UNIVERSIDAD PONTIFICIA DE SALAMANCA EN MADRID

Facultad de Informática

Escuela Universitaria de Informática

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software

ASIGNATURA	**Fundamentos de Programación I**	CÓDIGO	**106**
CONVOCATORIA	**Extraordinaria Septiembre 2002**	PLAN DE ESTUDIOS	**2000**
ESPECIALIDAD	**Común**	CURSO	**1**
TURNO	**Mañana**	CENTRO	**Escuela**
CARÁCTER	**Cuatrimestral (Primer cuatrimestre)**	CURSO ACADÉMICO	**2001-2002**

Soluciones propuestas

Preguntas teórico-prácticas

1. Búsqueda en arrays. Describa los distintos algoritmos de búsqueda que conozca.

   Apartado 9.3 del libro de texto

   Una función contínua en intervalo cerrado [a,b] toma el valor 0 en algún punto de ese intervalo si f(a)*f(b) < 0, es decir, si en ambos extremos los valores de f(a) y f(b) son de distintos signo. Escribir un algoritmo de búqueda que determine el valor v tal que f(v) = 0 con un error menor que 0,001.

```
real función Bolzano(E real : a,b)
inicio
      mientras f((a+b)/2) <> 0 hacer
         si f((a+b)/2) > 0 entonces
            a ← (a+b)/2
         si_no
            b ← (a+b)/2
         fin_si
     fin_mientras
     devolver((a+b)/2)
fin_procedimeinto
```

   Puntuación: 1,5 puntos

2. Concepto de registro: componentes y operaciones.

   Apartado 6.3 del libro de problemas

   Se dispone de un array de 1000 elementos que almacena números entre 1 y 10. Utilizando las estructuras de datos que estime más convenientes, diseñe un algoritmo que muestre los números más frecuentes por orden ascendente de frecuencia de aparición y a igual frecuencia de menor a mayor número.

```
tipos
  registro = elemento
    entero : n
    entero : f
  fin_registro

  array[1..10] de elemento : frecuencias
  array[1..1000] de entero : numeros
…

//Se supone la existencia de un procedimiento que rellena con
//números aleatorios el array de números

procedimiento ListarFrecuencias(E numeros : num)
```

```
var
   frecuencias : fr
   entero : i
inicio
   InicializaArrayFrecuencias(fr)
   desde i ← 1 hasta 1000 hacer
      //Por cada número del array se suma uno
      //a la frecuencia del elemento num[i] del
      //array de frecuencias
      fr[num[i]].f ← fr[num[i].f + 1
   fin_desde
   Ordenar(fr,10)
   desde i ← 1 hasta 10 hacer
      escribir(fr[num[i]].n, fr[num[i]].f)
   fin_desde
fin_procedimiento

procedimiento InicializaArrayFrecuencias(E/S frecuencias : fr)
//Inicializa el array de frecuencias
//en el campo n del elemento introduce el número
//se inicializa a 0 el campo frecuencia
var
   entero : i
inicio
   desde i ← 1 to 10 hacer
      fr[i].n ← i
      fr[i].f ← 0
   fin_desde
fin_procedimiento

procedimiento Ordenar(E/S frecuencias : fr;E entero : n)
var
   entero : i, j
inicio
   desde i ← 1 hasta n-1 hacer
      desde j ← 1 hasta n – i hacer
         si (fr[j].f < fr[j+1].f) o
            (fr[j].f < fr[j+1].f) y (fr[j].n < fr[j+1].n) entonces
            intercambio(fr[j],fr[j+1])
         fin_si
      fin_desde
   fin_desde
fin_procedimiento
```

Puntuación: 1,5 puntos

Preguntas prácticas

1. En un array de dos dimensiones se almacenan las notas finales de cuarenta alumnos en las cinco asignaturas de un curso académico. A partir de estos datos diseñe los módulos correspondientes para obtener:

   a) Número de alumnos que han suspendido todas las asignaturas.

   b) La asignatura con menor diferencia entre el número de aprobados y suspensos.

   c) El alumno que, habiendo aprobado todas las asignaturas tenga la nota más alta.

Puntuación: 3 puntos

```
tipos
   array[1..40,1..5] de real : notas
   //Array de 40 filas (alumnos) y
   //5 columnas (asignaturas)
…
```

a)

```
entero : funcion NumSuspensos(E notas : not)
var
   entero : n //Contador con el número de suspensos
   entero : i,j
   lógico : aprobada
inicio
   n ← 0
   desde i ← 1 hasta 40 hacer
      aprobada ← verdad
      j ← 1
      mientras aprobada y (j<5) hacer
         si not[i,j] < 5 entonces
             aprobada ← falso
         fin_si
         j ← j + 1
      fin_mientras
      //Si tiene alguna asignatura suspensa
      //aprobada es falso
      si no aprobada entonces
         n ← n + 1
      fin_si
   fin_desde
   devolver(n)
fin_función
```

b)

```
entero : función Diferencia(E notas : not)
var
   entero : j
   entero : dif //Diferencia entre aprobados y suspensos
   entero : max //Mayor diferencia
   entero : asig //Asignatura con mayor diferencia
inicio
   max ← 0
   desde j ← 1 hasta 5 hacer
      dif ← abs(NumAprobados(not,j) – NumSuspensos(not,j))
      si dif < max entonces
         max ← div
         asig ← j
      fin_si
   fin_desde
   devolver(asig)
fin_función

//Devuelve el número de aprobados de la asignatura
//situado en la columna col
entero : función NumAprobados(E notas : not; E entero : col)
var
   entero : i,n
```

```
inicio
   n ← 0
   desde i ← 1 hasta 40 hacer
      si not[col,i] >= 5 entonces
         n ← n + 1
      fin_si
   fin_desde
   devolver(n)
fin_función

//Devuelve el número de suspensos de la asignatura
//situado en la columna col
entero : función NumSuspensos(E notas : not; E entero : col)
var
   entero : i,n
inicio
   n ← 0
   desde i ← 1 hasta 40 hacer
      si not[col,i] <5 entonces
         n ← n + 1
      fin_si
   fin_desde
   devolver(n)
fin_función
```

c)

```
entero : función Alumno(E notas : not)
var
   entero : i,j
   real : max //Nota más alta
   entero : alum //Alumno con nota más alta
   lógico : aprobada
inicio
   max ← 0
   alum ← 0
   desde i ← 1 hasta 40 hacer
      //Comprueba si tiene aprobadas todas las asignaturas
      aprobada ← verdad
      j ← 1
      mientras aprobada y (j<5) hacer
         si not[i,j] < 5 entonces
             aprobada ← falso
         fin_si
         j ← j + 1
      fin_mientras
      //Si tiene alguna asignatura suspensa
      //aprobada es falso
      si aprobada entonces
         //Extrae la nota más alta
         desde j ← 1 hasta 5 hacer
            si not[i,j] > max entonces
               max ← not[i,]
               alum ← i
            fin_si
         fin_desde
      fin_si
   fin_desde
   //Devuelve 0 si ningún alumno ha aprobado tod
```

```
   devolver(alum)
fin_función
```

2. Se desea implementar conjuntos de números enteros mediante listas ordenadas simplemente enlazadas. Construir los procedimientos y funciones necesarios para determinar la unión, intersección, diferencia e igualdad entre dos conjuntos.

```
tipos
   entero = TipoElemento
   puntero_a nodo = lista
   registro = nodo
      TipoElemento : info
      lista : sig
   fin_registro

procedimiento Unión(E lista : c1,c2 ; E/S lista : u)
var
   TipoElemento : e1, e2
inicio
   u ← nulo
   mientras (c1 <> nulo) y (c2 <> nulo) hacer
      e1 ← c1↑.info
      e2 ← c2↑.info
      si e1 < e2 entonces
         InsertarOrdenado(u,e1)
         c1 ← c1↑.sig
      si_no
         si e2 < e1 entonces
          InsertarOrdenado(u,e2)
          c2 ← c2↑.sig
         si_no
           InsertarOrdenado(u,e1)
           c1 ← c1↑.sig
           c2 ← c2↑.sig
         fin_si
      fin_si
   fin_mientras
   mientras c1 <> nulo hacer
      e1 ← c1↑.info
      InsertarOrdenado(u,e1)
   fin_mientras
   mientras c2 <> nulo hacer
      e2 ← c2↑.info
      InsertarOrdenado(u,e2)
   fin_mientras
fin_procedimiento

procedimiento Intersección(E lista : c1,c2 ; E/S lista : i)
var
   TipoElemento : e1, e2
inicio
   i ← nulo
   mientras (c1 <> nulo) y (c2 <> nulo) hacer
      e1 ← c1↑.info
      e2 ← c2↑.info
```

```
      si e1 < e2 entonces
         c1 ← c1↑.sig
      si_no
         si e2 < e1 entonces
          c2 ← c2↑.sig
         si_no
           InsertarOrdenado(i,e1)
           c1 ← c1↑.sig
           c2 ← c2↑.sig
         fin_si
      fin_si
   fin_mientras
fin_procedimiento

procedimiento Diferencia(E lista : c1,c2 ; E/S lista : d)
var
   TipoElemento : e1, e2
inicio
   d ← nulo
   mientras (c1 <> nulo) y (c2 <> nulo) hacer
      e1 ← c1↑.info
      e2 ← c2↑.info
      si e1 < e2 entonces
         InsertarOrdenado(d,e1)
         c1 ← c1↑.sig
      si_no
         si e2 < e1 entonces
          InsertarOrdenado(d,e2)
          c2 ← c2↑.sig
         si_no
           c1 ← c1↑.sig
           c2 ← c2↑.sig
         fin_si
      fin_si
   fin_mientras
   mientras c1 <> nulo hacer
      e1 ← c1↑.info
      InsertarOrdenado(d,e1)
   fin_mientras
   mientras c2 <> nulo hacer
      e2 ← c2↑.info
      InsertarOrdenado(d,e2)
   fin_mientras
fin_procedimiento

lógico : función Igualdad(E lista : c1,c2)
var
   TipoElemento : e1, e2
   lógico : igual
inicio
   igual ← verdad
   mientras (c1 <> nulo) y (c2 <> nulo) y igual hacer
      e1 ← c1↑.info
      e2 ← c2↑.info
      si e1 <> e2 entonces
         igual ← falso
```

```
      si_no
        c1 ← c1↑.sig
        c2 ← c2↑.sig
      fin_si
   fin_mientras
   devolver(igual y (c1 = nulo) y (c2 = nulo))
fin_procedimiento

procedimiento InsertarOrdenado(E/S lista : l; E TipoElemento : e)
var
   lista : aux,ant
   lógico : encontrado
inicio
   aux ← l
   encontrado ← falso
   mientras no encontrado y (aux <> nulo)
      si aux↑.info >= e entonces
         encontrado ← verdad
      si_no
         ant ← aux
         aux ← aux↑.sig
      fin_si
   fin_mientras
   si aux = l entonces
      Insertar(l,e)
   si_no
      Insertar(ant↑.sig,e)
   fin_si
fin_procedimiento

procedimiento Insertar(E/S lista : l; E TipoElemento : e)
var
   lista : aux
inicio
   reservar(aux)
   aux↑.info ← e
   aux↑.sig← l
   l ← aux
```

Puntuación: 4 puntos