Capítulo 15. Inicialização, tempo de CPU e DOS

29 Out 2024

OBJETIVOS DO CAPÍTULO

- Inicializar variáveis e constantes junto com suas definições
- Comando do FORTRAN: PARAMETER
- Medição do tempo de CPU
- Versões DEBUG e RELEASE de um programa-executável
- Vários comandos do DOS

15.1 programa15a.f90

1) Objetivos do programa:
   (a) usar o novo comando do FORTRAN: PARAMETER; e
   (b) ao definir variáveis e constantes, inicializar seus valores
2) No Fortran, **criar um projeto** com o nome **programa15a**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **programa15a.f90**
4) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 15.1**.
5) Comentários sobre o programa:
   (a) Na linha `INTEGER :: UNIT = 20` está sendo definida a variável UNIT como sendo do tipo inteiro e ela está sendo inicializada com o valor 20.
   (b) Na linha `REAL*8, PARAMETER :: Pi = 3.141592653589793d+0` está sendo definida a constante Pi, através do comando PARAMETER, como sendo do tipo real de precisão dupla e ela está sendo inicializada com seu valor.
   (c) Variáveis inicializadas podem ser alteradas dentro do programa.
   (d) Constantes definidas com o comando PARAMETER, e inicializadas com um valor, não podem ter este valor alterado dentro do programa. Isso gera erro de compilação.
   (e) Diversas variáveis podem ser inicializadas em uma mesma linha de programa. Basta separá-las por vírgula.
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 um dado que corresponde à variável SAIDA. Usar, por exemplo, o dado mostrado na Figura 15.1.**
9) Executar o programa através de **Build, Execute**. O resultado deve ser o mostrado na Figura 15.2.
10) **Executar novamente o programa com outro dado e analisar os novos resultados**.

Tabela 15.1 Programa15a.f90

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

INTEGER :: UNIT = 20

REAL*8, PARAMETER :: Pi = 3.141592653589793d+0

VER = SYSTEM("Notepad DADOS.TXT" )

OPEN(1, file = "DADOS.TXT" )
READ(1,*) SAIDA
CLOSE(1)

OPEN(UNIT, file = SAIDA )

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

CLOSE(UNIT)

TEXTO = "Notepad " // SAIDA

VER = SYSTEM( TEXTO )

END
```

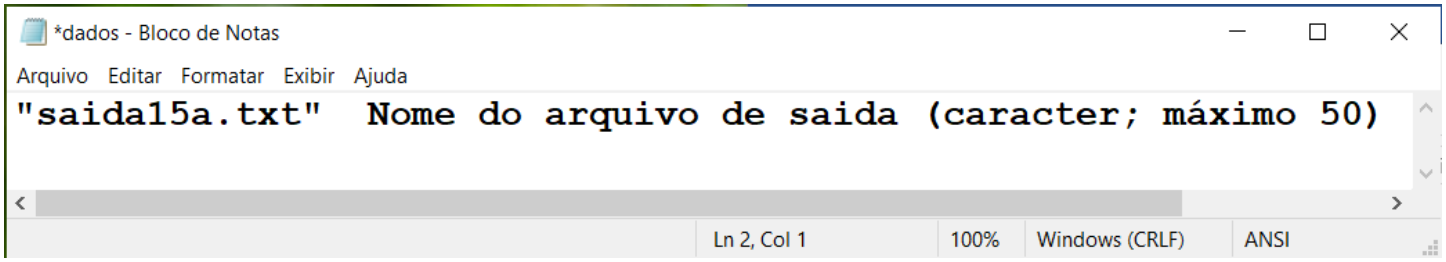

Figura 15.1 Exemplo de arquivo de dados para o programa15a.f90.

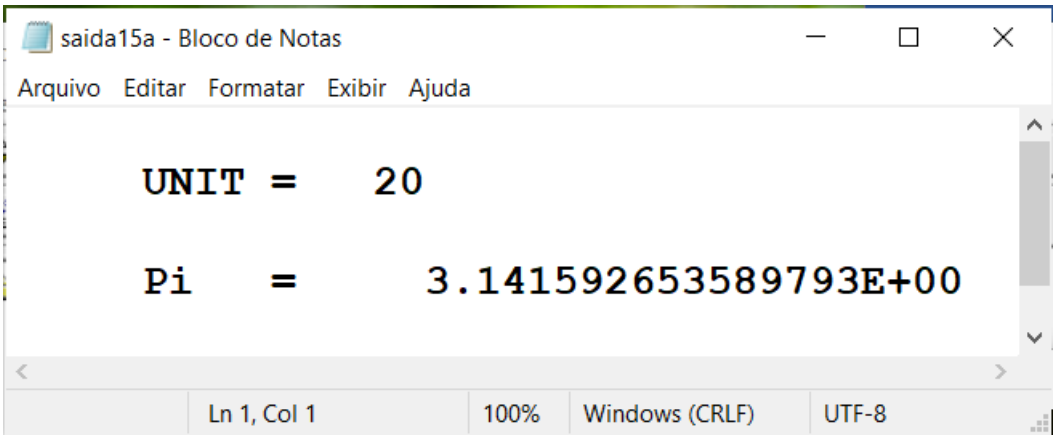

Figura 15.2 Resultado do programa15a.f90 para os dados da Figura 15.1.

15.2 programa15b.f90

1) Objetivos do programa:
   (a) Medir o tempo de CPU usado na execução do programa; e
   (b) mostrar a diferença, em termos de tempo de CPU, das versões DEBUG e RELEASE de um programa-executável.
2) No Fortran, **criar um projeto** com o nome **programa15b**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **programa15b.f90**
4) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 15.2**.
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 TIMEF zera a contagem de tempo. As chamadas sucessivas irão registrar 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 da execução de um programa geralmente resulta em diferenças de até ± 0.05 segundo em relação a um valor médio.
   (f) Um procedimento que ajuda a melhorar a precisão da medida do tempo de CPU é repetir, a execução da parte de interesse do programa, por um determinado número de vezes, de tal forma que o tempo total de CPU aumente e minimize a imprecisão de sua medida. Isto é feito com o ciclo j no programa15b.f90.

Tabela 15.2 Programa15b.f90

```
USE PORTLIB
IMPLICIT NONE
INTEGER VER, PASSOS, I, repeticoes, j
CHARACTER(50) SAIDA
CHARACTER(10) t1b,t2b

INTEGER :: UNIT = 20

REAL*8 T1, T2, SOMA

VER = SYSTEM("Notepad DADOS.TXT" )

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

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

CLOSE(1)

T1 = TIMEF()
CALL DATE_AND_TIME(TIME = t1b)

do j = 1, repeticoes
   SOMA = 0.0D0
   DO I = 1, PASSOS
      SOMA = SOMA + I
   END DO
end do

T2 = TIMEF()
CALL DATE_AND_TIME(TIME = t2b)

OPEN(UNIT, file = SAIDA )

write(*,*) "t1b = ", t1b
write(*,*) "t2b = ", t2b

WRITE(UNIT,11) repeticoes, PASSOS, SOMA, T1, T2, T2-T1, (t2-t1)/repeticoes
11 FORMAT( 1/, "*** PRIMEIRA SOMA ***", &
           1/, 5X, "repetições = ", I6,   &
           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 total = T2 - T1 (segundos)= ", 1PE25.10E3, &
           1/, 5X, "Tempo de CPU médio por repetição (segundos) =  ", 1PE25.10E3 )

T1 = TIMEF()
CALL DATE_AND_TIME(TIME = t1b)

do j = 1, repeticoes
   SOMA = 0.0D0
   DO I = 1, PASSOS
      SOMA = SOMA + I
   END DO
end do

T2 = TIMEF()
CALL DATE_AND_TIME(TIME = t2b)

write(*,*) "t1b = ", t1b
write(*,*) "t2b = ", t2b

WRITE(UNIT,12) repeticoes, PASSOS, SOMA, T1, T2, T2-T1, (t2-t1)/repeticoes
12 FORMAT( 1/, "*** SEGUNDA SOMA ***", &
           1/, 5X, "repetições = ", I6,   &
           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 total = T2 - T1 (segundos)= ", 1PE25.10E3, &
           1/, 5X, "Tempo de CPU médio por repetição (segundos) =  ", 1PE25.10E3 )

CLOSE(UNIT)

VER = SYSTEM( "Notepad " // SAIDA )

END
```

(g) 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 DATE_AND_TIME (que é usada no programa15b.f90), entre outras.

(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 do que a versão RELEASE, que geralmente usa entre 1/3 a ½ do tempo de CPU da versão DEBUG. Além disso, a versão DEBUG geralmente resulta em um programa-executável cujo arquivo precisa de mais memória em disco do que a versão RELEASE.

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

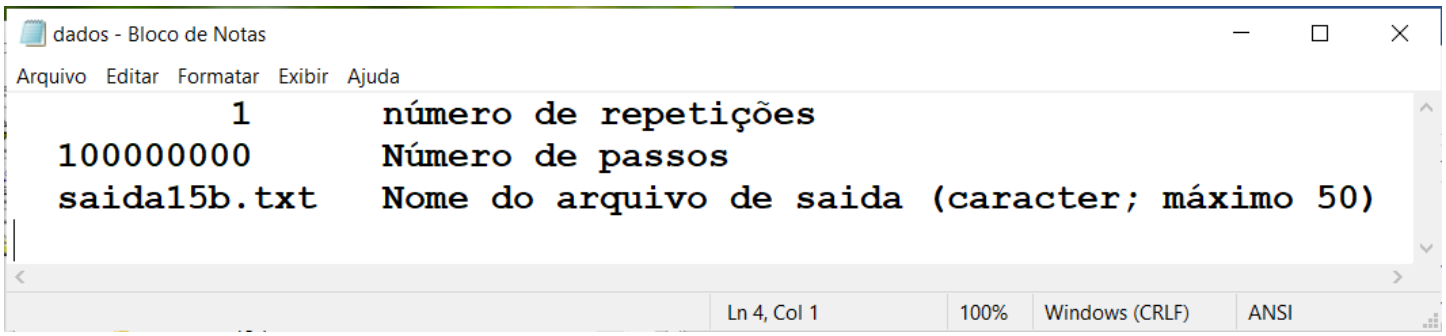

Figura 15.3 Exemplo de arquivo de dados para o programa15b.f90.

9) Executar o programa através de **Build, Execute**. O resultado é mostrado na Figura 15.4 para a função TIMEF. Os tempos de processamento se referem à execução do programa em notebook Samsung com processador Intel i3, e a sua precisão é de milésimos de segundo. 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 0.026 s. A cada execução do programa, tanto os valores do tempo de CPU quanto suas diferenças podem variar dentro da precisão geralmente esperada de 0.05 s.

10) Na Figura 15.5 são mostrados os resultados obtidos com a sub-rotina DATE_AND_TIME. Os valores representam hora, minuto, segundo e milésimos de segundo. A primeira diferença entre t2b e t1b é de 0.584 s e a segunda é de 0.561 s. Estas diferenças devem ser comparadas aos dois valores da Figura 15.4. Portanto, a diferença entre os dois resultados de DATE_AND_TIME é de 0.023 s

11) **Executar novamente o programa com 10 repeticoes.** Notar que agora os resultados mostrados nas Figuras 15.6 e 15.7 têm precisão de décimos de milésimos de segundo. A diferença entre os dois resultados do TIMEF é de 0.0091 s. Na Figura 15.7, a primeira diferença entre t2b e t1b é de 0.4410 s por repetição, e a segunda é de 0.4324 s. Estas diferenças devem ser comparadas aos dois valores da Figura 15.6. Portanto, a diferença entre os dois resultados de DATE_AND_TIME é de 0.0086 s. Esta precisão pode continuar a ser aumentada ao se aumentar o número de repetições dos cálculos.

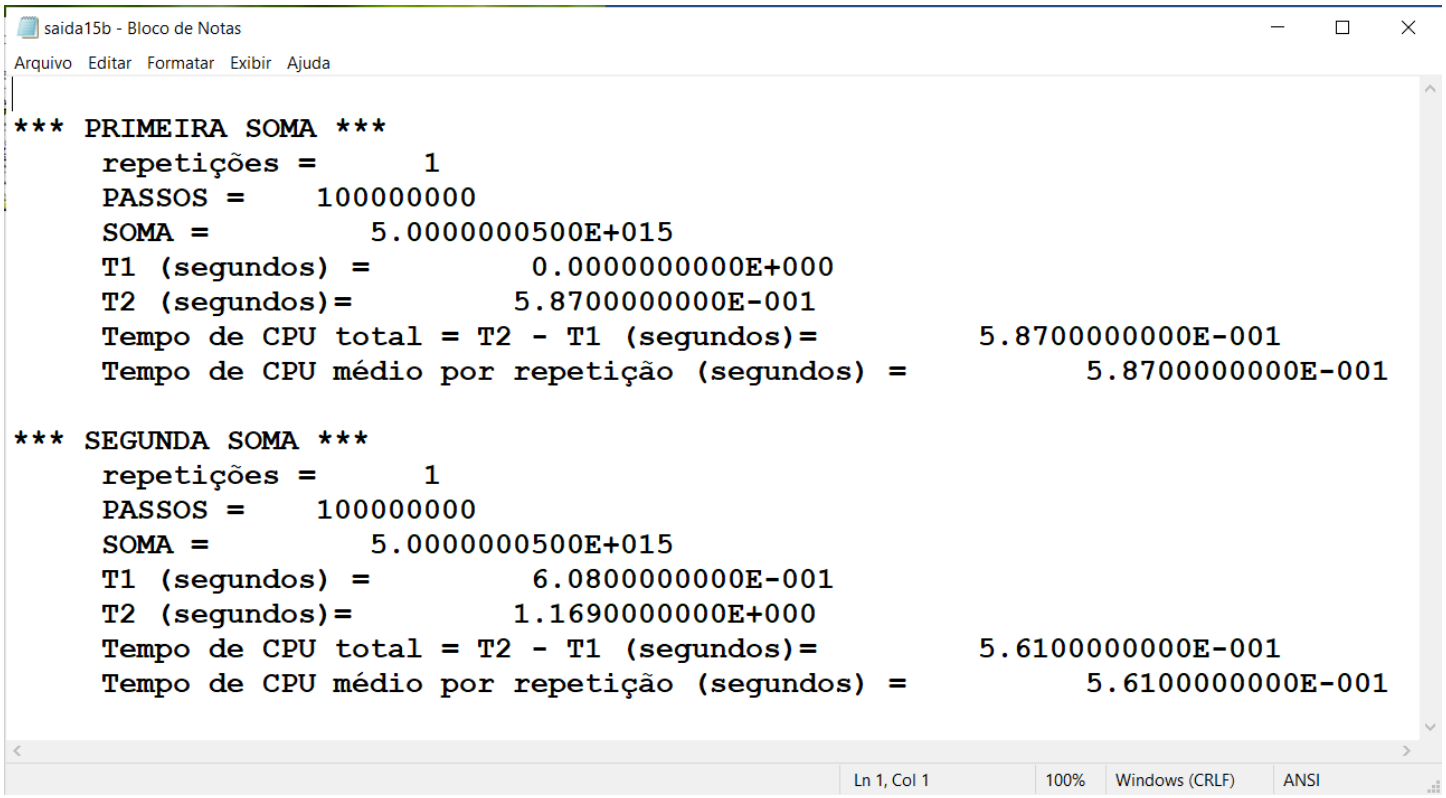

Figura 15.4 Resultado TIMEF do programa15b.f90 para os dados da Figura 15.3, versão DEBUG.

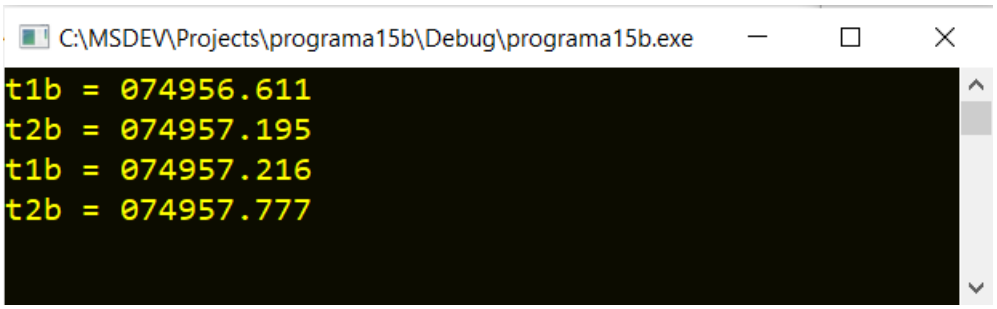

Figura 15.5 Resultado DATE_AND_TIME do programa15b.f90 para os dados da Figura 15.3, versão DEBUG.

12) **Executar novamente o programa com outros dados e analisar os novos resultados**. Utilizar, por exemplo, PASSOS = 10 milhões, 1 milhão e 1 bilhão; para cada valor de PASSOS, testar com repeticoes = 1 e 10.
13) **Verificar que dentro da pasta do projeto existe uma subpasta chamada DEBUG**. Notar que o tamanho do arquivo programa15b.exe é de 193 kB.
14) 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**.
15) **Verificar que, agora, dentro da pasta do projeto também existe uma subpasta chamada RELEASE**, e que o tamanho do arquivo programa15b.exe é de 136 kB, portanto, menor que a versão DEBUG.

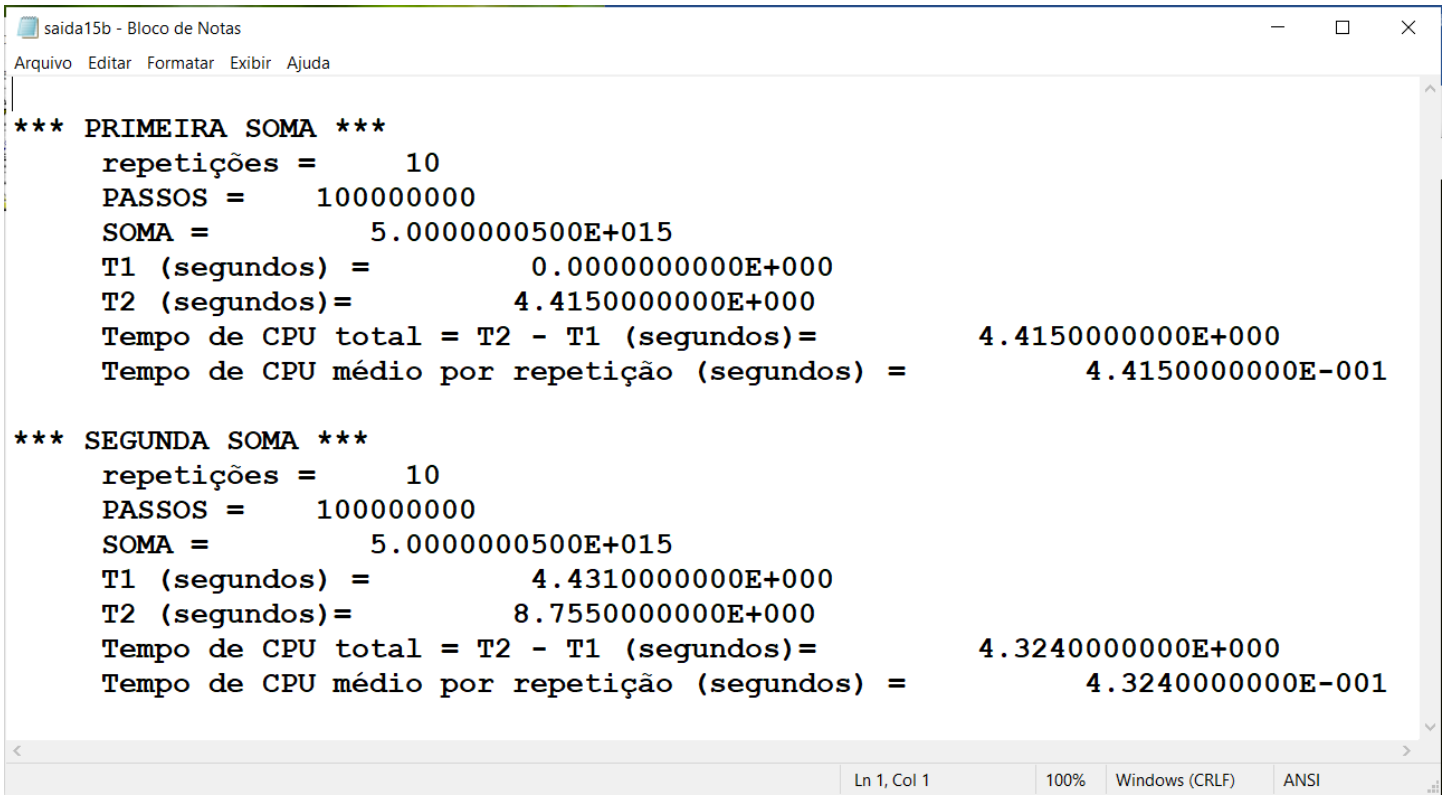

Figura 15.6 Resultado TIMEF do programa15b.f90 para 10 repetições, versão DEBUG.

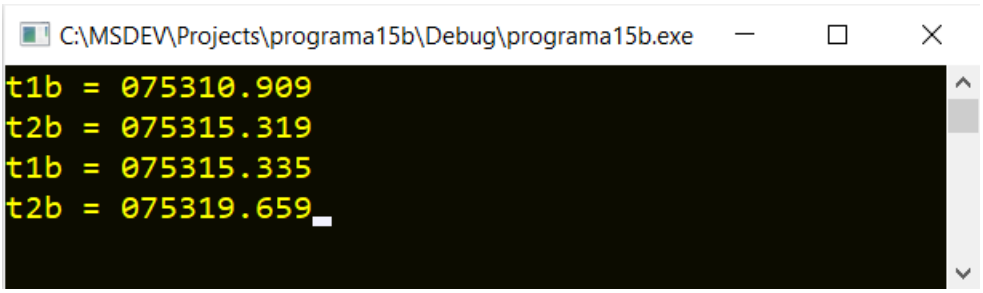

Figura 15.7 Resultado DATE_AND_TIME do programa15b.f90 para 10 repetições, versão DEBUG.

16) Executar o programa através de **Build, Execute com os dados da Figura 15.3**. Os resultados são mostrados nas Figuras 15.8 e 15.9. Novamente, os dois tempos de CPU da Figura 15.8 deveriam ter exatamente o mesmo valor, mas há uma diferença entre eles de 0.006 s. Comparando-se os tempos de processamento, verifica-se que a versão RELEASE é cerca de 32% da versão DEBUG da Figura 15.6.
17) Na Figura 15.9, a primeira diferença entre t2b e t1b é de 0.136 s, e a segunda é de 0.143 s. Estas diferenças devem ser comparadas aos dois valores da Figura 15.8. Portanto, a diferença entre os dois resultados de DATE_AND_TIME é de 0.007 s.
18) **Executar novamente o programa com outros dados e analisar os novos resultados**. Utilizar, por exemplo, PASSOS = 10 milhões, 1 milhão e 1 bilhão; para cada valor de PASSOS, testar com repeticoes = 1 e 10.

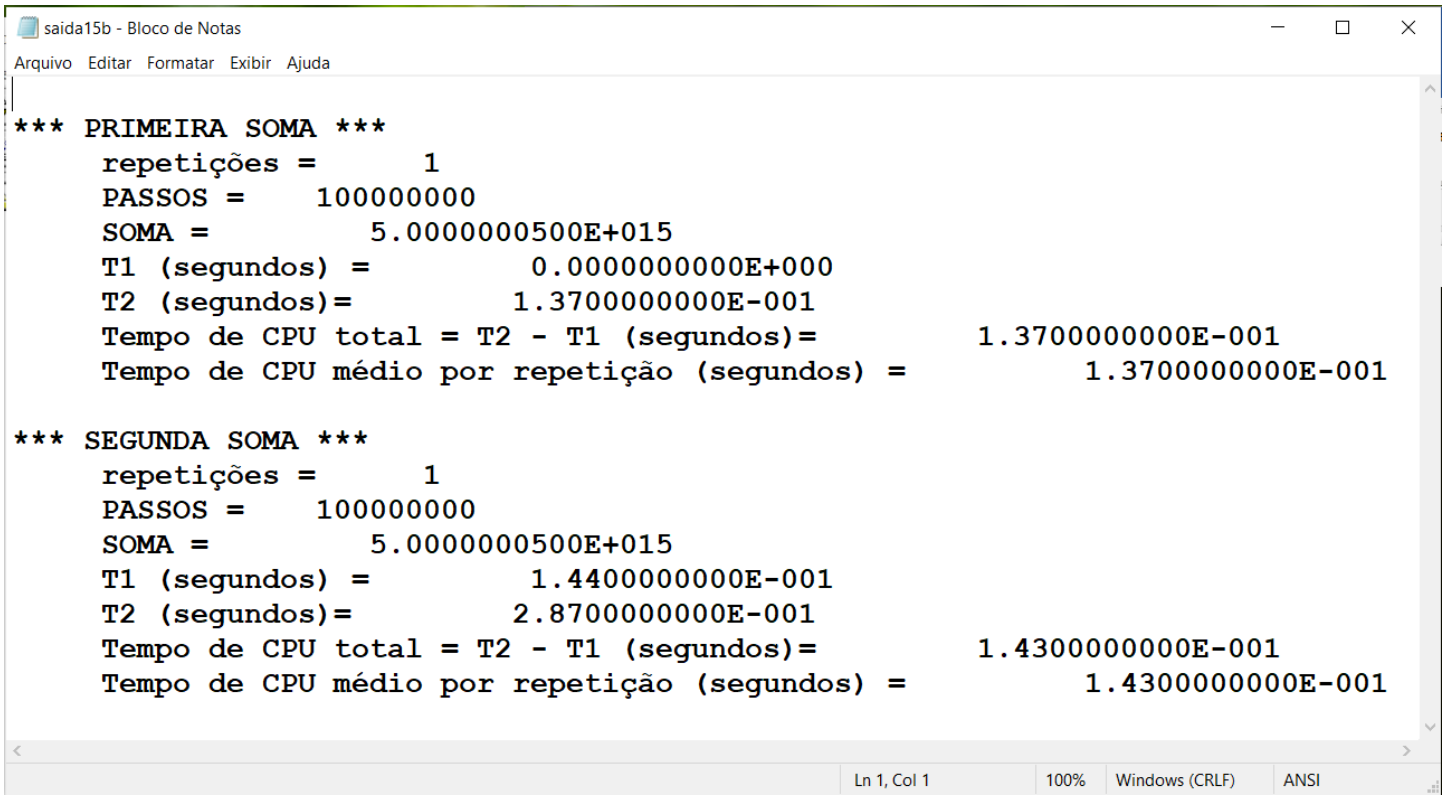

Figura 15.8 Resultado TIMEF do programa15b.f90 para os dados da Figura 15.3, versão RELEASE.

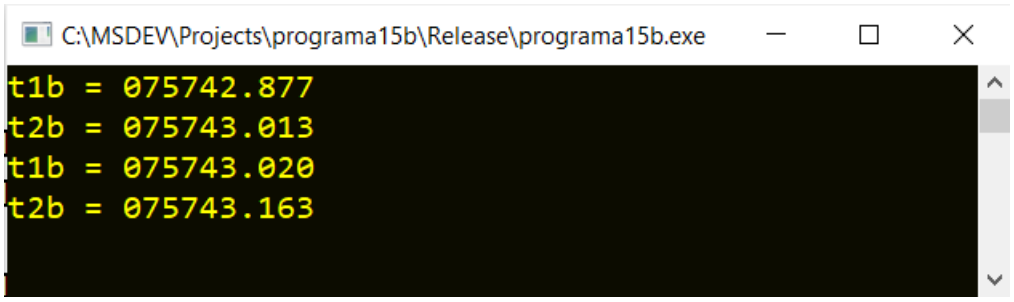

Figura 15.9 Resultado DATE_AND_TIME do programa15b.f90 para os dados da Figura 15.3, versão RELEASE.

15.3 programa15c.f90

1) Objetivo do programa: utilizar comandos do DOS durante a execução do programa.
2) No Fortran, **criar um projeto** com o nome **programa15c**
3) No Fortran, **criar e inserir** no projeto o programa-fonte **programa15c.f90**
4) No Fortran, **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 = ", A, &
          2/, "SAIDA      = ", A  )

CLOSE(UNIT)

! edição de comandos no arquivo EXECUTA.BAT

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

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

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

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

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

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

CLOSE (UNIT)

! fim

DOS = SYSTEM ( "EXECUTA.BAT" )

DOS = SYSTEM( "Notepad C:\Windows\Temp\FORTRAN\NOVO.TXT" )

END
```

5) Comentários sobre o programa:

(a) O aplicativo CMD.EXE é um interpretador de comandos do Windows, que emula o antigo sistema operacional DOS, precursor do Windows. Para ativá-lo, primeiro pesquisar no Windows por cmd.exe. Em seguida, executá-lo.

(b) Este aplicativo pode ser útil, por exemplo, na execução automática de outros aplicativos.

(c) Usar o comando Help para ver a lista de comandos deste aplicativo. Os comandos podem ser digitados em letras maiúsculas ou minúsculas.

(d) O comando do DOS chamado MKDIR é usado para criar uma nova pasta. Ele é usado na forma:

MKDIR PASTA

onde PASTA é o nome da pasta a ser criada, incluindo o caminho completo desde a raiz do HD (hard disk). Se o caminho não é especificado, a pasta é criada dentro da pasta na qual o comando é executado.

(e) O comando do DOS chamado COPY é usado para fazer cópia de um arquivo. 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 a pasta de cada arquivo, incluindo o caminho completo desde a raiz do HD (hard disk). Se as pastas e caminhos não são especificados, ARQ1 deve existir na pasta na qual o comando é executado, e ARQ2 é gerado na mesma pasta.

(f) 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 a sua pasta, incluindo o caminho completo desde a raiz do HD (hard disk). Se a pasta e caminho não são especificados, ARQ deve existir na pasta na qual o comando é executado.

(g) O comando do DOS chamado CD é usado para mudar a execução do programa para outra pasta. Ele é usado na forma: CD PASTA

onde PASTA é o nome da pasta para a qual passa a ser executado o programa, incluindo o caminho completo desde a raiz do HD (hard disk). Subentende-se que a pasta existe dentro da pasta na qual o comando é executado.

(h) 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 (e), acima.

(i) Ao se executar um arquivo com extensão ".BAT", são executados todos os comandos DOS dentro deste arquivo, linha por linha, de cima para baixo.

(j) Exemplos de aplicação dos comandos acima são apresentados no programa15c.f90.

(k) Existem diversos outros comandos do DOS que podem ser empregados em função do objetivo desejado; por exemplo: DIR, CLS, DATE, TIME, RMDIR, TYPE, HELP, EXIT, FC, GOTO, FIND, FOR, IF, PAUSE, MOVE, PRINT, SHUTDOWN, START, TITLE, VER e VOL.

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 15.10.**

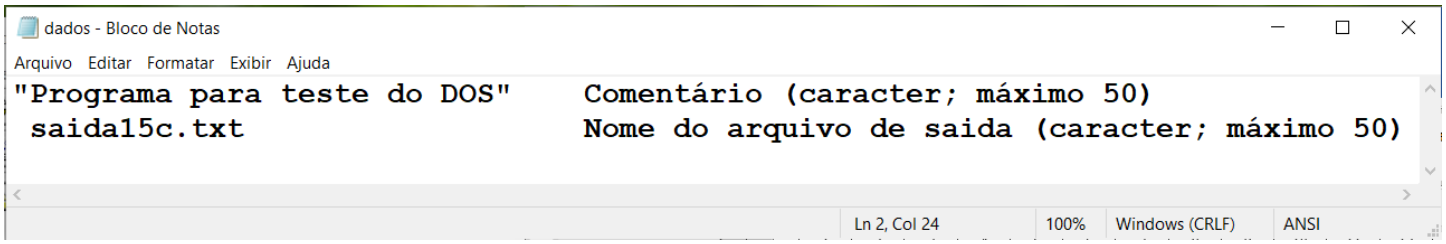

Figura 15.10 Exemplo de arquivo de dados para o programa15c.f90.

9) Algoritmo do programa:
   (a) ocorre a abertura do arquivo "DADOS.TXT"
   (b) são lidos os dois dados
   (c) cria-se 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 do 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:\Windows\Temp\FORTRAN\"
10) Executar o programa através de **Build, Execute**. **Analisar os resultados.** A Figura 15.11 mostra o conteúdo do **arquivo "EXECUTA.BAT"**, gerado pelo programa15c.f90; **verificar sua existência na pasta do projeto**. A Figura 15.12 mostra os comandos que foram executados no DOS, como resultado da execução do arquivo "EXECUTA.BAT". A Figura 15.13 mostra o conteúdo do **arquivo** de resultado do programa15c.f90; deve-se notar que seu nome é **"NOVO.TXT"** e que ele se localiza na pasta "C:\Windows\Temp\FORTRAN\"; **verificar sua existência**; além disso, o arquivo de saída foi eliminado da pasta do projeto.

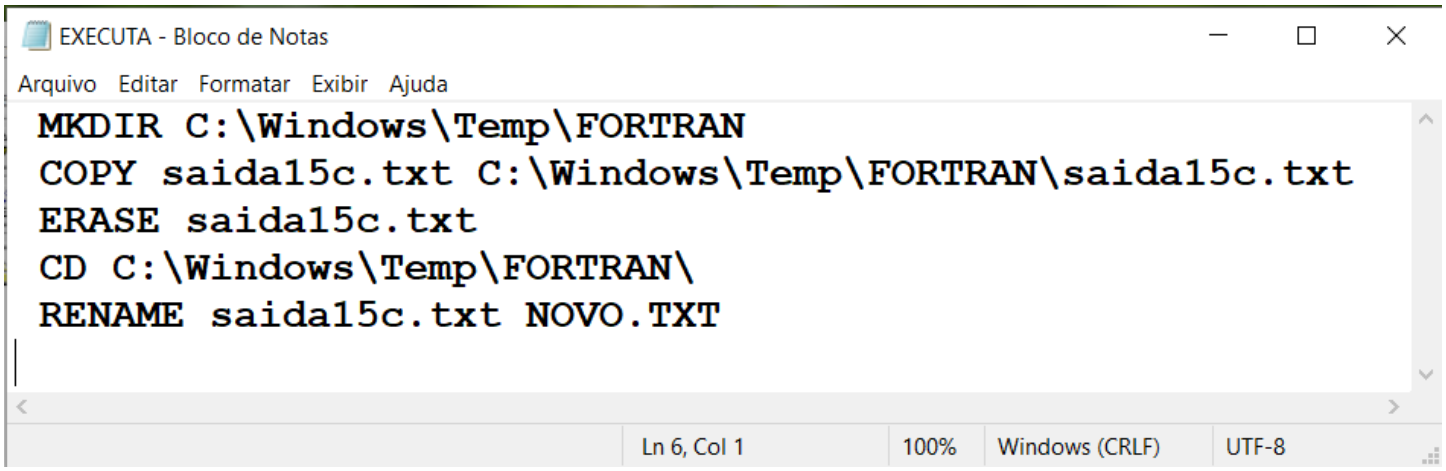

Figura 15.11 Conteúdo do arquivo “EXECUTA.BAT” gerado pelo programa15c.f90.

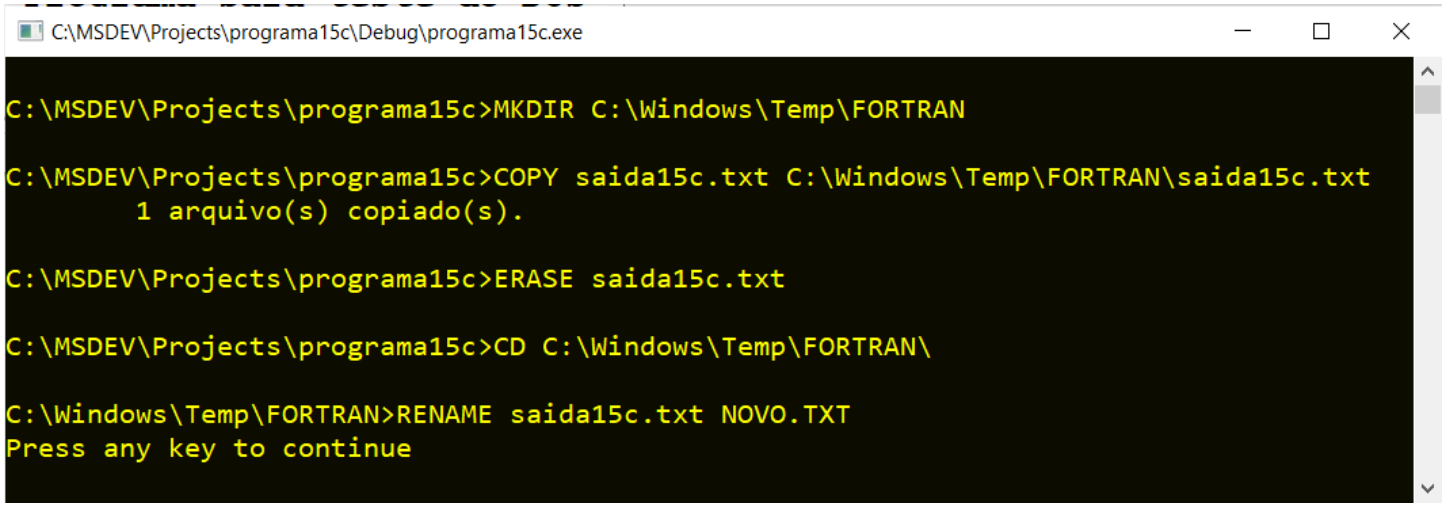

Figura 15.12 Janela DOS resultante da execução do programa15c.f90.

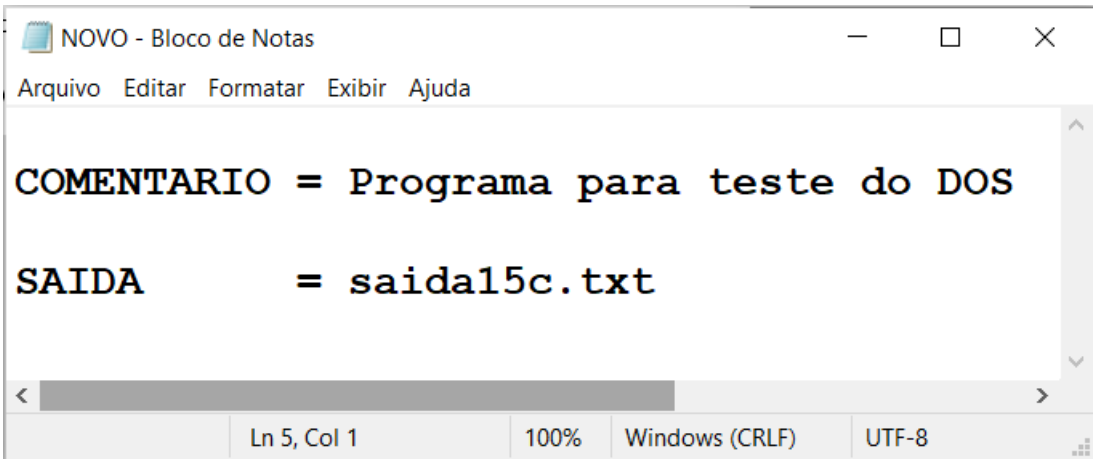

Figura 15.13 Resultado da execução do programa15c.f90.

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 na mesma linha do programa;

(d) inicializar duas constantes do tipo inteiro na mesma linha do programa; e

(e) escrever em 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, as instruções:

```
WRITE(*,*) "Espere alguns segundos e pressione a tecla ENTER"
READ(*,*)
```

Analisar o efeito da instrução READ vazia sobre o tempo de CPU.

Exercício 15.6

Adaptar o programa15c.f90, Tabela 15.3, visando generalizar o nome da pasta

```
"C:\Windows\Temp\FORTRAN\"
```

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