Programação Estruturada

Conceitos Elementares

Prof. Edson Alves

Faculdade UnB Gama

2020

Sumário

Principais características

Fortran

- Fortran (*IBM Mathematical FORmula TRANslation System*) é uma linguagem de programação desenvolvida na década de 1950
- Até os dias atuais é uma das principais (ou a principal) linguagem utilizada em programação científica
- O primeiro compilador foi desenvolvido na IBM, por uma equipe liderada por John W. Backus, nos anos de 1954 a 1957
- O ISO/IEC 1539-1:1997 contém o padrão Fortran 95, um dos mais populares da linguagem
- Fortran apresenta notável performance em computação numérica, o que levou a sua adoção em pesquisas científicas e aplicações computacionalmente intensivas, como meteorologia, física, engenharia, etc

GFortran

- O projeto GNU Fortran (GFortran) consiste em um *front-end* de compilador e bibliotecas de *run-time* para o GCC que dêem suporte à linguagem Fortran
- Ele é totalmente compatível com o padrão Fortran 95 e incluí suporte legado ao formato Fortran 77
- Em distribuições Linux com suporte ao apt, ele pode ser instalado com o comando

```
$ sudo apt-get install gfortran
```

- Para testar a instalação, insira o seguinte comando no terminal:

```
$ f95 -v
```

Hello World!

```
1 ! Implementação do Hello World em Fortran
2 program hello
3
4     write(*,*) 'Hello, World!'
5
6 end program hello
```

Compilação, linkedição e execução

- Para compilar um código Fortran (extensões `.f90`) é preciso invocar o GFortran, utilizando a *flag* `-c`:

```
$ f95 -c hello.f90
```

- No processo de linkedição é preciso indicar, os código-objetos que comporão o executável e, opcionalmente, o nome deste executável (opçaõ `-o`):

```
$ f95 hello.o -o hello
```

- É possível executar ambas etapas em um só comando:

```
$ f95 hello.f90 -o hello
```

- Para rodar o executável criado, basta usar os mesmo mecanismos disponíveis em Linux para invocar um programa como, por exemplo, indicar seu caminho:

```
$ ./hello
```

Variáveis em Fortran

- Em Fortran, as variáveis simbolizam regiões de memória, as quais podem ser lidas ou escritas
- Cada variável é identificada por um **nome**, que deve iniciar com um caractere alfabético e conter apenas caracteres alfanuméricos ou o símbolo '_'
- Em Fortran não há distinção entre caracteres maiúsculos e minúsculos
- Assim como nas linguagens imperativas, uma variável identifica tanto o endereço da região de memória quanto o valor armazenado
- Qual dos dois valores será utilizado depende do contexto (se é um *l-value* ou um *r-value*

Declaração de variáveis e tipos de dados

- Uma variável pode ser declarada em Fortran usando a seguinte syntaxe

```
tipo_do_dado :: nome_da variavel [= valor_inicial]
```

- Os principais tipos de dados em Fortran são: **real**, **integer**, **complex** e **character**
- O valor inicial é opcional
- Strings podem ser declaradas indicando-se o número de caracteres que a compõe

```
character (len = N) :: s    ! string de N caracteres
```

- Para declarar **constantes**, isto é, variáveis com permissão para leitura apenas), é utilizada a palavra-chave **parameter**:

```
complex, parameter :: pi = 3.1415
```

- No caso de constantes, o valor inicial é mandatório
- A expressão **implicit none** determina que todas as variáveis devem ser declaradas antes de seu uso, e é boa prática sempre utilizá-la no início dos programas

Exemplo de declaração e uso de variáveis em Fortran

```
1  ! Computa a área de um círculo de raio r
2  program area
3
4      implicit none
5
6      real, parameter :: pi = 3.141592    ! Declaração de constante
7      real :: A                           ! Declaração de variável real
8      integer :: r                        ! Declaração de variável inteira
9
10     r = 8                ! Define um valor para r por meio de atribuição
11
12     A = pi * r ** 2      ! Área do círculo
13
14     write(*,*) 'Area = ', a
15
16 end program area
```

Operadores aritméticos e funções intrínsecas

- Sendo uma linguagem voltada para computação científica, Fortran tem suporte para uma série de operadores aritméticos
- No caso das expressões com mais de um operador, o operador de menor precedência é computado antes do de menor precedência
- Além disso, há um bom número de funções *intrísecas* da linguagem, disponíveis sem a necessidade de importar arquivos ou bibliotecas externas
- Boa parte destas funções são relacionadas às funções matemáticas e manipulação numérica

Operador	Precedência	Operação
**	1	Expoenciação
*	2	Multiplicação
/	2	Divisão
+	3	Adição
-	3	Subtração

Funções intrísecas úteis

Função	Retorno
`abs(a)`	Valor absoluto de a
`sin(w)`	Seno de `w`
`cos(w)`	Cosseno de `w`
`tan(x)`	Tangente de `w`
`sqrt(x)`	Raiz quadrada de `x`
`conjg(z)`	Conjugado complexo de `z`
`log10(x)`	Logaritmo em base 10 de `x`
`mod(r1, r2)`	Resto da divisão de `r1` por `r2`
`max(r1, r2, ...)`	Maior dentre todos os argumentos
`min(r1, r2, ...)`	Menor dentre todos os argumentos

Legenda: `r`: real ou inteiro, `z`: complexo, `w`: real ou complexo, `x`: real, `a`: qualquer tipo.

Exemplo de uso de funções intrísecas e operadores aritméticos

```
1  ! Computa a forma polar do complexo c
2  program polar
3
4      implicit none
5
6      complex :: c = complex (0.5, sqrt(3.0)/2)
7
8      real :: p, theta         ! Parâmetros da forma polar
9
10     write(*,*) 'c = ', c
11
12     ! Converte c para a forma polar c = p(cos(theta) + isen(theta))
13     p = sqrt(real(c) ** 2 + aimag(c) ** 2)
14     theta = atan(aimag(c)/real(c))
15
16     ! p = 1, theta = pi/3 = 60°
17     write(*,*) 'p = ', p, ' theta = ', theta
18
19 end program polar
```

Operadores lógicos e relacionais

- Fortran também tem suporte para variáveis booleanas, cujos valores possíveis são verdadeiro (`.TRUE.`) e falso (`.FALSE.`)
- As variáveis booleanas são declaradas com o tipo **`logical`**:

```
logical :: T = .true., F = .false.
```

- Variáveis boolenas ou expressões que resultem em valores booleanos podem ser combinadas com os operadores lógicos **e** (`.and.`), **ou** (`.or`) ou **não** (`.not`)
- Os operadores relacionais são apresentados em duas formas
- A primeira delas é a em notação símbolica: `<`, `<=`, `==`, `>=`, `>`, `/=`
- A segunda é por meio de com operadores semelhantes aos operadores lógicos: `.lt.`, `.le.`, `.eq.`, `.ge.`, `.gt.`, `.ne.`

Exemplo de uso de operadores relacionais

```
1  ! Verifica se um competidor pode ou não participar da Maratona
2  ! no ano de 2020
3  program maratona
4
5      implicit none
6
7      integer :: inicio = 2017, nascimento = 1995
8      logical :: primeira_graduacao = .true.
9      logical :: ok
10
11     ok = nascimento >= 1997 .or. (primeira_graduacao .and. inicio > 2015)
12
13     write(*,*) 'Pode participar? ', ok
14
15 end program maratona
```

Saída de dados

- A função **write** permite a escrita de uma lista (`list`) de dados em um fluxo (`stream`), de acordo com a formatação dada em `label`:

```
write(stream, label) list
```

- O fluxo pode ser um número associado a um arquivo, uma variável do tipo **character** ou o símbolo `*`, que indica o valor padrão (em geral, o terminal)
- O rótulo (`label`) é o inteiro identificador do formato, ou `*` para formato livre
- A sintaxe para a declaração de um rótulo é

```
label format (format_descriptors)
```

- Os descritores de formato são uma lista de itens, separados por vírgula, que determinam como a saída deve ser apresentada

Exemplos de descritores de formato

Descritor	Efeito
nIw	Imprime os próximos n inteiros, com tamanho de w caracteres cada
$nFw.d$	Imprime os próximos n números complexos ou reais, em ponto fixo, com w caracteres, e d dígitos na parte decimal
$nEw.d$	Imprime os próximos n números complexos ou reais, em ponto flutuante, com w caracteres, e d dígitos na parte decimal
Aw	Imprime a variável não-numérica A, com w caracteres de tamanho

Referências

1. **BURKARDT**, John. Source Codes in Fortran 90, acesso em 31/01/2020.
2. **PADMAN**, Rachael. Computer Physics: Self-study guide 2 – Programming in Fortran 95, University of Cambridge, Departament of Physics, 2007.
3. GNU Fortran. Welcome to the home of GNU Fortran, acesso em 29/01/2020.
4. **SHALOM**, Elad. *A Review of Programming Paradigms Througout the History – With a Suggestion Toward a Future Approach*, Amazon, 2019.
5. **SHENE**, C. K. LOGICAL Type and Variables, acesso em 31/01/2020.
6. Wikipédia. Fortran, acesso em 29/01/2020.