



## Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Febrero de 2000	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

## Soluciones al examen

### Preguntas teórico-prácticas

#### Ejercicio 1.

- a) Estructuras de control repetitivas. Representación, características y comportamiento de cada una de ellas.

*Apartado 4.6. del libro de texto.*

- b) Se desea un algoritmo que permita calcular la media aritmética de una serie de números enteros positivos. El usuario introducirá números por teclado hasta que introduzca el número  $-1$  y a continuación se visualizará la media de todos los números anteriores con excepción de dicho número  $-1$ . Codifique el algoritmo empleando cada una de las estructuras repetitivas.

```
//Con una estructura de tipo Repetir
algoritmo Media1
var
    entero : conta,n
    real : media
inicio
    conta ← 0
    media ← 0
    leer(n)
    repetir
        conta ← conta + 1
        media ← media + n
        leer(n)
    hasta_que n = -1
    media ← media / conta
    escribir(media)
fin

//Con una estructura de tipo Mientras
algoritmo Media2
var
    entero : conta,n
    real : media
inicio
    conta ← 0
    media ← 0
    leer(n)
    mientras n <> -1 hacer
        conta ← conta + 1
        media ← media + n
        leer(n)
    fin_mientras
    si conta <> 0 entonces
        media ← media / conta
    si_no
        media ← 0
```



### Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Febrero de 2000	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

```
fin_si
  escribir(media)
fin
```

Con una estructura de tipo desde, no sería posible procesar una serie indefinida de números, puesto que este tipo de estructuras se repite un número fijo de veces.

De los algoritmos diseñados, indique cuales serán más convenientes y cuáles más inconvenientes de utilizar razonando la respuesta.

Si al menos se tuviera que procesar un número, la estructura más conveniente sería la primera (Repetir). La estructura Mientras se debería utilizar si fuera posible que no se procesara ningún elemento. Para un problema de este tipo la estructura Desde no sería la adecuada.

**Puntuación: 1,5 puntos.**

#### Ejercicio 2.

- a) Programación modular. Explique el concepto de programación modular. Explique las diferencias entre los distintos tipos de módulos que conozca. Explique las distintas formas de intercambiar la información entre el programa principal y los módulos.

Capítulo 5 del libro de texto

- b) Se tiene almacenado en un array de registros los nombres y las edades de una serie de personas. Se desea ordenar por nombre utilizando el método de inserción directa los elementos contenidos en el array comprendidos entre dos posiciones LimiteInferior y LimiteSuperior. Diseñe las estructuras de datos necesarias para almacenar la información así como un procedimiento que realice la ordenación por el método indicado entre las dos posiciones (dichas posiciones se pasarán como argumentos al procedimiento).

**Puntuación: 1,5 puntos.**

```
const
  MaxEl = 100 // Número máximo de elementos de la lista
tipos
  registro
    cadena : nombre
    entero : edad
  fin_registro = persona
  array[1..MaxEl] de persona : vector
var
  vector : v;
  entero : LimiteInferior, LimiteSuperior
...
...
procedimiento Ordenar(E/S vector : v; E entero : LimSup, LimInf)
var
  entero : i, j
  persona : aux
  lógico : sw
inicio
  desde i ← LimInf + 1 hasta LimSup hacer
    aux ← v[i]
    j ← i - 1
```



### Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Febrero de 2000	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

```

sw ← falso
mientras no sw y (j >=Inf) hacer
  si aux.nombre < v[j].nombre entonces
    v[j+1] ← v[j]
    j ← j - 1
  si_no
    sw ← verdad
  fin_si
fin_mientras
v[j+1] ← aux
fin_desde
fin_función
  
```

### Pruebas prácticas

#### Problema 1

En un array de n x m posiciones se almacenan aleatoriamente caracteres X y O. Diseñe las estructuras de datos el programa principal y los módulos necesarios que permitan:

- Indicar por cada fila y columna la cantidad máxima de caracteres X contiguos situados en cada una de ellas.
- Indicar en que fila y en qué columna se encuentra la cantidad máxima de caracteres X contiguos.

X	X	O	X	X	2
X	O	O	O	O	1
O	X	O	X	X	3
O	X	X	X	X	4
X	X	X	O	O	3
X	X	O	O	X	2
X	O	X	X	X	3
3	4	2	2	2	

El mayor número de X está en la columna 2

El mayor número de X está en la fila 4

**Puntuación: 3 puntos.**

```

algoritmo Problema1
const
  n = 4
  m = 5
  Ultimo = 5
tipos
  array[1..Ultimo,1..Ultimo] de carácter = tabla
  array[1..Ultimo] de enteros = vector
var
  tabla : t
  entero : i, j, x
  vector : MaximosFila,MaximosColumna

inicio
  // Carga del la tabla con X y O
  ....
  ....

  //Calcular el número de X por filas
  
```



### Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Febrero de 2000	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

```
desde i ← 1 hasta n hacer
    MaximosFila[i] ← XPorFila(t,i,m)
fin_desde

//Calcular el número de X por columnas
desde j ← 1 hasta m hacer
    MaximosColumna[j] ← XPorColumna(t,j,n)
fin_desde

//Escribir la tabla
desde i ← 1 hasta n hacer
    //Escribir una fila
    desde j ← 1 hasta m hacer
        escribir(t[i,j])
    fin_desde
    //Escribir el máximo de esa fila
    escribir(MaximosFila[i])
fin_desde

//Escribir los máximos de las columnas
desde j ← 1 hasta m hacer
    escribir(MaximosColumna[j])
fin_desde
escribir('Fila con mas X:',MaximoVector(MaximosFila,n))
escribir('Columna con mas X:',MaximoVector(MaximosColumna,m))
fin

entero : función XPorFila(E tabla t ; entero : c,m)
var
    entero : seguidos,mayor,j
inicio
    seguidos ← 0
    mayor ← 0
    desde j ← 1 hasta m hacer
        si t[f,j] = 'X' entonces
            si seguidos = 0 entonces
                seguidos ← 1
            si_no
                seguidos ← seguidos + 1
            fin_si
        si_no
            si seguidos >= mayor entonces
                mayor ← seguidos
                seguidos ← 0
            fin_si
        fin_si
    fin_desde
    si seguidos > mayor entonces
        devolver(seguidos)
    si_no
```



### Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Febrero de 2000	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

```
    devolver(mayor)
  fin_si
fin_función

entero : función XPorColumna(t:tablac,n:entero)
var
  entero : seguidos,mayor,i
inicio
  seguidos ← 0
  mayor ← 0
  desde i ← 1 hasta n hacer
    si t[i,c] = 'X' entonces
      si seguidos = 0 entonces
        seguidos ← 1
      si_no
        seguidos ← seguidos + 1
    fin_si
  si_no
    si seguidos >= mayor entonces
      mayor ← seguidos
      seguidos ← 0
    fin_si
  fin_si
  fin_desde
  si seguidos > mayor entonces
    devolver(seguidos)
  si_no
    devolver(mayor)
fin_función

entero : función MaximoVector(v:vector n : entero)
var
  entero : i,max
inicio
  max ← 1
  desde i ← 2 hasta n hacer
    si v[i] > v[max] entonces
      max ← i
    fin_si
  fin_desde
  devolver(max)
fin_función
```

### Problema 2

Un taller de reparaciones de ordenadores mantiene en un archivo directo por transformación (se deja al alumno la elección de la función hash) de claves un histórico de las reparaciones realizadas. Se prevé que el archivo contenga un máximo de 5000 registros, a los que se deberá añadir una cantidad suficiente de posiciones para almacenar las colisiones que se pudieran originar. La estructura de campos del archivo es la siguiente:

Campo	Formato	Observaciones
Fundamentos de la Programación (106)		Febrero 2000 - Mañana



### Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Febrero de 2000	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

RefEquipo	Entero	Campo clave con la referencia del equipo
NomPropietario	Cadena	Nombre del propietario del equipo
NumEntradas	Entero	Número de veces que el equipo ha entrado en el taller
UltimaFecha	Cadena (aaaa-mm-dd)	Fecha de la última entrada del equipo en el taller.

Cada vez que se finaliza la reparación de un equipo, se genera un registro en un archivo secuencial con la siguiente información.

Campo	Formato	Observaciones
RefEquipo	Entero	Campo clave con la referencia del equipo
NumReparacion	Entero	Referencia de la reparación
FechaReparacion	Cadena (aaaa-mm-dd)	Fecha de la reparación

El archivo está ordenado por el campo RefEquipo.

Semestralmente se actualiza el archivo histórico de reparaciones con los datos contenidos en el archivo secuencial. Diseñe un algoritmo con las estructuras de datos, programa principal y módulos necesarios para realizar dicha actualización teniendo en cuenta que:

- Puede que un equipo del archivo secuencial no exista en el archivo directo. En dichos casos se dará un alta en el archivo histórico, el campo NumEntradas se pondrá a 1 y el campo NomPropietario se dejará en blanco.
- Si el equipo ya existe en el archivo directo, se incrementará en 1 el campo NumEntradas.
- Puede que en el semestre un mismo equipo haya entrado varias veces en el taller. En esos casos el campo NumEntradas se incrementará en consecuencia y en el campo UltimaFecha aparecerá la última fecha de reparación del equipo.

**Puntuación: 4 puntos.**

```
algoritmo problema2
const
    AreaPrincipal = 5000
    UltimoRegistro = 6000 //5000 mas un 20% para colisiones
tipos
    registro
        entero : RefEquipo, NumEntradas
        cadena : propietario, UltimaFecha
        lógico : libre //se usa para controlar si una posición está vacía
    fin_registro = Equipo
    archivo_d de Equipo = Historico
    registro
        entero : RefEquipo, NumReparacion
        cadena : FechaReparacion
    fin_registro = Reparacion
    archivo_s de Reparacion = Reparaciones
var
    Reparaciones : Arep
    Historico : Ahis
    Equipo : Req
    Reparacion : Rrep
    entero : p
inicio
    abrir (AREp, lectura, 'REPARACIONES')
```



### Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Febrero de 2000	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

```
abrir(AHis,lectura/escritura,'HISTORICO')

leer(ARep, RRep)
mientras no fda(ARep) hacer
  p ← buscar(AHis, RRep.RefEquipo)
  si p = 0 entonces
    //El equipo no se encuentra, por lo que se da un alta
    p ← hash(RRep.RefEquipo)
    leer(AHis, RHis, p)
    si no RHis.libre entonces
      //Hay que buscar sitio en la zona de colisiones
      p ← AreaPrincipal
      repetir
        p ← p + 1
        leer(AHis, RHis, p)
      hasta que RHis.libre o (p = UltimoRegistro)
    fin_si
    si no RHis.libre entonces
      //El area de colisiones estaba llena
    si_no
      //Se ha encontrado un hueco libre en el area princial
      //o en la zona de colisiones
      //se da un alta
      RHis.RefEquipo ← RRep.RefEquipo
      RHis.Propietario ← ''
      RHis.UltimaFecha ← RRep.FechaReparacion
      RHis.NumEntradas ← 1
      RHis.libre ← falso
      escribir(AHis,RHis,p)
    fin_si
  si_no
    //Se ha encontrado el equipo, se procede a modificar los campos
    leer(AHis,RHis,p)
    RHis.NumEntradas ← RHis.NumEntradas + 1
    //Si la fecha de reparación es posterior, se modifica
    si RRep.FechaReparacion > RHis.UltimaFecha entonces
      RHis.UltimaFecha ← RRep.FechaReparacion
    fin_si
    escribir(AHis,RHis,p)
  fin_si
  leer(ARep,RRep)
fin_mientras
cerrar(AHis,ARep)
fin

entero : funcion Hash(E entero : clave)
inicio
  devolver(clave mod AreaPrincipal + 1)
fin_si

entero : funcion Buscar(E/S historico : AHis; E entero : clave)
```



### Cuadernillo de examen

ASIGNATURA	Fundamentos de la programación	CÓDIGO	106
CONVOCATORIA	Febrero de 2000	PLAN DE ESTUDIOS	1996
ESPECIALIDAD	Común	CURSO	1º
TURNO	Mañana	CENTRO	Facultad/Escuela
CARÁCTER	Anual	CURSO ACADÉMICO	1999/2000
DURACIÓN APROXIMADA	2 horas y media		

```
var
  Equipo : R
  entero : p
inicio
  p ← hash(clave)
  leer(AHis,R,p)
  si R.libre entonces
    devolver(0)
  si_no
    si R.RefEquipo = clave entonces
      devolver(p)
    si_no
      //De estar, está en la zona de colisiones
      p ← AreaPrincipal
      repetir
        p ← p + 1
        leer(AHis,R,p)
      hasta_que (R.RefEquipo = clave) o R.libre o (p = UltimoRegistro)
      si (R.RefEquipo = clave) entonces
        devolver(p)
      si_no
        devolver(0)
      fin_si
    fin_si
  fin_si
fin_función
```