_sea código hacer
      3  : provincia ← 'Almería'
      8  : provincia ← 'Barcelona'
      13 : provincia ← 'Ciudad Real'
      28 : provincia ← 'Madrid'
      26 : provincia ← 'La Rioja'
      50 : provincia ← 'Zaragoza'
   fin_según
   escribir(provincia)
fin
```

Puntuación: 1,5 puntos

2. Subprogramas. Tipos de subprogramas: características y diferencias entre ellos. Paso de argumentos: características y diferencias entre el paso de argumentos por valor y por referencia.

Documentación del tema 4. Diapositivas 11-22 y 26-35

Aplicación

Se desea diseñar un subprograma que reciba como argumento un número entero y devuelva la suma de sus dígitos. Realizar dos versiones del subprograma, una como función y otra como procedimiento.

```
entero función SumaDígitos(valor entero: n)
var
   entero: suma
inicio
   suma ← 0
   mientras n > 0 hacer
      suma ← suma + n mod 10
      n ← n div 10
   fin_mientras
   devolver(suma)
fin_función

procedimiento SumaDígitos(valor entero: n; ref entero:suma)
inicio
   suma ← 0
   mientras n > 0 hacer
      suma ← suma + n mod 10
      n ← n div 10
   fin_mientras
fin_procedimiento
```

Puntuación: 1,5 puntos

3. 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? ¿Cómo se pasan los argumentos de tipo array a los subprogramas? Almacenamiento de arrays de una y de dos dimensiones en memoria

Documentación del tema 5. Diapositivas 4-12, 22-28, 44-45, 47-48 y 56-57

Aplicación

Se tiene almacenado en un array de registros de n elementos la información de una serie de personas. Por cada persona se almacena su DNI, su nombre y su edad.

- Definir las estructuras de datos necesarias para almacenar esa información.

```
const
   n = ...
tipos
   registro = persona
      cadena : dni, nombre
      entero : edad
   fin_registro
   array[1..n] de persona = personas
```

- Codificar un subprograma que obtenga el DNI de la persona de mayor edad del array.

```
cadena función MayorPersona(valor personas: v; valor entero : n)
var
   entero: mayor,i
inicio
   mayor ← 1
   desde i ← 2 hasta n hacer
      si v[i].edad > v[mayor].edad entonces
         mayor ← i
      fin_si
   fin_desde
   devolver(v[mayor].dni)
fin_función
```

Puntuación: 2 puntos

Preguntas prácticas

Un despacho de abogados guarda información de la facturación que han hecho sus miembros a lo largo de los 12 meses del año en un array de dos dimensiones de forma que en el elemento `i,j` del array se almacena el total de las minutas que ha facturado el abogado `i` en el mes `j`. En la actualidad en el despacho trabajan 10 abogados

Se pide:

1. Declarar las estructuras de datos necesarias para almacenar la información y realizar los puntos que aparecen a continuación.

```
tipos
   array[1..10, 1..12] de real = tabla
   array[1..10] de real = vector
```

Puntuación: 0,5 puntos

2. Codificar un subprograma que devuelva el total de horas que ha facturado el despacho a lo largo del año.

```
real función TotalHoras(valor tabla:t)
var
   real : total
```

```
   entero : i,j
inicio
   total ← 0
   desde i ← 1 hasta 10 hacer
      desde j ← 1 hasta 12 hacer
         total ← total + t[i,j]
      fin_desde
   fin_desde
   devolver(total)
fin_función
```

Puntuación: 1 punto

3. Codificar un subprograma que devuelva el total de horas que ha facturado el despacho en un mes determinado que pasará como argumento al subprograma.

```
real función HorasAbogado(valor tabla:t; valor entero : mes)
var
   real : total
   entero : i
inicio
   total ← 0
   desde i ← 1 hasta 10 hacer
         total ← total + t[i,mes]
   fin_desde
   devolver(total)
fin_función
```

Puntuación: 1 punto

4. Codificar un subprograma que rellene un vector de 10 elementos con el total que ha facturado cada abogado a lo largo del año, de forma que en el elemento 1 del vector aparezca el total de facturación del abogado 1, en el elemento 2 la facturación del abogado 2, etc.

```
procedimiento FacturaciónTotal(valor tabla:t; ref vector: totales)
var
   entero : i,j
inicio
   desde i ← 1 hasta 10 hacer
      totales[i] ← 0
      desde j ← 1 hasta 12 hacer
         totales[i] ← totales[i] + t[i,j]
      fin_desde
   fin_desde
fin_procedimiento
```

Puntuación: 1,5 puntos

5. Codificar un subprograma que obtenga el número del abogado que más ha facturado a lo largo del año.
Puntuación: 1 punto

```
entero función MayorFacturación(valor vector: totales)
var
   entero : i,max
inicio
   max ← 1
   desde j ← 1 hasta 12 hacer
      si totales[i] > totales[max] entonces
         max ← i
      fin_si
   fin_desde
   devolver(max)
fin_función
```