

Capítulo 15. INICIALIZAÇÃO, TEMPO DE CPU E DOS

OBJETIVOS DO CAPÍTULO

- Inicializar variáveis e constantes junto com suas definições
- Versões DEBUG e RELEASE de um programa-executável
- Comandos do FORTRAN: PARAMETER, Formato A<X>
- Função do FORTRAN: TIMEF
- Comandos do DOS: MKDIR, COPY, ERASE, CD, RENAME, “arquivo”.BAT

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**

15.1 programa15a.f90

- 1) Objetivos do programa:
 - (a) usar os novos comandos do FORTRAN: PARAMETER e formato A<X>
 - (b) ao definir variáveis, inicializar seus valores; e
 - (c) escrever variáveis do tipo caracter com tamanho exato de seu conteúdo, usando o comando FORMAT.
- 2) No Fortran, seguindo o [procedimento-padrão](#), **criar um projeto** com o nome **programa15a**
- 3) No Fortran, seguindo o [procedimento-padrão](#), **criar e inserir** no projeto o programa-fonte **programa15a.f90**
- 4) Dentro do espaço de edição do Fortran, na subjanela maior, **copiar** exatamente o texto em vermelho mostrado na **Tabela 15.1**.

Tabela 15.1 Programa15a.f90

```
USE PORTLIB
IMPLICIT NONE
INTEGER VER, X
CHARACTER(50) SAIDA, TEXTO, COMENTARIO

INTEGER :: UNIT = 20

REAL*8, PARAMETER :: Pi = 3.14159E+00
```

```

VER = SYSTEM("Notepad DADOS.TXT" )

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

READ(1,*) COMENTARIO
READ(1,*) SAIDA

CLOSE(1)

OPEN(UNIT, file = SAIDA )

WRITE(UNIT,10) UNIT, Pi
10 FORMAT( /, 5X, "UNIT = ", I4, &
          2/, 5X, "Pi   = ", 1PE25.15 )

WRITE(UNIT,11) COMENTARIO
11 FORMAT( /, A50, " = COMENTARIO" )

X = LEN(TRIM(ADJUSTL(COMENTARIO)))

WRITE(UNIT,12) TRIM(ADJUSTL(COMENTARIO))
12 FORMAT( /, A<X>, " = COMENTARIO" )

CLOSE(UNIT)

TEXTO = "Notepad " // SAIDA

VER = SYSTEM( TEXTO )

END

```

5) Comentários sobre o programa:

- (a) Na linha `INTEGER :: UNIT = 20` está sendo definida a variável UNIT como sendo do tipo inteiro e está sendo atribuído o valor 20 a ela. Ou seja, ela está sendo inicializada com o valor 20.
- (b) Na linha `REAL*8, PARAMETER :: Pi = 3.14159E+00` está sendo definida a variável Pi como sendo do tipo real de dupla precisão, está sendo atribuído o valor 3.14159 a ela. Ou seja, ela está sendo inicializada com o valor 3.14159. Além disso, ela está sendo definida como uma constante através do comando PARAMETER.
- (c) Variáveis inicializadas podem ser alteradas dentro do programa.

- (d) Variáveis inicializadas e definidas como constantes, com o comando PARAMETER, não podem ser alteradas dentro do programa. Isso gera erro de compilação.
- (e) Diversas variáveis podem ser inicializadas numa mesma linha de programa. Basta separá-las por vírgula.
- (f) Na linha `WRITE (UNIT, 12) TRIM (ADJUSTL (COMENTARIO))` escreve-se a variável COMENTARIO no arquivo definido por UNIT, com o formato definido pelo número 12. O uso das funções TRIM e ADJUSTL permite eliminar espaços em branco existentes no conteúdo da variável COMENTARIO, conforme visto no Capítulo 4.
- (g) Na linha `12 FORMAT (/, A<X>, " = COMENTARIO")` o identificador A, usado para escrever variáveis do tipo caracter, está sendo usado numa forma avançada, que permite escrever variáveis do tipo caracter com o tamanho exato de seu conteúdo. Isto é, sem espaços em branco, que ocorrem devido ao pré-dimensionamento que é necessário fazer ao se definir uma variável caracter, no caso CHARACTER (50) SAIDA, TEXTO, COMENTARIO. Na sintaxe A<X>, X é uma variável inteira, definida por X = LEN (TRIM (ADJUSTL (COMENTARIO))), cujo valor resulta da aplicação da função LEN.
- 6) Executar **Build, Compile** para compilar o programa.
- 7) Gerar o programa-executável fazendo **Build, Build**.
- 8) Antes de executar o programa, é necessário criar o arquivo de dados e inserir nele os respectivos dados. No caso do programa15a.f90, é necessário **criar o arquivo “DADOS.TXT” e inserir os dois dados que correspondem às variáveis COMENTARIO e SAIDA. Usar, por exemplo, os dados mostrados na Figura 15.1.**

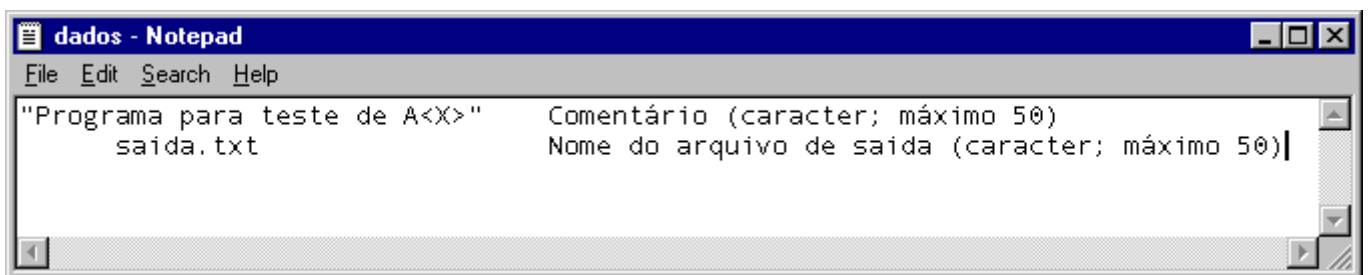


Figura 15.1 Exemplo de arquivo de dados para o programa15a.f90.

- 9) Executar o programa através de **Build, Execute**. O resultado deve ser o mostrado na Figura 15.2.
- 10) **Executar novamente o programa com outros dados e analisar os novos resultados.**
- 11) No Fortran, para fechar o projeto atual, executar **File, Close Workspace**.

15.2 programa15b.f90

- 1) Objetivos do programa:
 - (a) usar uma nova função do FORTRAN: TIMEF; e
 - (b) mostrar a diferença, em termos de tempo de processamento, das versões DEBUG e RELEASE de um programa-executável.

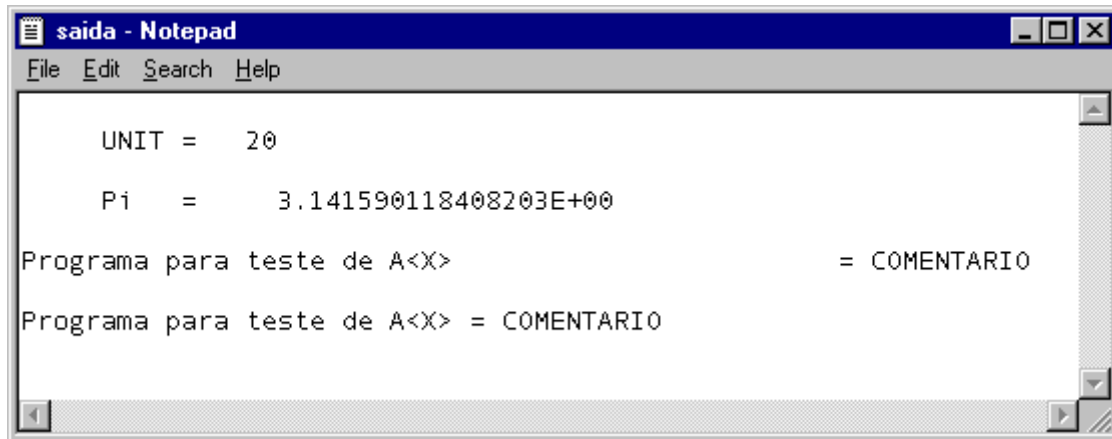


Figura 15.2 Resultado do programa15a.f90 para os dados da Figura 15.1.

- 2) No Fortran, seguindo o [procedimento-padrão](#), **criar um projeto** com o nome **programa15b**
- 3) No Fortran, seguindo o [procedimento-padrão](#), **criar e inserir** no projeto o programa-fonte **programa15b.f90**
- 4) Dentro do espaço de edição do Fortran, na subjanela maior, **copiar** exatamente o texto em vermelho mostrado na **Tabela 15.2**.

Tabela 15.2 Programa15b.f90

```

USE PORTLIB
IMPLICIT NONE
INTEGER VER, PASSOS, I
CHARACTER(50) SAIDA, TEXTO

INTEGER :: UNIT = 20

REAL*8 T1, T2, SOMA

VER = SYSTEM("Notepad DADOS.TXT" )

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

READ(1,*) PASSOS
READ(1,*) SAIDA

```

```

CLOSE (1)

T1 = TIMEF()

SOMA = 0.0D0
DO I = 1, PASSOS
    SOMA = SOMA + I
END DO

T2 = TIMEF()

OPEN(UNIT, file = SAIDA )

WRITE(UNIT,11) PASSOS, SOMA, T1, T2, T2-T1
11 FORMAT( 1/, "*** PRIMEIRA SOMA ***", &
          1/, 5X, "PASSOS = ", I12, &
          1/, 5X, "SOMA = ", 1PE25.10E3, &
          1/, 5X, "T1 (segundos) = ", 1PE25.10E3, &
          1/, 5X, "T2 (segundos)= ", 1PE25.10E3, &
          1/, 5X, "Tempo de CPU = T2 - T1 (segundos)= ", 1PE25.10E3 )

T1 = TIMEF()

SOMA = 0.0D0
DO I = 1, PASSOS
    SOMA = SOMA + I
END DO

T2 = TIMEF()

WRITE(UNIT,12) PASSOS, SOMA, T1, T2, T2-T1
12 FORMAT( 1/, "*** SEGUNDA SOMA ***", &
          1/, 5X, "PASSOS = ", I12, &
          1/, 5X, "SOMA = ", 1PE25.10E3, &
          1/, 5X, "T1 (segundos) = ", 1PE25.10E3, &
          1/, 5X, "T2 (segundos)= ", 1PE25.10E3, &
          1/, 5X, "Tempo de CPU = T2 - T1 (segundos)= ", 1PE25.10E3 )

CLOSE(UNIT)

TEXTO = "Notepad " // SAIDA

```

```
VER = SYSTEM( TEXTO )
```

```
END
```

5) Comentários sobre o programa:

- (a) A função TIMEF faz parte da biblioteca PORTLIB. Ela é usada para medir o tempo de processamento ou tempo de CPU do programa entre dois pontos desejados. O tempo de CPU é o tempo efetivamente gasto pelo processador do computador executando um programa ou parte de um programa.
 - (b) A função TIMEF mede o tempo de CPU em segundos.
 - (c) A função TIMEF é usada como na linha `T1 = TIMEF()` do programa, onde T1 deve ser uma variável do tipo real dupla.
 - (d) Dentro de um programa, a primeira chamada da função de TIMEF zera a contagem de tempo. As chamadas sucessivas, registram o tempo total transcorrido entre a zeragem e um ponto específico do programa. Desta forma, o tempo de processamento entre dois pontos é igual à diferença entre os tempos registrados nestes dois pontos.
 - (e) Como se poderá perceber nos exemplos, a função TIMEF não tem precisão muito elevada. A repetição de um programa pode gerar diferenças de até ± 0.05 segundo.
 - (f) No FORTRAN, a medição do tempo de processamento de um programa também pode ser feita com as funções DTIME e ETIME, e a sub-rotina CPU_TIME.
 - (g) No FORTRAN 95, por default, uma variável do tipo inteiro não pode ter valor maior do que $2^{31}-1$, que corresponde a 2 147 483 647, ou seja, mais de 2 bilhões.
 - (h) Por default, quando se compila e se gera o executável de um programa, obtém-se a chamada versão DEBUG. Ela é útil para se encontrar erros de edição ou de uso dos comandos do FORTRAN, isto é, erros de sintaxe ou erros de compilação. Mas, em termos de tempo de processamento, a versão DEBUG é mais lenta (podendo chegar a 50%) do que a versão RELEASE. Além disso, a versão DEBUG geralmente resulta num programa-executável cujo arquivo precisa de mais memória em disco do que a versão RELEASE. Para definir o tipo de versão do programa, no menu principal do Fortran, deve-se executar “Build, Set Default Configuration...”, escolher a opção desejada e clicar no botão OK. Depois, deve-se compilar e gerar o executável do programa.
- 6) Executar **Build, Compile** para compilar o programa.
 - 7) Gerar o programa-executável fazendo **Build, Build**.
 - 8) Antes de executar o programa, é necessário criar o arquivo de dados e inserir nele os respectivos dados. No caso do programa15b.f90, é necessário **criar o arquivo “DADOS.TXT” e inserir os dois**

dados que correspondem às variáveis PASSOS e SAIDA. Usar, por exemplo, os dados mostrados na Figura 15.3, onde PASSOS é igual a 100 milhões.

- 9) Executar o programa através de **Build, Execute**. O resultado é mostrado na Figura 15.4. Os tempos de processamento se referem à execução do programa num microcomputador Pentium III de 750 MHz. Analisando-se o programa, deve-se perceber que os dois tempos de CPU deveriam ter exatamente o mesmo valor, mas na Figura 15.4 nota-se que há uma diferença entre eles de 0.01 s. A cada execução do programa, tanto os valores do tempo de CPU quanto suas diferenças podem variar.

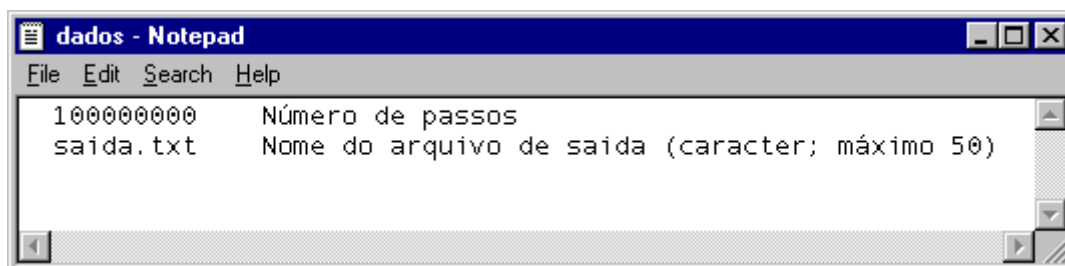


Figura 15.3 Exemplo de arquivo de dados para o programa15b.f90.

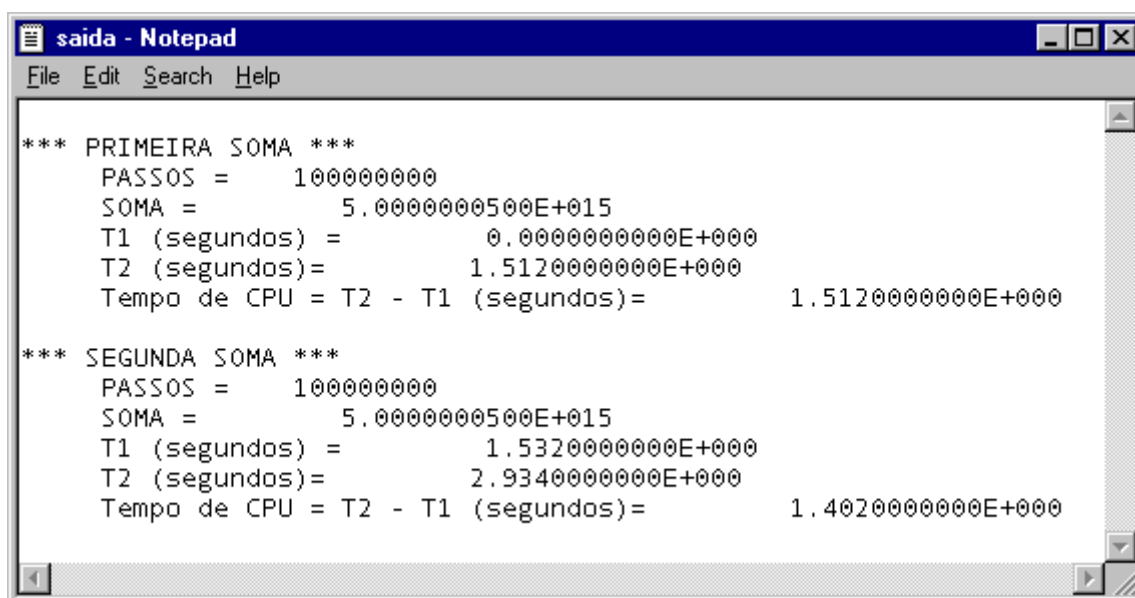
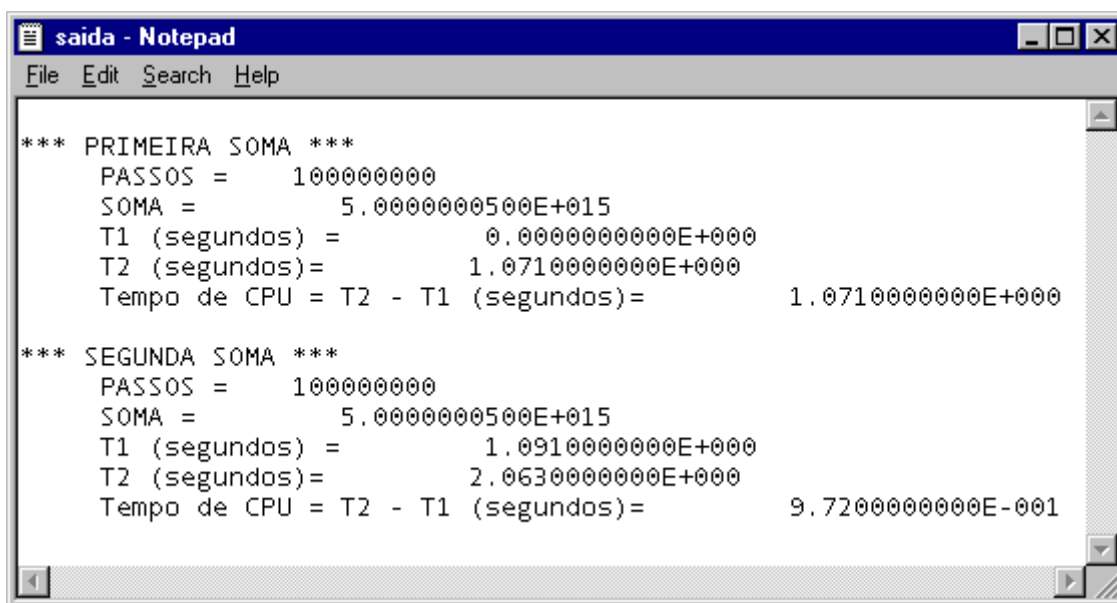


Figura 15.4 Resultado do programa15b.f90 para os dados da Figura 15.3, versão DEBUG.

- 10) **Executar novamente o programa com outros dados e analisar os novos resultados.** Utilizar, por exemplo, PASSOS = 10 milhões, 1 milhão e 1 bilhão.
- 11) **Verificar se dentro do diretório do projeto existe um subdiretório chamado DEBUG.** Se não, isto é, se o diretório existente for chamado RELEASE, onde se lê DEBUG, nos itens 12 a 14, abaixo, deve-se ler RELEASE e vice-versa.
- 12) Mudar a versão do programa para RELEASE. Para fazer isso, executar **Build, Set Default Configuration...**, escolher a opção **RELEASE**, clicar no botão **OK**. Depois, executar **Build**,

Compile para compilar novamente o programa. Gerar o novo programa-executável fazendo **Build, Build**.

- 13) **Verificar se dentro do diretório do projeto também existe um subdiretório chamado RELEASE.**
- 14) Executar o programa através de **Build, Execute**. O novo resultado é mostrado na Figura 15.5. Novamente, os dois tempos de CPU deveriam ter exatamente o mesmo valor, mas na Figura 15.5 nota-se que há uma diferença entre eles de 0.099 s. Comparando-se os tempos de processamento, verifica-se que a versão DEBUG é mais lenta cerca de 43% do que a versão RELEASE.



```
saida - Notepad
File Edit Search Help

*** PRIMEIRA SOMA ***
PASSOS = 100000000
SOMA = 5.00000000500E+015
T1 (segundos) = 0.0000000000E+000
T2 (segundos)= 1.0710000000E+000
Tempo de CPU = T2 - T1 (segundos)= 1.0710000000E+000

*** SEGUNDA SOMA ***
PASSOS = 100000000
SOMA = 5.00000000500E+015
T1 (segundos) = 1.0910000000E+000
T2 (segundos)= 2.0630000000E+000
Tempo de CPU = T2 - T1 (segundos)= 9.7200000000E-001
```

Figura 15.5 Resultado do programa15b.f90 para os dados da Figura 15.3, versão RELEASE.

- 15) **Executar novamente o programa com outros dados e analisar os novos resultados.** Utilizar, por exemplo, PASSOS = 10 milhões, 1 milhão e 1 bilhão.
- 16) No Fortran, para fechar o projeto atual, executar **File, Close Workspace**.

15.3 programa15c.f90

- 1) Objetivo do programa: utilizar comandos do DOS durante a execução do programa.
- 2) No Fortran, seguindo o [procedimento-padrão](#), **criar um projeto** com o nome **programa15c**
- 3) No Fortran, seguindo o [procedimento-padrão](#), **criar e inserir** no projeto o programa-fonte **programa15c.f90**
- 4) Dentro do espaço de edição do Fortran, na subjanela maior, **copiar** exatamente o texto em vermelho mostrado na **Tabela 15.3**, que é o programa15c.f90.

Tabela 15.3 Programa15c.f90

```

USE PORTLIB
IMPLICIT NONE
INTEGER DOS
CHARACTER(50)  SAIDA, COMENTARIO
INTEGER :: UNIT = 20

DOS = SYSTEM("Notepad DADOS.TXT" )

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

READ(1,*) COMENTARIO
READ(1,*) SAIDA

CLOSE(1)

OPEN(UNIT, file = SAIDA )

WRITE(UNIT,11) COMENTARIO, SAIDA
11 FORMAT(1/, "COMENTARIO = ", A50, &
          2/, "SAIDA      = ", A50  )

CLOSE(UNIT)

! edição de comandos no arquivo EXECUTA.BAT

OPEN(UNIT, file = "EXECUTA.BAT" )

WRITE(UNIT,*) "MKDIR C:\TEMP\FORTRAN"

WRITE(UNIT,*) "COPY " // TRIM(SAIDA) // " C:\TEMP\FORTRAN\" // TRIM(SAIDA)

WRITE(UNIT,*) "ERASE " // TRIM(SAIDA)

WRITE(UNIT,*) "CD C:\TEMP\FORTRAN\"

WRITE(UNIT,*) "RENAME "// TRIM(SAIDA) // " NOVO.TXT"

CLOSE (UNIT)

! fim

```

```
DOS = SYSTEM ( "EXECUTA.BAT" )
```

```
DOS = SYSTEM( "Notepad C:\TEMP\FORTRAN\NOVO.TXT" )
```

```
END
```

5) Comentários sobre o programa:

- (a) O comando do DOS chamado MKDIR é usado para criar um novo diretório. Ele é usado na forma:

```
MKDIR DIRETORIO
```

onde DIRETORIO é o nome do diretório a ser criado, incluindo o caminho completo desde a raiz do HD (hard disk) ou disquete. Se o caminho não é especificado, o diretório é criado dentro do diretório no qual o comando é executado.

- (b) O comando do DOS chamado COPY é usado para copiar um arquivo em outro. Ele é usado na forma:

```
COPY ARQ1 ARQ2
```

onde ARQ1 é o nome do arquivo a ser copiado em outro arquivo com o nome de ARQ2. Junto a ARQ1 e ARQ2 deve-se definir o diretório de cada arquivo, incluindo o caminho completo desde a raiz do HD (hard disk) ou disquete. Se os diretórios e caminhos não são especificados, ARQ1 deve existir no diretório no qual o comando é executado, e ARQ2 é gerado no mesmo diretório.

- (c) O comando do DOS chamado ERASE é usado para eliminar ou deletar um arquivo. Ele é usado na forma:

```
ERASE ARQ
```

onde ARQ é o nome do arquivo a ser eliminado. Junto a ARQ deve-se definir o seu diretório, incluindo o caminho completo desde a raiz do HD (hard disk) ou disquete. Se o diretório e caminho não são especificados, ARQ deve existir no diretório no qual o comando é executado.

- (d) O comando do DOS chamado CD é usado para mudar a execução do programa para outro diretório. Ele é usado na forma:

```
CD DIRETORIO
```

onde DIRETORIO é o nome do diretório para o qual passa a ser executado o programa, incluindo o caminho completo desde a raiz do HD (hard disk) ou disquete. Subentende-se que o diretório existe dentro do diretório no qual o comando é executado.

- (e) O comando do DOS chamado RENAME é usado para mudar o nome de um arquivo. Ele é usado na forma:

```
RENAME ARQ1 ARQ2
```

onde ARQ1 é o nome do arquivo existente, e ARQ2 é o novo nome. Aqui valem os mesmos comentários para ARQ1 e ARQ2 feitos no item (b), acima.

- (f) Ao se executar um arquivo com extensão “.BAT”, são executados todos os comandos DOS dentro deste arquivo, linha por linha, de cima para baixo.
 - (g) Exemplos de aplicação dos comandos acima são apresentados no programa.
 - (h) Existem diversos outros comandos do DOS que podem ser empregados em função do objetivo desejado.
- 6) Executar **Build, Compile** para compilar o programa.
 - 7) Gerar o programa-executável fazendo **Build, Build**.
 - 8) Antes de executar o programa, é necessário criar o arquivo de dados e inserir nele os respectivos dados. No caso do programa15c.f90, é necessário **criar o arquivo “DADOS.TXT” e inserir os dois dados que correspondem às variáveis COMENTARIO e SAIDA. Usar, por exemplo, os dados mostrados na Figura 15.1.**
 - 9) Algoritmo do programa:
 - (a) ocorre a abertura do arquivo “DADOS.TXT”
 - (b) são lidos os dois dados
 - (c) cria-se o arquivo o arquivo de saída; escreve-se nele os dois dados lidos; e fecha-se este arquivo
 - (d) cria-se o arquivo “EXECUTA.BAT”; dentro dele, são escritas diversas instruções DOS; fecha-se este arquivo
 - (e) acessa-se o DOS para executar as instruções contidas no arquivo “EXECUTA.BAT”
 - (f) acessa-se o DOS para abrir, com o aplicativo Notepad, o arquivo “NOVO.TXT” localizado em “C:\TEMP\FORTRAN\”
 - 10) Executar o programa através de **Build, Execute. Analisar os resultados.** A Figura 15.6 mostra o conteúdo do **arquivo “EXECUTA.BAT”**, gerado pelo programa15c.f90; **verificar sua existência no diretório do projeto.** A Figura 15.7 mostra os comandos que foram executados no DOS, como resultado da execução do arquivo “EXECUTA.BAT”. A Figura 15.8 mostra o conteúdo do **arquivo** de resultado do programa15c.f90; deve-se notar que seu nome é **“NOVO.TXT”** e que ele se localiza no diretório “C:\TEMP\FORTRAN\”; **verificar sua existência**; além disso, o arquivo de saída foi eliminado do diretório do projeto.
 - 11) Encerrar a sessão seguindo o [procedimento-padrão](#).

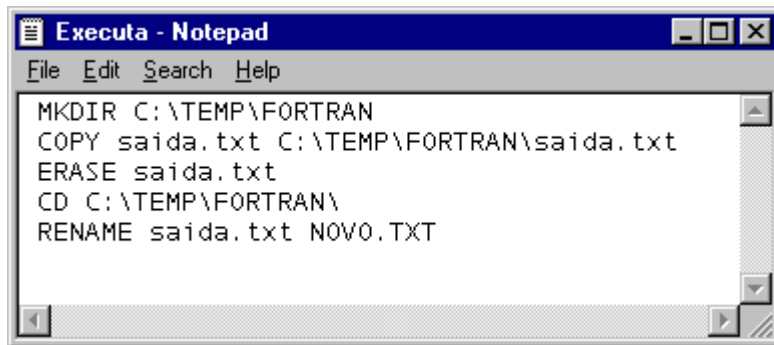


Figura 15.6 Conteúdo do arquivo “EXECUTA.BAT” gerado pelo programa15c.f90.

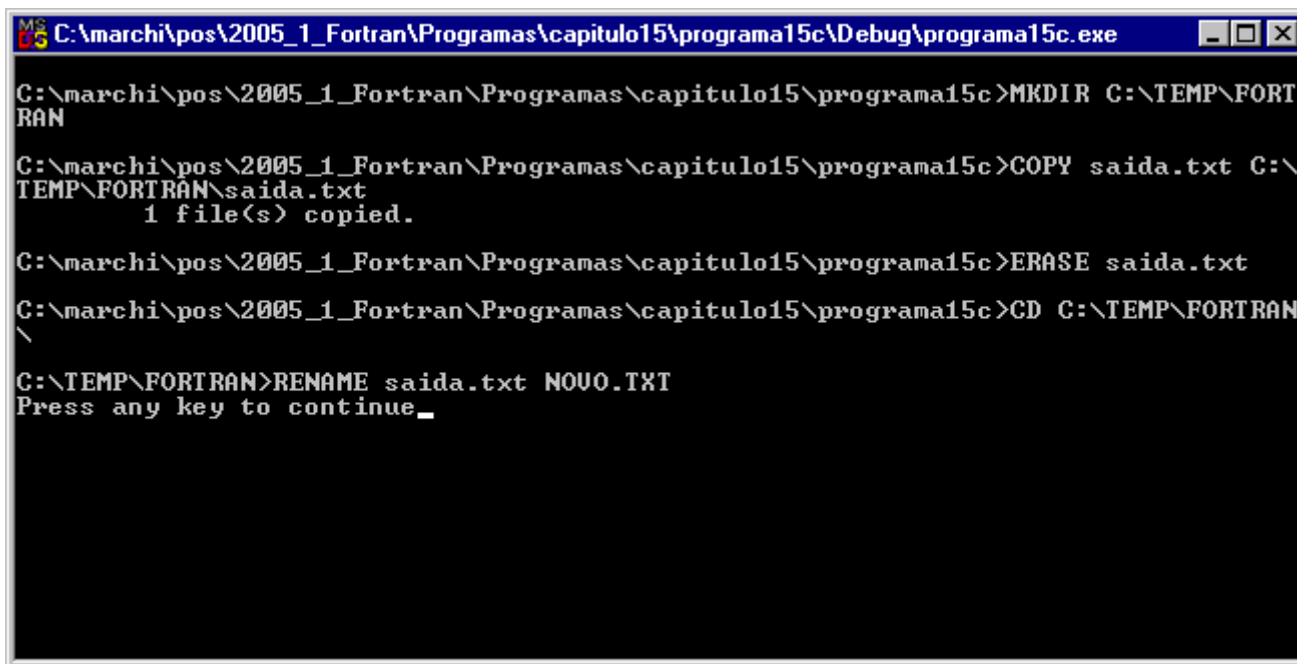


Figura 15.7 Janela DOS resultante da execução do programa15c.f90.

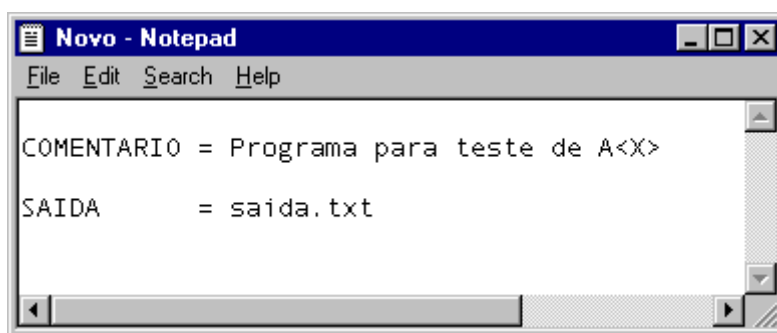


Figura 15.8 Resultado da execução do programa15c.f90.

15.4 EXERCÍCIOS

Exercício 15.1

Adaptar o programa15a.f90, Tabela 15.1, para:

- (a) inicializar uma variável do tipo caracter;
- (b) inicializar uma constante do tipo caracter;
- (c) inicializar duas variáveis do tipo inteiro numa mesma linha de programa;
- (d) inicializar duas constantes do tipo inteiro numa mesma linha de programa; e
- (e) escrever num arquivo os conteúdos das variáveis e constantes dos itens (a) a (d).

Exercício 15.2

Adaptar o programa15b.f90, Tabela 15.2, para usar a função DTIME junto com TIMEF e comparar o tempo de CPU medido por cada função.

Exercício 15.3

Adaptar o programa15b.f90, Tabela 15.2, para que a variável SOMA seja do tipo inteiro. Notar a redução do tempo de CPU que ocorre.

Exercício 15.4

Adaptar o programa15b.f90, Tabela 15.2, para obter e escrever o tempo de CPU gasto entre a primeira e a última chamada da função TIMEF.

Exercício 15.5

Adaptar o programa15b.f90, Tabela 15.2, para incluir, antes da última chamada da função TIMEF, a instrução

READ(*,*) “Espere alguns segundos e pressione a tecla ENTER”

Analisar o efeito desta instrução vazia no tempo de CPU.

Exercício 15.6

Adaptar o programa15c.f90, Tabela 15.3, visando generalizar o nome do diretório

“C:\TEMP\FORTRAN”

para qualquer nome que o usuário defina através do arquivo de dados.