Linguagem de Programação Fortran

Antônio André Cunha da Silva, Bárbara Elen Silva de Souza

Instituto de Ciências Exatas e Naturais – Universidade Federal do Pará (UFPA)

Belém – PA – Brasil

andyvolhp@gmail.com, bah.ccomp@gmail.com

RESUMO. *A linguagem Fortran foi a primeira linguagem de alto nível a ser largamente utilizada. Foi desenvolvida para lidar com problemas de simulação matemáticas, que envolvem essencialmente problemas da área cientifica. Neste artigo será mostrado, primeiramente, um breve histórico e as principais aplicações da linguagem, então, serão abordadas algumas das suas características, tipos de dados, alocação dinâmica de memória, comandos de controle de fluxo, procedimentos, encapsulamento, uso de bibliotecas e tratamento de exceções. Um exemplo desenvolvido na linguagem será mostrado e em seguida serão dadas informações quanto aos principais ambientes de desenvolvimento e o material de apoio disponível da Internet. Por fim, uma análise crítica da linguagem como um todo.*

ABSTRACT. *Fortran was the first high-level language to be widely used. Was developed to deal with mathematical simulation problems, wich essentially envolve problems of the science. This article will show, first, a brief history and major applications of the language, then, will be addressed some of its features, data types, dynamic memory allocation, flow control commands, procedures, packaging, use of libraries and exception handling. An example developed in the language will be shown and then will be given some information about the major development environments and supporting material available on the Internet. Finally, a critical analysis of language as a whole.*

1. Histórico

Antes da década de 50 os computadores operavam essencialmente problemas matemáticos. Para tratar desses problemas existia uma linguagem extremamente rudimentar, chamada de Assembler. Um programa escrito nessa linguagem era de difícil compreensão, e precisava ser reescrito para cada computador. Dessa forma, havia a necessidade de uma linguagem de fácil entendimento e menos dependente de plataforma. Justamente por essa necessidade, surgiu a primeira linguagem de programação de alto nível, chamada de Formula Translator (Tradutor de Fórmulas), ou simplesmente, FORTRAN.

FORTRAN foi desenvolvido por um grupo de engenheiros da IBM, liderados

pelo cientista da computação, John Backus. A partir deste momento, Fortran passou a ser largamente utilizada dentro de diversas áreas cientificas como: matemática, física, química, engenharia, meteorologia, estatística, astronomia, entre outras.

Devido a sua grande utilização, Fortran sofreu diversas atualizações e revisões. A sua primeira versão padronizada é a FORTRAN 66, depois houve a segunda atualização: FORTRAN 77 na década de 70. Estes dois padrões eram muito rigorosos, possuíam uma série de limitações.

Na década de 90 Fortran recebeu novos padrões, menos rigorosos. Esses novos padrões tornaram possível a alocação dinâmica de memória para vetores, o uso de ponteiros, tipos compostos, alguns conceitos de orientação a objetos, entre outros.

Mais recentemente, foram lançadas duas novas versões, que utilizam conceitos mais aprofundados de orientação a objetos e tratamento de exceções, são elas: FORTRAN 2003 e FORTRAN 2008.

A linguagem Fortran continua a ser utilizada devido à ampla biblioteca matemática disponível, sendo que estas podem também ser exportadas para programas escritos em outras linguagens, como C/C++.

2. Classe de Problemas Indicados

Como já citado anteriormente, Fortran é uma linguagem largamente utilizada na área científica. Possui uma ótima abordagem para operações matemáticas, e serve essencialmente para simulações de experimentos reais nas diversas áreas da ciência. Como lançamentos de foguetes e controle de tráfego aéreo, por exemplo.

A abordagem de Fortran para esses tipos de problema é muito boa, por existir uma ampla biblioteca matemática que trata de Análise Numérica, Álgebra Linear, Raízes de Polinômios, Equações Transcendentais, Interpolação de dados, Técnica de Mínimos Quadrados, Integração Numérica, Equações Diferenciais Ordinárias, entre outras.

3. Características Gerais

Existem dois formatos básicos em Fortran, o formato fixo e o formato livre. O primeiro diz respeito às versões mais antigas (FORTRAN 66 e 77), no qual havia rigorosas regras quanto à indentação do código. Por exemplo, as seis primeiras colunas de cada linha

não eram destinadas ao código-fonte em si. No formato livre, disponível a partir do FORTRAN 90, estas restrições entraram em desuso. Neste artigo, serão abordadas características referentes à linguagem FORTRAN 90 ou superiores.

Um programa Fortran é organizado através do uso de um programa principal, procedimentos e módulos. Nos procedimentos estão trechos de códigos reutilizáveis, nos módulos estão partes de programas independentes que podem ser importadas no programa principal. Procedimentos e Módulos são opcionais na construção de um programa.

Fortran não é case-sensitive e suas palavras-chaves não são reservadas, desta maneira, pode-se criar variáveis que possuem o mesmo nome que palavras-chaves.

Nos próximos tópicos serão mostradas as principais ferramentas e estruturas utilizadas na programação Fortran, tais como: tipos de dados, estruturas de controle, sub-rotinas, funções, encapsulamento e tratamento de exceções.

4. Tipos de Dados

4.1. Tipos Primitivos

Os tipos primitivos disponíveis na linguagem Fortran são: INTEGER, para números inteiros; REAL, para números com ponto flutuante; CHARACTER*i, para sequências alfanuméricas; COMPLEX, para números complexos e LOGICAL, que pode assumir um valor true ou false. Existe também o tipo REAL*8, que é um tipo REAL com dupla precisão. Variáveis desse tipo são muito utilizadas em cálculos na engenharia.

As variáveis podem não ser declaras de nenhum tipo, desta maneira, assume-se para variáveis não declaradas o tipo inteiro, caso comecem com as letras i, j, k, l, m e n, e tipo ponto flutuante, para as que iniciam com as demais letras.

4.2. Tipos Compostos

Dentre os tipos compostos, temos os que combinam variáveis homogêneas (vetores e matrizes) e os que combinam variáveis heterogêneas. Estes últimos, na versão FORTRAN 2003, podem utilizar o conceito de extensão de tipos de dados.

4.2.1. Vetores e Matrizes

Um vetor ou matriz pode ser declarado por meio do uso da palavra-chave DIMENSION. Em seguida, devem ser informados os tamanhos de cada dimensão entre parênteses e separados por vírgula. Operações entre vetores e matrizes podem ser feitas de maneira muito intuitiva no Fortran.

Podem-se somar, subtrair, multiplicar, dividir ou exponenciar vetores inteiros com uma simples operação, desde que os vetores tenham o mesmo tamanho. Quando os vetores não possuem o mesmo tamanho, podem ser realizadas operações com seções dos vetores, como no esquema abaixo:

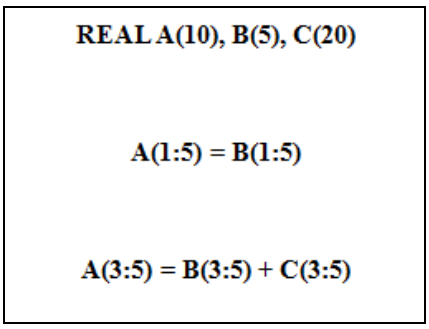

Figura 1. Seções de vetores

Na figura acima as posições de 1 a 5 do vetor B são copiadas para posições de 1 a 5 do vetor A. E os valores da posição 3 a 5 do vetor B são somados aos valores da posição 3 a 5 do vetor C e são armazenados nas posições de 3 a 5 do vetor A.

4.2.2. Tipos definidos pelo usuário

Tipos compostos heterogêneos podem ser definidos pelo usuário através da palavra-chave TYPE. A seguir um exemplo de como deve ser implementado.

```
!Declarando um tipo pessoa
TYPE :: pessoa
  CHARACTER(len=14) :: nome
  CHARACTER(len=14) :: telefone
  INTEGER :: idade
  REAL :: peso
  CHARACTER :: sexo
END TYPE pessoa
```

Figura 2. Tipo definido pelo usuário

Um tipo “pessoa” foi criado com cinco atributos, três do tipo cadeia de caracteres, um do tipo inteiro e um do tipo ponto flutuante. A seguir é mostrado como uma variável desse tipo pode ser instanciada e a sua inicialização, semelhante ao construtor de outras linguagens de programação.

```
!Declarando uma variável tipo pessoa
TYPE (pessoa) :: john
```

Figura 3. Variável tipo pessoa

```
!Inicializando a variável
john = pessoa('John','3227-4444',20,65.5,'M')
```

Figura 4. Inicialização da variável

Para acessar qualquer atributo de uma variável criada, usa-se o delimitador “%”(percentagem). Em outras linguagens, esse delimitador costuma ser o “.” (ponto).

```
!Acessando valores do tipo criado
john%peso = 63.4
PRINT *, john%telefone
```

Figura 5. Acessando atributos

Um tipo definido pelo usuário, em versões mais recentes do Fortran, como no Fortran 2003, pode ser estendido para novos tipos. A declaração é feita através da palavra-chave EXTENDS, o que permitirá a utilização de técnicas de polimorfismo.

4.3. Verificação de Tipos

A verificação de tipos na linguagem Fortran é dita fracamente tipada, pois é possível atribuir qualquer valor a uma variável de tipo numérico, não importando se o tipo é inteiro ou de ponto flutuante. Porém há o risco de ocorrer perda de dados com essas atribuições. A verificação somente é feita em tempo de compilação quando se tenta atribuir um tipo não-numerico a um tipo numérico, ou vice-versa.

5. Alocação Dinâmica de Memória

5.1. Alocação Dinâmica de Vetores e Matrizes

Alocação dinâmica de vetores e matrizes pode ser feita através do uso das palavras-chaves ALLOCATABLE E ALOCABLE. Essa operação só possibilita definir o tamanho de um vetor ou matriz em tempo de execução. A memória também pode ser desalocada através do uso das palavras-chaves DESALLOCATABLE e DESALOCABLE, visando um melhor uso da memória do computador. Tratamentos de exceções devem ser empregados para evitar possíveis valores absurdos passados como parâmetros para os tamanhos dos vetores.

5.3. Equivalence e Common

A declaração EQUIVALENCE, em determinado tipo de dado, permite a alocação de uma única área de memória para várias variáveis, estrutura semelhante a Union de outras linguagens de programação. O mesmo pode ser feito com blocos de programas, mas estes utilizam da declaração COMMON.

6. Comandos de Controle de Fluxo

6.1. Comandos de Decisão

Os comandos por decisão são semelhantes aos encontrados em outras linguagens. Existe o IF END IF, usado para selecionar determinado bloco de execução quando uma expressão relacional é verdadeira, e o SELECT CASE, onde uma variável é comparada com vários valores, quando a comparação tem resultado verdadeiro, o bloco referente a ela é executado**.**

6.2. Comandos de Repetição

Para comandos de repetição, existe o END DO limitado, que é parecido com a estrutura FOR encontrada em algumas linguagens. Nessa estrutura temos como parâmetros três valores: o primeiro é o valor inicial do *loop*, o segundo é o valor final do *loop* e o terceiro é o incremento. O fim do *loop* é dado quando se chega ao valor final do *loop*.

Outra estrutura de repetição é o DO WHILE, na qual o fim do loop ocorre quando determinada expressão lógica resulta em um valor falso.

Existem alguns comandos parecidos com CONTINUE e BREAK de outras linguagens, são eles CYRCLE e EXIT, respectivamente.

6.3. Go To

Utiliza-se o comando GO TO quando se tem por objetivo avançar ou recuar no código do programa de forma não sequencial. Para que esse avanço ou recuo sejam feitos, a linha de código precisa ter um rótulo, para que possa ser identificada. Esse comando requer um cuidado em sua utilização, pois pode prejudicar a legibilidade do código

```
1  PROGRAM teste_goto
2    IMPLICIT NONE
3    INTEGER :: fase = 0
4    10 continue O 10 representa o rótulo desta linha.
5    IF (fase < 3) THEN
6      fase = fase + 1
7      PRINT *, 'Fase', fase
8      !Utiliza-se GO TO 'rótulo da linha' para avançar ou recuar no código
9      !No caso abaixo, o GO TO leva para a linha de rótulo 10.
10     !Fazendo com que a variável 'fase' seja incrementada a cada loop.
11     GO TO 10
12   END IF
13   PAUSE
14 END PROGRAM teste_goto
```

Figura 6. Pequeno programa ilustrando a utilização do GO TO

7. Procedimentos

7.1. Sub-rotinas

As sub-rotinas executam determinado trecho de código especifico, e tem por objetivo evitar a reescrita de código. Uma sub-rotina deverá ter um nome e uma lista de argumentos, que pode ser vazia. Para chamar uma sub-rotina no programa principal, usa-se o comando CALL, seguido do nome da sub-rotina e a lista de argumentos.

7.2. Funções

Uma função é um procedimento semelhante a uma rotina, diferenciando-se pelo fato de que uma função retorna uma variável de algum tipo. Este tipo deve ser declarado como tipo da função, o usuário deve especificar qual será o retorno deste procedimento. A chamada de uma função no programa principal é feita apenas com o nome da função e a lista de argumentos.

7.3. Funções Intrínsecas

As funções intrínsecas são internas aos compiladores Fortran. Geralmente incluem

operações matemáticas básicas, como cálculos trigonométricos, operações entre matrizes, exponenciação, radiação, entre outras.

8. Encapsulamento

O Encapsulamento consiste em proteger partes do programa, evitando que possam ser acessadas por qualquer parte do programa. Geralmente, o encapsulamento é feito em uma parte separada do programa principal, chamada de módulo e declarado através das palavras-chaves MODULE END MODULE. Os componentes de um módulo podem ser usados por um programa principal ou outro módulo através da palavra-chave USE. Quando existe alguma função ou atributo em um módulo que não deve estar visível para outras partes do programa, usa-se a palavra-chave PRIVATE, caso o contrário, usa-se a palavra-chave PUBLIC.

9. Bibliotecas

Existe uma ampla quantidade de bibliotecas numéricas na linguagem Fortran. Uma biblioteca é um trecho de código externo, ou seja, um módulo. As bibliotecas numéricas do Fortran possuem uma série de sub-rotinas de métodos matemáticos. Uma das bibliotecas mais usadas no Fortran é a Biblioteca Numérica IMSL, muito utilizada para cálculos de raízes de equações complexas, regressão não linear e integração numérica.

10. Tratamento de Exceções

Existem tratamentos de exceções especialmente para cálculos com números reais usando os procedimentos internos da biblioteca IEEE. Segue abaixo um exemplo de uso de tratamento de exceções, onde se utiliza o método *ieee_set_halting_mode* para verificar se houve overflow em uma operação aritmética.

```
1  PROGRAM exception
2  USE, INTRINSIC :: ieee_exceptions, ONLY: &
3  ieee_overflow, ieee_get_flag, ieee_set_halting_mode
4  IMPLICIT NONE
5  REAL :: x, y, z
6  LOGICAL :: overflow_flag = .false.
7  PRINT *, 'Enter values for x and y to be multiplied:'
8  READ *, x, y
9
10 ! Irá fazer o programa continuar, mesmo que ocorra um erro
11 CALL ieee_set_halting_mode(ieee_overflow, .false.)
12
13 ! Faz a multiplicação
14 z = x * y
15
16 ! Verifica se houve overflow
17 CALL ieee_get_flag(ieee_overflow, overflow_flag)
18 IF (overflow_flag) THEN
19    PRINT *, 'An overflow occurred in the product x*y'
20  ELSE
21    PRINT *, 'No overflow occurred in the product x*y'
22 END IF
23 PRINT *, ''
24 print *, "The product was computed as ", z
25 end program exception
```

Figura 7. Tratamento de Exceções

11. Exemplo Original

Devido ao grande uso de Fortran em tratamento de problemas científicos na área da física, foi proposto como exemplo original deste trabalho, um jogo que simula o conceito de lançamento oblíquo. O programa foi desenvolvido de forma orientada a objetos.

O algoritmo do programa consiste em fornecer a velocidade e ângulo corretos para atingir um alvo a uma determinada distância. Os cálculos foram feitos em procedimentos. Foram usadas também estruturas de repetição e decisão.

12. Ambientes de Desenvolvimento

Existe uma variedade de compilares para programas Fortran. Na plataforma Linux, pode-se encontrar o GFortran. No Windows, temos o G95. Existem também compiladores com ambientes que permitem o desenvolvimento gráfico, como o Intel Visual Fortran e o Compaq Fortran.

13. Material Disponível na Internet

Na internet, há uma grande quantidade de material referente à Fortran 90/95. Materiais referentes às versões mais modernas são mais difíceis de ser encontrados,

principalmente sobre a versão 2008.

14. Análise Crítica e Considerações Finais

A linguagem Fortran pode ser considerada de fácil aprendizagem, pois possui uma boa legibilidade e é muito expressiva devido ao grande número de bibliotecas disponíveis. O entendimento de rotinas em Fortran se torna viável, devido às facilidades que podem se obter em cálculos numéricos.

15. Referências Bibliográficas

ATKINSON, Cyril P.; PACITTI, Tércio. *Programação e Métodos Computacionais*. 3 ed. Rio de Janeiro: LTC, 1980. 431 p. vol. 1.

CHAPMAN, S. *Introduction to Fortran 90/95.* 1 ed. New York: McGraw-Hill Professi, 1998. 555p.