2 ELEMENTOS BÁSICOS DO FORTRAN

Neste capítulo estuda-se inicialmente a estruturação básica envolvida na criação de um código executável em Fortran, seguida da descrição dos elementos que compõe a linguagem.

2.1 A SINTAXE DA LINGUAGEM

Em Fortran há três formatos básicos de arquivos envolvidos no processo de criação de um código executável:

Programa-Fonte. Trata-se do programa e/ou dos subprogramas escritos pelo programador, usando algum tipo de editor de texto, de acordo com as regras definidas pela linguagem de programação de alto nível. O programa-fonte pode estar estruturado por diferentes *unidades de programa*, as quais serão discutidas em mais detalhes no capítulo 9.

O envolvimento ativo do programador ocorre somente nesta etapa do processo de criação do código executável. As etapas a seguir são usualmente realizadas de forma automática pelo sistema de compilação.

Programa-Objeto. Trata-se do programa-fonte compilado pelo compilador. Esta é a transcrição realizada pelo compilador a partir do programa-fonte fornecido pelo programador para uma linguagem de baixo nível, como Assembler ou outro código diretamente interpretável pela CPU. O programa-objeto não pode ser diretamente executado; é necessário ainda passar-se pela fase de *ligação*, *interconexão* ou *linkagem* (transliteração do termo *linking*).

Programa executável. Após a fase de compilação, onde os programas-objeto são criados, o agente de compilação aciona o *interconector* (*linker*), o qual consiste em um programa especial que agrupa estes programas-objeto com outros objetos contidos em bibliotecas do sistema, de forma a criar um arquivo final, o programa executável, o qual pode ser então executado pelo programador.

Este processo está representado na figura 2.1.

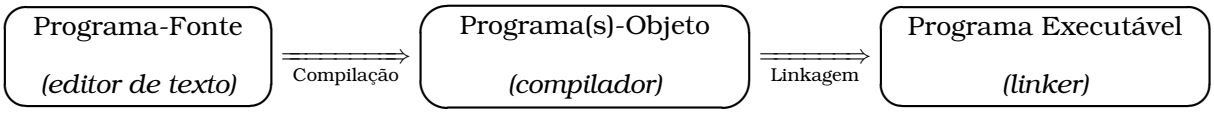

Figura 2.1: Processo de criação de um programa executável.

Nesta seção será apresentada a estrutura básica de um programa-fonte em Fortran. A estrutura de uma linguagem de programação assemelha-se a de uma língua indo-europeia, a qual é composta a partir de elementos básicos, as letras, as quais são combinadas para formar palavras, as quais são colocadas de forma adjacente separadas por espaços em branco ou por pontos, formando assim uma oração ou frase. Estas por sua vez compõe parágrafos que se juntam para formar as demais estruturas de um texto completo, destinado a transmitir ao leitor um pensamento, história ou raciocínio. Este texto é usualmente autoconsistente e em diferentes graduações independente de outros textos. O conjunto de regras que determina como todos estes elementos são compostos e empregados é denominado a **sintaxe** da linguagem.

A sintaxe do Fortran é estruturada de uma forma semelhante. Os elementos básicos da linguagem são:

caracteres alfanuméricos: compostos pelas 26 letras latinas maiúsculas ou minúsculas

```
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

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
```

juntamente com os 10 numerais arábicos

```
0 1 2 3 4 5 6 7 8 9
```

e o caractere de grifo (*underscore*) "_".

caracteres especiais: símbolos que são reconhecidos pelo compilador e que podem ou não possuir função sintática. A tabela 2.1 lista os caracteres especiais suportados pela linguagem.

Tabela 2.1: Caracteres especiais do Fortran. C.F.S: com função sintática; S.F.S: sem função sintática.

	Nome (C.F.S.)		**Nome (C.F.S.)**		**Nome (S.F.S.)**
=	Igual	:	Dois pontos	\	Barra invertida
+	Soma		Espaço em branco	$	Cifrão
-	Subtração	!	Exclamação	?	Ponto de interrogação
*	Multiplicação	%	Porcentagem	{	Chave à esquerda
/	Divisão	&	E comercial (*ampersand*)	}	Chave à direita
(	Parênteses esquerdo	;	Ponto e vírgula	~	Til
)	Parênteses direito	<	Menor que	‘	Crase
[	Colchete esquerdo	>	Maior que	^	Acento circunflexo
]	Colchete direito	’	Apóstrofe	|	Linha vertical
,	Vírgula	“	Aspas	#	Cerquilha
.	Ponto decimal			@	Arroba

Os símbolos básicos listados acima combinam-se para formar os **vocábulos léxicos** (***lexical tokens*** ou simplesmente ***tokens***), os quais correspondem às palavras em uma linguagem natural e que fornecem unidade basica de informação do Fortran. Tokens adjacentes são usualmente separados por espaços em branco ou pelo final da linha. Sequências de tokens formam uma **declaração** (***statement***) ou **comando** (ou **instrução**). Estas declarações ou comandos correspondem às frases em uma linguagem natural. Da mesma forma, declarações ou comandos podem ser agrupados para formar o que correspondem aos parágrafos e que no Fortran são denominadas as **unidades de programa**, a partir das quais será composto o programa-fonte. Esta estruturação será discutida em maiores detalhes nas seções a seguir.

2.2 Vocábulos léxicos *(tokens)*

Os caracteres aceitos pelo Fortran podem ser combinados de diferentes maneiras em diferentes contextos para a formação de um determinado token. Um token pode ser um **rótulo**, uma **palavra-chave** (***keyword***), um **nome**, uma **constante**,[1] um **operador**,[2] ou um **separador**. Alguns dos tipos de tokes listado acima serão discutidos a seguir; em particular, os separadores são

```
/ ( ) [ ] (/ /) , = => :  ::  ; %
```

os quais irão aparecer ao longo deste texto.

As palavras-chave (*keywords*) são nomes que possuem significados especiais no Fortran. Palavras-chave adjacentes devem ser separadas por um ou mais espaços em branco ou pelo final da linha. Contudo, como em qualquer linguagem natural, há exceções. As palavras-chave adjacentes listadas na tabela 2.2 não precisam ser separadas por espaços. Assim, "`end if`" tem o mesmo significado que "`endif`".

Por outro lado, com relação aos rótulos, nomes e constantes, o uso de espaços em branco entre tokens adjacentes irá depender do contexto. Para facilitar a compreensão, pode existir

[1] Discutidas no capítulo 3.
[2] Discutidos no capítulo 4.

Tabela 2.2: Palavras-chave adjacentes para as quais espaços em branco são opcionais.

`block data`	`double precision`	`else if`	`else where`
`end associate`	`end block`	`end block data`	`end critical`
`end do`	`end enum`	`end file`	`end forall`
`end function`	`end if`	`end interface`	`end module`
`end procedure`	`end program`	`end select`	`end submodule`
`end subroutine`	`end type`	`end where`	`go to`
`in out`	`select case`	`select type`	

mais do que um espaço em branco entre tokens adjacentes sem que o significado sintático seja alterado. Por exemplo, a expressão "`x*y`" é composta por três tokens: "`x`", "`*`" e "`y`" e significa o produto de `x` por `y`. O mesmo resultado será obtido se esta expressão for escrita "`x * y`" (com espaços em branco) ou "`x* y`". Por outro lado, "`real x`" é sintaticamente distinto de "`realx`".

2.3 Nomes válidos em Fortran

Um programa em Fortran faz referência a muitas entidades distintas, tais como programas, subprogramas, módulos, variáveis, etc. Os nomes destas entidades devem consistir em caracteres alfanuméricos e conter de 1 a 63 destes caracteres. Não há restrições na composição destes nomes, exceto que o primeiro caractere deve ser uma letra. Os seguintes exemplos são válidos:

```
A
A_COISA
X1
MASSA
Q123
TEMPO_DE_VOO
```

Já estes exemplos **não** são válidos:

```
1A        (começa com numeral)
A COISA   (contém espaço em branco)
$SINAL    (contém caractere não alfanumérico).
```

É importante enfatizar que o Fortran é *insensível a maiúsculas ou minúsculas*, isto é, para o compilador, o nome `TEMPO_DE_VOO` é idêntico a `tempo_de_voo` ou `Tempo_de_Voo`.

Sugestões de uso & estilo para programação

Existem convenções estilísticas para a definição de nomes de distintos tipos de objetos da linguagem. Essas convenções não fazem parte do padrão mas são outrossim sugestões que visam uma melhor compreensão do código. Uma regra universal consiste em empregar nomes mnemônicos. Por exemplo, `massa` ou `tempo_de_voo`. Algumas dessas convenções serão mencionadas ao longo do texto.

2.4 Formatação do programa-fonte

As declarações ou comandos que irão compor um programa-fonte em Fotran são escritas ao longo de linhas em um editor de texto. O Fortran adota o **formato livre** para essas linhas:

- No formato livre não há uma coluna específica para iniciar a linha. Pode-se começar a escrever o código a partir da coluna 1 e a linha de código pode se estender até a coluna 132. Além disso, os caracteres em branco são irrelevantes em qualquer lugar do código, exceto quanto estiverem sendo utilizados entre apóstrofes. Neste caso, cada caractere em branco será incluído na composição final de uma *constante de caracteres*.[3]

- Cada linha de texto é encerrada pelos registros *end-of-line* (EOL, *final de linha*) ou *carriage-return/line-feed* (CR/LF, *retorno-de-carro/nova-linha*), automaticamente inseridos pelo editor de texto e que usualmente permanecem invisíveis. Ao contrário de outras linguagens,

[3] Discutidas na seção 3.1.4.

como o C, não existe no Fortran um caractere especial para marcar o final de uma instrução ou declaração.

- Mais de uma instrução pode ser colocada na mesma linha. O **separador de instruções ou declarações** é o ponto e vírgula (;). Múltiplos “;” seguidos em uma linha, com ou sem brancos, são considerados como um separador simples. Desta forma, a seguinte linha de texto é interpretada como 3 linhas de código em sequência:

```
A = 0; B = 0 ; C = 0
```

  O caractere “;” não pode iniciar uma linha, exceto quando esta for a continuação de uma linha anterior.

- O caractere *ampersand* “&” é a **marca de continuação**; isto é, ele indica que a linha com instruções imediatamente posterior é a continuação da linha onde o “&” foi digitado. A continuação de linhas pode ser empregada para facilitar a leitura do código ou porque as 132 colunas disponíveis para a linha não foram suficientes para escrever toda a instrução. São permitidas até 255 linhas adicionais de código.

  Como exemplo, a linha de código

```
X = (-Y + ROOT_OF_DISCRIMINANT)/(2.0*A)
```

  também pode ser escrita com linhas adicionais usando o “&”:

```
X =                                 &
      (-Y + ROOT_OF_DISCRIMINANT)   &
      /(2.0*A)
```

  O caractere de continuação é usualmente (mas não necessariamente) o último caractere sintaticamente significativo não nulo em uma linha, exceto se for seguido pelo caractere de comentário (a seguir). Em certas situações, pode ser necessário continuar um token na próxima linha. Neste caso, além de se incluir “&” ao final da linha, o primeiro caractere não nula da próxima linha também deverá ser “&”. Isto usualmente ocorre com constantes de caractere. Por outro lado, nenhuma linha pode conter somente “&” ou ter “&” como o único caractere não nulo antes da *marca de comentários*.

- A **marca de comentários** é o ponto de exclamação “!” Para entrar com comentários em qualquer ponto do código-fonte, o usuário deve digitar o ponto de exclamação “!” em qualquer coluna de uma linha. Todo o restante da linha será desprezado pelo compilador. Por exemplo:

```
X = Y/A - B ! Soluciona a equação linear.
```

  Como um comentário sempre se estende até o final da linha, não é possível inserir comentários entre instruções em uma mesma linha.

2.5 O PROGRAMA “ALÔ MAMÃE”

Como primeiro exemplo de um programa em Fortran, considere o seguinte código-fonte:

Listagem 2.1: Primeiro programa em `Fortran`.

```
program primeiro
implicit none
print *, "Alô Mamãe"
end program primeiro
```

A discussão mais detalhada a respeito das unidades de programa do Fortran será realizada no capítulo 9. O exemplo acima ilustra a forma mais simples de um programa em Fortran, composta somente pelo **programa principal**. A estrutura do programa principal é a seguinte:

```
PROGRAM <nome_do_programa>
<declarações de nomes de variáveis>
<comandos executáveis>
END PROGRAM <nome_do_programa>
```

Comparando com o exemplo do programa `primeiro`, a declaração

```
PROGRAM primeiro
```

é sempre a primeira instrução de um programa. Ela serve para identificar o nome do código ao compilador.

Em seguida vêm as `<declarações de nomes de variáveis>`, as quais podem conter mais de uma linha. Neste ponto, são definidos os nomes das variáveis a ser usadas pelo programa. Caso não seja necessário declarar variáveis, como no programa `primeiro`, é recomendável incluir, pelo menos, a instrução `implicit none`. Esta instrui o compilador a exigir que todas as variáveis usadas pelo programa tenham o seu tipo explicitamente definido.[4]

Sugestões de uso & estilo para programação

É fortemente recomendado que a declaração `implicit none` seja sempre incluída. Desta forma, se for incluído algum nome (de variável, função, *etc*) que não foi explicitamente declarado, o compilador encerra o processamento do código-fonte emitindo uma mensagem de erro. O uso deste recurso torna o código-fonte *fortemente tipado* (*strong typed*) e pode evitar erros de programação potencialmente sérios.

Após declaradas as variáveis, vêm os `comandos executáveis`. No caso do programa `primeiro`, o único comando executável empregado foi

```
print *, "Alô Mamãe"
```

o qual tem como consequência a impressão do texto "Alô Mamãe" na tela do monitor, o qual é a chamada *saída padrão.*

Finalmente, o programa é encerrado com a instrução

```
end program primeiro
```

Os recursos utilizados no programa `primeiro` serão explicados com mais detalhes neste e nos próximos capítulos da apostila.

2.6 ENTRADA E SAÍDA PADRÕES

O Fortran possui três comandos de entrada/saída de dados. A maneira mais direta de definir valores de variáveis ou de exibir os valores destas é através dos dispositivos de entrada/saída padrões.

O dispositivo padrão de entrada de dados é o teclado. O comando de leitura para o programa ler os valores das variáveis é:

```
READ *, <lista de nomes de variáveis>
READ(*,*) <lista de nomes de variáveis>
```

onde na `lista de nomes de variáveis` estão listados os nomes das variáveis que deverão receber seus valores via teclado. O usuário deve entrar com os valores das variáveis separando-as por vírgulas.

O dispositivo padrão de saída de dados é a tela do monitor. Há dois comandos de saída padrão de dados:

```
PRINT *, ['<mensagem>'[,]][<lista de nomes de variáveis>]
WRITE(*,*) ['<mensagem>'[,]][<lista de nomes de variáveis>]
```

O programa a seguir instrui o computador a ler o valor de uma variável real a partir do teclado e, então, imprimir o valor desta na tela do monitor:

[4] Os tipos de variáveis suportados pelo Fortran são discutidos no capítulo 3.

```
program le_valor
implicit none
real :: a
print *, "Informe o valor de a:"
read *, a
print *, "Valor lido:",a
end program le_valor
```

No programa acima, foi declarada a variável real `a`, cujo valor será lido pelo computador, a partir do teclado, e então impresso na tela do monitor. É importante salientar que a instrução `implicit none` deve ser sempre a segunda linha de um programa, sub-programa ou módulo, aparecendo antes do restante das declarações de variáveis.

Esta seção apresentou somente um uso muito simples dos recursos de Entrada/Saída de dados. Uma descrição mais completa destes recursos será realizada no capítulo 10.