

Informática. Exame de Teoría. 10 de decembro de 2007

Nome:

1. (1 PUNTO) Dado un real de dobre precisión en Fortran: Cantos bytes ocupa en memoria RAM? Cómo se pode declarar?

SOLUCIÓN Un real de dobre precisión ocupa 8 bytes (64 bits) e pódese declarar como `real(8) :: x`.

2. (1 PUNTO) Qué significa a sentencia `implicit integer (a-c) real (d-z)`? Dada a declaración de vector `integer, dimension(-500:103) :: x`, cantos elementos ten este vector?

SOLUCIÓN a) A sentencia indica que en adiante, no subprograma actual, as variábeis que comecen polas letras a-c serán enteiras por defecto (é dicir, se non se declaran explícitamente doutro tipo) e as restantes serán reais por defecto. b) O vector ten $N = 103 - (-500) + 1 = 103 + 500 + 1 = 604$ elementos.

3. (0.5 PUNTOS) Escribe as declaración dun vector dinámico e dun vector estático en Fortran. Qué diferenza hai entre eles?

SOLUCIÓN a) Vector estático: `real, dimension(10) :: x`. Vector dinámico:

```
real, dimension(:), allocatable :: x
...
n = ...
allocate(x(n))
...
deallocate(x)
```

b) A diferenza é que no estático hai que saber exactamente o número de elementos (que debe ser unha constante) na declaración, constante igual en tódalas execucións do programa. Nembargantes, no vector dinámico non sabemos o número de elementos na declaración, senón que se coñece posteriormente durante a execución do programa (e pode ser distinta en execucións distintas), momento en que se reserva memoria con `allocate(...)`. Finalmente, hai que libera-la memoria reservada con `deallocate(...)`.

4. (0.5 PUNTOS) Se $x = y = z = 1$, cálc a orde de execución e o valor final da expresión $x - 5 > 3.5$.or. $y/z - 3 == 1$?

SOLUCIÓN Primeiro execútanse os operadores aritméticos, logo os relacionais e logo os lóxicos. O resultado é lóxico e vale `.false.`:

```
x - 5 > 3.5 .or. y/z - 3 == 1
1 - 5 > 3.5 .or. 1/1 - 3 == 1
-4 > 3.5 .or. 1 - 3 == 1
-4 > 3.5 .or. -2 == 1
.false. .or. .false.
.false.
```

5. (2 PUNTOS) Traduce a Fortran e a diagrama de fluxo o seguinte fragmento de pseudocódigo:

```

x = (y - 5)/2
se x < 4
    repetir
        x = x - 1
    ata que x == 0
se non se x < 10
    mentres x > 4
        x = x - 2
    fin mentres
se non
    dende i = x ata 0 paso -1
        x = x + i
    fin dende
fin se

```

SOLUCIÓN En Fortran:

```

x = (y - 5)/2
if(x < 4) then
    do
        x = x - 1
        if(x == 0) exit
    end do
else if(x < 10) then
    do
        if(x <= 4) exit
        x = x - 2
    end do
else
    do i = x, 0, -1
        x = x + i
    end do
end if

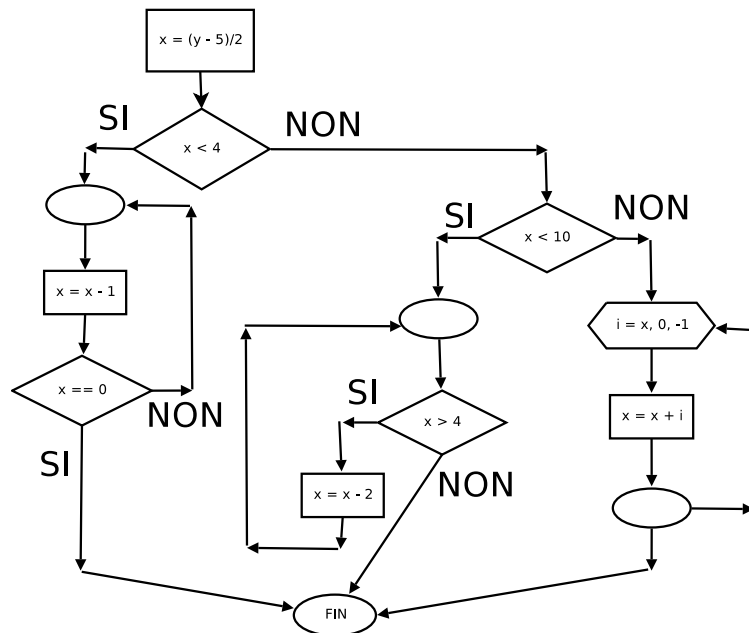
```

En diagrama de fluxo (ver figura en páxina seguinte):

6. **(2 PUNTOS)** Qué algoritmo de ordeamento se basea en intercambiar pares de elementos consecutivos? Cál deles intercambia elementos non consecutivos? Qué algoritmo de ordeamento sitúa na dereita do elemento central os elementos maiores ca éste e na súa esquerda os elementos menores, repetindo o procedemento coas listas esquerda e dereita? Por qué o algoritmo Shell é máis eficiente que o da burbulla?

SOLUCIÓN

- a) O algoritmo de intercambio directo (burbulla) é o que intercambia elementos consecutivos, situando en cada percorrido do vector o elemento meirande no extremo dereito dos elementos chequeados.
- b) O algoritmo Shell é o que intercambia elementos non consecutivos, separados por un incremento que vai decrecendo a medida que avanza o ordenamento: por iso se chama algoritmo de ordeamento por incrementos decrecentes.



c) O algoritmo quicksort é o que intercambia elementos á esquerda do elemento central meirandes co central con elementos á dereita do mesmo pero menores co mesmo: deste modo sitúa unha lista de elementos menores co central á esquerda, e outra lista con elementos meirandes á dereita, procedendo logo recursivamente.

d) O algoritmo Shell (complexidade $\mathcal{O}(n^{1,2})$) é máis eficiente co da burbulla (complexidade $\mathcal{O}(n^2)$) porque no principio realiza un ordeamento a grande escala (de grano grosso) situando os elementos meirandes ao final do vector e os pequenos ao comezo, graciñas a que pode intercambiar elementos non consecutivos. Nas etapas finais, cando o incremento é pequeno, realiza o ordeamento de grao fino.

7. (1 PUNTO) Escribi-la sentencia en Fortran necesaria para que un subprograma sen argumentos comparta dúas variábeis enteiras e unha real (supoñendo declaración implícita) con outros subprogramas.

SOLUCIÓN A sentencia `common`: `common /nome_bloque/ n, m, x`

8. (2 PUNTOS) Escribe unha sentencia en Fortran que abra un arquivo para escribir nel sen sobrescribilo en caso de que exista. Escribe outra sentencia que abra un arquivo existente para ler del. Qué teriamos que facer neste último caso para escribir no final do arquivo? Escribe unha sentencia que almacene no arquivo un valor real con 5 cifras decimais, outro valor real en formato exponencial con 3 cifras decimais, unha cadea de caracteres e un valor enteiro na columna 70.

SOLUCIÓN

a) `open(1, file = "datos.dat", status = "new", err = 1)`: así da erro se o arquivo xa existe.

b) `open(1, file = "datos.dat", status = ".old", err = 1)`: así da erro se o arquivo non existe.

c) Hai que ler todo o arquivo e logo, cando chegamos ao final, escribir nel:

```
open(1, file = "datos.dat", status = "old", err = 1)
do
```

```
      read (1, *, end = 1) x, y, z  
end do  
1 write (1, *) a, b, c
```