Cuadernillo de examen

Asignatura:	**Fundamentos de Programación I**			Código:	**106**
Titulación:	**Ingeniería Informática / Ingeniería Técnica en Informática**			Plan de estudios:	**2000/2002**
Curso:	**1º**	Carácter:	**Obligatoria**	Especialidad:	**Común**
Convocatoria:	**Febrero 2007**	Curso académico:	**2006/2007**	Turno:	**Mañana**
Duración aproximada:	**2 horas y media**				

Solución propuesta

Preguntas teóricas

1. Estructuras selectivas. Enumere y explique el funcionamiento de las distintas estructuras de control selectivas. Describa gráficamente con un diagrama de flujo cada una de ellas. Describa en que casos es adecuado utilizar una estructura selectiva múltiple.

Capítulo 4 del libro de texto.

Aplicación

Se desea codificar un subprograma que reciba como argumento el código postal de alguna de las siguientes localidades de la comunidad de Madrid y devuelva el nombre de dicha localidad. Los códigos postales que puede recibir serán 28100 (Alcobendas), 28410 (Manzanares el Real), 28720 (Bustarviejo) y 28722 (El Vellón). Cualquier otro código postal recibido devolverá la cadena "Código postal erróneo".
Realice dos versiones de la función, una utilizando estructuras selectivas múltiples y otra utilizando estructuras selectivas dobles.

```
cadena función CódigosPostales(valor cadena : CP)
inicio
   si CP = "28100" entonces
      devolver("Alcobendas")
   si_no
      si CP = "28410" entonces
         devolver("Manzanares el Real")
      si_no
         si CP="28720" entonces
            devolver("Bustarviejo")
         si_no
            si CP="28722" entonces
               devolver("El Vellón")
            si_no
               devolver("Código postal erróneo")
            fin_si
         fin_si
      fin_si
   fin_si
fin_función

cadena función CódigosPostales(valor cadena : CP)
inicio
   según_se CP hacer
      "28100" : devolver("Alcobendas")
      "28410" : devolver("Manzanares el Real")
      "28720" : devolver("Bustarviejo")
      "28722" : devolver("El Vellón")
   si_no
      devolver("Código postal erróneo")
   fin_según
fin_función
```

Puntuación: 1,5 puntos

2. Estructuras de datos. ¿Qué es un array? ¿Cómo se referencian los elementos de un array? ¿Cómo se almacena los vectores y las tablas en memoria? ¿Qué es un registro? ¿Cómo se referencian los elementos de un registro? ¿Qué operaciones se pueden hacer con un registro? ¿Qué operaciones se pueden hacer con los elementos de un registro?

Apartados 7.2, 7.3, 7.6, 7.7. del libro de texto

Aplicación

Se desea almacenar en un array de registros información sobre una 1500 de personas. Por cada persona se almacenará su DNI, su nombre y apellidos, su edad y un registro anidado con información bancaria que contendrá el nombre de la entidad y el número de cuenta. El array está ordenado de forma ascendente por el número de cuenta.

- Realice la declaración del array de registros.
- Diseñe un subprograma que busque en el array de registros utilizando el método de búsqueda binaria un número de cuenta que se pasará como argumento. El subprograma **sólo buscará entre las posiciones 100 y 1000 del array**.

```
...
tipos
   registro : persona
      cadena : DNI, nombre, apellidos
      entero : edad
      registro : datosBancarios
         cadena : entidad, numCuenta
      fin_registro
   fin_registro

   array[0..1500] de persona : personas

entero función BuscarPersona(valor persona:v; valor tipoElemento:el;
                             valor entero:primero, último)
var
   entero: izq, der, cen
inicio
   izq ← primero
   der ← último
   repetir
      cen ← (izq + der) div 2
      si v[cen].datosBancarios.numCuenta > el.datosBancarios.numCuenta entonces
         der ← cen – 1
      si_no
         izq ← cen + 1
      fin_si
   hasta_que (v[cen].datosBancarios.numCuenta = el. datosBancarios.numCuenta)
                       o (izq > der)
   si v[cen].datosBancarios.numCuenta = el.datosBancarios.numCuenta entonces
       devolver(cen)
   si_no
       devolver(0)
   fin_si
fin_función
```

Puntuación: 1,75 puntos

3. Archivos. Describa los distintos tipos de organización de archivos. Enumere y describa la utilidad de las distintas órdenes necesarias para trabajar con archivos secuenciales.

Apartados 9.3, 9.4, 9.6.1., 9.6.2., 9.6.3. y 9.9.1. del libro de texto

Aplicación

- Codifique un procedimiento que permita guardar en un archivo secuencial el array de registros del ejercicio enterior.
- Codifique un procedimiento que permita borrar de dicho archivo una persona a partir de su DNI que se pasará como argumento al procedimiento.

```
procedimiento GuardarPersonas(valor personas : v; valor entero : n)
var
```

```
   archivo_s de persona : A
   entero : i
inicio
   crear('ARCHIVO.DAT')
   abrir(A, 'ARCHIVO.DAT', escritura)
   desde i ← 1 hasta n hacer
      escribir(A, v[i])
   fin_desde
   cerrar(A)
fin_procedimiento

procedimiento BorrarPersona(valor cadena : DNI)
var
   archivo_s de persona : A, Aux
   persona : R
inicio
   abrir(A, 'ARCHIVO.DAT', lectura)
   crear('AUXILIAR.DAT')
   abrir(AUX, 'AUXILIAR.DAT', escritura)
   leer(A,R)
   mientras no fda(A) hacer
      si R.DNI <> DNI entonces
         escribir(Aux, R)
      fin_si
      leer(A,R)
   fin_mientras
   cerrar(A,AUX)
   borrar('ARCHIVO.DAT')
   renombrer('AUXILIAR.DAT', 'ARCHIVO.DAT')
fin_procedimiento
```

Puntuación: 1,75 puntos

Preguntas prácticas

Una empresa de medición de audiencias televisivas guarda en un array de registros información de los **215** programas que actualmente emiten por televisión las distintas cadenas gratuitas de ámbito nacional (TV1, TV2, Telecinco, Antena3, Cuatro y La Sexta). Por cada programa se guarda el nombre del programa y la cadena que lo emite. Diariamente almacena en un array de dos dimensiones el número de espectadores que ha tenido cada programa cada una de las **24** horas del día. Un programa puede emitirse a lo largo de distintas horas y, evidentemente, tendrá 0 espectadores en las horas en las que no se emite. La información está almacenada de forma que el número de espectadores del programa X del array de registros está en la fila X del array de dos dimensiones.

Se pide:

1. Realizar todas las declaraciones de las estructuras de datos necesarias para realizar los puntos que aparecen a continuación.

```
tipos
   //Registro y array de registros con los programas que se emiten
   registro = programa
      cadena : nombre, emisora
   fin_registro
   array[0..215] de programa = programas

   //Tabla con las audiencias de cada programa
   array[1..215, 1..24] de entero = tabla

   //Registro y array de registros las audiencias de las cadenas de televisión
   registro = TV
      cadena : nombre
      entero : numEspectadores  //Se utiliza para saber el total de espectadores y
                                //obtener el porcentaje de audiencia
      real : porcentaje
   fin_registro
   array[0..6] de TV = TVs
```

Puntuación: 0,5 puntos

2. Codifique una función que permita saber el nombre del programa más visto a una hora determinada que será pasada como argumento a la función.

```
cadena función elProgramaMásVisto(valor tabla : t; valor programas : p ;
                                  valor entero : hora)
var
   entero : maxCol //Posición del máximo de la columna hora
inicio
   maxCol ← máximoColumna(t, 215, hora)
   devolver programas[maxCol].nombre
fin_función

//Calcula la posición del mayor valor de la columna col
//entre los n elemento de la misma
entero función máximoColumna(valor tabla : t; valor entero :n,col)
var
   entero : i, máx
inicio
   máx ← 1
   desde i ← 2 hasta n hacer
      si t[i,col] > t[máx,col] entonces
         máx ← i
      fin_si
   fin_desde
   devolver(máx)
fin_función
```

Puntuación: 1,25 puntos

3. Codifique una función que permita saber el número de espectadores que ha tenido un programa determinado a lo largo del día.

```
//Calcula el número de espectadores del programa situaado en la posición númPrograma
//Consideramos que el número de espectadores es el momento de mayor audiencia del
//programa, es decir el máximo de la fila
entero función numeroDeEspectadores(valor tabla : t; valor entero : númPrograma)
var
   entero : máxFila //Posición del máximo de la fila númPrograma
inicio
   máxFila ← máximoFila(table, 24, númPrograma)
   devolver(t[númPrograma, máxFila]
fin_función


//Calcula la posición del mayor valor de la fila fila
//entre los n elemento de la misma
entero función máximoFila(valor tabla : t; valor entero :n,fila)
var
   entero : i, máx
inicio
   máx ← 1
   desde i ← 2 hasta n hacer
      si t[fila,i] > t[fila,máx] entonces
         máx ← i
      fin_si
   fin_desde
   devolver(máx)
fin_función
```

Puntuación: 1,25 puntos

4. Generar un array de registros que almacene cada una de las cadenas y el porcentaje de audiencia que ha tenido ese día. Sacar por pantalla dicho array ordenado de mayor a menor por el porcentaje de audiencia.

```
procedimiento audiencias(valor tabla : t; valor programas : p; ref TVs : cadenas)
var
   entero : i, espectadores, pos
   entero : totalEspectadores  //Total de espectadores de todas las cadenas
inicio
   //Llena el array Cadenas con el nombre de las cadenas
   //e inicializa el número de espectadores
   desde i ← 1 hasta 6 hacer
      cadenas[i].númEspectadores ← 0
      según_sea i hacer
         1 : cadenas[i].nombre ← "TV1"
         2 : cadenas[i].nombre ← "TV2"
         3 : cadenas[i].nombre ← "Antena3"
         4 : cadenas[i].nombre ← "Cuatro"
         5 : cadenas[i].nombre ← "Telecinco"
         6 : cadenas[i].nombre ← "La Sexta"
      fin_segun
   fin_desde

   //Calcula el total de audiencia de cada cadena
   //El total de cada cadena será la suma de las audiencias de cada uno
   //de los programas de la misma
   totalEspectadores ← 0
   desde i ← 1 hasta 215 hacer
      //Se calcula el número de espectadores del programa i
      espectadores ← numeroDeEspectadores(t,i)

      //Se busca la cadena del programa i y se suma el número de espectadores
      pos ← buscar(cadenas, 6, p[i].emisora

      //Se acumula el número del espectadores del programa al de la cadena
      cadenas[i].númEspectadores ← cadenas[i].númEspectadores + espectadores

      //Se acumula el número de espectadores del programa i a total de espectadores
      totalEspectadores ← totalEspectadores + espectadores
   fin_desde

   //Calcular el porcentaje de cada cadena sobre el total de espectadores
   desde i ← 1 hasta 6 hacer
      cadenas[i].porcentaje ← cadenas[i].númEspectadores * 100 / totalEspectadores
   fin_desde

   ordenarAudiencias(cadenas,6)

   //Escribir el ranking de audiencias
   desde i ← 1 hasta 6 hacer
      escribir(cadenas[i].nombre, cadenas[i].porcentaje)
   fin_desde
fin_procedimiento

//Se hace la búsqueda secuencial con centinela
entero función Buscar(valor TVs : v; valor entero : n; valor cadena : nombre)
var
   entero: i
inicio
   i ← n
   v[0] ← el
   mientras el.porcentaje <> v[i] .porcentaje hacer
      i ← i - 1
   fin_mientras
   devolver(i)
fin_función
```

```
//Se ordena por inserción binaria
procedimiento ordenarAudiencias(ref cadenas:v;valor entero : n)
//El vector v tiene elementos entre 1 y n
var
   entero : i,j
inicio
   desde i ← 2 hasta n hacer
      v[0] ← v[i]
      j ← i - 1
      mientras v[j].porcentaje < v[0].porcentaje hacer
          v[j+1] ← v[j]
          j ← j - 1
      fin_mientras
      v[j+1] ← v[0]
   fin_desde
fin_procedimiento
```

Puntuación: 2 puntos