Livro Texto:

Fortran estruturado, Autores - Harry Farrer e outros, 1992, Editora LTC.

Pode usar outros que tenham na biblioteca

Programa do Curso:

- Introdução ao processamento de dados
- Linguagem Fortran
- A evolução da linguagem Fortran
- Sintaxe de programas em Fortran
- Formatos livres e fixos
- Comandos de edição, compilação e execução
- Constantes variáveis e expressões
- Constantes inteiras, reais e caracteres
- Variáveis inteiras, reais e caracteres
- Comandos de especificação de variáveis e constantes
- Comandos de designação de áreas na memória
- Expressões aritméticas, relacionais e lógicas
- Comandos de controle de fluxo
- Comando GO TO
- Comandos IF lógico e bloco
- Comandos de entrada e saída
- Registros, arquivos e unidades
- Comandos OPEN e CLOSE
- Comandos READ
- Comandos WRITE
- Comandos FORMAT e especificação de formato
- Subprogramas
- Argumentos de Subprogramas
- Funções Intrínsecas
- Subrotinas e comando SUBROUTINE
- Aplicações simples
- Solução de equações de segundo grau
- Cálculo de médias e desvio padrão
- Multiplicação de matrizes

História do Fortran:

A linguagem fortran foi a primeira linguagem de alto nível usada para a programação de computadores. Foi proposta em 1953 por J. Backus, mas seu primeiro compilador só foi liberado em 1957. Seu nome é formado pelas letras da palavra Formula Translation, que indicam sua principal inovação na época: a facilidade de transcrição de fórmulas matemáticas para serem usadas nos computadores. O fortran foi um programa verdadeiramente revolucionário, antes dele todos os programas de computadores eram lentos, tendenciosos e originavam muitos erros.

Nos primeiros tempos, o programador podia escrever o algoritmo desejado como uma série de equações algébricas padronizadas e o compilador fortran podia converter as declarações em linguagem de máquina que computador reconhecia e executava.

O desenvolvimento continuou em 1962, com o lançamento do fortran IV, que tinha muitos melhoramentos e por isso tomou-se a versão mais utilizada nos 15 anos seguintes. Em 1966 o fortran IV foi adaptado como um padrão ANSI e passou a ser conhecido como fortran 66. Outra atualização importante aconteceu em 1977. Nesta versão foram incluídas novas características, que permitiram escrever e guardar mais. facilmente programas estruturados.

Foram introduzidas novas estruturas, como o bloco IF e foi a primeira versão de fortran em que as variáveis "character" (caracteres) eram realmente fáceis de manipular. No entanto, esta versão ainda era limitada em termos de estruturas de informação e, também por só permitir a codificação de algumas figuras de programação estruturada.

O melhoramento seguinte foi importante e deu origem ao fortran 90. Este incluía todo o fortran 77 como base e com mudanças significativas, mudanças nas operações sobre tabelas (array); na parametrização das funções intrínsecas, permitindo assim utilizar uma sequência de caracteres maior, como também usar mais do que dois tipos de precisão para variáveis do tipo Real e Complex; houve um aperfeiçoamento da computação numérica com a inclusão de um conjunto de funções numéricas.

No conjunto, os novos aspectos contidos no fortran 90 fazem com que esta, seja considerada a linguagem mais eficiente da nova geração de supercomputadores, e asseguram que o fortran continuará a ser usado com sucesso por muito tempo.

Utilização da linguagem Fortran:

A utilização da linguagem fortran para resolver um problema faz-se como em outras linguagens, através de sucessivas fases:

1) Descrição do problema: é a descrição e a delimitação adequada do problema a ser resolvido, caracterizando-o de maneira clara e completa .
2) Desenvolvimento de um algoritmo: objetiva a descrição, geralmente desenvolvido por etapas, do processamento a ser realizado para resolver o problema.
3) Transcrição do algoritmo para a linguagem fortran: transcrever o algoritmo para a linguagem fortran utilizando as regras e recursos oferecidos.
4) Digitação do programa: isto é feito diretamente no computador no próprio editor de texto do programa, seguindo as regras de cada compilador.
5) Processamento do programa: o compilador fortran verifica a correção e sintaxe do código e traduz para a linguagem de máquina. Existem diversas versões de compiladores fortran: gfortran, ifort, fl32, f77, f90, g77 entre outros, mas a sintaxe básica para a compilação é:

$$\underbrace{gfortran}_{compilador} \ldots \underbrace{modelo.for}_{programa\ fonte} \ldots \underbrace{-o}_{parâmetro\ de\ saída} \ldots \underbrace{modelo.exe}_{executável}$$

O texto de um programa escrito em fortran deve ficar disposto em linhas. As linhas devem ter 72 colunas, e cada coluna só pode conter 1 caractere (letra, número ou sinal especial).

Cada linha pode conter um comentário, declaração ou parte do programa.

Se a linha tiver um comentário, a coluna 1 deve obrigatoriamente ter a letra C ou o sinal !, e as colunas 2 a 72 o comentário.

Se a linha contiver um comando ou uma declaração, as suas colunas devem ser agrupadas da seguinte maneira:
- coluna 1 a 5 é reservada para o número do comando ou da declaração quando houver, não deve ser usado para escrever o programa.
- coluna 6 é reservada para indicar a continuação da linha anterior.
- colunas 7 a 72 reservadas para a escrita do programa, não deve ultrapassar a linha 72.

EX.1:
!1 2 3 4 5 6 7.....72

```
! programa para escrever numeros impares
! declarações
        Integer numero

!programa
        Do i =1,43,2
         Numero=i
        Print *, numero
       Enddo
        End
```

Ex. 2:

```
1 2 3 4 5 6 7.....72 ! programa para calcular uma equação qualquer
! declarações
              Integer I, medida(l20), alturaesq(120), alturadir(l20),
            & espessura(120), comprimento(l20)
! programa
              Do 10 i =1,43,2
              medida (i) = (lado(i)*alturaesq(i)*alturadir(i)*espessura(i)**2)/ Comprimento(i)
    10        Enddo
              End
```

Itens fundamentais:

1) Constantes

Conforme seu tipo, a constante é classificada como numérica, lógica ou literal, entre outras.

1a) Constante numérica:

A representação de uma constante numérica, é feita no sistema decimal, pode ser um número inteiro, ou com parte decimal. Chama-se de real ao número com parte decimal, sendo que esta é separada da parte inteira por um ponto, **(Atenção, em fortran sempre a separação é por um ponto, e não por uma virgula como usamos).** Chama-se de inteiro o número sem parte decimal, este nunca é acompanhado de ponto.

Ex. 3:

25 → constante inteira
3.14 → constante real
.7 → constante real
1.0 → constante real
1 → constante inteira

Como na matemática, o fortran admite a existência de constante reais com expoente decimal, isto é, um fator 10 elevado a um expoente inteiro.

Ex. 4:
$7{,}8 \times 10^{-3} \rightarrow$ 7.8E3
$35{,}93 \times 10^{-2} \rightarrow$ 35.93E-2

Tanto a constante, quanto o expoente decimal podem ser positivos ou negativos, conforme o sinal que os preceda.

Ex.5:
Constantes válidas

Inteiro	Real
15, +15, -15	15.1, +15.1, -15.1, .1, -.6

Constantes inválidas
12,5 (a vírgula não é um caractere de separação,
3.462.308 (as classes com centena, milhão, não são separados por ponto, nem por qualquer outro caractere)

Cada compilador tem um limite para a maior ou menor constante inteira e real, que ele pode armazenar. Valores maiores que esse limite causam erros fatais de execução. Neste compilador que está sendo usado os valores são da ordem de ±IE37.

1b) Constante lógica

É um valor lógico, que só pode ser falso ou verdadeiro, usado em proposições lógicas. Só existem 2 constantes deste tipo:

.TRUE. $\rightarrow$ verdadeiro.

.FALSE.$\rightarrow$ falso.

1c) Constante literal

É qualquer sequência de caracteres (letras, dígitos ou símbolos especiais) que formem um literal. Esta sequência de caracteres, também conhecida por string, deve estar entre apóstrofos.

Ex.6:
'luciana'
'guarda-roupa'
'25/0111983'

Exercício de fixação 1:

Identificar o tipo de cada urna das constantes:

a) 21 d)0.21E2

b) 'bola' e) 45.

c).true. f) ‘0.2’

2) Variáveis:

Uma variável é a representação simbólica dos elementos de um certo conjunto. Cada variável corresponde urna posição de memória, que pode variar (ou não) com o tempo. Toda a variável é identificada por um identificador, cuja formação obedece a determinadas regras.

2a) Formação dos identificadores

Um identificador é formado por um conjunto de caracteres (não muitos, vale o bom senso), sendo que o primeiro deve obrigatoriamente ser urna letra, os demais podem ser letras ou dígitos, não sendo permitidos o uso de caracteres especiais.

Ex. 6:

Identificadores permitidos	Identificadores não - permitidos
A	5B → começa com número
Nota	E(13) → contém o caractere especial **( )**
X5	A:B → contém o caractere especial**:**
A32B	Matrícula → contém o acento no **i**

Exercício de fixação 2:

Assinalar os identificadores válidos:

( ) valor ( ) x2 ( ) 3x4 ( ) xyz ( ) 'nota' ( ) salario-minimo ( ) nota*do*aluno

( ) Maria ( ) ah! ( ) alb2c3 ( ) km/h ( ) sala215

2b) Declaração das variáveis:

As variáveis (como as constantes) dependendo do valor nela armazenadas podem ser classificadas como: inteira, real, lógica ou literal (entre outras).

O primeiro caracter do identificador de uma variável define implicitamente o tipo desta variável segundo a seguinte convenção:

I, j, k, l, m, n → representam variáveis inteiras (ex.: iano, imes, idia)

Restante → representam variáveis reais (ex.: ano, mes, dia)

Embora seja permitida esta declaração implícita, para maior clareza do programa é aconselhável que as variáveis sejam declaradas explicitamente, o que pode ser feito da seguinte maneira:

Tipo da variável	Identificador
integer	iano
real	ano
logical	tipo

As variáveis literais podem armazenar um sequência, com um número variável de caracteres e tem a seguinte declaração:

character nome* 5
ou
character*5 nome

String nome com 5 caracteres,
ex.: nome = 'maria'

Exercício de fixação 3:

Supondo que as variáveis: NOME, PROF, IDADE e SAL serão utilizadas para armazenar o nome, profissão, idade e salário de uma pessoa, escrever o conjunto de declarações necessárias para criar essas variáveis e associar as mesmas aos respectivos tipos básicos.

3) Comentários

É um texto ou simplesmente uma frase utilizada para melhor clareza do programa. A linha do comentário de ter o símbolo ***!*** ou a letra na ***C*** na primeira coluna.

EX.7:

```
! programa para o calculo da nota final dos alunos da turma 25 ! mat - matrícula do aluno
! nota - total de pontos no semestre
! cod - código do aluno na disciplina
```

4) Expressões aritméticas

Denomina-se expressão aritmética àquela cujos operadores são aritméticos e cujos operandos são constante e/ou variáveis numéricas.

O conjunto de operadores e as operações associadas são:

operador	operação
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
**	Potenciação

EX.8:

a) x+y	b) x-y	c) 2*nota
d) total/n	e) soma**2	f) (F1 +G ** 2) ** 0.5 – H
g) p ** 0.5	h) a * b /M	

IMPORTANTE: A notação utilizada para as operações obedece as seguintes regras

a) Não é permitido omitir o sinal de multiplicação, como comumente fazemos na matemática.

b) Os operadores seguem uma relação de prioridades:

Prioridade	Operação
1°	Potenciação
2°	Multiplicação ou divisão
3°	Adição ou subtração

Para se obter uma sequência de cálculos diferentes, vários níveis de parênteses podem ser usados, não sendo permitido o uso de chaves ou colchetes.

EX.9:

$\sqrt{P(P-A)(P-B)(P-C)}$ em fortran → (P*(P-A)*(P-B)*(P-C))**0.5

$-A-B\left(C+\frac{D}{E-1}-F\right)$ em fortran → -A –B * (C +D/ (E-1) – F) +G

As expressões podem ser reais, inteiras ou mistas, dependendo de seus operandos.

I/I + M * * 2 - N → tipo inteiro } I, J, M, N →inteiro

A/B + C * 4. - X → tipo real } A, B, C, D e X → real

I/A + C *4 - D →tipo mista

Deste modo, o tipo do resultado de uma expressão depende do tipo dos operandos envolvidos, ou seja:

Tipo de operação	Tipo de Resultado
Inteiro	Inteiro
Real	Real
Mista	Real

OBS.: É bom destacar que o resultado de uma divisão de um inteiro / inteiro, é um inteiro, a parte decimal do quociente é truncada.

10/3 = o resultado é o número inteiro 3

10./3.= o resultado é o número real 3.333 ...

5) Funções

Além das operações básicas anteriormente citadas, podem-se usar nas expressões aritméticas algumas funções muito comuns na matemática, são elas:

Nome	Definição	Tipo	
		Parâmetro	Resultado
SIN (x)	Seno (radianos). Se x for complexo, a parte real é assumida como valor em radianos	Real ou complexo.	REAL*4
ASIN (x)	Arcoseno (rad.). Retoma valores na $(-\pi/2, \pi/2)$	Real, |x| ≤1	REAL*4
COS (x)	Coseno (rad.) Se x for complexo, a parte real é assumida como valor em radianos.	Real ou complexo	REAL*4
ACOS (x)	Arcocosseno (radianos) Retoma valores na faixa $(0, \pi)$	Real, |x| ≤1	REAL*4
TAN (x)	Tangente (radianos)	Real	REAL*4
ATAN (x)	Arcotangente (rad.). Retoma valores na faixa $(-\pi/2, \pi/2)$	Real	REAL*4
SINH (x)	Seno Hiperbólico (rad.)	Real	REAL*4
COSH (x)	Coseno Hiperbólico (rad.)	Real	REAL*4
TANH (x)	Tangente Hiperbólica (rad.)	Real	REAL*4
ALOG10 (x)	Logaritmo de x na base 10	Real	Real
ALOG (x)	Logaritmo neperiano de x (x > 0)	Real	Real
EXP (x)	número e (base dos logaritmos neperianos) elevado a x	Real	Real
ABS (x)	valor absoluto de x	Real	Real
IABS (x)	valor absoluto de x	Inteiro	Inteiro
IFIX (x)	Conversão de real para inteiro, truncando	Real	Inteiro
FLOAT (x)	Conversão de inteiro para real	Inteiro	Real
DBLE (x)	Converte para dupla precisão	Real	Real (dupla precisão)
CMPLX (x)	Converte para o tipo complexo	Real	Complexo
SIGN (x,y)	Fornece valor positivo de x se y ≥ 0 e negativo de x se y < 0	Real	Real
MOD (x,y)	Resto da divisão de x por y	Inteiro	Inteiro
AMOD (x,y)	Resto da divisão de x por y	Real	Real
SQRT (x)	Raiz quadrada de x (x ≥ 0)	Real	Real

a) a função atua sobre um argumento numérico que é o resultado obtido após a avaliação da expressão aritmética entre parênteses

b) existem outras funções além dessa que podem ser encontradas no manual do compilador.

EX.10:
X + SIN (A + B +C)
IFIX (A/2.) * 100 + T
X + ALOG (Y) - ABS (A+B)
H ** 2 - G * F * SIGN (1.,C + D)

Resolver em sala:

Sendo A e B variáveis inteiras e X e Y variáveis reais, quais os resultados fornecidos por cada um das funções, sendo:

A=10	P=1.8	M=6	B=3
C=D= -0.7	N=3	X=2.5	F=8
I=H=-1	Y = 1.2	G = 4.3	T = 12

a) $\sqrt{P(P-A)(P-B)(P-C)}$

b) $- A - B(C + \frac{D}{E-1} - F) + G$

c) I/I + M ** 2 - N

d)A/B + C * 4.-X

e) I/A+C*4-D

f)X + SIN (A + B +C)

g) IFIX (A/2.) * 100 + T

h)X + ALOG (Y) - ABS (A +B)

i) H ** 2 - G * F * SION (1.,C + D)

j)Quociente e resto de inteiro por inteiro

k) Quociente e resto em que os operando são reais

l) Truncamento (B^2 + X), Truncamento (A/3 + 1) e Truncamento (X - 3,2)

6) Expressões lógicas

Denomina-se expressão lógica à expressão cujos operadores são lógicos e cujos operandos são relações, variáveis e/ou constantes do tipo lógico.

6a) Relações

Uma relação é a comparação entre 2 valores do mesmo tipo básico, isto é, só se deve comparar, inteiro com inteiro, real com real e caractere com caractere.

Os operadores relacionais que indicam a comparação a ser realizada entre os termos da relação são:

.GT**.** → Maior que **.**GE**.** → Maior ou igual a **.**LT**.** → Menor que

.LE**.** → Menor ou igual a **.**EQ**.** → Igual a **.**NE**.** → Diferente de

O resultado obtido é sempre um valor lógico.

EX.11:

A **.NE.** B
NOME**. EQ.** 'JOAO'
B**2-4.*A*C.**LT.** 0
X **.EQ.** 1

X = INTEIRO
A,B,C = REAL
NOME = LITERAL

Exercício de fixação 4:

Dada as variáveis X, Y, Z e as variáveis literais COR e NOME, complete as lacunas

Variáveis					Relações		
X	Y	Z	COR	NOME	X**2+Y **.GT.** Z	COR **.EQ.** 'AZUL'	NOME **.NE.** 'JOSE'
1	2	5	'AZUL'	'PAULO'	**. FALSE.**	**. TRUE .**	**.TRUE .**
4	3	1	'VERDE'	'JOSE'			
1	1	2	'BRANCO'	'PEDRa'			
1	2	1	'AZUL'	'JOSE'			

Além das relações, temos os operadores lógicos:

.AND. = para a conjunção

.OR. = para a disjunção

.NOT. = para a negação

O valor lógico das seguintes conjunções é:

P	Q	R	S	P**.AND.**S	P**.AND.**R	Q**.AND.**S	Q**.AND.**R
.T.	.F.	.F.	.T.	.T.	.F.	.F.	.F.

O valor lógico das seguintes disjunções é:

P	Q	R	S	P**.OR.**S	P**.OR.**R	Q**.OR.**S	Q**.OR.**R
.T.	.F.	.F.	.T.	.T.	.T.	.T.	.F.

O valor lógico das seguintes negações é:

P	Q	R	S	**.NOT.**P	**.NOT.**Q	**.NOT.**S	**.NOT.**R
.T.	.F.	.F.	.T.	.F.	.T.	.F.	.F.

EX.11:

a) A+B .EQ. 0 **.AND.**C **.NE.** 1
b) TESTE **.OR.** A *C **.GT.** B
c) **.NOT.** TESTE**. AND.** COR**. EQ.** 'AZUL'

A, B, C → inteiro
TESTE → lógico
COR → literal

6b) Prioridades

Conforme visto no exemplo pode-se termais de um operador lógico em uma expressão, e conforme os valores envolvidos as operações lógicas alteram o resultado final. Assim, como acontece com as operações aritméticas, também existe um relação de prioridades entre os operadores.

Prioridade	Operador
1ª	Aritmético
2ª	Relacional
3ª	**.NOT.**
Prioridade	Operador
4ª	**.AND.**
5ª	**.OR.**

Resolver em sala:

Dadas as variáveis inteiras X, Y e Z (valendo 2, 5 e 9) e a variável literal NOME (contendo o string 'maria') e a variável lógica SIM (contendo o valor **.F.**) faça um programa que calcule as seguintes expressões.

a) X + Y **.GT.** Z **. AND.** NOME **.EQ.** 'maria'

b) SIM .OR. Y .GE. X

7) Expressões literais

Uma expressão literal é aquela formada por operadores literais e operandos que são constantes elou variáveis do tipo character.

Supondo que A e B são variáveis literais e que o símbolo // é o operador de junção (concatenação) de literais, a expressão A//E fornece como resultado um único literal formado pelo conteúdo de A seguido pelo conteúdo de B.

EX.13:
Se A contém o literal 'bola' e B contém o literal 'preta' , A//B será 'bolapreta'

8) Comando de atribuição

Este comando permite que forneça um valor a uma certa variável na qual está sendo armazenado um valor compatível com o tipo de variável. A forma geral é:

Identificador = expressão

Ex.14:

K= 1	A=B	COR = 'verde'	MEDIA = SOMA/N
TESTE = .false	TOTAL = SQRT (N) + X**2 + Y		

Exercício de fixação 5:

Fazer um programa que calcule as seguintes expressões:

a) A **EQ.** 1. **.AND.**TESTE

b) NOME **.EQ.** 'PEDRO' **.OR.** COR **.NE.** 'BRANCO'

c) C **.LT.** 10 . **.OR.** TESTE**. AND.** COR **.EQ.** 'PRETO'

d) A**2 + C** (1./3.) **.EQ.** 3 **.. AND.** (A + IFIX (B+C) **.GT.** 13. **.OR.** NOME **.EQ.** 'ANA')

e) P3 = Pl//P2

Sendo:	A, B, C = 1., 4.5 e 8.	NOME = 'tania'	COR = 'branco'
	Pl = 'boa-'	P2 = 'tarde'	TESTE =.T.

9) Comando DATA

Este comando permite que se atribuam valores iniciais às variáveis. Sua forma geral é:

DATA L1/Cl/, L2/c2/, ... ,Ln/cn/

Sendo L_i os identificadores (nomes das variáveis)

C_i os valores (constantes numéricas, lógicas e/ou literais) que serão atribuídos as variáveis.

As constante contidas em c podem ser precedidas de k*, sendo k um valor inteiro positivo que indica o número de vezes que a constante que ele precede deve ser atribuída. O comando DATA deve vir logo após as declarações e antes de qualquer outro comando.

EX.15:

```
Integer C, E
Character cor*5
DATA A, B, C, E /3.25, -0.8, 2*1/, G, H /.TRUE., .FALSE./, COR /'verde'/
```

Nesse exemplo, quando o programa começar a executar as variáveis receberão os seguintes valores:

A= 3.25	G= .T.	B= -0.8	H= .F.
C= 1	COR= 'verde'	E= 1	

10) Arquivos

Conjunto de dados, texto de programas, biblioteca de rotinas e outras informações podem ser armazenadas em meios secundários, tais como pendrive, cds, dvds, etc. As principais operações que podem ser feitas em um arquivo através de um programa são: leitura e escrita de um registro.

10a) Comando de abertura de arquivo (ascci) - comando OPEN

Inicialmente vamos aprender a abrir um arquivo. A regra geral é:

Open (**u**, file = **'nome do arquivo'**, status = **'estado'**, form = '**formato'**)

Sendo:

u = é um número inteiro, positivo que identifica a unidade de entrada ou saída;

file = é o nome do arquivo que de ser colocado entre apóstrofos

status = indica o estado do arquivo no momento da abertura, há 3 opções de status:

old → indica que o arquivo aberto já existe e contém alguma informação.

new → indica que o arquivo aberto é novo e não contém nenhuma informação.

unknown → nenhuma informação sobre o arquivo é conhecida

form = indica se o arquivo é formatado (formatted) ou não (unformatted)

EX.16:

```
Open (1, file = 'lista.txt', status = 'unknown')
```

Nesse exemplo o arquivo lista.txt é um arquivo com status desconhecido e indicado pelo número 1.

10b) Comando de leitura de arquivo (ascci) - comando READ

Um arquivo após ser aberto pode ser lido, ou escrito. O formato geral para a leitura é:

```
Read (ua,f)
```

Sendo:

ua = número que corresponde ao arquivo aberto

f = número que corresponde ao formato do arquivo

EX.17:

```
Read (1,12) variavel_1, variavel_2, ....
```

10c) Especificação de formatos de arquivo (ascci) - comando FORMAT

Como foi dito; é através do FORMAT que é descrito como os dados estão dispostos em um meio de entrada e como eles deverão ser distribuídos em um meio de saída. A regra geral é:

f FORMAT (especificações)

Sendo:

f →um número que indica o formato e que deve ser o mesmo que aparece na leitura ou escrita, e que deve ser colocada a esquerda da 6ª coluna de um programa;

Especificações é o conjunto de informações que descreve como é o formato dos dados que serão lidos/escritos. Deve-se sempre considerar ao se ler ou escrever um conjunto de dados que:

- Em um arquivo de leitura:

a) as posições deixadas em branco à direita do número são interpretadas como zero, já que o número deve ser colocado a mais à direita do campo utilizado;

b) o sinal de + para valores positivos pode ser omitido;

c) os zeros colocados a esquerda de um número são desprezados;

- Em um arquivo de escrita:

a) se o número for negativo o sinal - aparecerá na resposta;

b) se o numero for positivo nenhum sinal é colocado na resposta

Chama-se de letras de edição aquelas utilizadas nas especificações dos formatos. Estas letras de edição variam conforme o tipo da variável utilizada.

a) Iw

A letra de edição I especifica como os valores inteiros aparecem nos arquivos de entrada ou saída e w é um numero que indica quantas posições de memória serão utilizadas.

Ex.18:
O inteiro 2 corresponde a declaração i1.
O inteiro -2 corresponde a declaração i2.

Obs. Se a declaração de saída tiver menos posições que a necessária, o campo será substituído por ***. Ou seja, se na declaração de -2 tivesse sido declarado o formato como i1, a resposta que apareceria seria *** ao invés do número, se isso tivesse acontecido na leitura seria lido apenas o número 2.

b) Fw.d

A letra de edição F especifica como os valores reais aparecem nos arquivos de entrada ou saída, w é um numero que indica quantas posições de memória serão utilizadas e d indica quantas casas decimais serão consideradas.

Ex.19:
O real 2.5 corresponde a declaração F3.1
O real -1.82 corresponde a declaração F5.2

c) Ew.d

A letra de edição E especifica como os valores reais com expoente decimal aparecem nos arquivos de entrada ou saída, w é um numero que indica quantas posições de memória serão utilizadas e d indica quantas casas decimais serão consideradas.

Ex.20:
O real 25.346718E-03 corresponde a declaração E13.6

d)Lw

A letra de edição L especifica como os valores lógicos aparecem nos arquivos de entrada ou saída, w é um numero que indica quantas posições de memória serão utilizadas.

Ex.21:

Se a resposta de uma expressão estiver contida na variável lógica ano e esta for verdadeira, a resposta que aparecerá na tela será T.

e) Aw

A letra de edição a especifica como os caracteres aparecem nos arquivos de entrada ou saída, w é um numero que indica quantas posições de memória serão utilizadas.

Obs.:

a)O número de caracteres que uma variável pode conter é indicado na declaração do programa.

b) Se o valor de w atribuído no FORMAT for maior que o valor indicado na declaração, só serão lidos os caracteres indicados na declaração, sempre a direita do string.

c) Se o valor for menor, serão incluídos valores em branco.

Ex.22:
Character nome*7
O character nome corresponde a declaração a7.

Obs: Se colocarmos a5S, por exemplo, ele lerá ou escreverá 'riana'. Se colocarmos a9, ele lerá ou escreverá ' mariana'.

f) wx

A letra de edição x especifica como os espaços em branco aparecem nos arquivos de entrada ou saída, w é um numero que indica quantas posições de memória serão utilizadas.

Ex.23:
Suponha que queremos ler o seguinte registro:

1	2		3	5	.	6	

O formato de leitura seria:
Format(i2,1x,f4.1,1x)

Exercício de fixação 6:

Qual o fomato dos seguintes arquivos? Considere ~~b~~ como sendo 1 espaço em branco:
a) ana ~~b b b~~ 2.5~~ b~~ 2.5~~ b~~ 2~~ b~~ 3

b) 2.8␣ 1.0␣␣1.␣ -2.8␣ marcia
c) ana␣ maria␣ 20␣ anos␣ secretaria

Resolver em sala

Fazer um programa que leia o raio R a altura H de um cone reto e calcule a área lateral, a área total e o volume do cone.

Geratriz $\rightarrow$ G = $\sqrt{H^2 + R^2}$

Área lateral $\rightarrow$ AL = 2 x π x R x G

Área total $\rightarrow$ AL +π x R^2

Volume $\rightarrow$ V= $\frac{\pi x R^2 x H}{3}$

Use R = 3.0 e H = 4.0

11) Estrutura Condicional Simples

Texto de um algoritmo:

Se condição B

Então sequência A de comandos

Fim Se

Algoritmo convertido para linguagem fortran:

If (condição B) **Then**

Sequência A de comandos

Endif

Ex. 24:
Sendo x =1.0, y =2.0; x, y, soma = reais, avalie o valor da soma se x for maior que y:

```
real x,y,soma
soma = 0.0
if( x.ge.y) then
    soma=x
endif
```

* nesse caso soma igual a 0.0, pois x não é maior que y.

12) Estrutura Condicional Composta

Texto de um algoritmo:

Se condição B Então

sequência A de comandos

Senão

sequência B de comandos

Fim Se

Algoritmo convertido para linguagem fortran:

If (condição B) **then**

Sequência A de comandos

else

Sequência b de comandos

endif

Ex. 25:
Sendo x =1.0, y =2.0, x, y, soma =reais, avalie o valor de soma se x for maior que y.

```
real x, y, soma
soma = 0.0
if (x .ge.y) then
      soma =x
else
      soma = y
endif
```

* nesse caso soma igual a 2.0, pois x não é maior que y.

Exercício de fixação 7:

Fazer um programa que leia três valores inteiros, determine e imprima o menor deles.

13) Estrutura de Repetição:

Uma estrutura de repetição permite que uma sequência de comandos se repita até determinada condição de parada ser satisfeita.

Existem três formas de interrupção:

1) interrupção no início

Há duas maneira de se fazer a interrupção no início, uma é usando os comandos *if, go to e continue*; a outra é usando *do – enddo*.

a) 1º Modo:

```
r1    if (condição A) go to r2
            sequência de comandos B
             go to r1
r2    continue
```

Ex.26:

```
Programa para somar valores pares de 100 até 200
Soma = 0
Par = 100
10    if (par. GT. 200) go to 20
          soma = soma +par
          par = par +2
          go to 10
20 continue
```

b) 2º Modo:

O fortran possui uma única estrutura de repetição: é o comando **DO**.

```
Do var = nl, n2, n3
    Sequência de comandos
enddo
```

sendo: n1 = valor inicial ; n2 = valor final; n3 = incremento, se este for omitido é usado 1 como incremento.

Ex.27

```
Programa para somar valores pares de 100 até 200
    Soma = 0
    Do par =100,200,2
       soma = soma +par
    enddo
```

2) interrupção no meio

```
r1    continue
        sequência de comandos A
if (condição A) go to r2
        sequência de comandos B
go to r1
r2    continue
```

Ex.28.

```
Programa para somar valores pares de 100 até 200.
      Soma = 0
      Par = 98
10    continue
      par = par +2
```

```
      if (par. GT. 200) go to 20
      soma = soma +par
      go to 10
20    continue
```

3) interrupção no fim

```
r1    continue
      sequência de comandos A
      if (condição A) go to r1
```

Ex.29.

Programa para somar valores pares de 100 até 200

```
      Soma = 0
      Par = 100
10    continue
      soma = soma +par
      par = par +2
      if (par .LE. 200) go to 10
```

Exercício de fixação 8:

Uma pesquisa coletou algumas características físicas de uma população, como:
Sexo (masculino ou feminino); Cor dos olhos (azul, verde, castanho ou preto); Idade;
Para cada habitante foi digitada uma linha com os correspondentes dados (total de habitantes: 30). Fazer um programa que escreva:
Porcentagem de indivíduos de cada sexo.
Quantos homens têm olhos verdes.
Quantas mulheres têm idade superior a 18 anos.
Quantos indivíduos têm olhos pretos e menos de 30 anos

14) Comando write.

write (u, f)

sendo: u → número que indica arquivo aberto para ser escrito,

f →número que identifica o formato dos dados de saída.

Ex.30:

```
      Open (1, file='dados.txt', form='formatted', status='old')
      Write (1,12) ano, dia, chuva
12    format (i4, 2x, i2, 2x, f6.1)
```

15) Leitura de arquivo com número desconhecido de linhas

read (u, f, end = n)

sendo: u → número que indica arquivo aberto para ser lido (o mesmo do arquivo open),

f → número que identifica o formato dos dados de entrada.

n → número que identifica o comando que finaliza a leitura.

Ex. 31

```
        real letra, letra2
        open (1, file = 'entrada.txt')
        do i =1, 100
        read (1,12,end = 19)letra
12      format (f6.1)
        letra2=letra2+letra
19      continue
        enddo
        print *, letra2
        end
```

16) Comando rewind

Este comando posiciona o acesso ao primeiro registro de um arquivo sequencial, isto é, a próxima leitura ou escrita será realizada no primeiro registro do arquivo.

Ex. 32

```
        open (1, file = 'entrada.txt')
        do i =1, 100
        read (1 ,12,end = 19)letra
12      format (f6.1)
        ic=ic+ 1
        continue
19      enddo
        rewind (1)
         do i =1,ic
         print *, letra
        enddo
```

Exercício de fixação 9:

A solução x, y para o sistema de equações lineares abaixo:

$$\begin{cases} ax + by = u \\ cx + dy = v \end{cases}$$

É dada por:

$$x = \frac{d}{ab - bc}u - \frac{b}{ad - bc}v$$

$$y = \frac{-c}{ad - bc}u + \frac{a}{ad - bc}v$$

Escrever um programa que:

Leia várias linhas, onde cada linha contém os parâmetros a, b, c, d, u e v do sistema. Calcule a solução x, y da cada sistema dado por seus parâmetros; Escreva os parâmetros lidos c os valores calculados.

17) Variáveis compostas

Variáveis compostas correspondem a posições de memória, identificadas por um único nome, individualizadas por índices e cujo conteúdo é de um mesmo tipo.

Para referenciar um elemento em uma variável composta, é necessário colocar-se o nome da variável, seguida entre parênteses, de um ou mais índices.

Ex.33

Supondo-se que as notas de 10 alunos estejam armazenadas em uma variável composta, identificada por nota,

60	70	90	60	55	91	99	47	74	86

Para referenciar o terceiro elemento deste conjunto pode-se escrever: nota (3) e o conteúdo armazenado nesta posição é 90.

Em um programa a declaração e leitura dessa variável seriam:

integer nota(10) → na declaração é necessário colocar o total de valores que a variável vai receber

Ex.34

```
do i = 1,10
    read(1, 12) nota (i) →no código coloca-se apenas o índice
enddo
```

CUIDADO!!! NA DECLARAÇÃO NÃO SE DEVE COLOCAR ÍNDICE, NO CÓDIGO NÃO DEVE SE COLOCAR O VALOR TOTAL.

Exercício de fixação 10:

Usando o conceito de variável composta resolva o seguinte problema:
Numa fábrica trabalham homens e mulheres divididos entre as classes:

A→ os que produzem até 30 peças por mês / B → os que produzem de 31 a 35 peças por mês / C → os que produzem mais de 35 peças por mês.

A classe A recebe salário - mínimo. / A classe B recebe salário - mínimo e mais 3% do salário mínimo por peça, acima das 30 iniciais. / A classe C recebe salário - mínimo e mais 5% do salário - mínimo por peça, acima das 35 iniciais.

Fazer um programa que leia as várias linhas, contendo cada uma:

nº do operário, nº de peças fabricadas por mês, sexo do operário.

Calcule e escreva:

1) o salário de cada operário

2) total da folha mensal de pagamento da fábrica

3) número total de peças fabricadas por mês

4) número do operário de maior salário

Extra:

1) médias de peças fabricadas pelos homens em cada classe

2) médias de peças fabricadas pelas mulheres em cada classe

18) Variáveis compostas com duas dimensões.

Variáveis compostas podem ter várias dimensões, os compiladores em geral aceitam até 7. Nesse exemplo vamos estudar variáveis com dois índices, como uma matriz.

Ex.35

Supondo-se que as notas de 3 turmas, de 4 alunos cada, estejam armazenadas em uma variável composta bidimensional, identificada pela variável nota:

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

```
integer nota(4,3) → 4 linhas e 3 colunas
do i = I,4
      read(1,12) (nota (i,j),j=1,3)
enddo
```

Exercício de fixação 11:

Ler um arquivo de dados e para cada coluna (1 a 3) calcular a média e o desvio padrão:

$$med = \frac{\sum_{i}^{n} n}{n} \text{ e } \sigma = \sqrt{\frac{1}{n-1}(x_i - med)^2}$$

19) Modularização

As ferramentas de modularização utilizadas são as sub-rotinas ou as funções. A Subrotina permite a construção de módulos subordinados a um programa principal. Um subprograma possui suas variáveis e comandos próprios, e para ser executado, deve ser ativado por um programa principal ou outro subprograma. A criação de um subprograma é

feita através de sua declaração, que é constituída de um cabeçalho e de um corpo. Para ativá-lo utiliza-se o comando **call** nome (lista de parâmetros).

Sendo: Call → uma palavra chave

Nome → é o nome da subrotina

Lista de parâmetros → são os parâmetros atuais que irão substituir os parâmetros formais na execução da subrotina.

Ex. 36:

```
      real 1, m, n
      open (1 ,file='entrada. txt' ,status='unknown')
      read (l,2) 1, m, n
2     format (3f8.1)
      if (l .gt. m .or. 1 .gt. n)then
          if (m .11. n)then
             call troca (1, m)
          else
            call troca (I, n)
          endif
      endif
      if (m .gt. n) call troca (m, n)
      write (* ,2) 1, m, n
      end
      subroutine troca(a,b) → depois do end do programa principal
      real a, b, aux → tem que fazer as declarações novamente
      aux = a
      a=b
      b = aux
      return → tem que ter este comando antes do end
      end → tem que ter end na subrotina
```

Exercício de fixação 12:

Usando o conceito de variável composta resolva o seguinte problema:

Fazer um programa que leia uma lista que contém o número do funcionário, a tarefa que executou e o tempo em segundos que levou para executar a tarefa. A seguir, criar uma subrotina que transforme o valor lido em segundos, em hora, minuto e segundo. Escrever a resposta num arquivo.

RESUMO DAS FUNÇÕES BÁSICAS

O texto de um programa escrito em fortran:

Comentário → c (na primeira coluna) ou ! (em qualquer coluna)

Continuação de linha → qualquer caracter na coluna 6

Texto do programa → entre as colunas 1 e 72 exceto para número de comandos

Número de comandos →colunas 1 a 5

Tipos de constantes ou variáveis:

Real → qualquer número com casa decimal (ponto)

Integer → qualquer número sem casa decimal

Character → qualquer palavra ou número lido como texto

Logical → qualquer variável ou constante que receba um valor lógico (verdadeiro ou falso)

Declaração de variáveis (ou constantes) simples

Real var 1, var2, var3

Integer var4,varS,var6

Character var7*nº (sendo n° o número de caracteres que a variável ou constante vai receber)

Logical var8,var9

Expressões aritméticas em ordem de prioridade (resultado numérico)

(1) Exponenciação →* *

(2) Divisão → /

(2) Multiplicação → *

(3) Soma → +

(3) Subtração → -

Expressões e operadores lógicos (resultado sempre lógico)

maior que (>) → .GT.

maior ou igual a (≥) → .GE.

menor que(<) → .LT.

menor ou igual a (≤)→ .LE.

igual a (=)→ .EQ.

diferente de (≠)→ .NE.

conjunção → .AND.

disjunção → .OR.

negação → .NOT.

Prioridades

(1) aritmético

(2) relacional

(3) .NOT.

(4) .AND.

(5) .OR.

Comando de abertura de arquivo → OPEN

Open (nº, file = 'nome', status = 'new', ou 'unknown' ou 'old')

nº → qualquer número inteiro

nome → nome do arquivo de entrada

Comando de leitura de arquivo → READ

Read (nº,f) var1, var2, ...

nº → mesmo número inteiro colocado no arquivo de entrada (open)

f → número que corresponde ao formato do arquivo de entrada

Comando de escrita de arquivo → WRITE

Write (nº,f)var1, var2, ...

nº → mesmo número inteiro colocado no arquivo de saída (open)

f →número que corresponde ao formato do arquivo de saída

obs.: para usar o write é necessário antes abrir um novo arquivo utilizando o open

Especificação de formatos de arquivo → FORMA T

Integer →IW (ex.: 100 → 13)

Real → FW.d (ex.: 100.3 → F5.1)

Exponencial → EW.d (ex.: 25.34E-03 → E9.2)

Character → aw (ex.: lucia →a5)

Espaços em branco →wx (ex.: _ _ _ → 3x)

Sendo W um número que indica quantas posições de memória serão utilizadas e d indica quantas casas decimais serão consideradas.

Estrutura Condicional Simples→ comando if , endif

```
If (condição B) Then
      Sequência A de comandos
      Sequência B de comandos
Endif
```

Estrutura Condicional Composta → comando if, else, endif

```
If (condição B) Then
      Sequência A de comandos
else
      Sequência B de comandos
Endif
```

Estrutura de Repeticão com interrupção no início → comando if, go to, continue

```
r1    if (condição A) go to r2
      sequência de comandos B
      go to rI
r2    continue
```

Estrutura de Repetição com interrupção no meio→ comando ir, go to, continue

```
r1    continue
      sequência de comandos A
      if (condição A) go to r2
      sequência de comandos B
      go to r1
r2    continue
```

Estrutura de Repetição com interrupção no fim → comando ir, go to, continue

```
r1    continue
      sequência de comandos A
      if (condição A) go to r1
```

Estrutura de Repetição → comando do, enddo

```
do i=n1,n2,n3
      Sequência de comandos
enddo
sendo: n1 = valor inicial
         n2 = valor final
          n3 = incremento, se este for omitido é usado 1 como incremento
```

Leitura de arquivo com número desconhecido de linhas

```
      Do i = 1, 30
      Read(u, r, end = n) var1,var2
      Sequência de vários comandos n
n     continue
      enddo
```

sendo: u → número que indica arquivo aberto para ser lido (o mesmo do arquivo open),

f → número que identifica o formato dos dados de entrada.

n →número que identifica o comando que finaliza a leitura.

Variáveis compostas unidimensional

integer nota(10) → na declaração coloca-se o total de valores que a variável vai receber

do i = 1,10

read(1,12) nota (i) → no código do programa coloca-se apenas o índice

enddo

Variáveis compostas multidimensionais

integer mat (5,3) → na declaração coloca-se o total de valores que a variável vai receber

do i = 1,5

read(l,12) (mat (i,j),j = 1, 3) → no código do programa coloca-se apenas o índice

enddo

Modularização→ comando subroutine, call

if (1 .gt. m .or. 1 .gt. n) then

call troca (I, m)

end

subroutine troca(a,b) → depois do end do programa principal

real a, b, aux → declarações

sequencia de comandos

return → tem que ter este comando antes do end

end → tem que ter end na subrotina