PROGRAMAS EXTRAS – FORTRAN90/95

1. Character
2. Subrotinas e funções

Exemplo 1.1 Operações básicas com character.

```
PROGRAM CHARACTER_FUNCTIONS
! Program illustrating strings and character functions
   IMPLICIT NONE
   CHARACTER (LEN=72) SCHOOL
   SCHOOL = 'School of Mechanical, Metallurgical and Production Engineering'
   PRINT*,'1:',SCHOOL
   PRINT*,'2:',SCHOOL(11:20)
   PRINT*,'3:',SCHOOL(37:)
   PRINT*,'4:',LEN( SCHOOL ), LEN_TRIM( SCHOOL ), LEN( TRIM( SCHOOL ) )
   PRINT*,'5:',INDEX( SCHOOL, 'Civil' )
   PRINT*,'6:',SCAN( SCHOOL, 'PQR' ), SCAN( SCHOOL, 'pqr' )
   PRINT*,'7:',SCAN( SCHOOL, 'e' ), SCAN( SCHOOL, 'e', .TRUE. )
   PRINT*,'8:',VERIFY( SCHOOL, 'Superb scholars' )
   STOP
END PROGRAM CHARACTER_FUNCTIONS
```

Exemplo 1.2 Character arrays.

Define o arquivo de entrada: marks.dat

```
Name1    mark1
Name2    mark2
Name3    mark3
:        :
```

```
PROGRAM PRIZE_STUDENT
! Program finds the student or students with the top mark in the class
   IMPLICIT NONE
   INTEGER, PARAMETER :: NMAX = 100          ! maximum class size
   CHARACTER (LEN=15) NAME                   ! student name
   INTEGER MARK                              ! student mark
   INTEGER TOPMARK                           ! current top mark
   INTEGER NTOP                              ! number of students with top mark
   CHARACTER (LEN=15) TOPNAMES(NMAX)         ! names of students with top mark
   INTEGER IO                                ! file read status
   INTEGER I                                 ! a counter
   ! Set initial top mark to something negative
   TOPMARK = -1
   ! Open file containing student marks
   OPEN( 20, FILE = 'marks.dat' )
   ! Loop round, reading marks and seeing if we have a new top mark
   DO
      READ( 20, *, IOSTAT = IO ) NAME, MARK            ! read a mark
      IF ( IO /= 0 ) EXIT                              ! check for end of file
      IF ( MARK > TOPMARK ) THEN                       ! new top mark
         TOPMARK = MARK
         NTOP = 1
         TOPNAMES(NTOP) = NAME
      ELSE IF ( MARK == TOPMARK ) THEN                 ! equal to current top
         NTOP = NTOP + 1
         TOPNAMES(NTOP) = NAME
      END IF
   END DO
   ! Close the marks file containing marks
   CLOSE( 20 )
```

```
   ! Output the Roll of Honour
   PRINT *, 'Top mark is ', TOPMARK, ' achieved by:'
   DO I = 1, NTOP
      PRINT *, TOPNAMES(I)
   END DO
   STOP
END PROGRAM PRIZE_STUDENT
```

Versão melhorada usando a leitura do arquivo duas vezes, fazendo uso do comando Rewind.

```
PROGRAM PRIZE_STUDENT
! Program finds the student or students with the top mark in the class
   IMPLICIT NONE
   CHARACTER (LEN=20) NAME                    ! student name
   INTEGER MARK                               ! student mark
   INTEGER TOPMARK                            ! current top mark
   INTEGER IO                                 ! file read status
   ! Open the file containing student marks
   OPEN( 20, FILE = 'marks.dat' )
   ! First pass - identify the top mark
   TOPMARK = -1
   DO
      READ( 20, *, IOSTAT = IO ) NAME, MARK          ! read a mark
      IF ( IO /= 0 ) EXIT                            ! check for EOF
      IF ( MARK > TOPMARK ) TOPMARK = MARK           ! new top mark
   END DO
   REWIND( 20 )                                      ! go back to beginning
   ! Second pass - list only students with the top mark
   PRINT *, 'Top mark is ', TOPMARK, ' achieved by:'
   DO
      READ( 20, *, IOSTAT = IO ) NAME, MARK          ! read name and mark
      IF ( IO /= 0 ) EXIT                            ! check for EOF
      IF ( MARK == TOPMARK ) PRINT *, NAME           ! print best only
   END DO
   ! Close the marks file
   CLOSE( 20 )
   STOP
END PROGRAM PRIZE_STUDENT
```

Exemplo 1.3 Conjunto de caracteres em Fortran.

```
PROGRAM CHARACTER_SET
! Program prints (most of) the Fortran character set, 10 to a line
   IMPLICIT NONE
   INTEGER N
   CHARACTER (LEN=*), PARAMETER :: FMT = '( 10( 2X, I3, 1X, A1 ) )'
   WRITE( *, FMT ) ( N, CHAR(N), N = 20, 127 )
END PROGRAM CHARACTER_SET
```

Exemplo 1.4 Subrotina intrínseca com argumentos.

```
PROGRAM CLOCK
! Program asking the computer for date and time
   IMPLICIT NONE
   CHARACTER (LEN=8) DATE              ! date in format ccyymmdd
   CHARACTER (LEN=10) TIME             ! time in format hhmmss.sss
   CHARACTER (LEN=5) ZONE              ! time zone (rel to UTC) as Shhmm
   INTEGER VALUES(8)                   ! year, month, day, mins from UTC,
                                       ! hours, min, sec, msec
   CHARACTER (LEN=8) TIMESTRING        ! time in the format hh:mm:ss
   CHARACTER (LEN=10) DATESTRING       ! date in the format dd-mm-yyyy
```

```
   ! Ask the system for the date and time
   CALL DATE_AND_TIME( DATE, TIME, ZONE, VALUES )
   ! Convert to desired format
   TIMESTRING = TIME( 1:2 ) // ':' // TIME( 3:4 ) // ':' // TIME( 5:6 )
   DATESTRING = DATE( 7:8 ) // '-' // DATE( 5:6 ) // '-' // DATE( 1:4 )
   ! Output the desired time and date
   PRINT *, 'It is ', TIMESTRING, ' on ', DATESTRING
   STOP
END PROGRAM CLOCK
```

2. Subprogramas (Subrotinas e Funções)

Exemplo 2.1 Uso de funções aplicadas em problemas particulares

(a) Integração usando Regra do Trapézio:

$$\int_a^b f(x)dx \cong \frac{\Delta x}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{N-1} f(x_i) \right]$$

Onde, com N intervalos

$$\Delta x = \frac{b-a}{N}, x_i = a + i\Delta x$$

```
PROGRAM TRAPEZIUM_RULE
   IMPLICIT NONE
   REAL, EXTERNAL :: F                          !function to be integrated
   INTEGER N                                    !number of intervals
   REAL A, B                                    !limits of integration
   REAL INTEGRAL                                !value of integral
   REAL DX                                      !interval
   REAL X                                       !an ordinate
   INTEGER I                                    !a counter
   PRINT *, 'Input A, B, N'
   READ *, A, B, N
   DX = (B - A) / N                             ! calculate interval
   INTEGRAL = F(A) + F(B)                       ! contribution from ends
   DO I = 1, N - 1
      X = A + I * DX                            ! calculate intermediate point
      INTEGRAL = INTEGRAL + 2.0 * F(X)          ! add contribution to sum
   END DO
   INTEGRAL = INTEGRAL * DX / 2.0               ! convert sum to integral
   PRINT *, 'Integral = ', INTEGRAL
   STOP
END PROGRAM TRAPEZIUM_RULE
!=====================================================================
REAL FUNCTION F( X )                            ! Function to be integrated
   IMPLICIT NONE
   REAL X
   F = X ** 2
END FUNCTION F
```

(b) Método iterativo de Newton-Raphson para resolver f (x) = 0 :

$$x_{n+1} = x_n - \frac{f(x_n)}{f^{"}(x_i)}$$

```
PROGRAM NEWTON_RAPHSON
! Program finds a root of the equation f(x) = 0 by the Newton-Raphson method
   IMPLICIT NONE
   REAL, EXTERNAL :: F, DFDX                        ! function and its derivative
   REAL, PARAMETER :: TOLERANCE = 1.0E-6            ! tolerance for near-zero
   INTEGER, PARAMETER :: ITERMX = 200               ! maximum number of iterations
   REAL X                                           ! current x
   REAL FVALUE                                      ! current value of function f
   INTEGER :: ITER = 0                              ! current iteration number
   PRINT *, 'Input initial X'
   READ *, X                                        ! input first guess
   FVALUE = F( X )                                  ! value of function
   PRINT *, 'X, F(X) =', X, FVALUE
   ! Loop until root found or maximum iterations reached
   DO WHILE ( ABS( FVALUE ) > TOLERANCE .AND. ITER <= ITERMX )
      X = X - FVALUE / DFDX( X )                    ! update x by Newton-Raphson formula
      FVALUE = F( X )                               ! update value of function
      ITER = ITER + 1                               ! update iteration number
      PRINT *, 'X, F(X) =', X, FVALUE               ! output current values
   END DO
   ! Output answer (or warn if not converged)
   IF ( ABS( FVALUE ) > TOLERANCE ) THEN
      PRINT *, 'Not converged'
   ELSE
      PRINT *, 'Answer =', X
   END IF
   STOP
END PROGRAM NEWTON_RAPHSON
!=========================================
REAL FUNCTION F( X )
! This function should return the value of the function at X
   IMPLICIT NONE
   REAL X
   F = 16.0 * X * X - 4.0
END FUNCTION F
!=========================================
REAL FUNCTION DFDX( X )
! This function should return the derivative of the function at X
   IMPLICIT NONE
   REAL X
   DFDX = 32.0 * X
END FUNCTION DFDX
```

Exemplo 2.2 Uso de subrotinas.

```
PROGRAM EXAMPLE
! Program to swap two numbers
    IMPLICIT NONE
    EXTERNAL SWAP                      ! (optionally) declare routine to be used
    INTEGER I, J
    PRINT *, 'Input integers I and J'
    READ *, I, J
    CALL SWAP( I, J )
    PRINT *, 'Swapped numbers are ', I, J
    STOP
END PROGRAM EXAMPLE
!=======================================
SUBROUTINE SWAP( M, N )
    IMPLICIT NONE
    INTEGER M, N                       ! numbers to be swapped
    INTEGER TEMP                       ! temporary storage
    TEMP = M                           ! store number before changing it
```

```
   M=N
   N = TEMP
END SUBROUTINE SWAP
```

Exemplo 2.3 Especificando `INTENT para argumentos de subp`rogramas.

```
PROGRAM COORDINATES
! Program to convert from Cartesian to polar coordinates
   IMPLICIT NONE
   EXTERNAL POLARS
   REAL X, Y
   REAL R, THETA
   PRINT *, 'Input coordinates X and Y'
   READ *, X, Y
   CALL POLARS( X, Y, R, THETA )
   PRINT *, 'R, THETA =', R, THETA
   STOP
END PROGRAM COORDINATES
!========================
SUBROUTINE POLARS( X, Y, R, THETA )
! Subroutine transforming input (X, Y) to output (R, THETA)
   IMPLICIT NONE
   REAL, INTENT(IN) :: X, Y                ! cartesian coordinates (input)
   REAL, INTENT(OUT) :: R, THETA           ! polar coordinates (output)
   REAL, PARAMETER :: PI = 3.141593        ! the constant pi
   R = SQRT( X ** 2 + Y ** 2 )             ! radius
   THETA = ATAN2( Y, X )                   ! inverse tangent between -pi and pi
   IF ( THETA < 0.0 ) THETA = THETA + 2.0 * PI ! angle between 0 and 2 pi
   THETA = THETA * 180.0 / PI              ! convert to degrees
END SUBROUTINE POLARS
```

Exemplo 2.4 Subrotinas com arrys nos argumentos

Média: $\bar{X} = \frac{\sum X}{N}$; variança $\sigma^2 = \frac{\sum X^2}{N} - \bar{X}^2$, desvio padrão σ. Multiplique σ^2 Por $\frac{N}{N-1}$ para uma população livre.

```
PROGRAM EXAMPLE
! Program computes mean, variance and standard deviation
   IMPLICIT NONE
   EXTERNAL STATS                       ! subroutine to be used
   INTEGER NVAL                         ! number of values
   REAL, ALLOCATABLE :: X(:)            ! data values
   REAL MEAN, VARIANCE, STANDARD_DEVIATION             ! statistics
   INTEGER N                                           ! a counter
   ! Open data file
   OPEN( 10, FILE = 'stats.dat' )
   ! Read the number of points and set aside enough memory
   READ( 10, * ) NVAL
   ALLOCATE( X(NVAL) )
   ! Read data values
   READ( 10, * ) ( X(N), N = 1, NVAL )
   CLOSE( 10 )
   ! Compute statistics
   CALL STATS( NVAL, X, MEAN, VARIANCE, STANDARD_DEVIATION )
   ! Output results
   PRINT *,'Mean = ', MEAN
   PRINT *,'Variance = ', VARIANCE
```

```
   PRINT *,'Standard deviation = ', STANDARD_DEVIATION
   ! Recover computer memory
   DEALLOCATE( X )
   STOP
END PROGRAM EXAMPLE
!=================================================
SUBROUTINE STATS( N, X, M, VAR, SD )
! This works out the sample mean, variance and standard deviation
   IMPLICIT NONE
   INTEGER, INTENT(IN) :: N               ! array size
   REAL, INTENT(IN) :: X(N)               ! data values
   REAL, INTENT(OUT) :: M, VAR, SD        ! statistics
   ! Calculate statistics using array operation SUM
   M = SUM( X ) / N                       ! mean
   VAR = SUM( X * X ) / N - M ** 2          ! variance
   SD = SQRT( VAR )                         ! standard deviation
END SUBROUTINE STATS
```