Capítulo 9. CONJUNTOS/VETORES

16 Abr 2025

OBJETIVOS DO CAPÍTULO

- Conceitos de: conjunto/vetor, elemento, alocação dinâmica de memória
- Comandos novos do FORTRAN: DIMENSION, ALLOCATABLE, ALLOCATE, MINVAL, MAXVAL, SUM

9.1 programa9a.f90

1) No Fortran, **criar um projeto** com o nome **programa09**
2) No Fortran, **criar e inserir** no projeto o programa-fonte **programa9a.f90**
3) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 9.1**.
4) Objetivos do programa:
   a) Utilizar três novos comandos do FORTRAN: DIMENSION, ALLOCATABLE e ALLOCATE
   b) Determinar os valores mínimo, máximo e média de um conjunto de variáveis do tipo real
5) Comentários sobre o programa:
   a) Todas as variáveis do tipo inteiro, real e caracter que foram empregadas até o capítulo 8 podem ser denominadas de variáveis simples ou estáticas. Além delas, existem as chamadas variáveis compostas ou do tipo conjunto ou vetores, que são o tema deste capítulo 9.
   b) Neste programa são usados três novos comandos do FORTRAN: DIMENSION, ALLOCATABLE e ALLOCATE, que são empregados com variáveis do tipo conjunto. A sintaxe deles, isto é, a forma de utilizá-los é mostrada na Tabela 9.2.
   c) Em FORTRAN, conjunto é uma coleção de variáveis do mesmo tipo que são agrupadas em uma única variável mas com índice. Cada variável que compõe o conjunto é denominada de elemento ou componente. Uma variável do tipo conjunto também é chamada de vetor ou variável subscrita. As variáveis do tipo conjunto podem ser compostas por variáveis do tipo inteiro, real ou caracter. Por exemplo, as diversas notas de uma prova podem ser agrupadas em um único conjunto ou variável (NOTAS) para representá-las, conforme é mostrado na Tabela 9.1.
   d) O comando DIMENSION(:) é usado para definir a dimensão de uma variável do tipo conjunto. Os dois pontos indicam que a dimensão é variável e será especificada no programa.
   e) O comando ALLOCATABLE é usado para definir uma variável do tipo alocável, isto é, uma variável do tipo conjunto cuja quantidade de elementos que a compõe é definida posteriormente dentro do programa em função de alguma outra variável.
   f) O comando ALLOCATE é usado para definir quantos elementos compõem uma variável do tipo conjunto e reservar o espaço de memória correspondente no computador. Primeiro é necessário definir a quantidade

de elementos de um conjunto, através do comando ALLOCATE para, depois, atribuir valores aos seus componentes.

Tabela 9.1 Programa9a.f90.

```
INTEGER NOTA, QUANTIDADE_NOTAS
REAL, ALLOCATABLE, DIMENSION(:) :: NOTAS
REAL MINIMO, MAXIMO, SOMA, MEDIA

WRITE(*,*) "Entre com a quantidade de notas (inteiro)"
READ(*,*) QUANTIDADE_NOTAS

ALLOCATE ( NOTAS (QUANTIDADE_NOTAS) )

WRITE(*,*) "Entre com as notas (reais)"

DO NOTA = 1, QUANTIDADE_NOTAS
   READ(*,*) NOTAS(NOTA)
END DO

MINIMO = NOTAS(1)
MAXIMO = NOTAS(1)
SOMA   = NOTAS(1)

DO NOTA = 2, QUANTIDADE_NOTAS
   IF ( NOTAS(NOTA) < MINIMO ) MINIMO = NOTAS(NOTA)
   IF ( NOTAS(NOTA) > MAXIMO ) MAXIMO = NOTAS(NOTA)
   SOMA = SOMA + NOTAS(NOTA)
END DO

MEDIA = SOMA / QUANTIDADE_NOTAS

WRITE(*,*) "Nota minima = ", MINIMO
WRITE(*,*) "Nota maxima = ", MAXIMO
WRITE(*,*) "Nota media  = ", MEDIA

END
```

g) Na linha `REAL, ALLOCATABLE, DIMENSION(:) :: NOTAS` do programa, define-se a variável chamada NOTAS com as seguintes características: do tipo conjunto, devido ao comando DIMENSION(:); do tipo alocável, devido ao comando ALLOCATABLE; e do tipo real, ou seja, cada elemento da variável NOTAS poderá conter um número real, devido ao comando REAL. O duplo dois pontos que aparece nesta linha deve ser usado quando existe mais de uma definição para uma variável, onde cada definição deve ser separada por vírgula.

h) Na linha `ALLOCATE ( NOTAS (QUANTIDADE_NOTAS) )` do programa, utilizando-se o comando ALLOCATE e a variável QUANTIDADE_NOTAS, que é um dado do programa, define-se quantos elementos compõem a variável NOTAS e reserva-se o espaço de memória correspondente.

Tabela 9.2 Sintaxe de comandos para variáveis do tipo conjunto.

```
PARA DEFINIR O TIPO DE VARIAVÉL:
      REAL, ALLOCATABLE, DIMENSION(:) :: A, B
      INTEGER, ALLOCATABLE, DIMENSION(:) :: C, D
      CHARACTER(X), ALLOCATABLE, DIMENSION(:) :: E, F

PARA ALOCAR A MEMÓRIA DOS CONJUNTOS:
      ALLOCATE ( A(N1), B(N2), C(N3), D(N4), E(N5), F(N6))

onde X é um valor inteiro que define o número de caracteres; A, B, C, D, E e F são
conjuntos; e N1, N2, N3, N4, N5 e N6 são variáveis do tipo inteiro que definem a
quantidade de elementos de cada conjunto e seus valores devem ter sido definidos
previamente.
Os nomes de todas estas variáveis podem ser quaisquer.
```

i) Cada elemento de uma variável do tipo conjunto é referenciado por um número inteiro, chamado índice, que corresponde à ordem dele dentro do conjunto. Este índice deve ficar entre parênteses após o nome da variável do tipo conjunto. O índice ou o número do elemento pode ser representado por uma variável, por exemplo, na linha `READ(*,*) NOTAS(NOTA)` do programa, que está dentro de um ciclo. Neste ciclo, a variável NOTA corresponde a cada elemento da variável NOTAS.

j) Na linha `MINIMO = NOTAS(1)` do programa, está sendo inicializado o valor da variável MINIMO atribuindo a ela o valor do primeiro elemento da variável NOTAS. O mesmo ocorre nas duas linhas seguintes com as variáveis MAXIMO e SOMA. Inicializar o valor de uma variável significa estimar ou chutar seu valor com o objetivo de iniciar os cálculos; o valor inicial pode ser qualquer; se o valor não é definido ou inicializado, geralmente o programa assume o valor zero, o que pode gerar resultados incorretos, dependendo do cálculo a ser realizado.

k) No ciclo definido pela linha `DO NOTA = 2, QUANTIDADE_NOTAS` do programa, com o emprego de comandos IF, são determinados os valores MINIMO e MAXIMO dos elementos da variável NOTAS, assim como a SOMA dos valores de todos os elementos.

6) Algoritmo do programa:

a) Definir os tipos de todas as variáveis

b) Ler a quantidade de elementos da variável NOTAS, que é do tipo conjunto ou vetor

c) Alocar a memória para a variável NOTAS

d) Ler os valores de todos os elementos da variável NOTAS

e) Determinar os valores mínimo e máximo e a média da variável NOTAS

f) Escrever e identificar os valores mínimo e máximo e a média da variável NOTAS

7) Executar **Build, Compile** para compilar o programa.

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

9) Ao se executar o programa, através de **Build, Execute**, surge uma janela, mostrada na Figura 9.1, dentro da qual tem-se:

a) Na primeira linha, o comentário "Entre com a quantidade de notas (inteiro)", resultado do comando `WRITE(*,*) "Entre com a quantidade de notas (inteiro)"` do programa.

b) Na segunda linha, o programa pára e fica aguardando até que seja fornecido o valor da variável QUANTIDADE_NOTAS, resultado do comando `READ(*,*) QUANTIDADE_NOTAS` do programa. Para que o programa continue a sua execução é necessário **digitar 5**, por exemplo, e, em seguida, **clicar na tecla Enter**.

c) Na terceira linha, o comentário "Entre com as notas (reais)", resultado do comando `WRITE(*,*) "Entre com as notas (reais)"` do programa.

d) Na quarta linha, o programa pára e fica aguardando até que sejam fornecidos os valores de todos os elementos da variável NOTAS, resultado do comando `READ(*,*) NOTAS(NOTA)` do programa, que está dentro de um ciclo que começa no primeiro elemento e vai até o último do conjunto. Deve-se perceber que o comando READ é usado para ler o valor de apenas um elemento a cada vez; assim é necessário **digitar cada nota e**, em seguida, **clicar na tecla Enter** antes de se digitar uma nova nota. Usar, por exemplo, os valores 7, 8.5, 5.7, 4.9 e 9.

e) Em seguida são apresentados os resultados correspondentes à execução do programa.

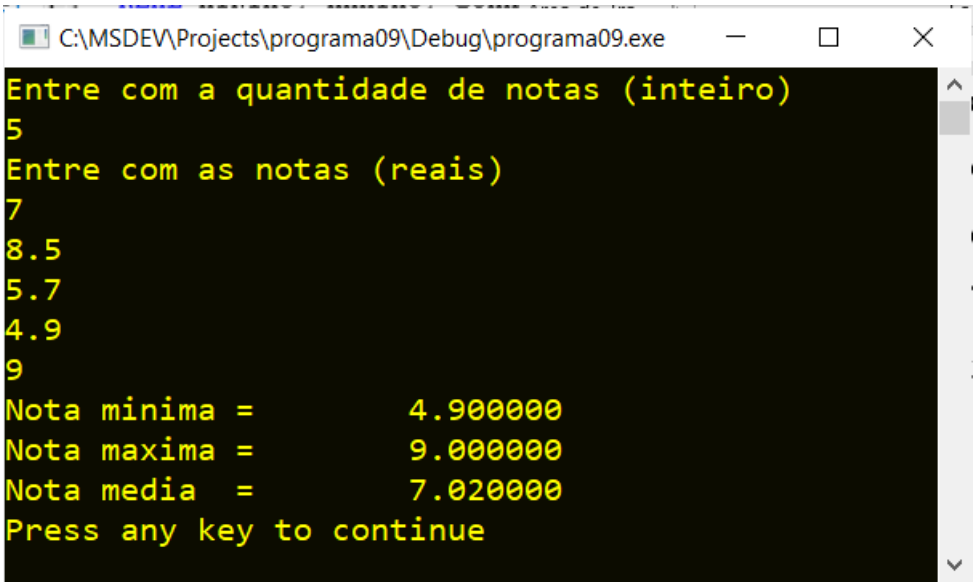

Figura 9.1 Resultado do programa9a.f90.

10) Até entender, **analisar** os resultados do programa9a.f90, mostrados na Figura 9.1, considerando cada linha do programa-fonte, as explicações descritas no item 5, acima, e a média obtida com o uso de uma calculadora.
11) **Executar** novamente o programa com outros valores. Até entender, **analisar** os novos resultados do programa9a.f90 considerando cada linha do programa-fonte, as explicações descritas no item 5, acima, e a média obtida com o uso de uma calculadora.

9.2 programa9b.f90

1) No Fortran, **criar e inserir** no projeto o programa-fonte **programa9b.f90**
2) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 9.3**.

Tabela 9.3 Programa9b.f90.

```
INTEGER NOTA, QUANTIDADE_NOTAS
REAL, ALLOCATABLE, DIMENSION(:) :: NOTAS
REAL MINIMO, MAXIMO, SOMA, MEDIA

WRITE(*,*) "Entre com a quantidade de notas"
READ(*,*) QUANTIDADE_NOTAS

ALLOCATE ( NOTAS (QUANTIDADE_NOTAS) )

WRITE(*,*) "Entre com as notas"

DO NOTA = 1, QUANTIDADE_NOTAS
   READ(*,*) NOTAS(NOTA)
END DO

MINIMO = MINVAL(NOTAS)
MAXIMO = MAXVAL(NOTAS)
SOMA   = SUM(NOTAS)

MEDIA = SOMA / QUANTIDADE_NOTAS

WRITE(*,*) "Nota minima = ", MINIMO
WRITE(*,*) "Nota maxima = ", MAXIMO
WRITE(*,*) "Nota media  = ", MEDIA

END
```

3) Objetivos do programa:
   a) Utilizar três novas funções matemáticas intrínsecas do FORTRAN: MINVAL, MAXVAL e SUM, que são utilizadas com variáveis do tipo conjunto
   b) Determinar os valores mínimo, máximo e a média de um conjunto de variáveis do tipo real
4) Comentários sobre o programa:
   a) Neste programa emprega-se exatamente o mesmo algoritmo do programa anterior, isto é, os dois programas fazem exatamente o mesmo. A diferença é que neste programa os valores mínimo, máximo e a média do conjunto de variáveis do tipo real são obtidos através de três novas funções matemáticas intrínsecas do FORTRAN: MINVAL, MAXVAL e SUM, que são utilizadas com variáveis do tipo conjunto. A sintaxe delas, isto é, a forma de utilizá-las é exemplificada na Tabela 9.3. Com isso, o que foi realizado com oito linhas de programa-fonte no programa9a.f90, empregando um ciclo e comandos IF, que são as seguintes:

```
MINIMO = NOTAS(1)
MAXIMO = NOTAS(1)
SOMA   = NOTAS(1)
DO NOTA = 2, QUANTIDADE_NOTAS
   IF ( NOTAS(NOTA) < MINIMO ) MINIMO = NOTAS(NOTA)
   IF ( NOTAS(NOTA) > MAXIMO ) MAXIMO = NOTAS(NOTA)
   SOMA = SOMA + NOTAS(NOTA)
END DO
```

   foram reduzidas a três linhas no programa9b.f90, que são as seguintes:

```
MINIMO = MINVAL(NOTAS)
MAXIMO = MAXVAL(NOTAS)
SOMA   = SUM(NOTAS)
```

   b) O comando B = MINVAL(A) determina o valor mínimo entre todos os elementos da variável do tipo conjunto A e atribui o resultado à variável simples B.
   c) O comando B = MAXVAL(A) determina o valor máximo entre todos os elementos da variável do tipo conjunto A e atribui o resultado à variável simples B.
   d) O comando B = SUM(A) calcula a soma dos valores de todos os elementos da variável do tipo conjunto A e atribui o resultado à variável simples B.
   e) Outras funções intrínsecas para variáveis do tipo conjunto podem ser vistas no **?InfoView, Reference, Procedures, Array Procedures**.
5) Executar **Build, Compile** para compilar o programa.
6) Gerar o programa-executável fazendo **Build, Build**.
7) Ao se executar o programa, através de **Build, Execute**, ocorre o mesmo já explicado no item 9 da seção 9.1 deste capítulo, e o resultado obtido é o mesmo já mostrado na Figura 9.1.

9.3 programa9c.f90

1) No Fortran, **criar e inserir** no projeto o programa-fonte **programa9c.f90**
2) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 9.4**.
3) Objetivos do programa:
   a) Mostrar como realizar operações com todos os elementos de um conjunto com um único comando
   b) Mostrar o uso de funções matemáticas intrínsecas do FORTRAN com variáveis do tipo conjunto
   c) Mostrar duas formas de escrever o conteúdo de variáveis do tipo conjunto

Tabela 9.4 Programa9c.f90.

```
INTEGER ELEMENTO, QUANTIDADE_ELEMENTOS
REAL, ALLOCATABLE, DIMENSION(:) :: CONJUNTO_A, CONJUNTO_B, CONJUNTO_C

WRITE(*,*) "Entre com a quantidade de elementos do CONJUNTO_A"
READ(*,*) QUANTIDADE_ELEMENTOS

ALLOCATE ( CONJUNTO_A(QUANTIDADE_ELEMENTOS), &
           CONJUNTO_B(QUANTIDADE_ELEMENTOS), &
           CONJUNTO_C(QUANTIDADE_ELEMENTOS) )

WRITE(*,*) "Entre com os valores do CONJUNTO_A"

DO ELEMENTO = 1, QUANTIDADE_ELEMENTOS
   READ(*,*) CONJUNTO_A(ELEMENTO)
END DO

CONJUNTO_B = CONJUNTO_A + 10

CONJUNTO_C = EXP(CONJUNTO_A)

WRITE(*,*) "CONJUNTO_B = ", CONJUNTO_B

DO ELEMENTO = 1, QUANTIDADE_ELEMENTOS
   WRITE(*,20) ELEMENTO, CONJUNTO_C(ELEMENTO)
   20 FORMAT(10X, "CONJUNTO_C(", I2, ") = ", 1PE15.3 )
END DO

END
```

4) Comentários sobre o programa:
   a) Na linha `REAL, ALLOCATABLE, DIMENSION(:) :: CONJUNTO_A, CONJUNTO_B, CONJUNTO_C` do programa, estão sendo definidas três variáveis (CONJUNTO_A, CONJUNTO_B e CONJUNTO_C) com

as seguintes características: do tipo conjunto, devido ao comando DIMENSION(:); do tipo alocável, devido ao comando ALLOCATABLE; e do tipo real, ou seja, cada elemento das três variáveis poderá conter um número real, devido ao comando REAL.

b) Na linha `CONJUNTO_B = CONJUNTO_A + 10` do programa, soma-se o valor 10 ao valor de cada elemento da variável CONJUNTO_A e atribui-se o resultado ao respectivo elemento da variável CONJUNTO_B. Portanto, em um único comando estão sendo feitos cálculos com todos os elementos de um conjunto. Para que isso seja possível, as duas variáveis do tipo conjunto têm que ter exatamente o mesmo número de elementos. Esta linha do programa poderia ser substituída de forma equivalente ao seguinte:

```
DO ELEMENTO = 1, QUANTIDADE_ELEMENTOS
   CONJUNTO_B(ELEMENTO) = CONJUNTO_A(ELEMENTO) + 10
END DO
```

c) Na linha `CONJUNTO_C = EXP(CONJUNTO_A)` do programa, calcula-se a exponencial do valor de cada elemento da variável CONJUNTO_A e atribui-se o resultado ao respectivo elemento da variável CONJUNTO_C. Assim, em um único comando estão sendo feitos cálculos com todos os elementos de um conjunto envolvendo uma função matemática intrínseca do FORTRAN, no caso a função exponencial (EXP). Mas também podem ser usadas outras funções como aquelas apresentadas nas Tabelas 5.5 e 5.6. Esta linha do programa também poderia ser substituída de forma equivalente ao seguinte:

```
DO ELEMENTO = 1, QUANTIDADE_ELEMENTOS
   CONJUNTO_C(ELEMENTO) = EXP(CONJUNTO_A(ELEMENTO))
END DO
```

d) Há duas formas de escrever os valores dos elementos de uma variável do tipo conjunto. A primeira é agir da mesma forma que no caso de uma variável simples. Por exemplo, como resultado da linha `WRITE(*,*) "CONJUNTO_B = ", CONJUNTO_B` do programa, serão escritos os valores de todos os elementos da variável CONJUNTO_B, na sequência do primeiro ao último elemento, no formato default do FORTRAN. A segunda forma é escrever o valor de cada elemento através de um ciclo, com ou sem formato de edição. Um exemplo disso é apresentado no último ciclo do programa9c.f90.

5) Algoritmo do programa:
   a) Definir os tipos de todas as variáveis
   b) Ler a quantidade de elementos das variáveis do tipo conjunto
   c) Alocar a memória para as três variáveis do tipo conjunto chamadas CONJUNTO_A, CONJUNTO_B e CONJUNTO_C
   d) Ler os valores de todos os elementos da variável CONJUNTO_A
   e) Somar o valor 10 ao valor de cada elemento da variável CONJUNTO_A e atribuir o resultado a cada elemento da variável CONJUNTO_B
   f) calcular a exponencial do valor de cada elemento da variável CONJUNTO_A e atribuir o resultado a cada elemento da variável CONJUNTO_C
   g) Escrever os valores de todos os elementos das variáveis CONJUNTO_B e CONJUNTO_C
6) Executar **Build, Compile** para compilar o programa.

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

8) Ao se executar o programa, através de **Build, Execute**, surge uma janela, mostrada na Figura 9.2, dentro da qual tem-se:

   a) Na primeira linha, o comentário “Entre com a quantidade de elementos do CONJUNTO_A”, resultado do comando `WRITE(*,*) "Entre com a quantidade de elementos do CONJUNTO_A"` do programa.

   b) Na segunda linha, o programa pára e fica aguardando até que seja fornecido o valor da variável QUANTIDADE_ELEMENTOS, resultado do comando `READ(*,*) QUANTIDADE_ELEMENTOS` do programa. Para que o programa continue a sua execução é necessário **digitar 6**, por exemplo, e, em seguida, **clicar na tecla Enter**.

   c) Na terceira linha, o comentário “Entre com os valores do CONJUNTO_A”, resultado do comando `WRITE(*,*) "Entre com os valores do CONJUNTO_A"` do programa.

   d) Na quarta linha, o programa pára e fica aguardando até que sejam fornecidos os valores de todos os elementos da variável CONJUNTO_A, resultado do comando `READ(*,*) CONJUNTO_A(ELEMENTO)` do programa, que está dentro de um ciclo que começa no primeiro elemento e vai até o último do conjunto. Deve-se perceber que o comando READ é usado para ler o valor de apenas um elemento a cada vez; assim é necessário **digitar cada valor e**, em seguida, **clicar na tecla Enter** antes de se digitar um novo valor. Usar, por exemplo, os valores 1, 2, 3, 4, 5 e 6.

   e) Em seguida são apresentados os resultados correspondentes à execução do programa.

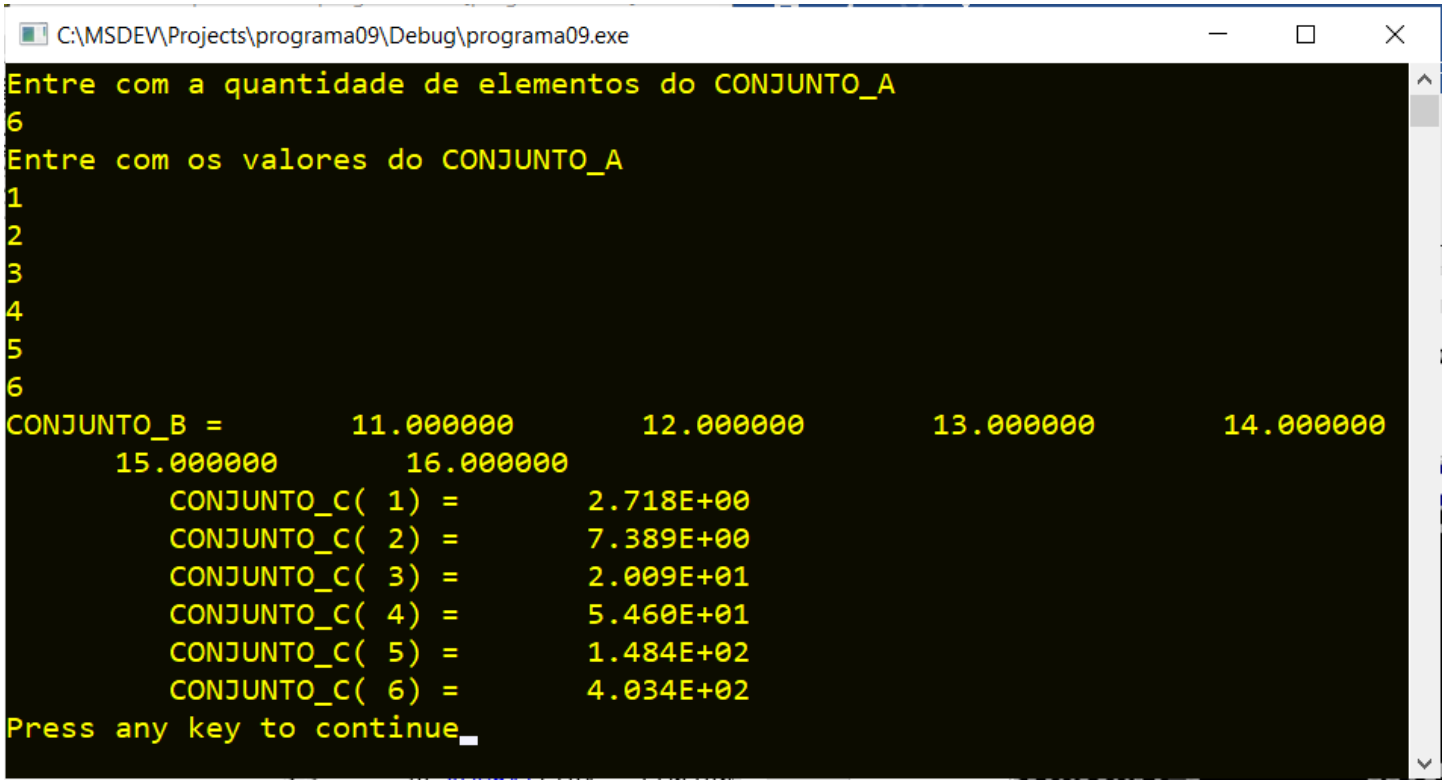

Figura 9.2 Resultado do programa9c.f90.

9) Até entender, **analisar** os resultados do programa9c.f90, mostrados na Figura 9.2, considerando cada linha do programa-fonte e as explicações descritas acima.

10) **Executar** novamente o programa com outros dados. Até entender, **analisar** os novos resultados considerando cada linha do programa-fonte e as explicações descritas acima.

9.4 programa9d.f90

1) No Fortran, **criar e inserir** no projeto o programa-fonte **programa9d.f90**
2) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 9.5**.
3) Objetivo do programa: aplicar os conceitos já vistos nas seções anteriores deste capítulo ao caso de uma progressão aritmética (P.A.) onde os valores dos elementos da variável do tipo conjunto são inteiros.

Tabela 9.5 Programa9d.f90.

```
INTEGER TERMO, N, SN, D
INTEGER, ALLOCATABLE, DIMENSION(:) :: A

WRITE(*,*) "Todas as variaveis sao do tipo inteiro"

WRITE(*,*) "Entre com o numero de termos da P.A."
READ(*,*) N

ALLOCATE ( A(N) )

WRITE(*,*) "Entre com o primeiro termo da P.A."
READ(*,*) A(1)

WRITE(*,*) "Entre com a diferenca entre dois termos subsequentes"
READ(*,*) D

DO TERMO = 2, N
   A(TERMO) = A(1) + (TERMO - 1) * D
END DO

SN = N * ( A(1) + A(N) ) / 2

DO TERMO = 1, N
   WRITE(*,10) TERMO, A(TERMO)
   10 FORMAT(3X, "A(", I3, ") = ", I8 )
END DO

WRITE(*,*) "Soma dos termos = ", SN

END
```

4) Algoritmo do programa:
   a) Definir os tipos de todas as variáveis
   b) Ler o número de termos da progressão aritmética (P.A.), denotado por N
   c) Alocar a memória para a variável do tipo conjunto chamada A
   d) Ler o valor do primeiro termo da P.A., denotado por A(1), e a diferença entre dois termos subsequentes, denotada por D
   e) Calcular os valores dos termos da P.A. e atribuí-los aos elementos 2 a N da variável A
   f) Calcular a soma dos valores de todos os elementos da P.A., denotada por SN
   g) Escrever os valores de todos os elementos da P.A. e a soma deles
5) Executar **Build, Compile** para compilar o programa.
6) Gerar o programa-executável fazendo **Build, Build**.
7) Ao se executar o programa, através de **Build, Execute**, surge uma janela, mostrada na Figura 9.3, dentro da qual são solicitados os dados (**usar, por exemplo, 10, 1 e 2**) e, em seguida, são apresentados os resultados da execução do programa.

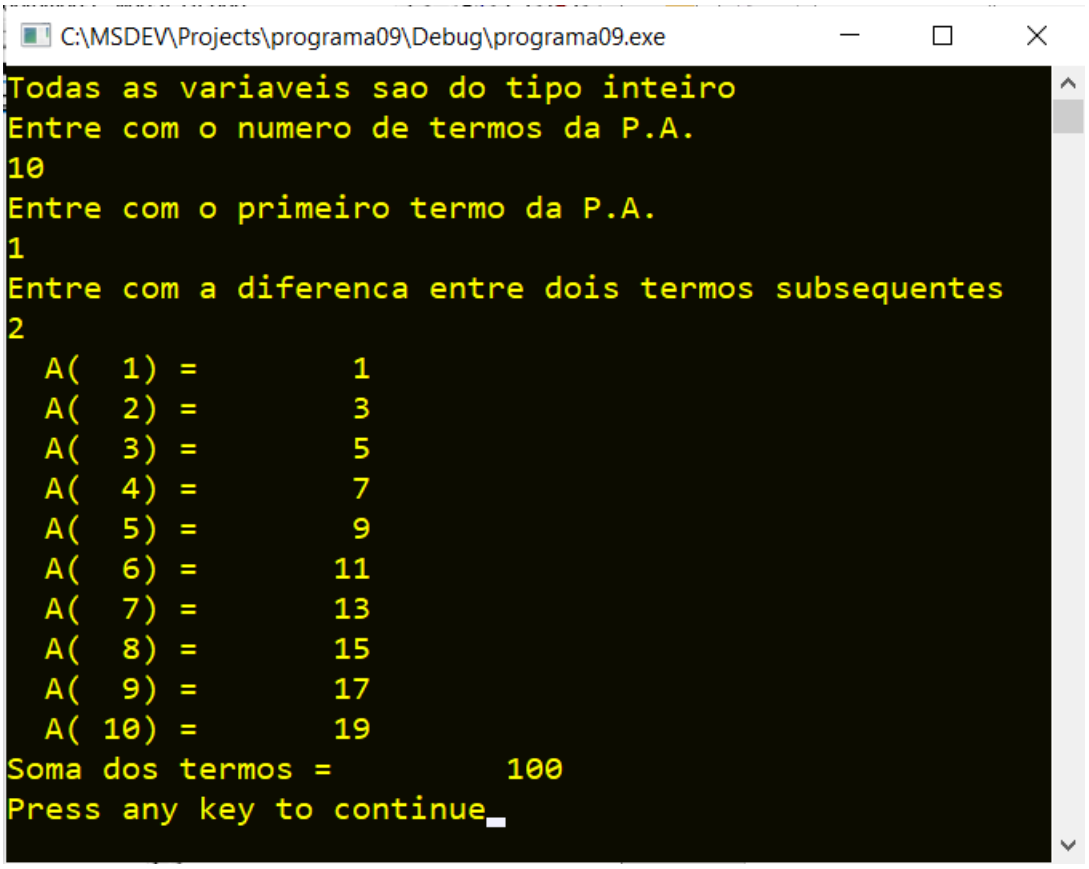

Figura 9.3 Resultado do programa9d.f90.

8) Até entender, **analisar** os resultados do programa9d.f90, mostrados na Figura 9.3, considerando cada linha do programa-fonte.
9) **Executar** novamente o programa com outros dados. **Usar, por exemplo, 30, 1 e 2**. Até entender, **analisar** os novos resultados considerando cada linha do programa-fonte.

10) **Executar** novamente o programa com outros dados. **Usar, por exemplo, 900, 1 e 2**. Até entender, **analisar** os novos resultados considerando cada linha do programa-fonte.

11) **Executar** novamente o programa com outros dados. Até entender, **analisar** os novos resultados considerando cada linha do programa-fonte.

9.5 programa9e.f90

1) No Fortran, **criar e inserir** no projeto o programa-fonte **programa9e.f90**
2) No Fortran, **copiar** exatamente o texto em vermelho mostrado na **Tabela 9.6**.

Tabela 9.6 Programa9e.f90.

```
INTEGER TERMO, N, SN, D
INTEGER, ALLOCATABLE, DIMENSION(:) :: A

WRITE(*,*) "Todas as variaveis sao do tipo inteiro"

WRITE(*,*) "Entre com o numero de termos da P.A."
READ(*,*) N

ALLOCATE ( A(-5:N) )

WRITE(*,*) "Entre com o primeiro termo da P.A."
READ(*,*) A(-5)

WRITE(*,*) "Entre com a diferenca entre dois termos subsequentes"
READ(*,*) D

DO TERMO = -4, N
   A(TERMO) = A(termo-1) + D
END DO

SN = (N+6) * ( A(-5) + A(N) ) / 2

DO TERMO = -5, N
   WRITE(*,10) TERMO, A(TERMO)
   10 FORMAT(3X, "A(", I3, ") = ", I8 )
END DO

WRITE(*,*) "Soma dos termos = ", SN

END
```

3) Objetivo do programa: exemplificar o uso de índices negativos para os elementos de uma variável do tipo conjunto.
4) Algoritmo do programa: é o mesmo da seção 9.4.
5) Executar **Build, Compile** para compilar o programa.
6) Gerar o programa-executável fazendo **Build, Build**.
7) Ao se executar o programa, através de **Build, Execute**, surge uma janela, mostrada na Figura 9.4, dentro da qual são solicitados os dados (**usar, por exemplo, 10, –10 e 2**) e, em seguida, são apresentados os resultados da execução do programa.
8) Até entender, **analisar** os resultados do programa9e.f90, mostrados na Figura 9.4, considerando cada linha do programa-fonte.
9) **Executar** novamente o programa com outros dados. Até entender, **analisar** os novos resultados considerando cada linha do programa-fonte.

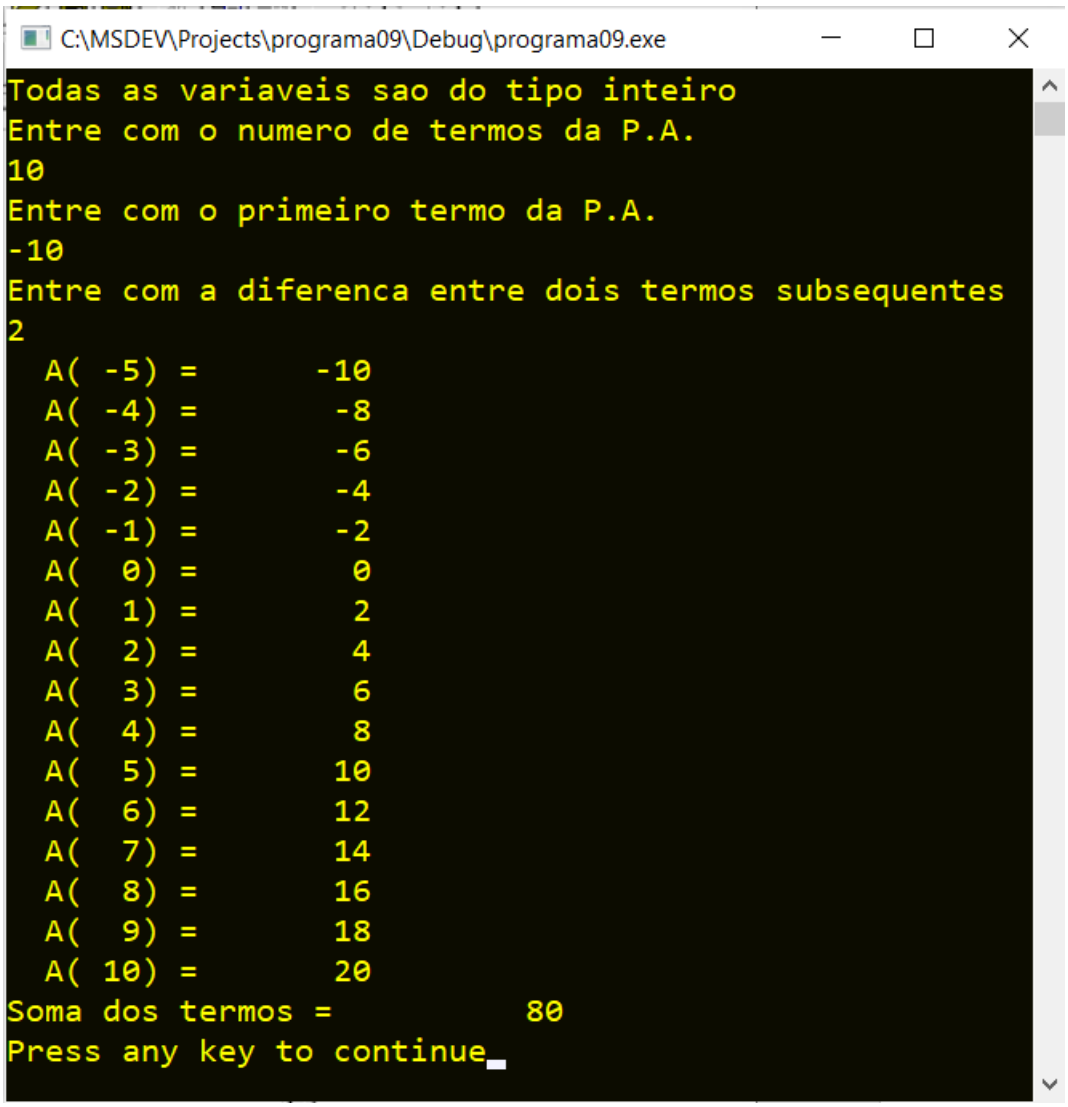

Figura 9.4 Resultado do programa9e.f90.

9.6 EXERCÍCIOS

Exercício 9.1

Adaptar o programa9d.f90 para calcular os valores reais dos termos da progressão aritmética e escrevê-los em arquivo.

Exercício 9.2

Adaptar o programa9d.f90 para calcular os valores reais dos termos de uma progressão geométrica e escrevê-los em arquivo.

Exercício 9.3

Editar um programa-fonte em FORTRAN para executar o seguinte algoritmo (passos):

1) Definir os tipos de todas as variáveis, sendo uma delas do tipo conjunto para caracteres, chamada NOMES
2) Ler a quantidade de elementos da variável NOMES
3) Alocar a memória para a variável NOMES
4) Ler os conteúdos de todos os elementos da variável NOMES
5) Escrever em arquivo o conteúdo de cada elemento da variável NOMES

Exercício 9.4

Adaptar o programa9b.f90 para:

1) Escrever em arquivo os valores das notas e os resultados calculados
2) Calcular e escrever o desvio-padrão (*D*) das notas, definido por

$$D = \sqrt{\frac{\sum_{i=1}^{N}(N_i - N_m)^2}{N-1}}$$

onde N é o número de notas, N_m é a média das notas, e N_i é cada nota

3) Executar o programa com as notas = 7, 8 e 10.
4) Resultados esperados: $N_m = 8.333$ $D = 1.528$

Exercício 9.5

Editar um programa-fonte em FORTRAN para executar o seguinte algoritmo (passos):

1) Ler os valores inteiros de cinco variáveis
2) Ordenar e escrever as cinco variáveis em ordem crescente de valor
3) Ordenar e escrever as cinco variáveis em ordem decrescente de valor

Exercício 9.6

Editar um programa-fonte em FORTRAN para executar o seguinte algoritmo (passos):

1) Ler os valores inteiros de cinco variáveis
2) Determinar e escrever quais variáveis têm valor par
3) Determinar e escrever quais variáveis têm valor ímpar
4) Calcular a soma dos valores pares
5) Calcular a soma dos valores ímpares

Sugestão: usar a função MOD; ver a Tabela 5.5 no Capítulo 5

Exercício 9.7

Adaptar o programa9d.f90 para fazer este exercício.

1) Criar um projeto com o nome **Cap9** e inserir nele o programa-fonte **Cap9.f90**
2) Criar o arquivo de saída **Cap9.TXT** e escrever nele o nome completo do aluno.
3) Com o comando READ, ler os valores que definem uma P.A.: N, A(1) e D; onde N é o número de termos, A(1) é o valor do primeiro termo, e D é a razão da P.A.
4) Determinar qual o número do termo que tem o menor valor positivo e atribuir à variável TERMO
5) Determinar qual o valor do termo que tem o menor valor positivo e atribuir à variável VALOR
6) Escrever no arquivo de saída os resultados de TERMO e VALOR junto com seus nomes.
7) Abrir o arquivo de saída com o aplicativo Bloco de Notas (Notepad).
8) Executar o programa com: N = 50 A(1) = –20 D = 7

   Resultados esperados: TERMO = 4 VALOR = 1

Exercício 9.8

Adaptar o programa9d.f90 para fazer este exercício.

1) Criar um projeto com o nome **Cap9b** e inserir nele o programa-fonte **Cap9b.f90**
2) Criar o arquivo de saída **Cap9b.TXT** e escrever nele o nome completo do aluno.
3) Com o comando READ, ler os valores que definem uma P.A.: N, A(1) e D; onde N é o número de termos, A(1) é o valor do primeiro termo, e D é a razão da P.A.
4) Calcular a soma dos valores positivos dos elementos da P.A. e atribuir à variável Pos
5) Calcular a soma dos valores negativos dos elementos da P.A. e atribuir à variável Neg
6) Escrever no arquivo de saída os resultados de Pos e Neg junto com seus nomes.
7) Abrir o arquivo de saída com o aplicativo Bloco de Notas (Notepad).
8) Executar o programa com: N = 50 A(1) = –20 D = 7

   Resultados esperados: Pos = 7614 Neg = –39