

Informática. Exame de Teoría.
Setembro, 2007

Nome:

1	2	3	4	5		6	7	8	9	10	T

PRIMEIRO PARCIAL:

1. (1 PUNTO) Explica a sintaxe e o comportamento dos bloques IF e IF/ELSE IF en Fortran.

SOLUCIÓN. Sintaxe do IF:

```
if(condicion) then
    sentencias
end if
```

As sentencias execútanse só se se cumpre a condición, que debe ser unha expresión de tipo lóxico. Sintaxe do IF/ELSE IF:

```
if(condicion1) then
    sentencias1
else if(condicion2) then
    sentencias2
else if(condicion3) then
    sentencias3
...
else if(condicionN) then
    sentenciasN
else
    sentenciasN+1
end if
```

Neste caso, avalíase condicion1. Se se cumpre, execútanse sentencias1 e sáese fora da estrutura de selección. En caso contrario, avalíase condicion2 e, se se cumpre, execútanse sentencias2 e sáese fora. Se non, avalíase condicion3, e así sucesivamente. Se non se cumpre ningunha das N condicións, execútanse sentenciasN+1 (isto último é opcional).

2. (1 PUNTO) Escribe en Fortran as sentencias correspondentes a unha estrutura iterativa indefinida que en cada iteración lea por teclado un número enteiro e o sume a un acumulador até que o usuario introduza o valor -1. Nese momento debe mostrar por pantalla o resultado da suma.

SOLUCIÓN.

```

suma = 0
do
  print *, "introduza un numero enteiro: "
  read *, n
  if(-1 /= n) then
    suma = suma + n
  else
    print *, "suma = ", suma
    exit
  end if
end do

```

3. (1 PUNTO) Explica brevemente cómo opera o algoritmo de ordeamento por inserción. Cómo operaría sobre o vector { 5 8 3 2 7 1 }

SOLUCIÓN. O algoritmo opera collendo un número X e buscando á súa esquerda un número inferior a el. Se o atopa, sitúa o número X á súa dereita (desprazando tódolos números entre eles unha posición cara á dereita. Se non o atopa, sitúa X no extremo esquerdo do vector a ordear. Co vector proposto, operaría así

```

5 8 3 2 7 1
3 5 8 2 7 1
2 3 5 8 7 1
2 3 5 7 8 1
1 2 3 5 7 8

```

4. (1 PUNTO) Escribir un segmento de código en Fortran cun programa principal e unha subrutina `s()` que use un bloque `common` chamado `bloque1` para compartir unha variábel enteira `n` e unha cadea de 10 caracteres `c` entre ambos subprogramas.

SOLUCIÓN.

```

program proba
integer :: n
character(len = 10) :: c
common /bloque1/ n, c
...
end program proba

subroutine s()
integer :: n
character(len = 10) :: c
common /bloque1/ n, c
...
end subroutine s

```

5. (1 PUNTO) Escribi-las sentencias en Fortran necesarias para abri-lo arquivo `datos.dat` e escribir un enteiro de 5 cifras, un real con expoñente de 4 cifras decimais na columna 10, dous espazos en branco, unha cadea de 20 caracteres e un real sen expoñente de 3 cifras decimais na columna 70.

SOLUCIÓN.

```

integer :: n
character(len = 20) :: c
real :: x
real(kind = 8) :: y
open(1, file = "datos.dat", err = 1)
write (1, 2) n, x, c, y
2  format(i5, t10, e10.4, 2x, a20, t70, f10.3)
stop
1  print *, "erro en open"
stop

```

SEGUNDO PARCIAL:

6. (1 PUNTO) Explicar brevemente para qué sirven os rexistros **Contador de Programa** e **Punteiro de Pila** (1 liña cada un). Ónde se atopan estes rexistros?

SOLUCIÓN. O contador de programa é o rexistro que almacena a dirección en memoria RAM da seguinte instrucción a executar polo microprocesador. O punteiro de pila é o rexistro que almacena a dirección en memoria RAM da instrucións onde comeza o subprograma que chamou ao subprograma que se está executando. Deste modo é posíbel conserva-lo rastro da secuencia de chamadas a subprogramas sucesivos e desfaze-la secuencia de chamadas. Ambos rexistros están no microprocesador (CPU).

7. (1 PUNTO) Escribir as sentencias en Fortran necesarias para declarar un módulo chamado `punto_2d` que conteña un tipo de datos derivado chamado `punto` con catro variábeis reais `x`, `y`, `r` e `a`. Tamén debe declarar unha variábel do tipo `punto`. Finalmente, debe ter dous subprogramas chamados `a_polares(x, y)` e `a_cartesianas(r, a)`, con tódolos argumentos reais, que convirtan a coordenadas polares e cartesianas respectivamente.

SOLUCIÓN.

```

module punto_2d
  type punto
    real :: x, y, r, a
  end type punto
  type(punto) :: p
contains
  subroutine a_polares(x, y)
    real, intent(in) :: x, y
    r = sqrt(x*x + y*y)
    a = atan(y/x)
    return
  end subroutine a_polares

  subroutine a_cartesianas(r, a)
    real, intent(in) :: r, a
    x = r*cos(a)
    y = r*sin(a)
    return

```

```

    end subroutine a_cartesianas
end module punto_2d

```

8. (1 PUNTO) Consideremo-lo seguinte fragmento de pseudocódigo, aplicado á xestión de listas enlazadas:

```

q, t : punteiros a nodo
ref : tipo X
band : loxica

q = p; band = SI
mentres q^.dato /= ref e band == SI
    se q^.enlace /= NULO
        t = q
        q = q^.enlace
    se non
        band = NON
    fin se
fin mentres

```

Onde **p** é o punteiro ao primeiro nodo da lista. Qué fai este fragmento? A qué apunta o punteiro **t** cando remata a execución? Cando se remata a estrutura iterativa **mentres**? Cando vale NON a bandeira **band**?

SOLUCIÓN. Este framgento busca un nodo con dato **ref** nunha lista simplemente enlazada. Se o atopa, q apunta a este nodo, e t ao nodo anterior. Se non o atopa, q vale nulo. A estrutura iterativa mentres remata cando se atopa o nodo con dato ref (en tal caso q apunta a ese nodo) ou ben se remata a lista (en tal caso q = nulo). A bandeira band vale NON cando atopamo-lo final da lista.

9. (1 PUNTO) Escribe un programa en Matlab que lea por teclado dous números enteiros x_1 e x_2 e calcule, de forma interactiva, os termos da sucesión $\{x_n\}_{n=3}^{n=100}$, definidos pola expresión:

$$x_n = \begin{cases} \frac{x_{n-1} - x_{n-2}}{2} & n \text{ par} \\ \frac{x_{n-1} + x_{n-2}}{2} & n \text{ impar} \end{cases}$$

SOLUCIÓN.

```

clear all
x2 = input('introduza un enteiro: ');
x1 = input('introduza outro enteiro: ');
for i = 1:100
    if rem(i, 2) == 1
        y = (x1 - x2)/2;
    else
        y = (x1 + x2)/2;
    end
end

```

```
fprintf('%f\n', y);  
% actualizacion de x_{n - 1} e de x_{n - 1} para a seguinte iteracion  
x2 = x1; x1 = y;  
end
```

10. (1 PUNTO) Qué é unha interrupción? Nomea posíbeis causas dunha interrupción.

SOLUCIÓN. Unha interrupción é unha detención asíncrona (é decir, que pode ocorrer en calquer momento, non se repite periódicamente) na execución dun programa. Provoca que se garde o estado de execución do programa nese instante e que se execute un programa de xestión da interrupción. Cando este programa remata, recupérase o estado de execución previo á interrupción e reanúdase a súa execución. Eventos que poden provocar interrupcións son: peticións do usuario (CTRL+ALT+supr), hibernación ou suspensión, operacións de E/S (p.ex., recepción de paquetes por rede, desbordamento de pila, conmutación entre programas que comparten o tempo de CPU, entre outras).