Control de programación en Fortran

Escribe un programa en fortran que lea do arquivo `matriz.dat` unha matriz cadrada a de orde n. Define un subprograma chamado `calculos()` (debes decidir o tipo e argumentos que necesita) que calcule a suma dos elementos pares de a e conte o número de elementos impares en a. O programa principal ten que chamar ó subprograma e visualizar os resultados na pantalla: a matriz a (unha fila en cada liña), a suma dos elementos pares e o número de elementos impares.

Por exemplo, se o arquivo `matriz.dat` contén:

```
4
1 2 4 6
4 3 7 5
4 8 0 5
6 4 7 1
```

onde a primeira liña do arquivo é a orde da matriz, o resultado sería a suma 38 e o número de elementos impares 7.

```
program control1
integer, dimension(:,:), allocatable ::a
open(1, file="matriz.dat",status="old", err=4)
read(1,*) n
allocate(a(n,n))
print*, "Matriz a:"
do i=1,n
  read(1,*) (a(i,j), j=1,n)
  print*, (a(i,j), j=1,n)
end do
close(1)
call calculos(a, n, nsuma, nimpares)
print*, "Suma pares = ", nsuma
print*, "Numero de impares= ", nimpares
deallocate(a)
stop
4 print*, "Erro abrindo matriz.dat"
end program control1

!*********************************
! Definicion de subprograma
subroutine calculos(b, m, suma, impares)
integer, dimension(m,m), intent(in) :: b
integer, intent(in) :: m
integer, intent(out) :: impares, suma
suma = 0
impares = 0
do i=1,m
  do j=1,m
    if(mod(b(i,j), 2) == 0) then
      suma = suma + b(i,j)
    else
      impares = impares + 1
    end if
  end do
end do
return
end subroutine calculos
```

Control de programación en Fortran

Escribe un subprograma `polinomio()` (debes decidir o tipo e argumentos) que calcule o n-ésimo polinomio de Tchevyshev para un valor de x, definido por:

$$P_n(x) = \prod_{k=1}^{n}(kx - 2) \tag{1}$$

Escribe un programa principal que pida ó usuario a orde do polinomio n e chame ao subprograma `polinomio()` para que calcule o n-ésimo polinomio para valores de x=0, 0.1, 0.2, 0.3,0.4,0.5,... mentres que o valor do polinomio sexa positivo. O programa ten que avaliar o número de polinomios calculados, se este número e maior que 10 mostrar a mensaxe "Menos de 10 polinomios" e en caso contrario mostrar a mensaxe "Máis de 10 polinomios". O programa principal ten que gardar nun arquivo chamado `control.dat` o valor de x e o do polinomio (un en cada liña). Por exemplo, se consideras $n = 7$ o arquivo `control.dat` debe conter:

```
 0.00000000      -128.000000
 0.100000001      -25.3955517
 0.200000003      -2.32243180
```

```
program control2
integer :: contador
print*, "Introduce n: "
read*, n
open(2, file="control.dat", status="new", err=5)
x=0; h=0.1; contador = 0
do
  pn=polinomio(n,x)
  if (pn > 0) exit
  write(2,*) x, pn
  contador = contador + 1
  x = x+h
end do
close(2)
if(contador < 10) then
  print*, "Menos de 10 polinomios"
else
  print*, "Mais de 10 polinomios"
end if
stop
5 print*, "Erro abrindo control.dat"
end program control2
!*************************************
! Definicion de subprograma
function polinomio(n, x)
real, intent(in):: x
integer, intent(in) :: n
  pnx = 1
  do k = 1, n
      pnx = pnx*(k*x-2)
  end do
  polinomio = pnx
return
end function polinomio
```

Control de programación en Fortran

Escribe un programa que lea por teclado repetidamente números enteiros ata que o número de elementos pares e impares sexa igual. Sexa n o número de elementos pares introducidos, tes que definir un subprograma chamado `xera_matriz()` (debes decidir o tipo e argumentos) que xere unha matriz b de orde $n \times 2n$, onde cada elemento $b_{ij} = ij$, $i = 1, \ldots, n$, $j = 1, \ldots, 2n$.

Por exemplo, se introduces os números 2,4,5,6,8,3,1,7, o programa debe rematar de ler números logo de introducir o 7. Nese caso $n = 4$, e o arquivo `matrizb.dat` contén:

```
1          2          3          4          5          6          7          8
2          4          6          8         10         12         14         16
3          6          9         12         15         18         21         24
4          8         12         16         20         24         28         32
```

```
program control3
integer, dimension(:,:), allocatable :: b
npar=0; nimpar=0
do
  print*, "Introduce numero: "
  read*, n
  if (mod(n,2) == 0) then
    npar = npar + 1
  else
    nimpar = nimpar + 1
  end if
  if (npar == nimpar) exit
end do
n=npar
allocate(b(n,2*n))
call xera_matriz(b, n)
open(3,file="matrizb.dat", status="new", err=6)
n2=2*n
do i=1,n
  write(3,*) (b(i,j), j=1,n2)
end do
close(3)
deallocate(b)
stop
6 print*, "Erro abrindo matrizb.dat"
end program control3
!*********************************
subroutine xera_matriz(a, m)
integer, dimension(m,m), intent(out) :: a
integer, intent(in)::m
m2=2*m
do i=1,m
  do j=1,m2
    a(i,j)=i*j
  end do
end do
return
end subroutine xera_matriz
```

Control de programación en Fortran

Define un subprograma chamado `cifras` (debes decidir o tipo e argumentos) que descompoña un número enteiro nas súas cifras. O programa principal ten que ler por teclado dous números x e y e almacenar no arquivo `cifras.dat` as cifras do número x (todas nunha única liña), as cifras do número y e noutra liñha as cifras comúns a ambos números. Usa vectores estáticos de 50 elementos. Por exemplo, se introduces os números $x = 123456$ e $y = 13682$, o arquivo `cifras.dat` contén:

```
Cifras x: 6 5 4 3 2 1
Cifras y: 2 8 6 3 1
Cifras comuns: 6 3 2 1
```

```
program control4
integer, dimension(50) :: xc, yc, cc
integer :: x, y
print*, "Introduce x e y: "
read*, x, y
call cifras(xc, 50, x, nxc)
call cifras(yc, 50, y, nyc)
open(2, file="cifras.dat", status="new", err=7)
write(2,*) "Cifras x: ", (xc(i), i=1,nxc)
write(2,*) "Cifras y: ", (yc(i), i=1,nyc)
ncc=0
do i=1, nxc
  do j=1,nyc
    if( xc(i) == yc(j)) then
        ncc = ncc + 1
        cc(ncc) = xc(i)
        exit
    end if
  end do
end do
write(2,*) "Cifras comuns: ", (cc(i), i=1,ncc)
close(2)
stop
7 print*, "Erro abrindo arquivo cifras.dat"
end program control4
!*****************************
subroutine cifras(v, n, x, nc)
integer, intent(in) ::x, n
integer, dimension(n), intent(out) :: v
integer, intent(out) :: nc
m=x
nc=1
do
  v(nc)=mod(m,10)
  m=m/10
  if (m == 0) exit
  nc = nc + 1
end do
print*, "Cifras de ", x , " son: ", (v(i), i=1,nc)
return
end subroutine cifras
```

Control de programación en Fortran

Escribe un programa que lea dos vectores **v** e **w** de dimensión n dende o arquivo `vectores.dat`. Define un subprograma chamado `construir_matriz()` (debes decidir o tipo e argumentos) que calcule unha matriz **a** de orde $n \times n$, onde cada elemento da matriz a_{ij} ven dado pola expresión:

$$a_{ij} = \begin{cases} \sum_{k=1}^{i} v_k^2 & j \leq i \\ w_j v_i & j > i \end{cases}$$

O programa principal ten que chamar ao subprograma `construir_matriz()` para calcular a matriz **a** e visualizala na pantalla (unha fila en cada liña). Por exemplo, se o arquivo `vectores.dat` conten os seguintes vectores:

```
5
2 3 4 5 6
2 1 5 4 3.1
```

onde a primeira liña é a dimensión dos vectores, a segunda o vector **v** e a terceira o vector **w**, a matriz calculada **a** sería:

```
4.00000000      2.00000000      10.0000000      8.00000000      6.19999981
13.0000000      13.0000000      15.0000000      12.0000000      9.29999924
29.0000000      29.0000000      29.0000000      16.0000000      12.3999996
54.0000000      54.0000000      54.0000000      54.0000000      15.5000000
90.0000000      90.0000000      90.0000000      90.0000000      90.0000000
```

```
program control5
real, dimension(:),allocatable ::v, w
real, dimension(:,:), allocatable ::a
open(2, file="vectores.dat", status="old", err=5)
read(2, *) n
allocate(v(n), w(n), a(n,n))
read(2,*)(v(i), i=1,n)
read(2,*)(w(i), i=1,n)
call construir_matriz(v,w, a, n)
print*, "Matriz a:"
do i=1,n
    print*, (a(i,j), j=1,n)
end do
deallocate(v,w,a)
stop
5 print*, "Erro abrindo vectores.dat"
end program control5
!*************************************
! Definicion de subprograma
subroutine construir_matriz(v, w, b, m)
real, dimension(m), intent(in) :: v, w
real, dimension(m,m), intent(out) :: b
integer, intent(in)::m
do i=1,m
  do j=1,m
    if (j <= i) then
        b(i,j)=0
        do k=1,i
            b(i,j)= b(i,j) + v(k)*v(k)
```

```
        end do
    else
        b(i,j)=w(j)*v(i)
    end if
  end do
end do
return
end subroutine construir_matriz
```