PROGRAMAS EXTRAS – FORTRAN90/95

2. Modulos – Modules

1. Input e Output

Exemplo 1.1 Output formatado e escrito em arquivo.

```
PROGRAM TRIG_TABLE
! Compiles a table of SIN, COS, TAN against angle in DEGREES
    IMPLICIT NONE
    INTEGER DEG                                ! angle in degrees
    REAL RAD                                   ! angle in radians
    REAL PI                                    ! mathematical pi
    CHARACTER (LEN=*), PARAMETER :: FMTHEAD = '( 1X, A3, 3( 2X, A7) )'
    CHARACTER (LEN=*), PARAMETER :: FMTDATA = '( 1X, I3, 3( 2X, F7.4 ) )'
                                              ! formats for headings and data
    PI = 4.0 * ATAN( 1.0 )
    WRITE( *, FMTHEAD ) 'Deg', 'Sin', 'Cos', 'Tan'
    DO DEG = 0, 80, 10
       RAD = DEG * PI / 180.0
       WRITE( *, FMTDATA ) DEG, SIN( RAD ), COS( RAD ), TAN( RAD )
    END DO
    STOP
END PROGRAM TRIG_TABLE
```

Forma alternativa:

```
   WRITE( *,'( 1X, A3, 3( 2X, A7 ) )' ) 'Deg', 'Sin', 'Cos', 'Tan'
   …
      WRITE( *,'( 1X, I3,3( 2X, F7.4 ) )' ) DEG, SIN( RAD ), COS( RAD ),TAN( RAD )
```

ou:

```
   100 FORMAT( 1X, A3, 3( 2X, A7
                                 ))
   110 FORMAT( 1X, I3, 3( 2X, F7.4 ) )
   …
   WRITE( *, 100 ) 'Deg', 'Sin', 'Cos', 'Tan'
   …
      WRITE( *, 110 ) DEG, SIN( RAD ), COS( RAD ), TAN( RAD )
   …
```

Escrevendo em arquivo:

```
   OPEN( 33, FILE = 'trig.out' )                    ! open file for output
   WRITE( 33, FMTHEAD ) 'Deg', 'Sin', 'Cos', 'Tan'
   …
      WRITE( 33, FMTDATA ) DEG, SIN( RAD ), COS( RAD ), TAN( RAD )
   …
   CLOSE( 33 )                                     ! close file (tidiness is a virtue!)
```

Exemplo 1.2 Ilustra saída de número em formato de ponto flutuante (fXX.X)

```
PROGRAM EXPTABLE
! Program tabulates EXP(X)
     IMPLICIT NONE
     INTEGER :: NSTEP = 15                              ! number of steps
     REAL :: XMIN = 0.0, XMAX = 3.0                     !interval limits
     REAL DELTAX                                        !step size
     REAL X                                             !current X value
     INTEGER I                                          !a counter
     ! Format specifiers
     CHARACTER (LEN=*), PARAMETER :: FMT1 = '( 1X, A4 , 2X,A10)'
     CHARACTER (LEN=*), PARAMETER :: FMT2 = '( 1X, F4.2, 2X, 1PE10.3 )'
     DELTAX = ( XMAX - XMIN ) / NSTEP                   ! calculate step size
     WRITE( *, FMT1 ) 'X', 'EXP'                        ! write headers
```

```
    DO I = 0, NSTEP
       X = XMIN + I * DELTAX                          ! set X value
       WRITE( *, FMT2 ) X, EXP( X )                   ! write data
    END DO
END PROGRAM EXPTABLE
```

Exemplo 1.3 Ilustrando READ formatado, sem avanço i/o e IOSTAT = especificador. Nota: Um arquivo de entrada text.dat é requerido: qualquer texto será possível.

```
 PROGRAM ANALYSE_TEXT
   IMPLICIT NONE
   INTEGER :: IO = 0                                 ! holds i/o status
   INTEGER :: NLETTERS = 0                           ! number of letters read
   INTEGER :: NWORDS = 0                             ! number of words read
   CHARACTER CH, LAST_CH                             ! successive characters
   CHARACTER (LEN=*), PARAMETER :: ALPHABET = &
          'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
   CHARACTER, PARAMETER :: SPACE=' '
   LAST_CH = SPACE
   ! Open the text file
   OPEN( 10, FILE = 'text.dat' )
   ! Read characters one-by-one until end of file is reached
   DO WHILE ( IO /= -1 )                              ! IO=-1 means EOF
      ! Read one character
      READ( 10, '( A1 )', IOSTAT = IO, ADVANCE = 'NO' ) CH
      IF ( IO == 0 ) THEN                            ! a character has been read
         PRINT *, 'Character = ', CH
         ! Is it a new word?
         IF (LAST_CH == SPACE .AND. CH /= SPACE ) NWORDS = NWORDS + 1
         ! Is it a letter of the alphabet or something else?
         IF ( INDEX( ALPHABET, CH ) /= 0 ) NLETTERS = NLETTERS + 1
         LAST_CH = CH                                ! update last character
      ELSE                                           ! end of line or end of file
         PRINT *, 'IO = ', IO
         LAST_CH = SPACE
      END IF
   END DO
   ! Close the text file
   CLOSE( 10 )
   ! Output the analysis
   PRINT *, 'Number of letters = ', NLETTERS
   PRINT *, 'Number of words = ', NWORDS
   STOP
END PROGRAM ANALYSE_TEXT
```

2. Modulos – Modules

Exemplo 2.1 Ilustra o uso de modules para compartilhar parâmetros e variáveis.

```
MODULE CONVERSION
! Length conversion factors
   IMPLICIT NONE
   REAL, PARAMETER :: MILES_TO_METRES  = 1609.0
   REAL, PARAMETER :: YARDS_TO_METRES  = 0.9144
   REAL, PARAMETER :: FEET_TO_METRES   = 0.3048
   REAL, PARAMETER :: INCHES_TO_METRES = 0.0254
END MODULE CONVERSION

PROGRAM DISTANCE
   USE CONVERSION ! make conversion factors available
   IMPLICIT NONE
   REAL METRES    ! distance in metres
```

```
    REAL AMOUNT      ! numerical quantity
    CHARACTER UNITS  ! units (i-inches, f-feet, y-yards, m-miles)
    PRINT *, 'Input amount and units (i-inches, f-feet, y-yard, m-mile)'
    READ *, AMOUNT, UNITS
    SELECT CASE (UNITS)
       CASE ( 'i' , 'I');METRES=AMOUNT*INCHES_TO_METRES
       CASE ( 'f' , 'F');METRES=AMOUNT*FEET_TO_METRES
       CASE ( 'y' , 'Y');METRES=AMOUNT*YARDS_TO_METRES
       CASE ( 'm' , 'M');METRES=AMOUNT*MILES_TO_METRES
    END SELECT
    PRINT *, 'Distance in metres = ', METRES
    STOP
END PROGRAM DISTANCE
```

Exemplo 2.2 Ilustrando modules e CONTAIN

```
MODULE PHYSICS
! Physical constants and some useful subprograms
   IMPLICIT NONE
   REAL, PARAMETER :: SPEED_OF_LIGHT=3.00E+08   !(m/s)
   REAL, PARAMETER :: PLANCKS_CONSTANT=6.63E-34 !(J s)
   REAL, PARAMETER :: GRAVITATIONAL_CONSTANT= 6.67E-11 !(N m2/kg2)
   REAL, PARAMETER :: ELECTRON_MASS=9.11E-31 !(kg)
   REAL, PARAMETER :: ELECTRON_CHARGE=1.60E-19 !(C)
   REAL, PARAMETER :: STEFAN_BOLTZMANN_CONSTANT =5.67E-08 !(W/m2/K4)
   REAL, PARAMETER :: IDEAL_GAS_CONSTANT=8.31E+00 !(J/K)
   REAL, PARAMETER :: AVOGADRO_NUMBER=6.02E+23 !(/mol)
   CONTAINS
      REAL FUNCTION PRESSURE( n, T, V )
      ! Computes pressure by ideal gas law (pV=nRT)
         REAL n, T, V ! moles, temperature, volume
         PRESSURE = n * IDEAL_GAS_CONSTANT * T / V
      END FUNCTION PRESSURE
      REAL FUNCTION RADIATION( T, AREA, EMISSIVITY )
      ! Computes radiative heat flux
         REAL T  ! thermodynamic temperature
         REAL AREA ! area of surface
         REAL EMISSIVITY
         RADIATION = EMISSIVITY * STEFAN_BOLTZMANN_CONSTANT * T ** 4 * AREA
      END FUNCTION RADIATION
END MODULE PHYSICS


PROGRAM IDEAL_GAS
! Program to test module <physics>
   USE PHYSICS                          ! access module
   IMPLICIT NONE
   REAL RMM                             !relative molecular mass
   REAL MASS                            !mass (kg)
   REAL TEMPERATURE                     !temperature (K)
   REAL VOLUME                          !volume (m3)
   REAL MOLES                           !moles of gas
   PRINT *, 'Input relative molecular mass'
   READ *, RMM
   PRINT *, 'Input mass(kg), temperature(K), volume(m3)'
   READ *, MASS, TEMPERATURE, VOLUME
   MOLES = 1000.0 * MASS / RMM          ! calculate moles of gas
   PRINT *, 'Pressure = ', PRESSURE( MOLES, TEMPERATURE, VOLUME ), 'Pa'
   STOP
END PROGRAM IDEAL_GAS
```