Programação Científica em FORTRAN 95

PROGRAMAÇÃO CIENTÍFICA EM FORTRAN 95

2ª Edição Revisada e Ampliada

Rudnei Dias da Cunha

Clube de Autores
2011

1ª edição: 2010
2ª edição (revisada e ampliada): 2011

Rudnei Dias da Cunha é professor associado da UFRGS, desempenhando suas atividades junto ao Departamento de Matemática Pura e Aplicada, do Instituto de Matemática desde 1994. É Bacharel em Ciências de Computação pela UFRGS (1988) e *Doctor of Philosophy in Computer Science* pela *University of Kent at Canterbury*, Reino Unido (1992). Exerceu as funções de programador de computadores e analista de sistemas no Centro de Processamento de Dados da UFRGS (1983-1994) e do Centro Nacional de Supercomputação da UFRGS (1994). Foi coordenador do Programa de Pós-Graduação em Matemática Aplicada da UFRGS (1999-2000) , vice-diretor (2001-2004) e diretor (2005-2012) do Instituto de Matemática da UFRGS.

Capa: imagem renderizada com o algoritmo "Ray Tracing" usando o software **GEOM**, desenvolvido pelo autor em FORTRAN 95.

C972p Cunha, Rudnei Dias da
Programação Científica em FORTRAN 95 / Rudnei Dias da Cunha. – 2.ed. - Porto Alegre : Ed. do Autor, 2011.

414 p. : il.
Inclui bibliografia.
Inclui quadros, tabelas, expressões e equações.
Inclui índice analítico e apêndice
ISBN 978-85-910277-1-2

1. Informática. 2. Ciência e Tecnologia de Computação. 3. Linguagem de Programação – Fortran 95. 4. Linguagem Fortran – Histórico. 5. Programação estruturada. 6. Lógica da linguagem. 7. Matrizes esparsas. 8. Equações diferenciais parciais. 9. Estruturas de dados. I. Título.23

CDU 004.43

CIP-Brasil. Dados Internacionais de Catalogação na Publicação.
Bibliotecário Responsável Sibila Francine Tengaten Binotto CRB-10/1743

Pluralitas non est ponenda sine neccesitate.
Guilherme de Occam, c. Séc. XIV.

APRESENTAÇÃO

A linguagem de programação FORTRAN 95 é uma das principais linguagens de programação no ambiente acadêmico e científico. Resultado do constante processo de revisão da linguagem, num trabalho encabeçado pela *International Standards Organization*, ela oferece uma série de modernos recursos de programação que facilitam a criação de *software* científico.

Este livro apresenta os conceitos da linguagem de forma global e progressiva, de tal forma que o leitor possa ter desde o primeiro capítulo uma ideia dos inúmeros recursos que a linguagem oferece. Ele é baseado no meu livro sobre a versão anterior da linguagem, "Introdução à Linguagem de Programação FORTRAN 90" (Editora da Universidade Federal do Rio Grande do Sul – UFRGS, 2005) e na experiência adquirida ao ministrar cursos à distância sobre a linguagem FORTRAN 90 a alunos de todo o Brasil, através do Centro Nacional de Supercomputação da UFRGS, no âmbito do Sistema Nacional de Processamento de Alto Desempenho – SINAPAD. Como resultado dessa experiência foram revisados os capítulos, buscando esclarecer melhor alguns tópicos nos quais percebi maior dificuldade de entendimento pelos alunos. Além disso, o presente livro traz um conjunto de exercícios propostos ao final de cada capítulo, e através da sua resolução espera-se que o leitor possa fixar os conceitos da linguagem.

Nessa segunda edição, foram corrigidos alguns tópicos e incluídos outros, de forma a facilitar ainda mais o entendimento da linguagem.

A linguagem FORTRAN continua viva e vibrante, e continuará sendo usada por muitas décadas. Por isso, encerro essa apresentação da mesma forma que na obra predecessora; como disse certa vez C.A.R. Hoare, um dos grandes cientistas da computação: *"Não sei como será a linguagem de programação do ano 2000, mas ela será chamada* FORTRAN*"*.

Porto Alegre, novembro de 2011.

Rudnei Dias da Cunha

PREFÁCIO À 1ª EDIÇÃO

A linguagem de programação Fortran vem há muito tempo sendo uma das mais importantes e utilizadas em ambientes científicos e tecnológicos.

Voltada para o uso de computadores com arquiteturas avançadas, permanece a linguagem a ser usada para desempenhar tarefas da área da computação numérica de alto desempenho.

O FORTRAN permite a criação de programas que primam pela velocidade de execução. Daí reside seu uso em aplicações computacionalmente intensivas como meteorologia, física, astronomia, geofísica, engenharia etc.

O livro "Programação Científica em FORTRAN 95" foi escrito por um professor e pesquisador, que vem se destacando por suas pesquisas e que a utiliza como ferramenta de trabalho e, por isso tem a abrangência, a profundidade e a precisão necessárias para ser usado tanto em cursos de graduação, como de pós-graduação.

O livro oferece aos leitores um texto que abrange assuntos ligados à disciplina de Cálculo Numérico, bem como tópicos avançados em computação científica, tais como a resolução de sistemas lineares pelos Gradientes-Conjugados, e a solução de um PVI por Runge-Kutta de 4ª ordem.

A comunidade científica tem com a edição deste livro um forte aliado, que, sem dúvida, será de grande utilidade nas salas de aula e ambientes de pesquisa.

Porto Alegre, dezembro de 2009.

Dalcidio Moraes Claudio
Prof. Titular PUCRS

SUMÁRIO

1 - INTRODUÇÃO

O surgimento dos computadores de uso geral, a partir de 1950, trouxe a necessidade de serem utilizados de forma mais eficiente. A dificuldade existente na programação de computadores até então, usando-se diretamente códigos binários – números escritos com os algarismos 0 e 1 – era um dos principais entraves na utilização dos computadores.

Com isso, surgiu o conceito de linguagens de programação de alto nível. A primeira a surgir foi a linguagem FORTRAN. Um grupo de pesquisadores da IBM, liderado por John Backus (CHIVERS e SLEIGHTHOLME, 2000) desenvolveu a linguagem entre 1954 e 1957. O resultado foi uma linguagem que atendia às necessidades daqueles usuários que precisavam resolver problemas de cunho científico, com uma significativa parcela de cálculo envolvida. O próprio nome da linguagem já exibia essa característica: FORTRAN é uma corruptela de *"FORmula TRANslation"*.

Em 1966, quando a primeira versão padronizada da linguagem estava disponível, FORTRAN já era a linguagem de escolha da comunidade científica. Dentre as principais características que a linguagem exibia, destacam-se, como citado em (CHIVERS e SLEIGHTHOLME, 2000): ampla disponibilidade de compiladores para diferentes computadores; facilidade para ensiná-la; independência do computador; eficiência em muitas implementações; e demonstrava as vantagens do uso de sub-rotinas e de compilação independente. Além dessas, destacamos

a simplicidade da linguagem, que colaborou para a sua disseminação na comunidade científica.

A linguagem foi modernizada a partir de 1970 e o novo padrão, FORTRAN 77, introduzia algumas pequenas modificações na estrutura dos comandos, como o uso de blocos `IF .. THEN .. ELSE`. No entanto, nessa época, existiam outras linguagens – como `C`, PASCAL, ADA e MODULA – que haviam introduzido novos conceitos ou sedimentado outros, como a tipagem explícita de variáveis; definição de novos tipos de dados, permitindo a criação de estruturas de dados mais adequadas para resolver problemas; alocação dinâmica de dados; subprogramas recursivos; controle de exceção (uma das principais características da linguagem ADA); e estabelecimento de módulos. FORTRAN 77 não oferecia quaisquer desses recursos e, por isso, acabou sendo relegada quase que a segundo plano.

Por meados da década de 1980, uma nova revisão da linguagem estava por vir. Batizada então de FORTRAN 8X, ela viria a incorporar praticamente todos os outros conceitos já citados (exceto a tipagem explícita e o controle de exceção). O processo de definição do novo padrão foi demorado, porém permitiu que muitas das ideias a serem inseridas nesse padrão fossem amadurecidas. Assim, surgiu a linguagem FORTRAN 90, que, na visão do autor, não perde em nada para linguagens como C e PASCAL, para os fins a que ela se destina. Mais do que isso, ela incorpora mecanismos para a manipulação de arranjos que não são oferecidos em qualquer outra linguagem e que em muito auxiliam no desenvolvimento de programas científicos.

Em 1996, foi feita mais uma revisão da linguagem, chamada de FORTRAN 95 (ver KERRIGAN, 1993), a qual incorporou pequenas modificações à linguagem FORTRAN 95, motivada pela necessidade de aproximar o padrão à linguagem HIGH PERFORMANCE FORTRAN – HPF, a qual é voltada para o uso de computadores de arquiteturas avançadas, vetoriais e/ou paralelas. A linguagem HPF pode ser considerada uma extensão da linguagem FORTRAN 95 e, atualmente, muitos fabricantes oferecem compiladores HPF baseados na linguagem FORTRAN 95.

Desde 1997, a linguagem encontra-se em processo de revisão. As novas versões da linguagem FORTRAN (ver ISO/IEC JTC1/SC22/WG5, 2003 e ISO/IEC JTC1/SC22/WG5/N1791, 2009), denominadas FORTRAN 2003 e FORTRAN 2008, incorporarão uma série de novos comandos que permitirão, entre outros, o controle de exceções, programação orientada a objetos e suporte à programação paralela. Essas revisões aumentarão ainda mais os recursos da linguagem e, consequentemente, seu uso pela comunidade científica.

1.1 Estrutura do livro

O Capítulo 2 apresenta as noções iniciais da linguagem, através de um exemplo com várias características de um programa típico em FORTRAN 95. O Capítulo 3 trata da notação utilizada, estrutura de um programa, formato livre e fixo e conjunto de caracteres da linguagem. No Capítulo 4, são apresentados os diferentes tipos de dados oferecidos pela linguagem, bem como os comandos para a definição de dados do usuário. O Capítulo 5 apresenta as regras sintáticas para formação de literais e expressões, além dos operadores predefinidos. No capitulo 6 são apresentados os comandos de

controle de fluxo de execução da linguagem (decisão e repetição). O Capítulo 7 apresenta os comandos existentes para a realização de operações de entrada e saída, como a manipulação de arquivos externos. O leitor terá, a partir de então, condições de montar programas simples em FORTRAN 95.

O Capítulo 8 apresenta os comandos para declaração de subprogramas (sub-rotinas e funções), bem como a criação de módulos (conjuntos de subprogramas) e de sobrecarga de operadores, a partir de subprogramas criados pelo leitor. No Capítulo 9, são apresentados todos os intrínsecos oferecidos pela linguagem. O Capítulo 10 apresenta as características da linguagem a serem incorporadas nas novas versões da linguagem FORTRAN, bem como o uso em computadores de alto desempenho, com HPF e a biblioteca MESSAGE-PASSING INTERFACE. O Capítulo 11 apresenta uma série de estudos de caso nos quais são trabalhados os conceitos apresentados nos capítulos anteriores. No Apêndice A é feita uma breve discussão sobre o processo de compilação de programas FORTRAN 95 em um ambiente UNIX ou LINUX, incluindo o uso de `Makefile`. O Apêndice B apresenta um conjunto de sugestões para a conversão de programas em FORTRAN 77 para FORTRAN 95. No Apêndice C, são apresentadas as respostas aos exercícios propostos.

Todos os exemplos e estudos de caso aqui apresentados foram testados usando os compiladores `pgf90` (Portland Group, Inc.), `ifc` (Intel Compilers), `Fortran Powerstation 1.0` (Microsoft) e FTN95 (Silverfrost).

2 - NOÇÕES INICIAIS

Neste capítulo, apresentaremos um programa completo em FORTRAN 95, de tal forma que o leitor tenha um primeiro contato com as características da linguagem, sem preocupar-se de imediato com regras sintáticas da mesma, as quais serão apresentadas nos capítulos subsequentes.

O problema a ser resolvido através deste programa-exemplo é o de determinar o maior valor de cada coluna de uma matriz de valores reais, os quais se encontram armazenados em um arquivo de dados externo. O resultado deste programa será um vetor (ou *arranjo*) contendo aqueles valores máximos, os quais serão exibidos ao final da execução do programa.

O arquivo de dados contendo os elementos da matriz está organizado na forma de *registros*, isto é, um conjunto de caracteres em cada linha do arquivo. O arquivo – cujo nome é `MATRIZ.DAT` – contém no primeiro registro o número de linhas e de colunas da matriz e, nos registros subsequentes, os valores de cada linha da matriz, como segue:

```
 6 4
10.0   3.0   4.0  5.0
 1.0  -4.0   7.0  4.0
 2.0   9.0   8.0  1.0
 6.0  20.0  11.0  1.0
 0.0   8.0   3.0  7.0
 1.0   2.0   5.0  2.0
```

O programa será escrito de tal forma que ele possa ser utilizado para qualquer arquivo de dados. Dessa forma, como o número de linhas e colunas da matriz pode ser diferente para cada arquivo de dados, o programa deve ser capaz de criar dinamicamente uma matriz com tamanho suficiente para armazenar os elementos listados no arquivo de dados. A possibilidade de se efetuar a *alocação dinâmica de dados* é uma das principais vantagens da linguagem FORTRAN 95 sobre as versões anteriores.

A fim de ilustrar algumas das potencialidades da linguagem, o programa utiliza uma sub-rotina para calcular o valor máximo de cada coluna da matriz. O leitor observará, no exemplo , o uso extensivo de *comentários*, isto é, sentenças precedidas pelo caracter ! ; o uso de comentários é incentivado, pois auxiliam no entendimento do programa.

O programa deve realizar os seguintes passos:

1. Verificar a existência ou não do arquivo de dados e, caso o mesmo exista:
    - 1.1. abrir o arquivo para leitura;
    - 1.2. ler do arquivo o número de linhas e de colunas da matriz;
    - 1.3. alocar as áreas de memória para a matriz e para o arranjo que conterá os valores máximos de cada coluna;
    - 1.4. ler do arquivo os elementos da matriz;
    - 1.5. fechar o arquivo;
    - 1.6. invocar a execução da sub-rotina que calcula os valores máximos de cada coluna;
    - 1.7. exibir os valores máximos.

A seguir apresentamos o programa em questão.

Exemplo 2.1 *Um primeiro programa.*

```
   ! Início do texto do programa
   PROGRAM PRIMEIRO_PROGRAMA
   ! Declaração das variáveis do
   ! programa:
   ! Número de linhas e colunas
   ! da matriz
   INTEGER :: N_LINHAS, N_COLUNAS

   ! Arranjo que conterá a matriz
   REAL, DIMENSION(:,:), &
      ALLOCATABLE :: MATRIZ

   ! Arranjo para armazenar os valores
   ! máximos de cada coluna
  REAL, DIMENSION(:), &
      ALLOCATABLE :: VALORES_MAXIMOS

  ! Variáveis de controle para acessar
  ! elementos da matriz
  INTEGER :: I, J

  ! Variável lógica usada para
  ! verificar a existência do arquivo
  LOGICAL :: EXISTE_ARQUIVO

  ! Inicialmente, consulte se o arquivo
  ! MATRIZ.DAT existe; em caso
  ! afirmativo, a variável lógica
  ! EXISTE_ARQUIVO conterá o valor
  ! .TRUE. após a chamada à função
  ! INQUIRE (embutida na linguagem)
     INQUIRE(FILE='MATRIZ.DAT', &
         EXIST=EXISTE_ARQUIVO)
```

```
IF (EXISTE_ARQUIVO) THEN
    ! Abra o arquivo MATRIZ.DAT e
    ! associe-o à unidade lógica
    ! de número "1"
    OPEN(UNIT=1,FILE='MATRIZ.DAT')

    ! Leia o número de linhas e de
    ! colunas da matriz no primei-
    ! ro registro do arquivo
    ! em formato livre (FMT=*)
    READ(UNIT=1,FMT=*)N_LINHAS, &
                      N_COLUNAS

    ! Aloque a matriz com N_LINHAS
    ! linhas e N_COLUNAS colunas;
    ! aloque também o arranjo
    ! VALORES_MAXIMOS contendo
    ! N_COLUNAS elementos
    ALLOCATE(MATRIZ(N_LINHAS, &
          N_COLUNAS))
    ALLOCATE(VALORES_MAXIMOS( &
          N_COLUNAS))

    ! Leia os elementos da matriz
    ! a partir do segundo registro do
    ! arquivo de dados, em formato
    ! livre
    DO I=1,N_LINHAS
       READ(1,FMT=*)MATRIZ(I,:)
    END DO
    ! Feche o arquivo de dados
    CLOSE(UNIT=1)

    ! Determine o valor máximo de
    ! cada coluna da matriz chamando
```

```
        ! a sub-rotina MAXIMO_COLUNAS
        CALL MAXIMO_COLUNAS(N_LINHAS, &
              N_COLUNAS,MATRIZ, &
              VALORES_MAXIMOS)

        ! Exiba os valores máximos
        ! no dispositivo padrão de saída,
        ! o monitor de vídeo
        PRINT *,'VALORES MÁXIMOS:'
        PRINT *,'COLUNA | VALOR MÁXIMO'
        DO I=1,N_COLUNAS
           WRITE(*,*)I,VALORES_MAXIMOS(I)
        END DO

        ! Desaloque a matriz e o arranjo
        ! alocados anteriormente
        DEALLOCATE(MATRIZ, &
              VALORES_MAXIMOS)

    ELSE
        ! O arquivo MATRIZ.DAT não existe
        ! ou não foi localizado, exiba
        ! mensagem de erro
        PRINT *,'Arquivo MATRIZ.DAT ' &
                 'não pode ser aberto'
    END IF

    ! Fim dos comandos executáveis no
    ! programa principal, isto é, fim do
    ! programa
CONTAINS
    ! Declaração de uma sub-rotina
    ! interna ao programa principal
    SUBROUTINE MAXIMO_COLUNAS(M,N,A, &
                MAXIMOS)
    ! Parâmetros de entrada da
```

```
    ! sub-rotina
    INTEGER, INTENT(IN) :: M, N
    REAL, DIMENSION(M,N), INTENT(IN):: A
    ! Parâmetros de saída da sub-rotina
    REAL, DIMENSION(N), INTENT(OUT) :: &
       MAXIMOS
    ! Variável local a sub-rotina
    INTEGER :: I
       DO I=1,N
          MAXIMOS(I) = MAXVAL(A(1:M,I))
       END DO
    ! Fim do texto da sub-rotina
    END SUBROUTINE MAXIMO_COLUNAS

! Fim do texto do programa
END PROGRAM PRIMEIRO_PROGRAMA
```

O exemplo apresenta um programa que é representativo dos programas escritos na linguagem FORTRAN 95. Escrito em *formato livre* (vide seção 3.3), ele contém comandos que permitem *declarar* variáveis (dados e seus respectivos tipos) e blocos de programa (o programa principal) e subprogramas (`PROGRAM` e `SUBROUTINE`). A estrutura básica de um programa em FORTRAN 95, bem como as regras para formação dos nomes de variáveis, são descritas no Capítulo 3.

Observe que existem diferentes *tipos de dados*, expressos através das palavras `INTEGER` e `REAL`; a linguagem oferece outros tipos além desses. Além disso, variáveis podem ser *escalares* (isto é, conter apenas um valor, como as variáveis `N_LINHAS`, `N_COLUNAS`, `I` e `J`), ou podem ser *arranjos* (variáveis contendo mais do que um valor, como matrizes e vetores, através do uso do atributo `DIMENSION` na declaração das variáveis `MATRIZ` e `VALORES_MAXIMOS`). A declaração

de variáveis é estudada no Capítulo 4; a manipulação de variáveis e de expressões, através dos diferentes operadores oferecidos pela linguagem, é descrita no Capítulo 5.
Além disso, o programa apresenta vários comandos ligados a *operações de entrada e saída de dados*. Dados podem ser lidos através do comando `READ` e impressos (em um arquivo em disco, monitor de vídeo ou impressora) com os comandos `WRITE` e `PRINT`. Para serem efetuadas operações de entrada e saída em um arquivo externo, deve-se inicialmente abrir o mesmo, com o comando `OPEN`; uma vez que ele não seja mais utilizado, deve-se fechá-lo (isto é, torná-lo indisponível para acesso pelo programa) através do comando `CLOSE`. Outros comandos, como o `INQUIRE`, permitem que se verifique a existência de um arquivo de dados (entre outras opções). Os comandos existentes para realizar operações de entrada e saída de dados são descritos no Capítulo 7.

A linguagem FORTRAN 95 permite a alocação dinâmica de dados, conforme citado anteriormente. Isso é realizado sobre dados declarados com o atributo `ALLOCATABLE` (ver Capítulo 5), como feito com as variáveis `MATRIZ` e `VALORES_MAXIMOS`; essas variáveis só podem ser utilizadas, no entanto, após terem sido criadas na memória, através do comando `ALLOCATE`. É extremamente recomendável que, uma vez que elas não sejam mais necessárias no programa, sejam removidas da memória, através do comando `DEALLOCATE`. A utilização desses comandos é descrita com mais detalhes na seção 5.3.

Praticamente todo programa escrito em linguagens de alto nível apresenta pelo menos um *subprograma*, o qual é um trecho de código que pode ser invocado em diferentes trechos do programa principal ou de outras sub-rotinas. Em FORTRAN 95,

existem dois tipos de subprogramas: uma sub-rotina, declarada com o comando `SUBROUTINE`, e uma função, declarada com o comando `FUNCTION`. Além disso, os subprogramas podem ser *internos* a um programa principal ou outro subprograma (quando declarados após a palavra-chave `CONTAINS`) ou *externos*. Os subprogramas podem também ser agrupados em módulos (`MODULE`), os quais facilitam o desenvolvimento de programas estruturados. A declaração e uso de subprogramas é discutida no Capítulo 8.

A linguagem FORTRAN 95 oferece também um grande conjunto de funções *intrínsecas*, as quais permitem calcular várias funções matemáticas (como funções trigonométricas, por exemplo), manipular diferentes tipos de dados e efetuar modificações sobre os mesmos, dentre outras. No exemplo 2.1, a sub-rotina `MAXIMO_COLUNAS` utiliza o intrínseco `MAXVAL`, o qual retorna o maior valor dentre aqueles elementos de um arranjo. Os intrínsecos oferecidos pela linguagem são descritos no Capítulo 9.

Uma das grandes vantagens da linguagem FORTRAN 95 sobre as demais é a possibilidade de se manipular um grande número de elementos de um ou mais arranjos simultaneamente, e expressar tal manipulação de forma simples. Por exemplo, é possível indicar todos os elementos de uma determinada dimensão de um arranjo através do operador `:` (vide seção 5.5.3); no exemplo 2.1, o comando `MAXVAL(A(1:M,I))` faz com que seja calculado o maior valor, dentre os elementos de `A`, armazenados na linha `1` até a linha `M`, na coluna `I`.

Uma vez que o programa tenha sido compilado corretamente por um compilador FORTRAN 95 (mais detalhes sobre a compilação de programas num ambiente UNIX ou LINUX são

apresentados no Apêndice 1), pode-se executá-lo e obteremos o seguinte resultado:

```
VALORES MAXIMOS:
COLUNA | VALOR MAXIMO
      1        10.000000
      2        20.000000
      3        11.000000
      4         7.000000
```

o qual pode ser verificado por inspeção como sendo correto, em função dos dados constantes no arquivo `MATRIZ.DAT`.

3 - ESTRUTURA DE UM PROGRAMA

Neste Capítulo, descreveremos a estrutura de um programa em FORTRAN 95, contendo declarações de variáveis, comandos executáveis, subprogramas e módulos; a formatação do texto do programa ou módulo usando formato livre ou fixo, bem como o conjunto de caracteres da linguagem.

3.1 Notação

Inicialmente, cabe ressaltar que um programa de computador é um conjunto de instruções escritas de acordo com as regras sintáticas de uma linguagem. Ao longo deste texto, será utilizada uma notação para descrever os comandos da linguagem, conforme segue:

- qualquer símbolo da linguagem – variável, tipo de dado e comando – aparece escrito em `teletipo`;
- uma palavra englobada por < > indica o nome de uma *variável*, *expressão* ou *comando* da linguagem, que deve *obrigatoriamente* aparecer naquela situação;
- uma ou mais palavras englobadas por [ ] indica que tais palavras são *opcionais* naquele contexto;
- mais de uma palavra, englobadas por < >, separadas por —, indica que uma dessas palavras deve ser escolhida;
- o símbolo `::=` indica que a palavra à esquerda receberá o valor da palavra ou expressão à direita.

3.2 Definição de um programa

Um programa ou código-fonte escrito na linguagem FORTRAN 95 é composto por *blocos*, os quais definem as diferentes seções do mesmo. Um programa FORTRAN 95 tem, no mínimo, uma seção chamada de *programa principal*, a qual é composta por comandos não-executáveis (declarações e comentários) e executáveis, e é englobada pelos comandos `PROGRAM <nome-do-programa>` e `END PROGRAM <nome-do-programa>`. A estrutura básica um programa em FORTRAN 95 pode, então, ser descrita como:

```
PROGRAM <nome-do-programa>
[ <declarações> ]
[ <comandos-executáveis> ]
END PROGRAM <nome-do-programa>
```

Semântica:

1. `<declarações>` é um conjunto de *comandos não-executáveis* que definem os *dados* e *subprogramas* a serem usados no programa;
2. `<comandos-executáveis>` é um conjunto de comandos que serão executados, na ordem em que aparecem listados e atendendo a quaisquer desvios em seu *fluxo de execução*, conforme expresso pelo programador, valendo-se dos diferentes comandos de *controle* de fluxo de execução (ver Capítulo

6 - CONTROLE DE FLUXO DE EXECUÇÃO).

Assim, o menor programa que pode existir em FORTRAN 95 é dado pelo exemplo 3.1.

Exemplo 3.1 *Menor programa em FORTRAN 95.*

```
PROGRAM MENOR_PROGRAMA
END PROGRAM MENOR_PROGRAMA
```

Além do programa principal, pode-se definir *subprogramas* (`SUBROUTINE` e `FUNCTION`) e módulos (Capítulo 8). Um subprograma é dito *interno* quando o seu código *pertencer* a uma outra seção (isto é, o seu código-fonte é expresso após o comando `CONTAINS` dentro daquela seção), ou ele é *externo*, quando o seu código-fonte for descrito separadamente – em um outro arquivo ou dentro de um *módulo*. Um módulo (seção 8.7) é uma seção englobada por `MODULE <nome-do-módulo>` e `END MODULE <nome-do-módulo>`, o qual contém declarações de variáveis e tipos de dados e/ou um conjunto de subprogramas, necessariamente externos a outras seções (programa principal e outros módulos).

Um subprograma interno só pode ser invocado dentro da seção na qual ele é declarado; um subprograma externo pode ser invocado em qualquer seção que dele faça uso. Se um subprograma externo estiver contido em um módulo, então na seção onde ele for invocado deverá constar implicitamente um comando `USE <nome-do-módulo>`.

3.3 Formato livre e fixo

Existem dois tipos diferentes de formatos que um programa pode se apresentar – livre e fixo. No formato *livre*, um comando pode iniciar em qualquer caracter da linha, a qual contém até 132 caracteres. Nesse formato, os comandos podem aparecer um em cada linha ou, então, mais de um comando numa mesma linha, quando eles devem ser separados por `;` (ponto-e-vírgula). Caso um comando não caiba numa linha, ele deve ser interrompido com o caracter `&`, e a continuação do comando deve aparecer na linha seguinte. Um comentário, identificado pelo caracter `!`, pode aparecer em qualquer posição numa linha. O exemplo ilustra o uso do formato livre.

Exemplo 3.2 *Um programa em FORTRAN 95 usando formato livre.*

```
PROGRAM TESTE
! Este é um programa de teste
INTEGER :: I, J
REAL, & ! este é um comentário
    DIMENSION(200,300) :: MATRIZ
    MATRIZ = 100.0
END PROGRAM TESTE
```

Já no formato *fixo*, herdado da linguagem FORTRAN 77, uma linha tem apenas 72 caracteres, e os comandos devem aparecer apenas a partir da coluna 7, um comando por linha. As colunas 1 à 5 são reservadas para se identificar um comando através de um rótulo numérico; e a coluna 6 é reservada para a colocação de um caracter de continuação (diferente de 0 ou espaço em branco). Comentários podem ser inseridos através do

caracter `C` ou `*`, na coluna 1, ou através do caracter `!` , o qual só será considerado como início de comentário se aparecer nas colunas 1 à 5 ou 7 à 72. O exemplo mostra como usar o formato fixo, com os mesmos comandos usados no exemplo 3.2.

Exemplo 3.3 *Um programa em FORTRAN 95 usando formato fixo.*

```
C        1         2         3
C234567890123456789012345678901234567890
      PROGRAM TESTE
     * ESTE E UM PROGRAMA DE TESTE
       INTEGER :: I, J
       REAL, ! este é um comentário
      +DIMENSION(200,300) :: MATRIZ
       MATRIZ = 100.0
       END PROGRAM TESTE
```

Observe que a declaração do arranjo `MATRIZ` utiliza o caracter `+` como caracter de continuação, colocado na coluna 6 da linha subsequente à linha onde aparece `REAL, ! Este` é um comentário.

Os compiladores disponíveis para a linguagem FORTRAN 95 distinguem entre esses dois formatos através da extensão do nome do arquivo. Um programa gravado num arquivo `teste.F` ou `teste.f` será interpretado como sendo em formato fixo; se for utilizada a extensão `F90` ou `f90`, como em `teste.F90`, então o programa será considerado como sendo em formato livre. Alguns compiladores, no entanto, permitem que se desative tal convenção, através do uso de uma *flag* específica de compilação; consulte o manual do compilador em uso para maiores informações.

3.4 Conjunto de caracteres da linguagem FORTRAN 95

Qualquer programa em FORTRAN 95 é escrito utilizando-se um conjunto de caracteres, incluindo as letras do alfabeto, os algarismos e outros caracteres especiais.

Letras: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Algarismos: 0 1 2 3 4 5 6 7 8 9

Caracteres especiais:

	espaço em branco
:	dois pontos
;	ponto-e-vírgula
=	igual
+	mais
−	menos
*	asterisco
/	barra diagonal
(	abre parêntese
)	fecha parêntese
,	vírgula
.	ponto
'	apóstrofo
"	aspas
!	ponto de exclamação
?	ponto de interrogação
%	porcentual
&	e comercial
<	menor do que
>	maior do que
_	sublinha

O uso de letras minúsculas num programa FORTRAN 95 é dependente do compilador. Caso seu uso seja aceito, então as letras minúsculas são consideradas como se fossem maiúsculas, com exceção de seu uso em constantes literais (ver Capítulo 5), em caracteres de edição (vide seção 7.3.2) e em parâmetros dos comandos `OPEN` e `INQUIRE` (vide seção 7.2).