Capítulo 17. MÓDULOS

2 Nov 2024

OBJETIVOS DO CAPÍTULO

- Utilizar módulos
- Comandos novos do FORTRAN: MODULE, END MODULE, PRIVATE, PUBLIC

17.1 projeto programa17a

1) Objetivos do programa:
   (a) exemplificar o uso e as características básicas de módulos em FORTRAN; e
   (b) usar dois novos comandos do FORTRAN: MODULE, END MODULE.
2) No Fortran, **criar um projeto** com o nome **programa17a**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **dados.f90**
4) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 17.1**.

Tabela 17.1 Programa-fonte dados.f90 do projeto programa17a

```
MODULE DADOS

  IMPLICIT NONE

  REAL*8  I, J

CONTAINS

  SUBROUTINE LE_DADOS

    WRITE(*,*) "Entre com o valor de I"
    READ(*,*) I

    WRITE(*,*) "Entre com o valor de J"
    READ(*,*) J

  END SUBROUTINE LE_DADOS

END MODULE DADOS
```

5) No Fortran, **criar e inserir** no projeto o programa-fonte **saida.f90**
6) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 17.2**.
7) No Fortran, **criar e inserir** no projeto o programa-fonte **principal.f90**

8) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 17.3**.

Tabela 17.2 Programa-fonte saida.f90 do projeto programa17a

```
MODULE SAIDA

  USE DADOS

  IMPLICIT NONE

  REAL*8 K

CONTAINS

  SUBROUTINE CALCULOS

    K = I + J

  END SUBROUTINE CALCULOS

  SUBROUTINE RESULTADOS

    USE PORTLIB

    INTEGER VER
    INTEGER C
    CHARACTER(20) B

    B = "teste de FORTRAN"
    C = 7

    OPEN(1, file = "SAIDA.TXT" )

    WRITE(1,3) I, J, K
    3 FORMAT( 2/, "sub-rotina RESULTADOS", &
              2/, "I = ", 1PE10.3, &
              2/, "J = ", 1PE10.3, &
              2/, "K = ", 1PE10.3  )

    WRITE(1,4) B, C
    4 FORMAT(1/, 5X, A,  "= B", &
             2/, 5X, I5, "= C"  )

    CLOSE(1)
```

```
    VER = SYSTEM("Notepad SAIDA.TXT" )

  END SUBROUTINE RESULTADOS

END MODULE SAIDA
```

Tabela 17.3 Programa-fonte principal.f90 do projeto programa17a

```
PROGRAM PROGRAMA17A

  USE SAIDA

  IMPLICIT NONE

  CALL LE_DADOS

  CALL CALCULOS

  CALL RESULTADOS

  WRITE(*,*) "MAIN: I, J, K = ", I, J, K

END PROGRAM PROGRAMA17A
```

9) Comentários sobre o programa:

(a) Um módulo é praticamente igual a um programa-principal. A maior diferença é que em um módulo não se pode ter comandos executáveis antes do comando CONTAINS, ao contrário do que ocorre no programa-principal.

(b) A definição de módulo em FORTRAN deve seguir a sintaxe mostrada na Tabela 17.4. O nome do módulo segue as regras válidas para variáveis em FORTRAN, não podendo ser igual a nenhum outro nome de módulo, variável ou sub-rotina do programa. No caso de não haver sub-rotinas no módulo, o comando CONTAINS não deve ser usado; um exemplo disso é um módulo usado para definir as variáveis globais do programa.

(c) O uso de módulos facilita muito a estruturação de programas próprios de grande porte.

(d) Para não haver problemas com definição de variáveis, deve-se usar o comando IMPLICIT NONE dentro de cada módulo.

(e) Um módulo pode ser usado dentro de uma sub-rotina, de outro módulo ou dentro de um programa-principal através do comando USE seguido do nome do módulo.

(f) Dentro de um módulo, as variáveis definidas antes do comando CONTAINS são reconhecidas por todas as sub-rotinas do módulo, ou seja, elas são variáveis globais do módulo onde estão definidas.

(g) Um programa-fonte pode conter um ou vários módulos em sequência, porém eles não podem depender um do outro.

(h) A primeira compilação dos programas-fonte que contêm módulos deve ser feita na seguinte ordem: (1) os módulos que não dependem de outros; (2) os módulos que dependem de outros que já foram compilados; e (3) o programa-principal.

Tabela 17.4 Sintaxe de módulos em FORTRAN.

```
MODULE NOME
  comandos USE e EXTERNAL
  definições de variáveis
CONTAINS
  sub-rotinas
END MODULE NOME
```

10) Executar **Build, Compile** para compilar o programa-fonte **dados.f90**. Repetir para **saida.f90** e **principal.f90**, nesta ordem.
11) Gerar o programa-executável fazendo **Build, Build**.
12) Executar o programa através de **Build, Execute**. **Usar**, por exemplo, **os valores 1 e 2 para as variáveis I e J, respectivamente**. Neste caso, os resultados da execução devem ser os mostrados nas Figuras 17.1 e 17.2.

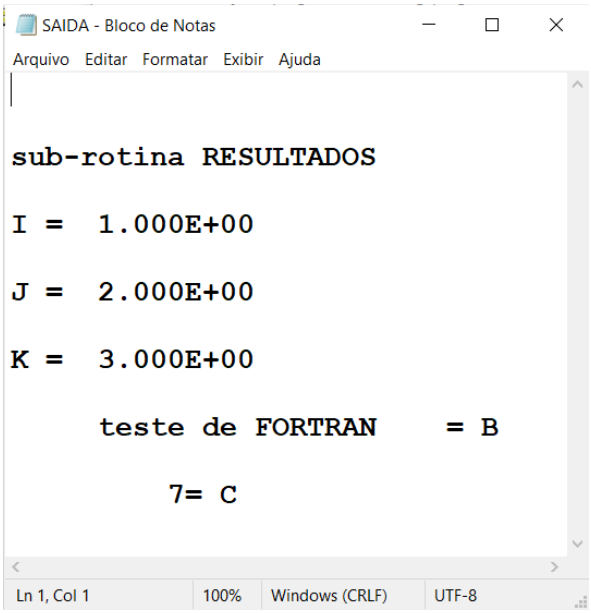

Figura 17.1 Arquivo com resultados do programa17a.exe.

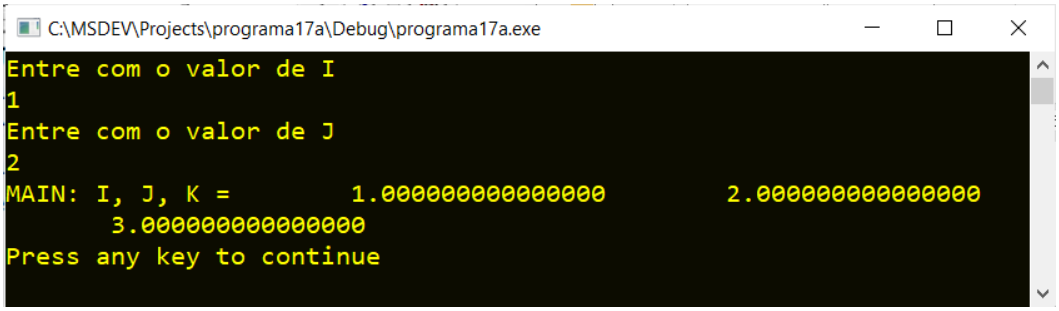

Figura 17.2 Janela DOS após a execução do programa17a.exe.

13) **Analisar os resultados** mostrados nas Figuras 17.1 e 17.2 considerando os três programas-fonte e os comentários do item 9, acima, bem como o seguinte:
   (a) Notar que as variáveis I e J, que foram definidas e seus valores lidos dentro do módulo DADOS, também são reconhecidas dentro do módulo SAIDA. Isso ocorre porque o módulo DADOS foi incluído no módulo SAIDA.
   (b) Além disso, notar que as variáveis I e J e seus valores também são reconhecidos dentro do programa-principal. Isso ocorre porque o módulo SAIDA foi incluído no programa-principal.

17.2 projeto programa17b

1) Objetivo do programa: entender o uso de módulos em programa composto por quatro módulos.
2) No Fortran, **criar um projeto** com o nome **programa17b**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **variaveis.f90** e **copiar** da **Tabela 17.5**.
4) No Fortran, **criar e inserir** no projeto o programa-fonte **dados.f90** e **copiar** da **Tabela 17.6**.
5) No Fortran, **criar e inserir** no projeto o programa-fonte **calculos.f90** e **copiar** da **Tabela 17.7**.
6) No Fortran, **criar e inserir** no projeto o programa-fonte **resultados.f90** e **copiar** da **Tabela 17.8**.
7) No Fortran, **criar e inserir** no projeto o programa-fonte **modulo.f90** e **copiar** da **Tabela 17.9**.
8) Comentários sobre o programa:
   (a) Ele é composto por quatro módulos, sendo cada um editado em programa-fonte diferente.
   (b) O módulo VARIAVEIS é usado para definir todas as variáveis usadas no programa.
   (c) O programa-principal incorpora apenas o módulo RESULTADOS. Mas este, tem incorporado dentro de si o módulo CALCULOS, que incorpora o módulo DADOS, que finalmente incorpora o módulo VARIAVEIS. Assim, todos os módulos estão também implicitamente inseridos dentro do programa-principal.
9) **Estudar os quatro módulos e o programa-principal** considerando os comentários do item 8 desta seção e o item 9 da seção anterior.
10) **Criar o arquivo dados.txt de acordo com a Figura 17.3.**
11) Executar **Build, Compile** para compilar o programa-fonte **variaveis.f90**. Repetir para **dados.f90, calculos.f90, resultados.f90** e **modulo.f90**, nesta ordem.
12) Gerar o programa-executável fazendo **Build, Build**.
13) Executar o programa através de **Build, Execute**.
14) **Analisar os resultados** mostrados na Figura 17.4.
15) **Executar** novamente o programa usando **tipo_de_calculo = 2** e **analisar** os novos resultados.
16) **Executar** novamente o programa usando **tipo_de_calculo = 3** e **analisar** os novos resultados.
17) **Executar** novamente o programa usando **tipo_de_calculo = 0** e **analisar** os novos resultados.

Tabela 17.5 Programa-fonte variaveis.f90 do projeto programa17b

```
module VARIAVEIS

! inclusão de módulos do Fortran90

USE PORTLIB ! para usar o comando SYSTEM

! ------------------------------------------------

! definição das variáveis globais do programa

implicit none

integer   :: n               ! número de elementos dos vetores
integer   :: i               ! número do elemento dos vetores
integer   :: tipo_de_calculo ! a realizar
integer   :: dos             ! acessa prompt dos

integer   :: local_maximo(1) ! posição do valor máximo do vetor b
integer   :: local_minimo(1) ! posição do valor mínimo do vetor b

real*8    :: soma            ! soma dos valores do vetor b
real*8    :: maximo          ! valor máximo do vetor b
real*8    :: minimo          ! valor mínimo do vetor b

! vetores com dimensão aberta e alocáveis
real*8,dimension(:),allocatable :: a, b

end module VARIAVEIS
```

Tabela 17.6 Programa-fonte dados.f90 do projeto programa17b

```
module DADOS

use VARIAVEIS

implicit none

contains

subroutine le_dados

  ! *** leitura dos dados ***

  ! mostra o conteúdo do arquivo "dados.txt" com o programa NOTEPAD
```

```
  dos = system( 'notepad dados.txt' )

  open(10,file='dados.txt')  ! abre o arquivo "dados.txt"

  read(10,*) tipo_de_calculo
  read(10,*) n

  ! aloca memória para os vetores a, b
  allocate ( a(n), b(n) )

  read(10,*) (a(i),i=1,n)

  close(10)

end subroutine le_dados

end module dados
```

Tabela 17.7 Programa-fonte calculos.f90 do projeto programa17b

```
module CALCULOS

use DADOS

implicit none

contains

subroutine opera_vetores

  ! *** realiza cálculos conforme tipo_de_calculo escolhido nos dados

  select case ( tipo_de_calculo)

    case ( 1 )
      b = a + 1

    case ( 2 )
      b = a * 2

    case ( 3 )
      b = a ** 2

    case default
      b = 1
```

```
  end select

  ! ------------------------------------------------

  ! *** emprega comandos usados com vetores e matrizes ***

  maximo = maxval(b) ! valor máximo do array

  local_maximo = maxloc(b) ! posição do valor máximo do vetor b

  minimo = minval(b) ! valor mínimo do array

  local_minimo = minloc(b) ! posição do valor mínimo do vetor b

  soma = sum(b)

  ! ------------------------------------------------

end subroutine opera_vetores

end module CALCULOS
```

Tabela 17.8 Programa-fonte resultados.f90 do projeto programa17b

```
module RESULTADOS

use CALCULOS

implicit none

contains

subroutine escreve

  ! *** escreve o vetor b em arquivo ***

  open(20,file='saida.txt')  ! abre o arquivo "saida.txt"

  do i = 1, n
    write(20,30) i, b(i)
  end do

  write(20,31) maximo, local_maximo, minimo, local_minimo, soma
```

```
  close(20)

  30 format ( i4, 5x, 1pe10.3 )
  31 format (/,'Valor máximo = ',1pe12.3,/, &
               'Posição      = ',i4,//,     &
               'Valor mínimo = ',1pe12.3,/, &
               'Posição      = ',i4,//,     &
               'Soma         = ',1pe12.3)

  ! ------------------------------------------------

  ! *** mostra o conteúdo do arquivo "saida.txt" ***

  dos = system( 'notepad saida.txt' )

  ! ------------------------------------------------

end subroutine escreve

end module RESULTADOS
```

Tabela 17.9 Programa-fonte principal modulo.f90 do projeto programa17b

```
program PROGRAMA17B

use RESULTADOS

implicit none

! ------------------------------------------------

  call le_dados

  call opera_vetores

  call escreve

! ------------------------------------------------

end   program PROGRAMA17B
```

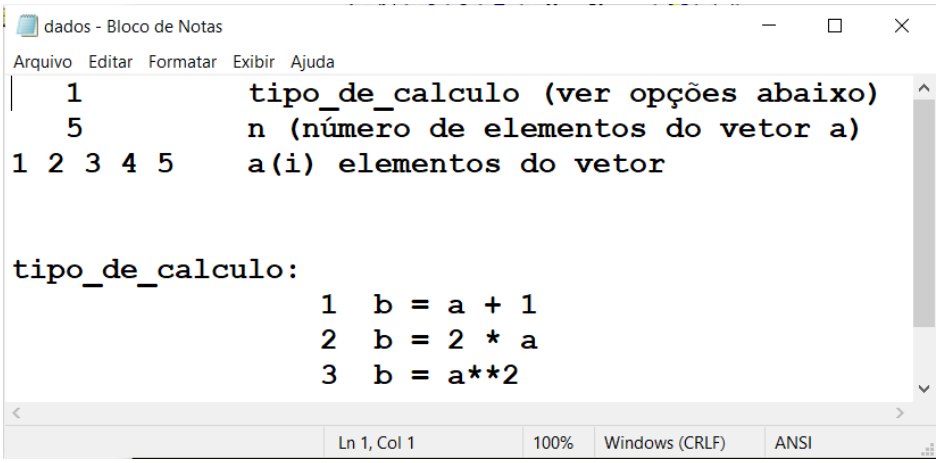

Figura 17.3 Arquivo de dados do programa17b.exe.

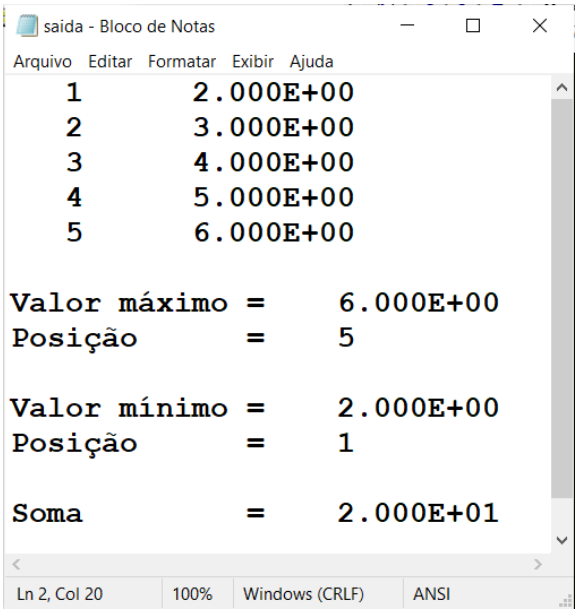

Figura 17.4 Arquivo de resultados do programa17b.exe.

17.3 projeto programa17c

1) Objetivos do programa:
   (a) definir variáveis públicas e privadas em módulos;
   (b) usar dois novos comandos do FORTRAN: PUBLIC e PRIVATE; e
   (c) entender o uso de módulos com variáveis públicas e privadas através de um programa-exemplo.
2) No Fortran, **criar um projeto** com o nome **programa17c**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **base.f90** e **copiar** da **Tabela 17.10**.
4) No Fortran, **criar e inserir** no projeto o programa-fonte **base2.f90** e **copiar** da **Tabela 17.11**.
5) No Fortran, **criar e inserir** no projeto o programa-fonte **main.f90** e **copiar** da **Tabela 17.12**.

Tabela 17.10 Programa-fonte base.f90 do projeto programa17c

```
module PRIMEIRO

  implicit none

  integer,parameter,private :: R = 11

  integer,private :: N

  integer,public  :: K

  integer :: L

contains

  subroutine UM

    N = 12

    K = 13

    L = 14

    write (10,*) 'rotina UM, modulo PRIMEIRO, R =', R

    write (10,*) 'rotina UM, modulo PRIMEIRO, N =', N

    write (10,*) 'rotina UM, modulo PRIMEIRO, K =', K

    write (10,*) 'rotina UM, modulo PRIMEIRO, L =', L

  end subroutine UM

end module PRIMEIRO
```

6) Comentários sobre o programa:

   (a) Dois novos comandos do FORTRAN, associados ao uso de módulos, são utilizados neste programa: PUBLIC e PRIVATE.

   (b) O comando PRIVATE é empregado para definir uma variável como privativa do módulo no qual ela é definida. Ou seja, ela só é reconhecida pelas sub-rotinas definidas dentro do próprio módulo. Ela não é reconhecida como variável dentro de outros módulos ou do programa-principal que utilizem o módulo no qual ela está definida. Um exemplo é dado na linha `integer,private :: N` do módulo PRIMEIRO: a variável N só é reconhecida como tal dentro do módulo PRIMEIRO; o mesmo ocorre com a variável

R. As variáveis R e N do módulo SEGUNDO são diferentes das variáveis R e N do módulo PRIMEIRO, embora tenham os mesmos nomes.

(c) O comando PUBLIC é empregado para definir uma variável como global. Isto é, ela é reconhecida pelas sub-rotinas definidas dentro do próprio módulo, e também dentro de outros módulos ou do programa-principal que utilizem o módulo no qual ela está definida. Um exemplo é dado na linha `integer,public :: K` do módulo PRIMEIRO: a variável K é reconhecida como tal dentro dos módulos PRIMEIRO e SEGUNDO, e do programa-principal.

Tabela 17.11 Programa-fonte base2.f90 do projeto programa17c

```
module SEGUNDO

  USE PRIMEIRO

  implicit none

  integer,parameter,private :: R = 22

  integer,private :: N

contains

  subroutine DOIS

    write (10,*) 'rotina DOIS, modulo SEGUNDO, R =', R

    write (10,*) 'rotina DOIS, modulo SEGUNDO, N =', N

    write (10,*) 'rotina DOIS, modulo SEGUNDO, K =', K

    write (10,*) 'rotina DOIS, modulo SEGUNDO, L =', L

  end subroutine DOIS

  subroutine TRES

    N = 23

    write (10,*) 'rotina TRES, modulo SEGUNDO, N =', N

  end subroutine TRES

end module SEGUNDO
```

(d) Todas as variáveis definidas no módulo, antes do comando CONTAINS, são assumidas como PUBLIC, a menos que sejam explicitamente definidas como PRIVATE. Um exemplo é dado na linha `integer :: L` do módulo PRIMEIRO: a variável L é entendida como PUBLIC.

(e) Mas todas as variáveis definidas em sub-rotinas são assumidas como PRIVATE.

Tabela 17.12 Programa-fonte principal main.f90 do projeto programa17c

```
program PROGRAMA17C

  use SEGUNDO
  use portlib

  implicit none
  integer dos
  integer,parameter :: R = 33
  integer :: N

  open ( 10, file = 'saida.txt' )

  call executa

  close (10)

  dos = system ( 'notepad saida.txt' )

contains

  subroutine executa

    call UM

    call DOIS

    call TRES

    call QUATRO

    N = 34

    write (10,*) 'rotina EXECUTA, PRINCIPAL, N =', N

  end subroutine executa

  subroutine QUATRO
```

```
    write (10,*) 'rotina QUATRO, PRINCIPAL, R =', R

    write (10,*) 'rotina QUATRO, PRINCIPAL, N =', N

    write (10,*) 'rotina QUATRO, PRINCIPAL, K =', K

    write (10,*) 'rotina QUATRO, PRINCIPAL, L =', L

  end subroutine QUATRO

end program PROGRAMA17C
```

7) **Estudar os dois módulos e o programa-principal** considerando os comentários do item 6 desta seção e da seção 17.1.
8) Executar **Build, Compile** para compilar o programa-fonte **base.f90**. Repetir para **base2.f90** e **main.f90**, nesta ordem.
9) Gerar o programa-executável fazendo **Build, Build**.
10) Executar o programa através de **Build, Execute**. O resultado deve ser o mostrado na Figura 17.5.
11) **Analisar os resultados**.

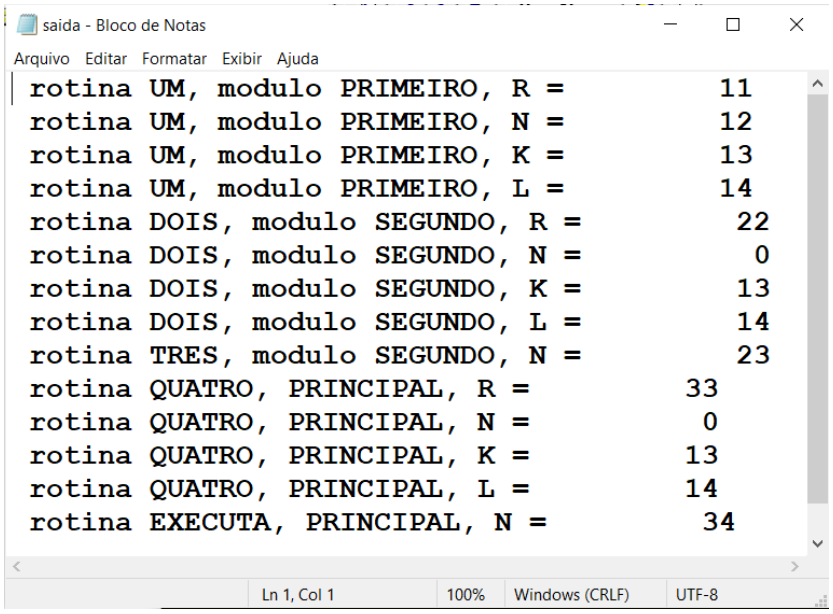

Figura 17.5 Arquivo com resultados do programa17c.exe.

17.4 EXERCÍCIOS

Exercício 17.1

(a) Transformar o programa11d.f90 em um módulo.

(b) Fazer o mesmo para o programa16c.f90.

(c) Criar um programa-principal para executar as rotinas destes dois módulos.

Exercício 17.2

Adaptar o programa17a para que os dados de I, J, B e C são fornecidos através de arquivo de dados.

Exercício 17.3

Adaptar o programa17b para incluir as opções 4 e 5 no tipo de cálculo, onde:

4: b = e^a

5: b = $ln(|a + 1|)$

Exercício 17.4

Adaptar o programa17c para que todos os comandos WRITE sejam usados apenas no programa-principal, mas escrevendo todas as variáveis da versão original.