

OBJETIVOS DO CAPÍTULO

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

Para inicializar as atividades deste capítulo, deve-se acessar o programa Fortran, no Windows, através de: **Start, Programs, Fortran PowerStation 4.0, Microsoft Developer Studio**

17.1 programa17a.f90

- 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, seguindo o [procedimento-padrão](#), **criar um projeto** com o nome **programa17a**
- 3) No Fortran, seguindo o [procedimento-padrão](#), **criar e inserir** no projeto o programa-fonte **dados.f90**
- 4) Dentro do espaço de edição do Fortran, na subjanela maior, **copiar** exatamente o texto em vermelho mostrado na **Tabela 17.1**.
- 5) No Fortran, seguindo o [procedimento-padrão](#), **criar e inserir** no projeto o programa-fonte **saida.f90**
- 6) Dentro do espaço de edição do Fortran, na subjanela maior, **copiar** exatamente o texto em vermelho mostrado na **Tabela 17.2**.
- 7) No Fortran, seguindo o [procedimento-padrão](#), **criar e inserir** no projeto o programa-fonte **principal.f90**
- 8) Dentro do espaço de edição do Fortran, na subjanela maior, **copiar** exatamente o texto em vermelho mostrado na **Tabela 17.3**.
- 9) Comentários sobre o programa:
 - (a) Um módulo é praticamente igual a um programa-principal. A maior diferença é que num módulo não se pode ter comandos executáveis antes do comando CONTAINS, ao contrário do que ocorre num 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 num 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 de grande porte próprios.
 - (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 seqüência.
 - (h) A 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; e (3) o programa-principal.
- 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**.

Tabela 17.1 Programa-fonte dados.f90

```
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
```

- 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.
- 13) **Analisar os resultados** mostrados nas Figuras 17.1 e 17.2 considerando os três programas-fonte e os comentários do item 9, acima.
- 14) No Fortran, para fechar o projeto atual, executar **File, Close Workspace**.

Tabela 17.2 Programa-fonte saida.f90

```
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/, "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

```

PROGRAM PROGRAMA17A

USE SAIDA

IMPLICIT NONE

CALL LE_DADOS

CALL CALCULOS

CALL RESULTADOS

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

END PROGRAM PROGRAMA17A

```

Tabela 17.4 Sintaxe de um módulo em FORTRAN.

```

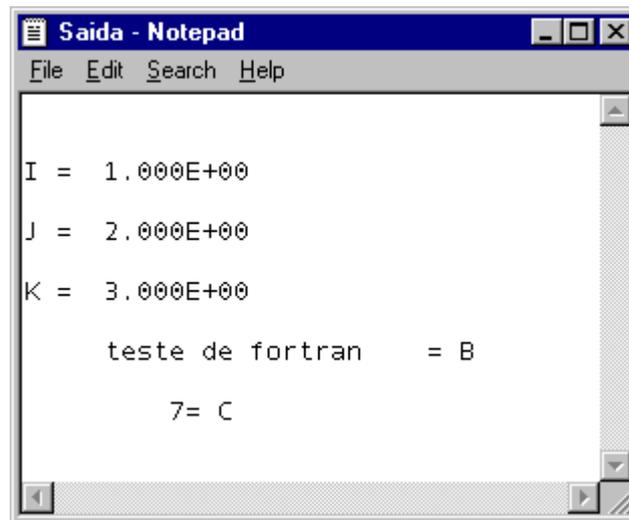
MODULE NOME
  comandos USE e EXTERNAL
  definições de variáveis
CONTAINS
  sub-rotinas
END MODULE NOME

```

17.2 programa17b.f90

- 1) Objetivo do programa: entender o uso de módulos num programa composto por quatro módulos.

- 2) No Fortran, seguindo o [procedimento-padrão](#), **criar um projeto** com o nome **programa17b**
- 3) **Acessar** o site <ftp://ftp.demec.ufpr.br/Disciplinas/Tm784/programa17b/>

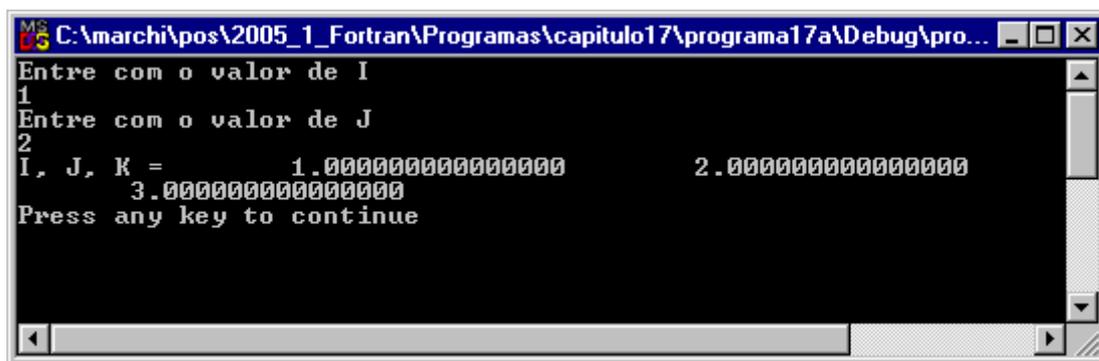


```

I = 1.000E+00
J = 2.000E+00
K = 3.000E+00
teste de fortran = B
7= C

```

Figura 17.1 Arquivo com resultados do programa17a.exe.



```

C:\marchi\pos\2005_1_Fortran\Programas\capitulo17\programa17a\Debug\pro...
Entre com o valor de I
1
Entre com o valor de J
2
I, J, K = 1.0000000000000000 2.0000000000000000
3.0000000000000000
Press any key to continue

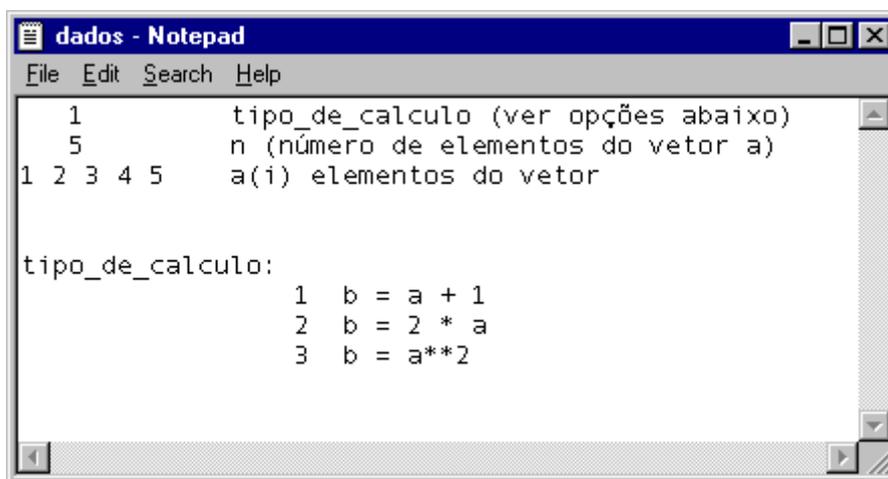
```

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

- 4) **Clicar** com o botão do lado direito do mouse sobre o arquivo calculos.f90
- 5) **Escolher** a opção Save Target As
- 6) Na opção Save in, **localizar** o diretório do projeto
- 7) **Clicar** no botão Save
- 8) **Repetir** os itens 4 a 7, acima, para os arquivos dados.f90, dados.txt, modulo.f90, resultados.f90 e variaveis.f90
- 9) Comentários sobre o programa:
 - (a) Ele é composto por quatro módulos, sendo cada um editado num programa-fonte.
 - (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. A ordem de compilação deve ser a inversa desta seqüência.

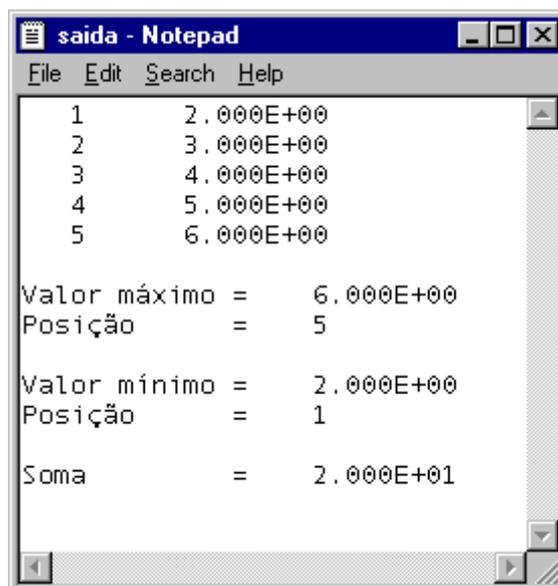
- 10) **Estudar os quatro módulos e o programa-principal** considerando os comentários do item 9 desta seção e da anterior.
- 11) Executar **Build, Compile** para compilar o programa-fonte **variaveis.f90**. Repetir para **dados.f90**, **calculos.f90**, **resultados.f90** e **principal.f90**, nesta ordem.
- 12) Gerar o programa-executável fazendo **Build, Build**.
- 13) Executar o programa através de **Build, Execute. Usar, os dados conforme mostrado na Figura 17.3**.
- 14) **Analisar os resultados** mostrados na Figuras 17.4.



```
dados - Notepad
File Edit Search Help
1      tipo_de_calculo (ver opções abaixo)
5      n (número de elementos do vetor a)
1 2 3 4 5  a(i) elementos do vetor

tipo_de_calculo:
      1  b = a + 1
      2  b = 2 * a
      3  b = a**2
```

Figura 17.3 Arquivo de dados do programa17b.exe.



```
saida - Notepad
File Edit Search Help
1      2.000E+00
2      3.000E+00
3      4.000E+00
4      5.000E+00
5      6.000E+00

Valor máximo = 6.000E+00
Posição     = 5

Valor mínimo = 2.000E+00
Posição     = 1

Soma        = 2.000E+01
```

Figura 17.4 Arquivo de resultados do programa17b.exe.

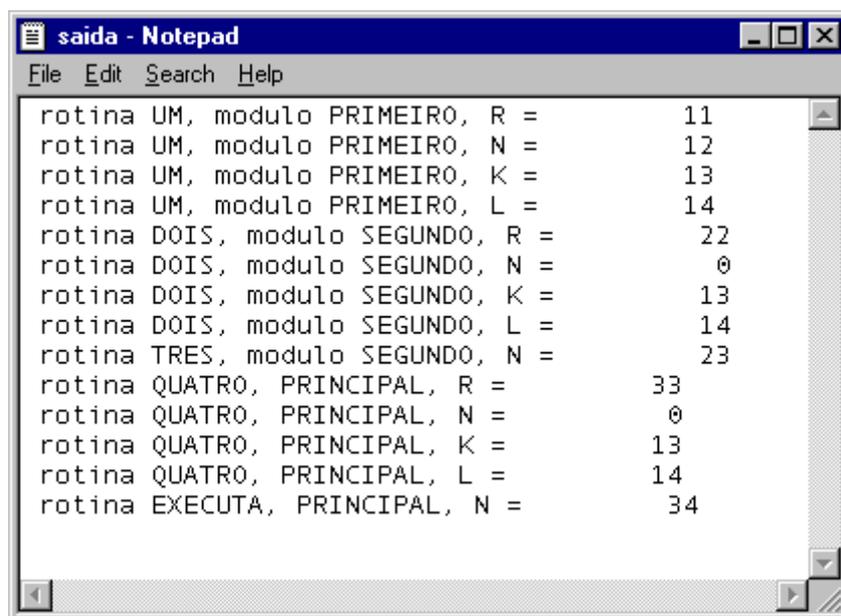
- 15) **Executar** novamente o programa usando **tipo_de_calculo = 2** e **analisar** os novos resultados.
- 16) **Executar** novamente o programa usando **tipo_de_calculo = 3** e **analisar** os novos resultados.
- 17) No Fortran, para fechar o projeto atual, executar **File, Close Workspace**.

17.3 programa17c.f90

- 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 num programa-exemplo.
- 2) No Fortran, seguindo o [procedimento-padrão](#), **criar um projeto** com o nome **programa17c**
- 3) **Acessar** o site <ftp://ftp.demec.ufpr.br/Disciplinas/Tm784/programa17c/>
- 4) **Clicar** com o botão do lado direito do mouse sobre o arquivo base.f90
- 5) **Escolher** a opção Save Target As
- 6) Na opção Save in, **localizar** o diretório do projeto
- 7) **Clicar** no botão Save
- 8) **Repetir** os itens 4 a 7, acima, para os arquivos base2.f90 e main.f90
- 9) Comentários sobre o programa:
 - (a) Dois novos comandos do FORTRAN, associados ao uso de módulos, são usados 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.
 - (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.
 - (d) Todas as variáveis definidas num 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 numa sub-rotina são assumidas como PRIVATE.

- 10) **Estudar os dois módulos e o programa-principal** considerando os comentários do item 9 desta seção e da seção 17.1.
- 11) Executar **Build, Compile** para compilar o programa-fonte **base.f90**. Repetir para **base2.f90** e **main.f90**, nesta ordem.
- 12) Gerar o programa-executável fazendo **Build, Build**.
- 13) Executar o programa através de **Build, Execute**. O resultado deve ser o mostrado na Figura 17.5.
- 14) **Analisar os resultados**.
- 15) Encerrar a sessão seguindo o [procedimento-padrão](#).



```
saida - Notepad
File Edit Search Help
rotina UM, modulo PRIMEIRO, R =      11
rotina UM, modulo PRIMEIRO, N =      12
rotina UM, modulo PRIMEIRO, K =      13
rotina UM, modulo PRIMEIRO, L =      14
rotina DOIS, modulo SEGUNDO, R =     22
rotina DOIS, modulo SEGUNDO, N =      0
rotina DOIS, modulo SEGUNDO, K =     13
rotina DOIS, modulo SEGUNDO, L =     14
rotina TRES, modulo SEGUNDO, N =     23
rotina QUATRO, PRINCIPAL, R =      33
rotina QUATRO, PRINCIPAL, N =      0
rotina QUATRO, PRINCIPAL, K =      13
rotina QUATRO, PRINCIPAL, L =      14
rotina EXECUTA, PRINCIPAL, N =      34
```

Figura 17.5 Arquivo com resultados do programa17c.exe.

17.4 EXERCÍCIOS

Exercício 17.1

- (a) Transformar o programa12d.f90 num módulo.
- (b) Fazer o mesmo para o programa16c.f90.
- (c) Criar um programa-principal para executar as rotinas destes dois módulos.