Cuadernillo de examen

ASIGNATURA:	**Fundamentos de Programación I**	CÓDIGO:	**106**
CONVOCATORIA:	**Febrero 2009**	PLAN DE ESTUDIOS:	**2000/2002**
CURSO:	**1º**	CURSO ACADÉMICO:	**2008/2009**
TURNO:	**Mañana**	TITULACIÓN:	**Ingeniería Informática** **Ingeniería Técnica en Informática**
CARÁCTER:	**Cuatrimestral**	ESPECIALIDAD:	**Común**
DURACIÓN APROXIMADA:	**2 horas y media**		

Soluciones propuestas

Nota importante: Los alumnos que hayan superado la evaluación continua, sólo tendrán que hacer las tres preguntas teórico prácticas. En ese caso, la puntuación de las preguntas teórico prácticas será de un punto cada una. Si no se ha superado la evaluación continua, se deberá realizar la totalidad del examen y la puntuación será la que aparece en cada enunciado.

Preguntas teórico-prácticas

1. Programación estructurada ¿en qué consiste? Teorema de la programación estructurada. Estructuras de control selectivas: explique las distintas estructuras de control selectivas, exponga sus diferencias y represente cada una de ellas mediante un diagrama de flujo.

Apartados 2.3, 4,3, 4.4 y 4.5 del libro de texto y apuntes de clase

Aplicación

Codifique y aplique la estructura selectiva más adecuada para resolver los siguientes problemas:

- Se desea codificar un algoritmo que permita averiguar el sueldo de un empleado a partir de su categoría. Se deberá proporcionar por teclado la categoría (un dato de tipo entero) y devolver el sueldo según el siguiente baremo: categoría 1 – 1250€, categoría 2 – 1500€, categoría 3 – 1750€, categoría 4 – 2100€, categoría 5 – 2500€.

```
algoritmo Pregunta1_1
var
   real : sueldo
   entero : categoría
inicio
   leer(categoría)
   según_sea categoría hacer
      1 : sueldo ← 1250
      2 : sueldo ← 1500
      3 : sueldo ← 1750
      4 : sueldo ← 2100
      5 : sueldo ← 2500
   fin_según
   escribir(sueldo)
fin
```

- Se desea codificar un algoritmo que permita averiguar la comisión de un vendedor a partir de las ventas efectuadas. Se deberá proporcionar por teclado las ventas (un dato de tipo real) y devolver la comisión según el siguiente baremo: entre 0 y 3000€ recibirá 0% de comisión, cuando las ventas sean mayores de 3000€ y menores 5400€ recibirá un 5% de comisión, cuando las ventas

sean mayores de 5400€ y menores de 6500€ recibirá un 10% de comisión, y si las ventas superan los 6500€ recibirá un 12% de comisión.

```
algoritmo Pregunta1_2
var
   real : ventas, comisión
inicio
   leer(ventas)
   si ventas < 3000 entonces
      comisión ← ventas
   si_no
      si ventas <5400 entonces
         comisión ← ventas * 0.05
      si_no
         si ventas < 6500 entonces
            comisión ← ventas * 0.1
         sin_no
            comisión ← ventas * 0,12
        fin_si
      fin_si
   fin_si
   escribir(comisión)
fin
```

Puntuación: 2 puntos

2. Programación modular. Ámbito de variables. Intercambio de información entre subprogramas. Paso de parámetros.

Apartados 6.3, 6.4 y 6.5 del libro de texto y apuntes de clase

Aplicación

Diseñe los dos módulos que aparecen a continuación. **Debe elegir el tipo de módulo más adecuado a cada caso**.

- Codifique un módulo que reciba una cantidad en segundos y lo convierta en horas, minutos y segundos.

```
procecimiento Pregunta2_1(valor entero:segundos;
                          ref real : hh,mm,ss)
inicio
   hh ← segundos div 3600
   mm ← segundos mod 3600 div 60
   ss ← segundos mod 3600 mod 60
fin_procedimiento
```

- Codifique un módulo que reciba el número de día y mes de una fecha del año 2009 y devuelva el número de días transcurrido desde el 1 de enero. Por ejemplo si el día es 3 y el mes 2, deberá devolver 34 (los 31 días de enero más los tres días que han pasado de febrero).

```
entero función númeroDeDías(valor entero : día, mes)
var
   entero : i,días
inicio
   días ← 0
   desde i ← 1 hasta mes – 1 hacer
```

```
        según_sea  i hacer
           2 : días ← días + 28
           4,6,9,11 : días ← días + 30
        si_no
           días ← días + 31
        fin_según
     fin_desde
     devolver(días + día)
fin_función
```

Puntuación: 1 punto

3. Ordenación de arrays. Enumere y describa los distintos métodos de ordenación que conozca.

Apartado 10.2 del libro de texto

Aplicación

Se desea almacenar en una estructura de datos interna la información sobre las clases impartidas por los N profesores de una Universidad. Por cada profesor se almacena información sobre su DNI, su nombre y las asignaturas que imparte (puede impartir hasta un máximo de 6 asignaturas). Por cada una de las asignaturas, además, se guardará el código de asignatura y el grupo dónde la imparte.

- Defina las estructuras de datos necesarias para almacenar dicha información.

```
tipos
   registro : asignatura
      cadena : código, grupo
   fin_registr
   registro : profesores
      cadena : DNI, nombre
      array [1..6] de asignatura : asignaturas
   fin_registro
   array[0..n] de profesor : profesores
```

- Codifique un procedimiento que permita ordenar el array por el nombre del profesor.

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

Puntuación: 2 puntos

Preguntas prácticas

Una empresa de transportes almacena en un array de registros los viajeros que ha tenido en cada uno de los trayectos que sirve a lo largo de un mes. Cada elemento del array almacena el código del trayecto, una descripción del trayecto y un array de enteros con una posición por cada día del mes y que almacena en número de viajeros de ese día.

				Días						
		Código	**Descripción**	**1**	**2**	**3**	**4**	**5**	**...**	**N**
Trayectos	**1**									
	2									
	3									
	...									
	M									

La empresa tiene M trayectos y el array está ordenado por el código del trayecto.
Se pide:

1. Definir las estructuras de datos necesarias para poder realizar todos los puntos del problema.
   Puntuación: 0,5 puntos

```
tipos
   registro : trayecto
      cadena : código, descripción
      array [1..31] de entero : días
   fin_registro
   array[0..n] de trayecto : trayectos

   registro : pasajeroDía
      entero : día, pasajeros
   fin_registro
   array[0..31] de pasajeroDía : pasajerosDía
```

2. Diseñe un subprograma que devuelva el trayecto que ha tenido más pasajeros,
   Puntuación: 1 punto

```
entero función trayectoConMásPasajeros(valor trayectos: t;
        valor entero : m,n)
var
   entero :i,j, máximo
   array[1..m] de entero : totalPasajerosFila
inicio
   //Se acumulan los pasajeros de cada trayecto
   desde i ← 1 hasta m hacer
      totalPasajerosFila[i] ← 0
      desde j ← 1 hasta n hacer
         totalPasajerosFila[i] ← totalPasajerosFila[i] + t[i].días[j]
      fin_desde
   fin_desde
   //Se calcula el máximo del array totalPasajerosFila
   máximo ← totalPasajerosFila[1]
   desde i ← 2 hasta m hacer
      si totalPasajerosFila[i] > totalPasajerosFila[máximo] entonces
         máximo ← i
      fin_si
```

```
   fin_desde
   devolver(máximo)
fin_fución
```

3. Diseñe un subprograma en el que, a partir de un código de trayecto y de un día de mes, devuelva el número de viajeros que ha tenido dicho trayecto en dicho día. Tanto el código de trayecto como el día de mes se deberán pasar como argumentos a dicho subprograma.
Puntuación: 1,5 puntos

```
entero : función buscarNúmeroPasajeros(valor trayectos: t;
                                valor código : cadena;
                                valor día,m : entero)
var
   entero : p //Posición del trayecto dentro del array
inicio
   p ← buscarTrayecto(t, código, m)
   devolver(t[p].días[día])
fin

entero función buscarTrayecto(valor trayectos:t;
                                valor código:el;
                                valor entero:n)
var
   entero: izq, der, cen
inicio
   izq ← 1
   der ← n
   repetir
      cen ← (izq + der) div 2
      si t[cen].código > el.código entonces
         der ← cen – 1
      si_no
         izq ← cen + 1
      fin_si
   hasta_que (t[cen].código = el.código) o (izq > der)
   si t[cen].código = el.código entonces
       devolver(cen)
   si_no
       devolver(0)
   fin_si
fin_función
```

4. Diseñe un subprograma que permita almacenar en otro array de registros el total de pasajeros por cada día del mes. Dicho array deberá tener N posiciones y, por cada elemento se almacenará el número del día y el total de pasajeros de ese día. El subprograma deberá devolver el array ordenado descendentemente por el número de pasajeros del día.
Puntuación: 2 puntos

```
procedimiento obtenerPasajerosDía(valor trayectos : t;
                                  valor entero : m,n ;
                                  ref pasajerosDía : p)
var
   entero : i,j
inicio
```

```
    desde j ← 1 hasta n hacer
       p[j].día ← j
       p[j].pasajeros ← 0
       desde i ← 1 hasta m hacer
          p[j].pasajeros ← [j].pasajeros + t[i].días[j]
       fin_desde
    fin_desde
    ordenar(p,n)
 fin_desde

procedimiento ordenar(ref pasajerosDía:v;valor entero : n)
var
   entero : i,j
   lógico : ordenado
inicio
   i ← 0
   repetir
      i ← i + 1
      ordenado ← verdad
      desde j ← n hasta i+1 incremento -1 hacer
         si v[j-1].pasajeros < v[j].pasajeros entonces
            intercambiar(v[j],v[j-1])
            ordenado ← falso
         fin_si
      fin_desde
   hasta_que ordenado
fin_procedimiento
```