

Informática. Exame de Teoría.

Febreiro, 2007

PRIMEIRO PARCIAL:

1. (1 PUNTO) Cáis destas sentencias son obligatorias: `stop`, `program nome`, `end program nome`? Para qué serve cada unha?

SOLUCIÓN: As sentencias `program nome` e `stop` son optativas, e só a sentencia `end program nome` é obrigatoria. A primeira indica o comezo do programa principal, e o nome que se lle da. A segunda indica o remate da execución do programa, e pode haber varias sentencias `stop` nun programa. A terceira indica o remate do programa principal.

2. (1 PUNTO) Qué vantaxes ofrece un vector dinámico fronte a un vector estático en Fortran? Cómo podemos comprobar se se reservou memoria para un vector dinámico?

SOLUCIÓN: Nun vector estático temos que coñece-lo número de elementos no momento de declara-lo vector e, polo tanto, este número debe se-lo mesmo en todas as execucións do programa. Este número pode ser excesivo ou insuficiente para o número de elementos que queremos usar, e en tais casos desperdiciariamos memoria, ou non poderíamos executa-lo programa. Nun vector dinámico, o número de elementos non se ten que especificar na súa declaración, senón que podemos reservar memoria para el ao longo do programa. Deste modo, podemos usar exactamente o número de elementos que necesitemos en cada execución. Para determinar se se reservou memoria para un vector, podemos usa-la sentencia `allocated(..)`, que retorna `.true.` en tal caso.

3. (1 PUNTO) Escribir un segmento de código en Fortran que lea por teclado un número do 1 ao 7 e imprima por pantalla o nome do día de semana correspondente (luns-domingo).

SOLUCIÓN:

```
print *, "introduza un numero: "  
read *, n  
select case (n)  
case(1)  
    print *, "luns"  
case(2)  
    print *, "martes"  
...  
case default  
    print *, "desconhecido"  
end select
```

4. (1 PUNTO) Qué son os subprogramas internos en Fortran? Cómo se crean?

SOLUCIÓN: Son subprogramas (funcións externas ou subrutinas) que se atopan dentro doutro subprograma, logo da sentencia `contains`. Un subprograma interno non pode conter subprogramas internos. Dende o subprograma pódese chamar a tódolos seus subprogramas internos, e entre eles poden chamarse entre si, e poden acceder ás variábeis do subprograma ao que pertencen.

5. (2 PUNTOS) Explica brevemente (3 liñas) cómo funciona o algoritmo de ordeamento por selección. Cómo operaría este algoritmo co vector { 9, 3, 4, 6, 1 }?

SOLUCIÓN: Este algoritmo baséase en atopar-lo menor dos n números e intercambialo co número situado máis á esquerda, colocando o mínimo na primeira posición. Logo, atopa o menor dos $n - 1$ números da dereita, e intercámbiao co número situado na segunda posición. E así sucesivamente. Co vector indicado, o algoritmo operaría así:

```
9 3 4 6 1 => o minimo e o 1, intercambiaio co 9 e situa o 1 no extremo esquerdo
1 3 4 6 9 => os numeros xa estan ordeados, pero busca o minimo de 3 4 6 9 (3) e
            intercambiaio polo 3 (o vector non cambia)
1 3 4 6 9 => o minimo de 4 6 9 e 4, non varia
1 3 4 6 9 => o minio de 6 e 9 e 6, non varia, e remata
```

6. (2 PUNTOS) Escribir un algoritmo en pseudocódigo que lea por teclado un valor numérico x (clave). Logo, debe ler por teclado números repetidamente, comparalos con x e mostra-lo mesaxe superior á clave, inferior á clave ou igual á clave, según sexa, e neste último caso debe remata-la repetición.

SOLUCIÓN:

```
inicio
escribir "introduce a clave: "
ler clave
repetir
    escribir "introduce un numero: "
    ler n
    se n > clave
        escribir "superior a clave"
    se non se n < clave
        escribir "inferior a clave"
    se non
        escribir "igual a clave"
ata que n == clave
fin
```

7. (2 PUNTOS) 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:

```
character(len = 20) :: s
open(1, file = "datos.dat", status = "old", err = 1)
write (1, fmt = 2) n, x, s, y
2 format(i5, t10, e10.4, 2x, a20, t70, f8.3)
...
stop
1 print *, "erro en open"
stop
```

SEGUNDO PARCIAL:

8. (1 PUNTO) Escribe as sentencias en Fortran necesarias para declarar un punteiro `p` a enteiro, facer que apunte a unha variábel de tipo enteiro `x` con valor 5, facer que logo apunte a outra variábel enteira `y` con valor 10 e finalmente facer que non apunte a ningures.

SOLUCIÓN:

```
integer, target :: x = 5, y = 10
integer, pointer :: p
p => x
p => y
nullify(p)
```

9. Explica brevemente (3 liñas cada un):

- (1 PUNTO) As diferencias entre a memoria RAM e a memoria caché.

SOLUCIÓN: A memoria RAM está composta por celdas DRAM externas ao microprocesador e a memoria caché está composta por celdas SRAM internas ao microprocesador. As primeiras ocupan menos espazo, pero son máis lentas, xa que conteñen un transistor e un condensador e requiren refrescar periódicamente a carga do condensador, o cal leva tempo e reduce a súa velocidade. As celdas SRAM non teñen condensador e non requiren refresco algún, polo que son máis rápidas, pero ocupan máis espazo, de modo que a cantidade de celdas que se poden meter no microprocesador son menores, e a súa capacidade é inferior.

- (1 PUNTO) O funcionamento da lectura e escritura nun CD-RW .

SOLUCIÓN: Un CD-RW contén unha aleación de prata, antimonio, indio e teluro que ten dous estados estábeis: un cristalino, con alto índice de reflexión, e outro amorfo, cun índice de reflexión baixo. No proceso de escritura do CD-RW, o láser pode incidir sobre esta aleación cunha potencia alta, en cuxo caso transforma a aleación de cristalino a amorfo (escritura dun bit 1), ou ben cunha potencia media, transformando a aleación de amorfa a cristalina (escritura de bit 0). Se o láser incide cunha potencia baixa, non transforma o estado da aleación (lectura), obténdose un bit 1 se hai reflexión (estado cristalino) e un bit 0 se non a hai (estado amorfo).

- (1 PUNTO) As características do formato MP3 que lle permiten comprimir a información de audio. Qué factor de compresión pode acadar? **SOLUCIÓN:** O formato MP3 usa filtros para dividi-la sinal de audio en 32 canles de frecuencia, dedicando máis bits de resolución para aquelas canles (rangos de frecuencia) con meirande amplitude, e tamén usa máis bits de resolución nas canles nas que o oído humano ten máis sensibilidade. Deste modo acada un factor de compresión de 12:1 (8.33 %)

10. (2 PUNTOS) Explica brevemente o que fai o seguinte segmento de pseudocódigo. Qué estrutura de datos está implicada? Qué fan as funcións `incognita1(...)` e `incognita2(...)`? Qué situacións se xestionan nesta función? Fai un gráfico representando cada un destes casos.

funcion incognita1(p, x)	
p, q: punteiros a nodo	se (q == p) e (q == f)
x : tipo X	p = NULO
inicio	f = NULO
	se non
	se q == p
q = p	p = p^.enlace_der
mentres (q /= NULO) e (q^.dato /= x)	p^.enlace_esq = NULO
q = q^.enlace_der	se non
fin mentres	se q == f
retornar q	f = f^.enlace_esq
fin	f^.enlace_der = NULO
-----	se non
funcion incognita2(p, f, ref)	t = q^.enlace_esq
p, f, r, t: punteiros a nodo	r = q^.enlace_der
ref : tipo X	t^.enlace_der = r
inicio	r^.enlace_esq = t
	fin_se
q = incognita1(p, ref)	fin_se
	fin_se
	destrue(q)
	se non
	escribir "Elemento ref non atopado"
	fin_se
	fin

SOLUCIÓN: A primeira función busca un nodo con dato x nunha lista dobremente enlazada, retornando un punteiro ao nodo buscado (se o atopa) ou un punteiro nulo se non o atopa. A segunda función suprime o nodo con dato ref dunha lista dobremente enlazada, chamando primeiro á primeira función para que o busque.

- Se non o atopa, retorna un punteiro nulo e incognita2() mostra a mesaxe de erro.
- Se o atopa, incognita2() comproba se este nodo é o único da lista (igual ao primeiro e ao último). En tal caso, pon a nulo p e f (a lista queda baleira).
- Se q é o primeiro nodo pero non o último, p pasa a apuntar ao segundo nodo, e o seu enlace esquerdo faise nulo.
- Se q apunta a f pero non a p, f pasa a apuntar ao nodo anterior a q (penúltimo) e o enlace dereito de f faise nulo.
- Se q é distinto de p e de f, o nodo a borrar está no medio da lista: o enlace dereito do nodo anterior a q pasa a apuntar ao enlace dereito de q (r), e o enlace esquerdo do nodo seguinte a q pasa a apuntar ao enlace esquerdo de q (t).

Finalmente, incognita2() libera a memoria do nodo q que se desexa borrar

11. (1 PUNTO) Explica brevemente (2 liñas) as diferencias entre as funcións `round()`, `fix()`, `floor()` e `ceil()`. Cómo operan sobre o vector `[-1.1 -0.6 -0.1 0.1 0.6 1.1]`?

SOLUCIÓN: `round()` redondea cara o enteiro máis cercano. `fix()` redondea cara 0. `floor()` redondea cara $-\infty$ e `ceil()` redondea cara $+\infty$. Neste exemplo, temos:

```

round : -1    -1    0    0    1    1
fix :   -1    0    0    0    0    1
floor: -2    -1   -1    0    0    1
ceil:  -1    0    0    1    1    2

```

12. (1 PUNTO) Escribe os comandos de Matlab necesarios para calcular:

$$\int_0^{\infty} \frac{\sin^3 x}{1 + \cos x + e^x} dx \quad (1)$$

$$\sum_{n=1}^{\infty} \frac{2 + \cos \frac{\pi n}{5}}{n} \quad (2)$$

SOLUCIÓN:

```

syms x
int(sin(x)^3/(1 + cos(x) + exp(x)), x, 0, inf)
syms n
symsum((2 + cos(pi*n/5))/n, n, 1, inf)

```

13. (2 PUNTOS) Escribe un programa en Matlab que lea por teclado un número $x_i \in (0, 1)$ e calcule e imprima por pantalla:

$$x_{i+1} = \begin{cases} \frac{1}{1+x_i} & \text{mod}(i, 3) = 0 \\ \frac{1}{1+x_i^2} & \text{mod}(i, 3) = 1 \\ \frac{1}{1+x_i^3} & \text{mod}(i, 3) = 2 \end{cases}$$

SOLUCIÓN:

O programa debe calcular x_{i+1} iterativamente hasta que $|x_{i+1} - x_i| < 10^{-5}$. A función $\text{mod}(i, j)$ é o resto da división enteira i/j .

SOLUCIÓN:

```

clear all

xi = input('introduza un numero en (0, 1): ');
i = 0; error = inf;
while error > 1e-5
    n = rem(i, 3);
    if n == 0
        xi1 = 1/(1 + xi);
    elseif n == 1
        xi1 = 1/(1 + xi^2);
    else
        xi1 = 1/(1 + xi^3);
    end
    error = abs(xi1 - xi);
    fprintf('xi1 = %.3f, error = %.5f\n', xi1, error);
    xi = xi1; i = i + 1;
end

```