# Resolução do Exame de Programação e Computadores 09/01/1999

SMF / DEC, FEUP

Janeiro de 1999

Estas folhas contêm **uma** resolução do Exame de Programação e Computadores. Tenha em atenção que existem múltiplas resoluções alternativas, igualmente correctas, para estas questões.

Os programas transcritos estão brevemente comentados, mas eventualmente para os compreender deverá confrontar com o enunciado e meditar sobre o problema...

## Grupo I

(a)
```
PROGRAM exame_1a
  IMPLICIT NONE
  INTEGER ::  face, n, impares
  !  contador do número de impares
  impares = 0
  DO n = 1, 100 !  100 lançamentos
    CALL lancar_dado(face)
    !  se o lançamento for impar conta mais um:
    IF(MOD(face,2)/=0) impares = impares+1
  END DO
  WRITE(*,*) 'No.  de impares:', impares
  STOP
CONTAINS
  SUBROUTINE lancar_dado(face)
  ⋮
  END SUBROUTINE lancar_dado
END PROGRAM exame_1a
```

(b)
```
PROGRAM exame_1b
  IMPLICIT NONE
  INTEGER ::  face1, face2, n, seis
  !  contador do número de somas 6:
  seis = 0
  CALL lancar_dado(face1) !  1o.  lançamento
  DO n = 2, 100 !  99 lançamentos
    CALL lancar_dado(face2)
    IF(face1 + face2 == 6) seis = seis+1
    face1 = face2
  END DO
```

```
    WRITE(*,*) 'No.  de somas 6 em consecutivos:', seis
    STOP
  CONTAINS
    SUBROUTINE lancar_dado(face)
    ⋮
    END SUBROUTINE lancar_dado
  END PROGRAM exame_1a
```

NOTA: nos exercícios I.(a) e I.(b) seria também aceitável considerar o subprograma dado como externo e usar uma declaração "`INTERFACE ...END INTERFACE`" no programa principal.

# Grupo II

```
PROGRAM exame_2
  IMPLICIT NONE
  INTEGER ::  h1,m1, h2,m2 !  hora e min.  de entrada e saida
  INTEGER ::  total_min, total_h, preco
  WRITE(*,*) 'Hora e minuto de entrada:'
  READ(*,*) h1,m1
  WRITE(*,*) 'Hora e minuto de saida:'
  READ(*,*) h2,m2
  !  calcular o total de minutos dentro do parque:
  total_min = h2*60 + m2 - h1*60 - m1
  !  calcular o total de horas (arredondado para cima):
  total_h = total_min/60
  IF(MOD(total_min,60)>0) total_h = total_h + 1
  !  calcular o preco a pagar:
  IF(total_h <= 1) THEN
    preco = 100
  ELSE IF(total_h <= 3) THEN
    preco = 100 + (total_h-1)*130
  ELSE
    preco = 100 + 2*130 + (total_h-3)*200
  END IF
  WRITE(*,*) 'Preco:', preco
  STOP
END PROGRAM exame_2
```

# Grupo III

(a)
```
FUNCTION desde_1900(dia, mes, ano) RESULT(total)
  IMPLICIT NONE
  INTEGER, INTENT(IN) ::  dia, mes, ano
  INTEGER ::  total, dias, n,m
  !  'total' vai contar o no.  de dias desde 1 de Janeiro de 1900
  total = 0
  !  'n' percorre todos os anos desde 1900 até ao penúltimo:
  DO n = 1900, ano-1
```

```
    !  se um ano é bissexto conta + 366 dias; senão conta + 365:
    IF((MOD(n,4)==0.AND.MOD(n,100)/=0) .OR. MOD(n,400)==0) THEN
       total = total + 366
    ELSE
       total = total + 365
    END IF
  END DO
  !  'm' percorre todos os meses desde Janeiro até ao penúltimo:
  DO m = 1, mes-1
    !  calcula o no.  de dias de cada mes:
    SELECT CASE(m)
        CASE(4,6,9,11) !  abril, jun., set.  e nov.
            dias = 30
        CASE(2) !  fevereiro
            IF((MOD(ano,4)==0.AND.MOD(ano,100)/=0) .OR. &
               & MOD(ano,400)==0) THEN
               dias = 29 !  ano bissexto
            ELSE
               dias = 28 !  ano normal
            END IF
        CASE DEFAULT ! outros meses
            dias = 31
    END SELECT
    total = total+dias
  END DO
  !  por fim soma os dias do último mes:
  total = total + dia-1
  RETURN
END FUNCTION desde_1900
```

(b)
```
PROGRAM exame_3
  IMPLICIT NONE
  INTEGER ::  dia1,mes1,ano1, dia2,mes2,ano2, n
  WRITE(*,*) 'Introduza a 1a.data (dia, mes, ano):'
  READ(*,*) dia1,mes1,ano1
  WRITE(*,*) 'Introduza a 2a.data (dia, mes, ano):'
  READ(*,*) dia2,mes2,ano2
  !  calcula a diferenca entre dias desde 1900 em valor absoluto
  !  pois não sabemos qual das datas é posterior:
  n = ABS(desde_1900(dia1,mes1,ano1) - desde_1900(dia2,mes2,ano2))
  WRITE(*,*) 'Diferenca entre datas:', n
  STOP
CONTAINS
  FUNCTION desde_1900(...)  ...
  ⋮
  END FUNCTION desde_1900
END PROGRAM exame_3
```

# Grupo IV

(a) Apresentamos duas resoluções desta alínea: a primeira recorre intensivamente à função intrínsica `SUM` de FORTRAN 90 para somar valores de variáveis indexadas:

```
FUNCTION correlate(x,y) RESULT(rho)
  IMPLICIT NONE
  REAL, INTENT(IN) ::  x(:), y(:)  !  dois vectores reais
  REAL ::  rho, xm, ym
  !  nota:  SIZE(x) deve ser igual a SIZE(y) ...
  xm = SUM(x)/SIZE(x) !  média do vector x
  ym = SUM(y)/SIZE(y) !  média do vector y
  !  cálculo do coeficiente de correlação
  !  recorde-se do que significa usar +, *, -, **, ...  com vectores
  rho = SUM((x-xm)*(y-ym)) / SQRT(SUM((x-xm)**2)*SUM((y-ym)**2))
  RETURN
END FUNCTION correlate
```

A segunda resolução efectua todas as somas usando ciclos explícitos:

```
FUNCTION correlate(x,y) RESULT(rho)
  IMPLICIT NONE
  REAL, INTENT(IN) ::  x(:), y(:)  !  dois vectores reais
  REAL ::  rho, xm, ym, sxx, sxy, syy
  INTEGER ::  n, i
  !  calcular médias dos vectores x e y:
  n = SIZE(x) !  nota:  SIZE(x) deve ser igual a SIZE(y)
  xm = 0
  ym = 0
  DO i = 1, n
    xm = xm + x(i)
    ym = ym + y(i)
  END DO
  xm = xm/n !  média do vector x
  ym = ym/n !  média do vector y
  !  calcular as somas SXX SXY e SYY:
  sxx = 0
  syy = 0
  sxy = 0
  DO i = 1, n
    sxx = sxx + (x(i)-xm)**2
    syy = syy + (y(i)-ym)**2
    sxy = sxy + (x(i)-xm)*(y(i)-ym)
  END DO
  !  calcular o coeficiente de correlação:
  rho = sxy/SQRT(sxx*syy)
  RETURN
END FUNCTION correlate
```

Claro está, uma resolução mista que combine partes destas duas seria ainda aceitável.

Qualquer destes subprogramas calcula internamente a média dos vectores dados (como era pretendido); contudo, foram ainda aceites resoluções em que as médias eram calculadas externamente, pois o enunciado do problema poderia induzir essa interpretação.

Note-se ainda que o valor de correlação $\rho(cal, glu)$ dado no enunciado não foi calculado correctamente (embora o seu valor não fosse necessário para escrever o programa).

(b)
```
PROGRAM exame_4
  IMPLICIT NONE
  REAL, ALLOCATABLE ::  tab(:,:)  !  tabela de alimentos
  INTEGER ::  i,j,n
  REAL ::  rho_cal_glu, rho_cal_prot, rho_cal_lip

  WRITE(*,*) 'No.de alimentos?'
  READ(*,*) n

  ALOCATE(tab(n,4)) !  4 colunas:  cal, glu, prot, lip
  READ(*,*) ((tab(i,j), j=1,4), i=1,n)

  rho_cal_glu = correlate(tab(:,1), tab(:,2))
  rho_cal_prot = correlate(tab(:,1), tab(:,3))
  rho_cal_lip = correlate(tab(:,1), tab(:,4))
  WRITE(*,*) 'rho(cal,glu)=', rho_cal_glu
  WRITE(*,*) 'rho(cal,prot)=', rho_cal_prot
  WRITE(*,*) 'rho(cal,lip)=', rho_cal_lip
  STOP
CONTAINS
  FUNCTION correlate(...)
  ⋮
  END FUNCTION correlate
END PROGRAM exame_4
```

Para este programa principal os dados da tabela do exemplo deveriam ser introduzidos pela seguinte ordem:

- o nº de alimentos;
- os valores numéricos da tabela por linhas.

Assim a sequência de dados correspondente ao exemplo seria:

```
6, 239, 49, 1.2, 8, 354, 77, ..., 22, 4, 0.3, 1
```