Capítulo 11. SUB-ROTINAS

22 Abr 2025

OBJETIVOS DO CAPÍTULO

- Conceitos de: sub-rotina, programa-principal, projeto com diversos programas-fonte, passagem de argumentos
- Comandos novos do FORTRAN: PROGRAM, SUBROUTINE, CALL, CONTAINS

11.1 programa11a

1) No Fortran, **criar um projeto** com o nome **programa11a**
2) No Fortran, **criar e inserir** no projeto o programa-fonte **principal1.f90**
3) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.1**.

Tabela 11.1 Programa-fonte principal1.f90.

```
PROGRAM CAPITULO_11A

  REAL A, B, C

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

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

  CALL SOMA ( A, B, C )

  WRITE(*,*) "C = A + B = ", C

END PROGRAM CAPITULO_11A
```

4) No Fortran, **criar e inserir** no projeto o programa-fonte **rotina1.f90**
5) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.2**.
6) Objetivos do programa:
   a) Aplicar os novos comandos PROGRAM, SUBROUTINE e CALL do FORTRAN
   b) Utilizar uma sub-rotina externa ao programa-principal, apenas para realizar um cálculo
   c) Transferir valores entre o programa principal e a sub-rotina

Tabela 11.2 Programa-fonte rotina1.f90.

```
SUBROUTINE SOMA ( X, Y, Z )

  REAL X, Y, Z

  Z = X + Y

END SUBROUTINE SOMA
```

7) Comentários sobre o programa:
   a) Neste programa são usados três novos comandos do FORTRAN: PROGRAM, SUBROUTINE e CALL. Eles são aplicados com o que se denomina em programação de "sub-rotina".
   b) Sub-rotina também é um programa que é usado por outro programa ou por outra sub-rotina.
   c) O programa-principal é o programa-fonte que contém o algoritmo que se deseja executar e que usa sub-rotinas.
   d) Uma sub-rotina não funciona sozinha, ela precisa ser ativada ou chamada pelo programa-principal, ou por outra sub-rotina chamada pelo programa-principal.
   e) Uma sub-rotina pode estar contida dentro do mesmo programa-fonte que contém o programa-principal ou pode estar dentro de outro programa-fonte. O primeiro caso será exemplificado na seção 11.4 deste capítulo. Já o segundo caso é exemplificado na presente seção e nas duas seguintes. Neste caso, isto é, quando há mais de um programa-fonte que constitui um projeto, usa-se o comando PROGRAM para definir qual programa-fonte é o programa-principal.
   f) O comando USE é empregado para incluir uma biblioteca dentro do programa-principal, para que este possa chamar as sub-rotinas desejadas da biblioteca.
   g) Quando há mais de um programa-fonte no projeto, primeiro deve-se compilar cada um deles. Depois, deve-se fazer uma única lincagem para gerar o programa-executável.
   h) Cada sub-rotina pode depender de variáveis do programa-principal ou pode ser um programa totalmente independente. Neste caso, com poucas adaptações, um programa já existente pode ser transformado em uma sub-rotina de um outro programa.
   i) A principal vantagem de usar sub-rotinas é dividir um programa muito grande ou complexo em unidades menores, ou subprogramas, que são mais fáceis de implementar e que facilitam a detecção de erros.
   j) Cada sub-rotina deve ter um nome específico, que é definido com o comando SUBROUTINE. Este nome é usado para chamar ou usar cada sub-rotina no local desejado do programa. As sub-rotinas são ativadas ou chamadas através do comando CALL.

k) As sub-rotinas podem ser escritas pelo próprio programador ou por outros programadores, na forma de programas-fonte ou bibliotecas.

l) Na linha `PROGRAM CAPITULO_11A`, do programa-fonte principal1.f90, define-se o início e o nome do programa-principal como sendo CAPITULO_11A. E na última linha, com o comando `END PROGRAM CAPITULO_11A`, define-se o fim do programa-principal. O nome do programa não tem utilidade prática nenhuma. Deve-se perceber que ele é diferente do nome do projeto e do nome do programa-fonte.

m) Na linha `CALL SOMA ( A, B, C )`, do programa-fonte principal1.f90, chama-se a sub-rotina de nome SOMA e transfere-se a ela os valores das variáveis que estão entre parênteses, isto é, as variáveis A, B e C, que foram declaradas como variáveis reais no programa-principal.

n) Na linha `SUBROUTINE SOMA ( X, Y, Z )`, do programa-fonte rotina1.f90, define-se o início e o nome da sub-rotina como sendo SOMA e, ainda, quais as variáveis que são recebidas e devolvidas ao programa-principal, no caso as variáveis X, Y e Z. E na última linha, com o comando `END SUBROUTINE SOMA`, define-se o fim da sub-rotina SOMA.

o) As variáveis de uma sub-rotina que são recebidas e devolvidas ao programa-principal são denominadas de argumentos da sub-rotina. Elas têm que ser respectivamente do mesmo tipo das variáveis usadas no programa-principal que chama a sub-rotina, mas não precisam ter o mesmo nome. Não é obrigatório que as sub-rotinas tenham argumentos.

8) Algoritmo do programa:

a) No programa-principal, definir as variáveis A, B e C como sendo do tipo real

b) No programa-principal, ler os valores das variáveis A e B

c) No programa-principal, chamar a sub-rotina SOMA, transferindo a ela os valores atuais das variáveis A, B e C

d) Na sub-rotina SOMA, receber os valores das variáveis X, Y e Z, transferidos do programa-principal

e) Na sub-rotina SOMA, definir as variáveis X, Y e Z como reais

f) Na sub-rotina SOMA, realizar a soma das variáveis X e Y e atribuir o resultado à variável Z

g) Na sub-rotina SOMA, ao encontrar o fim da sub-rotina, voltar ao programa-principal no ponto onde a sub-rotina foi chamada, transferindo os valores atuais das variáveis X, Y e Z, da sub-rotina, para as variáveis A, B e C do programa-principal

h) No programa-principal, escrever o valor atual da variável C

i) No programa-principal, encerrar a execução do programa

9) Executar **Build, Compile** para compilar o programa-fonte rotina1.f90.

10) Executar **Build, Compile** para compilar o programa-fonte principal1.f90.

11) Gerar o programa-executável fazendo **Build, Build**.

12) Ao se executar o programa, através de **Build, Execute**, surge uma janela, mostrada na Figura 11.1, dentro da qual tem-se:

a) Na primeira linha, o comentário “Entre com o valor de A”, resultado da instrução `WRITE(*,*) "Entre com o valor de A"` do programa.

b) Na segunda linha, o programa pára e fica aguardando até que seja fornecido o valor da variável A, resultado da instrução `READ(*,*) A` do programa. Para que o programa continue a sua execução, é necessário **digitar 1**, por exemplo, e, em seguida, **clicar na tecla Enter**.

c) Na terceira linha, o comentário "Entre com o valor de B", resultado da instrução `WRITE(*,*) "Entre com o valor de B"` do programa.

d) Na quarta linha, o programa pára e fica aguardando até que seja fornecido o valor da variável B, resultado da instrução `READ(*,*) B` do programa. Para que o programa continue a sua execução, é necessário **digitar 2**, por exemplo, e, em seguida, **clicar na tecla Enter**.

e) Na quinta linha, é apresentado o resultado da soma das variáveis A e B, calculado dentro da sub-rotina SOMA.

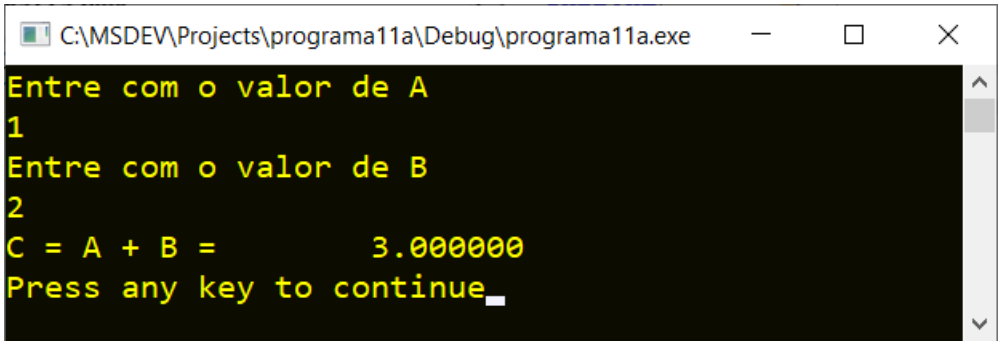

Figura 11.1 Resultado do programa11a.

13) Até entender, **analisar** o resultado do programa11a, mostrado na Figura 11.1, considerando cada linha dos dois programas-fonte envolvidos e as explicações descritas nos itens 7 e 8, acima.

14) **Executar** novamente o programa com outros dados, por exemplo, 7 e 50. Analisar o novo resultado.

11.2 programa11b

1) No Fortran, **criar um projeto** com o nome **programa11b**
2) No Fortran, **criar e inserir** no projeto o programa-fonte **principal2.f90**
3) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.3**.
4) No Fortran, **criar e inserir** no projeto o programa-fonte **rotinas2.f90**
5) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.4**.
6) Objetivos do programa:
   a) Implementar um programa-fonte com duas sub-rotinas
   b) Utilizar duas sub-rotinas externas ao programa-principal, para fazer os cálculos e escrever os resultados
7) Comentários sobre o programa:
   a) Um programa-fonte pode ser constituído por uma ou várias sub-rotinas. Um exemplo é o programa-fonte rotinas2.f90 que contém duas sub-rotinas.

b) Na linha `CALL FATORIAL ( INTEIRO )`, do programa-fonte principal2.f90, chama-se a sub-rotina de nome FATORIAL e transfere-se a ela o valor da variável que está entre parênteses, isto é, a variável INTEIRO, que foi declarada como variável do tipo inteiro no programa-principal.

Tabela 11.3 Programa-fonte principal2.f90.

```
PROGRAM CAPITULO_11B

  INTEGER INTEIRO
  REAL A, B, C

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

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

  CALL SOMA ( A, B, C )

  WRITE(*,*) "C = A + B = ", C

  WRITE(*,*) "Entre com um valor inteiro para calcular o seu fatorial"
  READ(*,*) INTEIRO

  CALL FATORIAL ( INTEIRO )

END PROGRAM CAPITULO_11B
```

8) Algoritmo do programa:

a) No programa-principal, definir as variáveis A, B e C como sendo do tipo real e a variável INTEIRO como do tipo inteiro

b) No programa-principal, ler o valor das variáveis A e B

c) No programa-principal, chamar a sub-rotina SOMA, transferindo a ela os valores atuais das variáveis A, B e C

d) Na sub-rotina SOMA, receber os valores das variáveis X, Y e Z, transferidos do programa-principal

e) Na sub-rotina SOMA, definir as variáveis X, Y e Z como reais

f) Na sub-rotina SOMA, realizar a soma das variáveis X e Y e atribuir o resultado à variável Z

g) Na sub-rotina SOMA, ao encontrar o fim da sub-rotina, voltar ao programa-principal no ponto onde a sub-rotina foi chamada, transferindo os valores atuais das variáveis X, Y e Z, da sub-rotina, para as variáveis A, B e C do programa-principal

h) No programa-principal, escrever o valor atual da variável C

i) No programa-principal, ler o valor da variável INTEIRO

j) No programa-principal, chamar a sub-rotina FATORIAL, transferindo a ela o valor atual da variável INTEIRO

k) Na sub-rotina FATORIAL, receber o valor da variável N, transferido do programa-principal

l) Na sub-rotina FATORIAL, declarar a variável N como inteiro bem como as variáveis auxiliares I e FAT

m) Na sub-rotina FATORIAL, realizar o cálculo do fatorial da variável N e atribuir o resultado à variável FAT se N ≥ 0; se N < 0, informar que não existe fatorial

n) Na sub-rotina FATORIAL, escrever o valor da variável FAT se N ≥ 0

o) Na sub-rotina FATORIAL, ao encontrar o fim da sub-rotina, voltar ao programa-principal no ponto onde a sub-rotina foi chamada, transferindo o valor atual da variável N, da sub-rotina, para a variável INTEIRO do programa-principal

p) No programa-principal, encerrar a execução do programa

Tabela 11.4 Programa-fonte rotinas2.f90.

```
SUBROUTINE SOMA ( X, Y, Z )

   REAL X, Y, Z

   Z = X + Y

END SUBROUTINE SOMA


SUBROUTINE FATORIAL ( N )

   INTEGER I, N, FAT

   IF ( N < 0 ) THEN
      WRITE(*,*) "Nao existe fatorial de ", N
   ELSE
      FAT = 1
      DO I = 2, N
         FAT = FAT * I
      END DO
      WRITE(*,*) "O fatorial de", N, " eh = ", FAT
   END IF

END SUBROUTINE FATORIAL
```

9) Executar **Build, Compile** para compilar o programa-fonte rotinas2.f90.

10) Executar **Build, Compile** para compilar o programa-fonte principal2.f90.

11) Gerar o programa-executável fazendo **Build, Build**.

12) Ao se executar o programa, através de **Build, Execute**, **usar 1, 2 e 4 como dados**, mostrados na Figura 11.2, que também apresenta os respectivos resultados.

13) Até entender, **analisar** os resultados do programa11b, mostrados na Figura 11.2, considerando cada linha dos dois programas-fonte envolvidos e as explicações descritas nos itens 7 e 8, acima.

14) **Executar** novamente o programa com outros dados, por exemplo: 3, 4 e –2. Os resultados são mostrados na Figura 11.3.

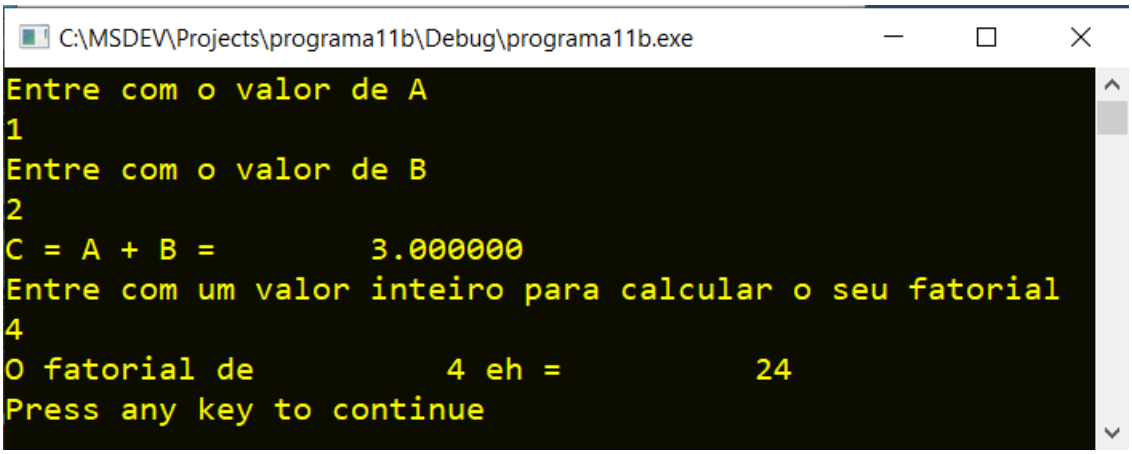

Figura 11.2 Resultados do programa11b para os dados do item 12.

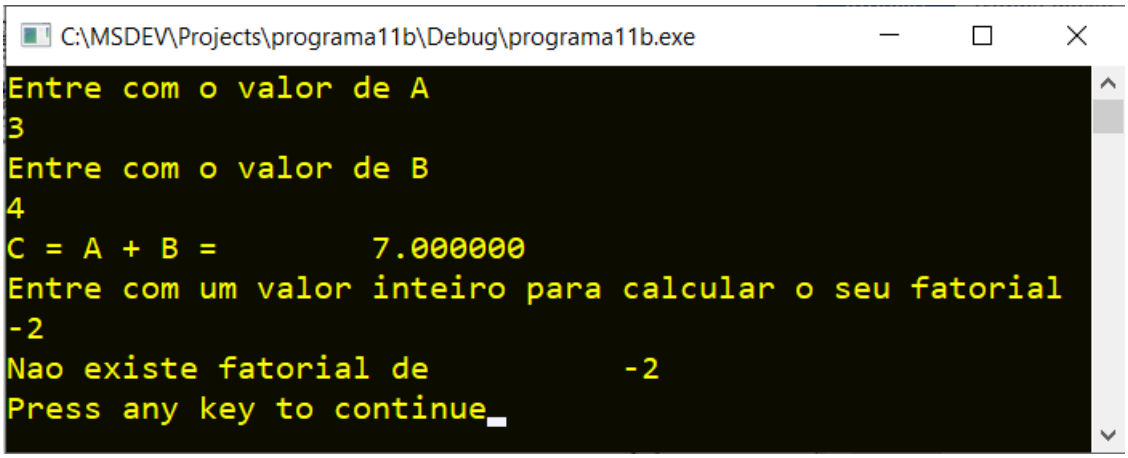

Figura 11.3 Resultados do programa11b para os dados do item 14.

11.3 programa11c

1) No Fortran, **criar um projeto** com o nome **programa11c**
2) No Fortran, **criar e inserir** no projeto o programa-fonte **principal3.f90**
3) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.5**.
4) No Fortran, **criar e inserir** no projeto o programa-fonte **rotinas2.f90**
5) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.4**. O conteúdo do arquivo rotinas2.f90 também pode ser copiado diretamente do projeto anterior, programa11b.
6) No Fortran, **criar e inserir** no projeto o programa-fonte **rotina3.f90**

7) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.6**.

Tabela 11.5 Programa-fonte principal3.f90.

```
PROGRAM CAPITULO_11C

  CALL OUTRAS

END PROGRAM CAPITULO_11C
```

8) Objetivos do programa:
   a) Implementar um programa com três sub-rotinas divididas em dois programas-fonte
   b) Fazer uma sub-rotina chamar outras sub-rotinas
   c) Utilizar uma sub-rotina sem argumentos

Tabela 11.6 Programa-fonte rotina3.f90.

```
SUBROUTINE OUTRAS

  INTEGER INTEIRO
  REAL A, B, C

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

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

  CALL SOMA ( A, B, C )

  WRITE(*,*) "C = A + B = ", C

  WRITE(*,*) "Entre com um valor inteiro para calcular o seu fatorial"
  READ(*,*) INTEIRO

  CALL FATORIAL ( INTEIRO )

END SUBROUTINE OUTRAS
```

9) Comentários sobre o programa:
   a) O programa11c faz exatamente o mesmo que o programa11b, da seção anterior. A diferença é que tudo o que antes era feito no programa-principal agora é feito pela sub-rotina OUTRAS, que é chamada pelo programa-principal cuja única função dele é essa chamada.

b) Uma sub-rotina pode chamar uma ou diversas sub-rotinas. Por exemplo, a sub-rotina OUTRAS chama as sub-rotinas SOMA e FATORIAL.

10) Algoritmo do programa: é o mesmo da seção 11.2, item 8.
11) Executar **Build, Compile** para compilar o programa-fonte rotinas2.f90.
12) Executar **Build, Compile** para compilar o programa-fonte rotina3.f90.
13) Executar **Build, Compile** para compilar o programa-fonte principal3.f90.
14) Gerar o programa-executável fazendo **Build, Build**.
15) Ao se executar o programa com **Build, Execute**, e usar os dados 1, 2 e 4, obtém-se os mesmos resultados da Figura 11.2.
16) Até entender, **analisar** os resultados do programa11c, mostrados na Figura 11.2, considerando cada linha dos três programas-fonte envolvidos e as explicações pertinentes.

11.4 programa11d

1) No Fortran, **criar um projeto** com o nome **programa11d**
2) No Fortran, **criar e inserir** no projeto o programa-fonte **principal4.f90**
3) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.7**. Deve-se perceber que quase todo o conteúdo do arquivo principal4.f90 já foi digitado no projeto programa11c e, portanto, ele pode ser copiado dos arquivos rotinas2.f90 e rotina3.f90.
4) Objetivos do programa:
   a) Implementar um programa com sub-rotinas inseridas dentro do mesmo programa-fonte do programa-principal
   b) Utilizar sub-rotina de biblioteca
5) Comentários sobre o programa:
   a) O programa11d é praticamente idêntico ao programa11c, da seção anterior. Há apenas duas diferenças. A primeira é que as três sub-rotinas (OUTRAS, SOMA e FATORIAL), que antes estavam dentro de dois programas-fonte (rotinas2.f90 e rotina3.f90), agora estão inseridas dentro do mesmo programa-fonte do próprio programa-principal.
   b) O comando CONTAINS do FORTRAN é usado para separar o fim do programa-principal do início das sub-rotinas contidas dentro do mesmo programa-fonte, conforme pode-se ver na Tabela 11.7.
   c) A segunda diferença é que, antes da chamada da sub-rotina OUTRAS, foi inserido uma chamada da sub-rotina TDATE, que é uma sub-rotina pertencente à biblioteca MSIMSLMS. E, para utilizar esta biblioteca, no início do programa-principal foi empregado o comando USE junto com o nome da biblioteca.
   d) A biblioteca MSIMSLMS contém muitas sub-rotinas com diversas finalidades. A lista completa das sub-rotinas desta biblioteca, informações detalhadas e exemplos sobre cada uma delas podem ser vistos no manual online do Fortran em: **? InfoView, IMSL Libraries Reference.**

e) Uma vantagem da estrutura do programa11d, com todas as sub-rotinas no mesmo programa-fonte, é que todas as variáveis definidas antes do comando CONTAINS podem ser usadas em todas as sub-rotinas, sem redefini-las e sem passá-las por argumentos. Isso permite que a data seja escrita no programa-principal e nas sub-rotinas, como exemplificado nas sub-rotinas OUTRAS e FATORIAL.

f) Comparando-se os programas das seções 11.1 a 11.4, deve-se perceber que existe grande flexibilidade nas estruturas possíveis de se usar.

6) Algoritmo do programa: é o mesmo da seção 11.2, item 8, acrescido, da chamada da sub-rotina TDATE e da escrita da data corrente no programa-principal e em duas sub-rotinas.

Tabela 11.7 Programa-fonte principal4.f90.

```
PROGRAM CAPITULO_11D

  USE MSIMSLMS

  INTEGER DIA, MES, ANO

  CALL TDATE ( DIA, MES, ANO )

  WRITE(*,1) DIA, MES, ANO
  1 FORMAT (/, 5X, "PRINCIPAL - data de hoje: ", I2, "/", I2, "/", I4)

  CALL OUTRAS

CONTAINS

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

SUBROUTINE OUTRAS

  INTEGER INTEIRO
  REAL A, B, C

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

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

  CALL SOMA ( A, B, C )

  WRITE(*,*) "C = A + B = ", C

  WRITE(*,1) DIA, MES, ANO
```

```
  1 FORMAT (/, 5X, "OUTRAS - data de hoje: ", I2, "/", I2, "/", I4)

  WRITE(*,*) "Entre com um valor inteiro para calcular o seu fatorial"
  READ(*,*) INTEIRO

  CALL FATORIAL ( INTEIRO )

END SUBROUTINE OUTRAS

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

SUBROUTINE SOMA ( X, Y, Z )

  REAL X, Y, Z

  Z = X + Y

END SUBROUTINE SOMA

! ------------------------------------------------------------------
SUBROUTINE FATORIAL ( N )

   INTEGER I, N, FAT

   IF ( N < 0 ) THEN
      WRITE(*,*) "Nao existe fatorial de ", N
   ELSE
      FAT = 1
      DO I = 2, N
         FAT = FAT * I
      END DO
      WRITE(*,*) "O fatorial de", N, " eh = ", FAT
   END IF

   WRITE(*,1) DIA, MES, ANO
   1 FORMAT (/, 5X, "FATORIAL - data de hoje: ", I2, "/", I2, "/", I4)

END SUBROUTINE FATORIAL

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

END PROGRAM CAPITULO_11D
```

7) Executar **Build, Compile** para compilar o programa-fonte principal4.f90.

8) Gerar o programa-executável fazendo **Build, Build**.

9) Executar o programa através de **Build, Execute** com os dados mostrados na Figura 11.4.
10) Até entender, **analisar** os resultados do programa11d mostrados na Figura 11.4, considerando cada linha do programa-fonte e as explicações pertinentes.
11) Uma outra estrutura que pode ser usada é colocar o END PROGRAM no lugar do comando CONTAINS (e este não é usado). Em seguida, são colocadas todas as sub-rotinas no mesmo programa-fonte. Nesta estrutura, todas as variáveis definidas antes do END PROGRAM só podem ser usadas nas sub-rotinas se forem passadas por argumentos.

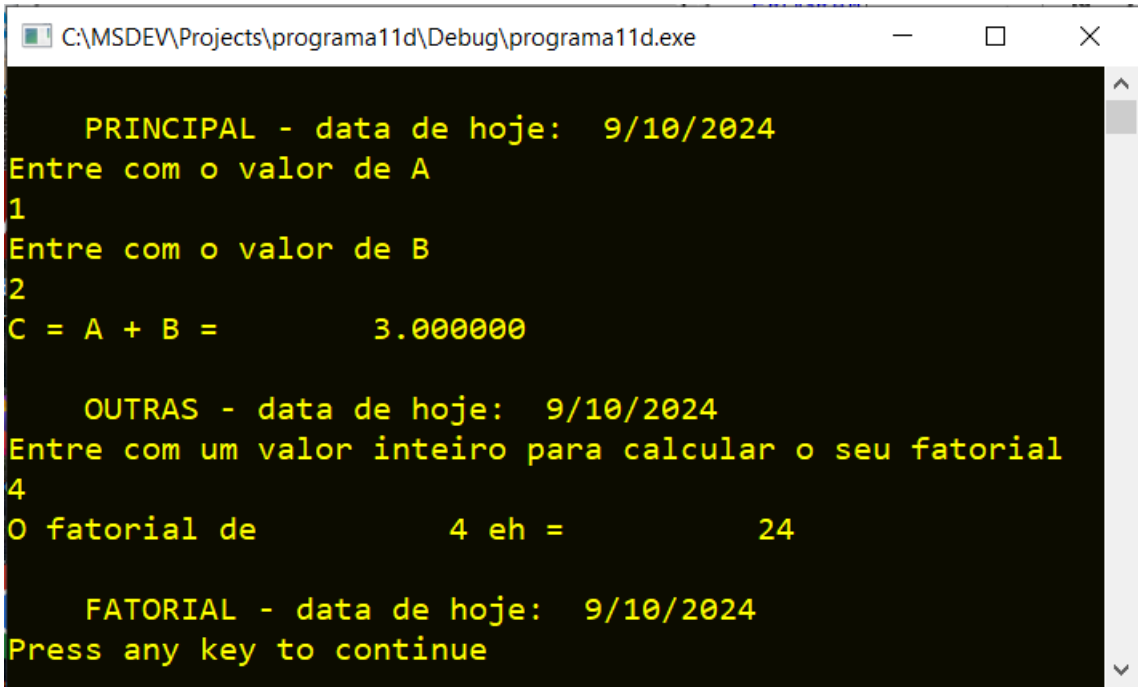

Figura 11.4 Dados e resultados do programa11d.

11.5 programa11e

1) No Fortran, **criar um projeto** com o nome **programa11e**
2) No Fortran, **criar e inserir** no projeto o programa-fonte **principal.f90**
3) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.8**.
4) No Fortran, **criar e inserir** no projeto o programa-fonte **rotina.f90**
5) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 11.9**.
6) Objetivo do programa: mostrar como passar vetor por argumento de sub-rotina.
7) Comentários sobre o programa:
   a) No programa-principal, o vetor é definido e dimensionado como mostrado no Capítulo 9.
   b) A sub-rotina que recebe o vetor deve dimensioná-lo com o número de elementos já definido no programa-principal.
8) Executar **Build, Compile** para compilar os programas-fonte principal.f90 e rotina.f90.
9) Gerar o programa-executável fazendo **Build, Build**.

Tabela 11.8 Programa-fonte principal.f90.

```
PROGRAM CAPITULO_11E

  integer N
  real, allocatable, dimension(:) :: vetor

  WRITE(*,*) "principal: entre com a quantidade de elementos do vetor"
  READ(*,*) N

  allocate ( vetor(N) )

  CALL dados ( N, vetor )

  WRITE(*,*) "principal: vetor = ", vetor

END PROGRAM CAPITULO_11E
```

Tabela 11.9 Programa-fonte rotina.f90.

```
SUBROUTINE dados ( M, conjunto )

  integer M, i
  real, dimension(M) :: conjunto

  write(*,*) "sub-rotina: entre com os valores do VETOR"

  write(*,*) "sub-rotina: M = ", M

  do i = 1, M
     write(*,*) "sub-rotina: entre com o elemento = ", i
     read(*,*) conjunto(i)
  end do

END SUBROUTINE dados
```

10) Executar o programa através de **Build, Execute** com os dados 3, 11, 12 e 13 mostrados na Figura 11.5.
11) Até entender, **analisar** os resultados do programa11e mostrados na Figura 11.5, considerando cada linha do programa-fonte e as explicações pertinentes.

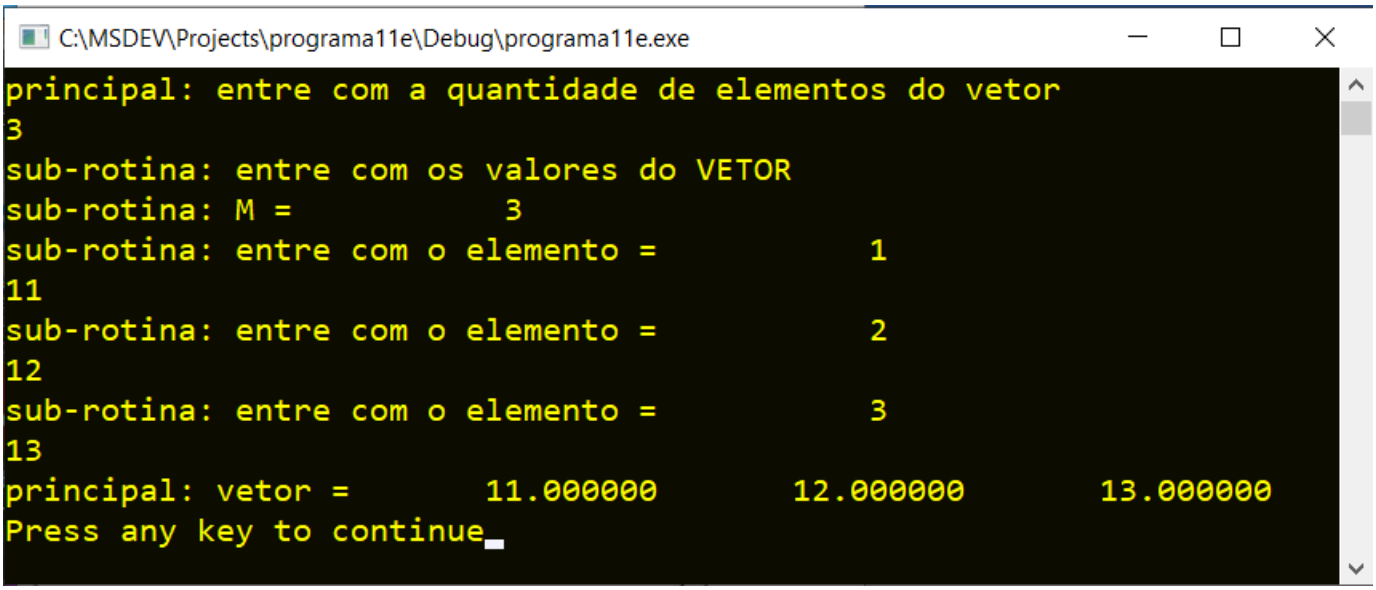

Figura 11.5 Dados e resultados do programa11e.

11.6 EXERCÍCIOS

Exercício 11.1

Com o Programa11b, verificar até que número se consegue calcular o fatorial.

Deverá ser até 12.

Confirmar isso comparando os resultados do programa em FORTRAN com o Excel ou uma calculadora científica.

Depois, confirmar que o fatorial de 13, fornecido pelo Programa11b, está incorreto; isso ocorre porque se passa do limite de valor dos inteiros, que é 2^{31}-1, ou seja 2,147,483,647.

Exercício 11.2

Uma forma de contornar o problema do limite dos números inteiros, do Exercício 12.1, é mudar a variável FAT de inteiro para real no Programa11b.

Testar isso, fazendo uma nova versão do programa11b.

Assim será possível calcular o fatorial de até 34.

Para 35 já deverá ocorrer overflow, isto é, será ultrapassado o limite dos números reais de precisão simples, que é 3.40×10^{38}.

Exercício 11.3

1) Criar um projeto com o nome **Cap11** e inserir nele o programa-fonte **Cap11.f90**
2) **Alterar o programa-fonte principal4.f90, da seção 11.4,** da seguinte forma:
   a) Adaptar a sub-rotina FATORIAL e sua chamada na sub-rotina OUTRAS para que o valor do fatorial (FAT) seja passado à sub-rotina OUTRAS.
   b) Implementar a sub-rotina ESCREVE para: criar o arquivo de saída **Cap11.TXT**; escrever neste arquivo o nome completo do aluno; e escrever nele os valores das variáveis A, B, C, INTEIRO e FAT junto com seus nomes como comentários. Esta nova sub-rotina deverá ser chamada dentro da sub-rotina OUTRAS, após a chamada da sub-rotina FATORIAL.
   c) Implementar a sub-rotina ARQUIVO para mostrar, com o aplicativo Bloco de Notas (Notepad), o conteúdo do arquivo criado pela sub-rotina ESCREVE. A sub-rotina ARQUIVO deverá ser chamada pela sub-rotina OUTRAS, após a chamada da sub-rotina ESCREVE.