UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

Sumário

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

APOSTILA DE FORTRAN

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

Capítulo 1: Noções Preliminares

1.1. Introdução

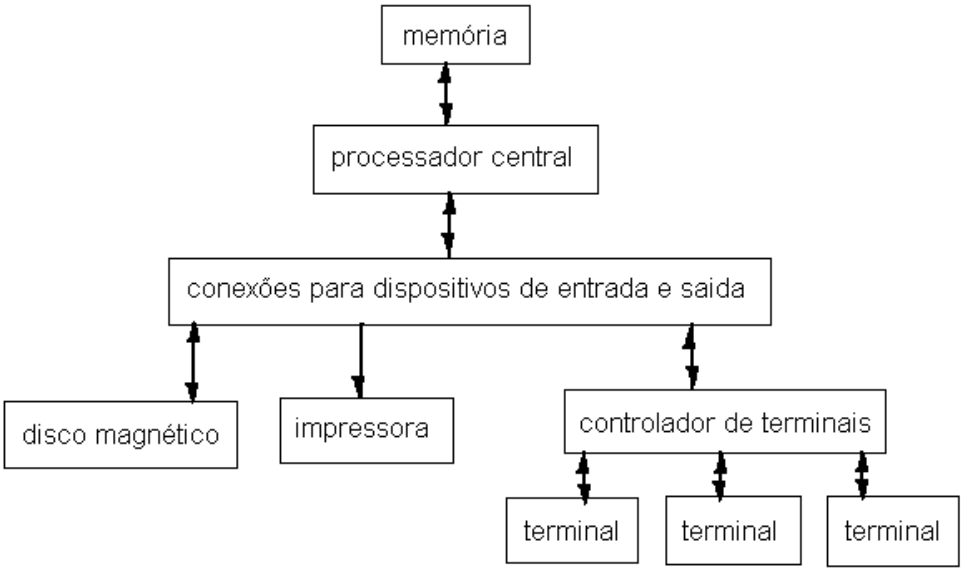
- "O que todo computador pode realmente fazer é seguir ordens muito simples, as quais foram cuidadosamente consideradas e refletidas por um programador e escritas em uma linguagem de programação, como o Fortran."

1.2. Computadores

- Há somente 4 tipos de instruções que um computador pode realizar:

1. Armazenamento e recuperação de informações da memória;
2. Cálculos;
3. Entrada e saída de dados;
4. Controle de programa.

- Esquema de um computador:



1.3. Algoritmos

- Um "algoritmo" é um conjunto de instruções passo-a-passo para resolver um problema.

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

- Um algoritmo correto deve possuir 3 qualidades:

1. Cada passo no algoritmo deve ser uma instrução que possa ser realizada.
2. A ordem dos passos deve ser precisamente determinada.
3. O algoritmo deve ter fim.

- Elaboração em grupos de 3 de um algoritmo para que um dos elementos do grupo desempenhe a seguinte tarefa:

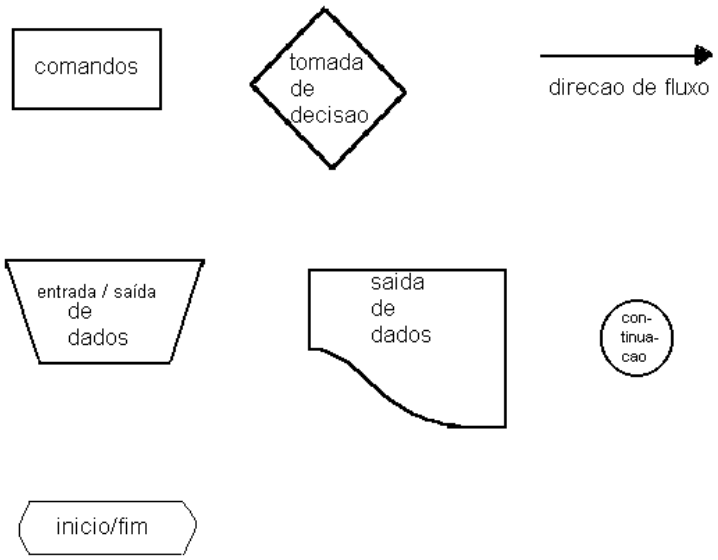
- Sair de uma posição na sala de aula, que será determinada pelo professor, até o interruptor das luzes e providenciar o desligamento destas.

As operações corporais possíveis devem ser determinadas antes do começo do trabalho em grupo, em conjunto entre os alunos e o professor. Após o término do trabalho dos grupos, o professor escolherá aleatoriamente trabalhos, os lerá em voz alta, determinando se os passos do algoritmo estão contidos no conjunto de operações permitidas. Se isto for verdade, um dos elementos do grupo lerá em voz alta os passos que serão executados por outro membro do grupo. Sugestões de operações permitidas: caminhar, girar o corpo (graus e sentido), levantamento de braço (graus e sentido), movimento de mão, cegueira do executor, etc.

1.4. Diagramas de Fluxo

- "Diagramas de Fluxo" fornecem uma representação gráfica de um algoritmo.

- Principais símbolos de diagramas de fluxo:



UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

1.5. Linguagens de Programação

- Um "programa de computador" é um algoritmo escrito numa linguagem de programação, como o Fortran.

- Existem 3 níveis de linguagens de programação:

1. Linguagens de baixo nível (ou de máquina).
2. Linguagens de nível intermediário (ou simbólica montadora).
3. Linguagens de alto nível (ou de compiladores).

1.5.1 Linguagem de Máquina

- Cada instrução é constituída de 2 partes:

código de operação	operando(s)

- O "código de operação" instrui o computador para realizar uma função específica: adição, subtração, leitura, escrita, comparação, movimento de dados, etc...

- O(s) operando(s) especifica(m) o endereço do dado a ser operado, endereço do dispositivo de entrada/saída, etc...

- Exemplo do trecho principal de um programa em linguagem de máquina para calcular

$$R = \frac{5A+16B}{C} - D$$

onde os valores de A, B, C, D e R estão armazenados nos endereços 1000, 1004, 1012, 1020 e 2050 da memória.

Código da Operação	Operando	Comentário
14	1000	carrega A no acumulador
12	=5	multiplica o acumulador por 5
15	3000	armazena conteúdo acumulado no endereço 3000
14	1004	carrega B no acumulador
12	=16	multiplica acumulador por 16
10	3000	adiciona acumulador com conteúdo de 3000
13	1012	divide acumulador pelo conteúdo de 1012
11	1020	subtrai do acumulador o conteúdo de 1020
15	2050	armazena acumulador em 2050

- Na realidade, um programa real em linguagem de máquina, para ser compreendido por ela, geralmente é escrito no sistema binário.

APOSTILA DE FORTRAN

1.5.2. Linguagem Simbólica Montadora

- As linguagens simbólicas procuram facilitar a tarefa do programador, criando mnemônicos para as operações.

- Os programas montadores traduzem um programa fonte (ou seja, em linguagem simbólica) em um programa objeto (ou seja, em linguagem de máquina).

- O trecho de programa visto acima, ficaria em uma linguagem simbólica parecido com:

Operação	Operando	Comentário
CAR	A	carrega A
MUL	5	multiplica por 5
ARM	TMP	armazena o resultado em TMP
CAR	B	carrega B
MUL	16	multiplica por 16
ADI	TMP	adiciona o resultado com o conteúdo de TMP
DIV	C	divide o resultado por C
SUB	D	subtrai o resultado de D
ARM	R	armazena o resultado em R

1.5.3. Linguagens de Compiladores

- Uma linguagem de compilador (ou de alto nível) tem por objetivo permitir ao programador se utilizar de uma linguagem muito próxima àquela utilizada no ambiente no qual se coloca a tarefa a ser realizada.

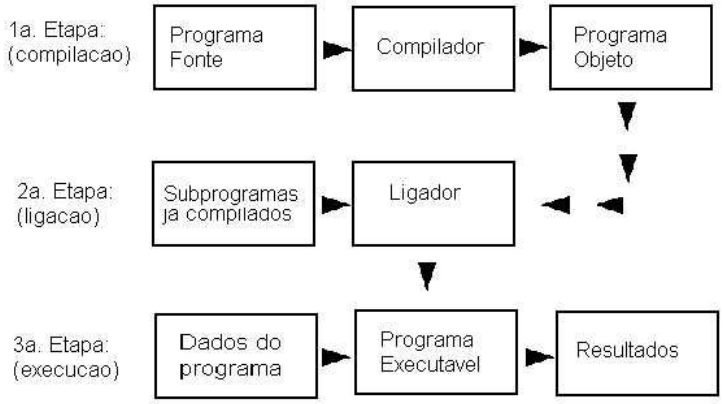
- Os programas compiladores traduzem um programa fonte, escritos em uma linguagem de alto nível, em um programa objeto em linguagem de máquina.

- O trecho de programa visto acima, ficaria na linguagem de alto nível Fortran idêntico a:

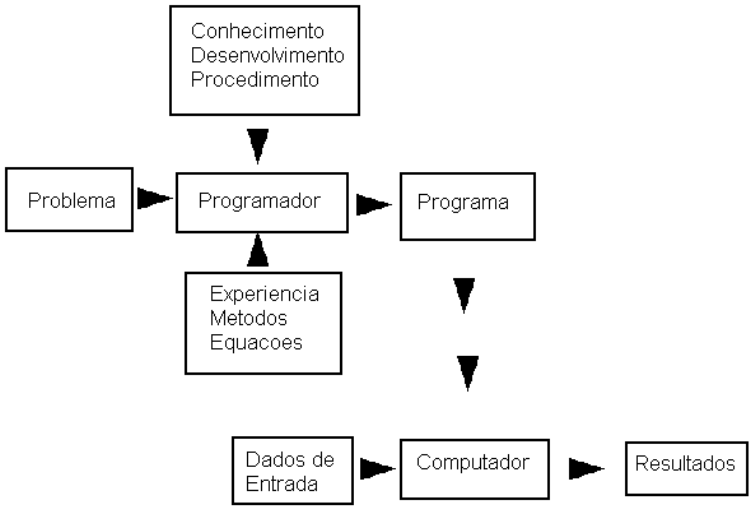
```
R = (5.0*A + 16.0*B) / C - D
```

- Etapas do processamento de um programa em linguagem de alto nível:

APOSTILA DE FORTRAN



1.6.Passos no Desenvolvimento de Programas



UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

Capitulo 2 - A Linguagem Fortran: Conceitos Básicos

2.1 Introdução

- FORTRAN = FORmula TRANslation (em português: tradução de fórmulas).

- O Fortran é uma linguagem simples cujo vocabulário consiste de um conjunto de palavras, conhecidas como "palavras-chave" (para se ter uma idéia, no Fortran77 eram cerca de 450; atualmente o número é bem maior).

- Um "comando" Fortran é uma sentença escrita nesta linguagem. Um programa consiste numa seqüência de comandos, cada um escrito numa linha.

2.2. Definições da Linguagem

- *Programa Principal* é uma unidade que pode chamar outras unidades de programa, mas que não pode ser chamado por estas. O programa principal é quem recebe o controle da execução no início da fase de execução.

- *Subprograma* é uma unidade de programa que é chamada por outra unidade de programa. Pode ser de dois tipos: subrotina ou função. Usualmente um subprograma recebe parâmetros de entrada e retorna parâmetros de saída.

- *Função Intrínseca* é uma função suprida pelo compilador e que realiza operações sobre números ou caracteres.

- *Programa Executável* é um programa já em linguagem de máquina que pode ser executado pelo computador. Consiste de um programa principal e subprogramas.

- *Comando Executável* é aquele que efetua cálculos, testes, providencia entrada ou saída de dados, altera o fluxo de execução ou atribui valores a variáveis.

- *Comando Não-Executável* é aquele que descreve as características de uma unidade de programa, dos dados ou de informações de edição.

- *Unidade de Programa* é uma seqüência de comandos terminando com um comando END.

- *Arquivos de Dados* são unidades de entrada e saída de dados, passíveis de serem lidos/escritos por um programa.

2.3. Itens sintáticos do Fortran

- *Sintaxe* é a parte da gramática que estuda a disposição das palavras na frase e a das frases no discurso.

- *Constantes* são valores ou números que ocorrem num programa.

- *Nome simbólico* é uma seqüência de uma ou mais letras ou dígitos, o primeiro dos quais deve ser uma letra.

APOSTILA DE FORTRAN

- *Variáveis* são valores ou números ou conjuntos de caracteres que ocorrem num programa e que podem sofrer variações durante a execução do programa. Na realidade uma variável é o nome de uma localização da memória.

- *Número de comando* é um número de um a cinco dígitos, um dos quais deve ser diferente de zero e é usado para identificar um comando. Deve estar localizado no início da linha.

- *Palavra chave* é uma seqüência de letras que identificam exclusivamente comandos Fortran.

- *Operador* é um ente similar a um operador matemático.

- Valores são obtidos da memória quando eles são usados no lado direito do sinal de igual.

- Observe que o sinal de "=" não indica uma igualdade e sim uma atribuição.

- O Fortran permite que você defina diferentes tipos de variáveis. Usa-se variáveis inteiras para armazenar números inteiros, variáveis reais para números reais, variáveis complexas para números complexos, variáveis caracter para palavras e frases, etc.

- Você pode definir os tipos de variáveis no seu programa através de "comandos de definição de tipo".

- Existem vários comandos Fortran para a definição de tipo de variável, entre os quais "CHARACTER", "INTEGER" e "REAL".

- Os cálculos podem ser feitos através de comandos de atribuição, escrevendo-se uma "expressão" no lado direito do sinal de igual.

- Símbolos de operações matemáticas (também conhecidos como "operadores aritméticos"): adição, "+"; subtração, "-"; multiplicação, "*"; divisão, "/"; potenciação, "**".

- Pode-se usar parênteses para agrupar termos em expressões Fortran, como é feito em álgebra.

2.4. Caracteres usados no Fortran

- O conjunto de caracteres Fortran é constituído de: letras (ou caracteres alfabéticos), dígitos (ou caracteres numéricos) e símbolos especiais como =, +, -, /, *, etc.

- É preciso tomar cuidado em relação aos símbolos especiais, pois estes podem variar de função dependendo do fabricante do compilador.

2.5. Uma visão geral do Fortran

- Exemplo de um programa para calcular a média das notas de uma prova de uma turma e determinar o número de notas acima e abaixo de 5,0. Vamos usar a equação abaixo para calcular a média da turma:

APOSTILA DE FORTRAN

$$MT = \frac{\sum_{i=1}^{NT} nota(i)}{NT},$$

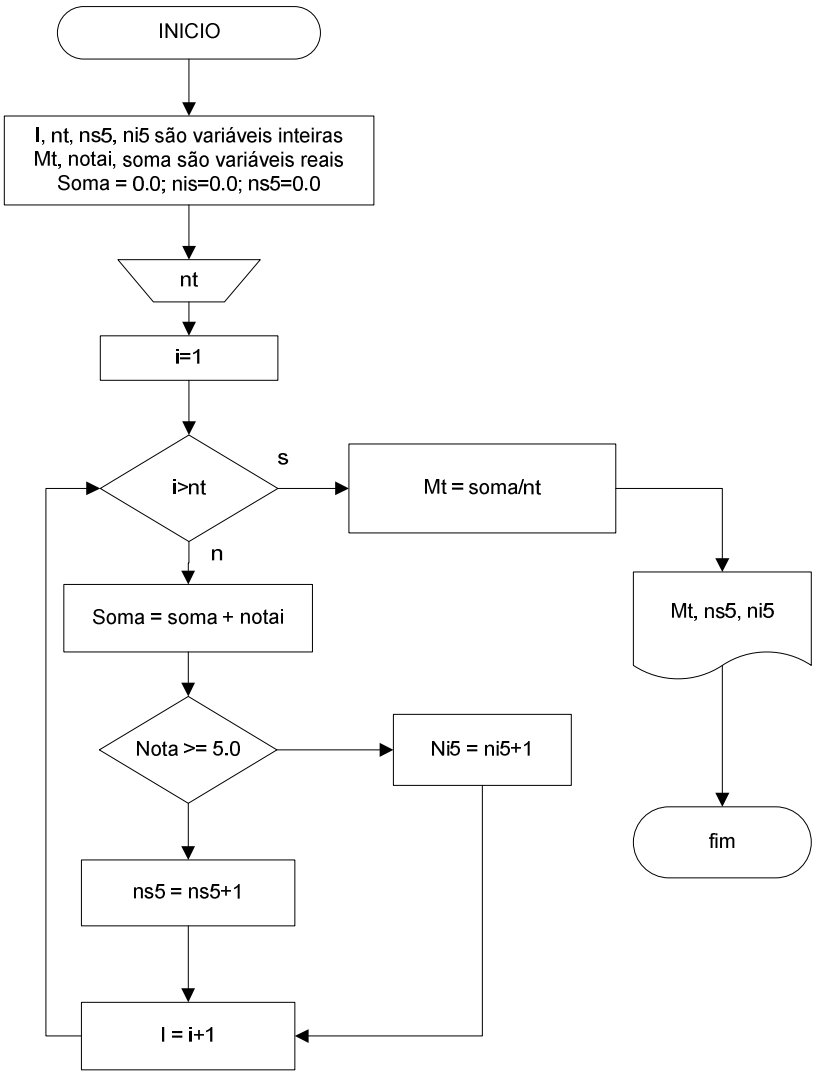
sendo que: MT = média da turma,
nota(i) = nota do aluno i,
NT = número total de alunos.

O código do programa poderia ser:

```
   program media
! programa para calcular a media de uma turma
! programador: FCLavarda
   implicit none
   integer i,nt,ns5,ni5
   real mt,nota,soma
   soma=0.0
   ns5=0
   ni5=0
   print *,'Digite o numero total de alunos: '
   read *, nt
   do i=1,nt
      print *,'Digite a nota do aluno ',i
      read *,nota
      soma=soma+nota
      if(nota>=5.0) then
         ns5=ns5+1
      else
         ni5=ni5+1
      endif
   enddo
   mt=soma/nt
   print *,'*** Resultados ***'
   print *,'Media da turma: ',mt
   print *,'Notas superiores ou iguais a 5.0 = ',ns5
   print *,'Notas inferiores a 5.0 = ',ni5
   stop
   end
```

- Veja figura a seguir contendo o fluxograma para o programa media.f90.

APOSTILA DE FORTRAN



- Uma "variável" Fortran é um nome para uma localização de memória. Variáveis no exemplo acima: nt, ns5, ni5, mt, nota, soma, i.

- O "valor" de uma variável é a informação armazenada na localização de memória que é representada pela variável.

- Um "comando de atribuição" é o tipo de comando que permite armazenar um valor em uma variável. Comandos de atribuição no exemplo acima:

soma=0.0
ns5=0
ni5=0
soma=soma+nota
ns5=ns5+1
ni5=ni5+1
mt=soma/nt

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

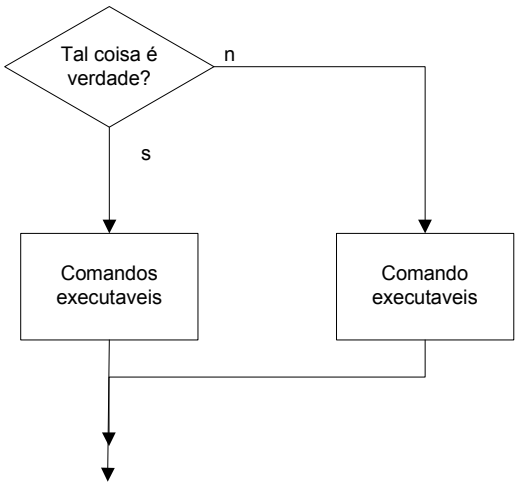
APOSTILA DE FORTRAN

- Quando você atribui um valor a uma variável, o valor anteriormente armazenado é destruído.

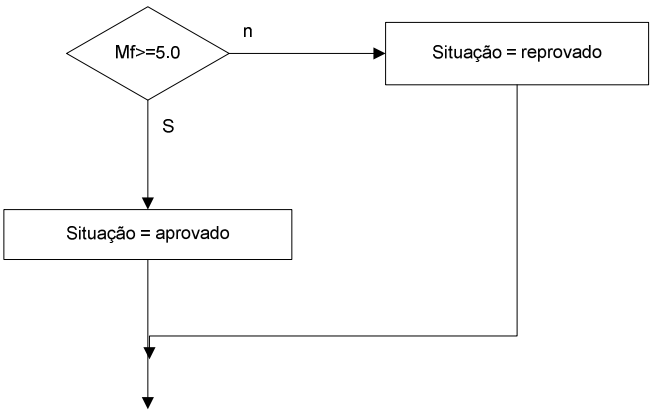
- Os comandos num programa Fortran são, normalmente, executados na ordem em que são escritos. Entretanto, pode-se alterar esta ordem de execução através dos "comandos de controle de fluxo".

- Um comando de controle que representa uma decisão é a construção "IF-THEN-ELSE", que é sempre terminado por um comando ENDIF.

- Veja figura a seguir para uma tomada de decisão e desvio do fluxo de execução.



- Veja figura a seguir para o fluxograma para o algoritmo que decide aprovar ou reprovar um aluno, baseado na sua média final.



- Comandos Fortran para o fluxograma anterior:

```
if(mf>=5.0)then
   situacao='aprovado'
else
   situacao='reprovado'
```

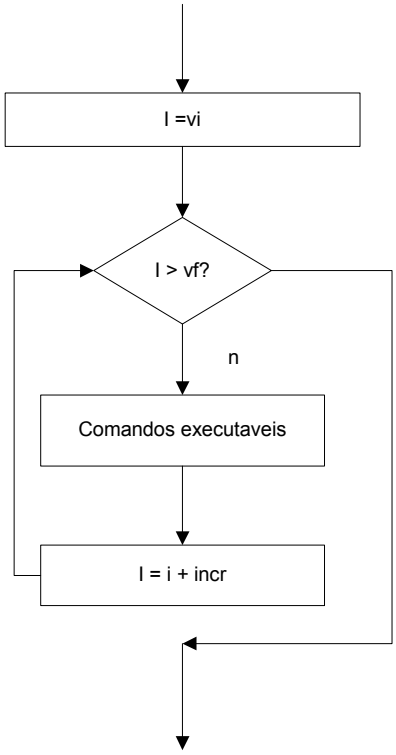
APOSTILA DE FORTRAN

endif

- Um comando de controle que representa um laço é o comando "DO", que é sempre terminado por um comando ENDDO.

- Em Fortran77, um comando DO pode terminar por um comando numerado.

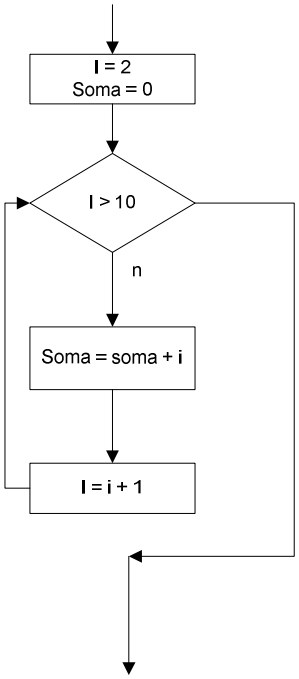
- Veja a seguir um fluxograma de um laço de repetição.



- Veja a seguir o fluxograma para o algoritmo para somar todos os números naturais de 1 a 10.

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

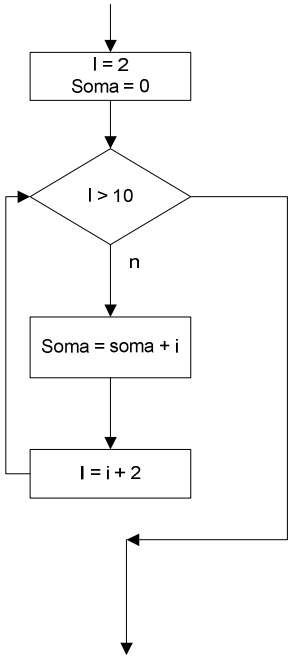


- Comandos Fortran para o fluxograma anterior:

```
soma=0
 do i=1,10,1
    soma=soma+i
 enddo
```

- Veja a seguir o fluxograma para o algoritmo para somar todos os números naturais pares de 1 a 10.

APOSTILA DE FORTRAN



- Comandos Fortran para o fluxograma anterior:

```
soma=0
do i=2,10,2
   soma=soma+i
enddo
```

- Para dizer ao computador para parar a execução de seu programa, você usa o comando "STOP".

- A última linha em qualquer unidade de programa Fortran é sempre o comando "END".

2.6. Codificação de programas em Fortran

2.6.1. Formato Livre (Fortran90 e posteriores)

- Cada comando é escrito numa linha.

- Cada linha de comando possui até 132 colunas. O editor de texto é capaz de trabalhar com um número maior de colunas (cuidado!).

- Continuação de comandos: muitas vezes um comando não cabe nas colunas disponíveis em uma linha (ou não se quer escrever tudo numa linha). Uma "linha de continuação" pode então ser definida simplesmente colocando-se um caractere & no final da linha e continua-se na de baixo. Pode-se fazer isto até 39 vezes seguidas, de modo que um comando pode ter até 40 linhas. Como sugestão de boa programação, as colunas [1,5] podem ficar em branco.

APOSTILA DE FORTRAN

- Para a codificação de arquivos de dados de entrada e saída, estes seguem a estrutura que porventura seja definida no próprio programa que vai ler/escrever neste arquivo.

2.6.2. Formato Fixo (Fortran77 e anteriores)

- Cada comando é escrito numa linha e cada linha, por sua vez, é dividida em "campos", que são conjuntos de colunas.

- Cada linha de comando possui 80 colunas. O editor de texto é capaz de trabalhar com um número maior de colunas (cuidado!).

- Cada linha é dividida em 4 campos, que são formados pelos conjuntos de colunas [1,5], [6], [7,72] e [73,80].

- *Campo de números de comando*, colunas [1,5]: um número de comando (qualquer número inteiro e maior que zero, consistindo de 5 dígitos) pode ser escrito nas colunas de 1 a 5. Brancos e zeros à esquerda são ignorados.

- *Campo de continuação de comandos*, coluna [6]: muitas vezes um comando não cabe nas 66 colunas disponíveis em uma linha. Uma "linha de continuação" pode então ser definida simplesmente colocando-se um caractere qualquer diferente de branco e zero na coluna 6. Pode-se fazer isto até 19 vezes seguidas, de modo que um comando pode ter até 20 linhas. As colunas [1,5] devem ficar em branco.

- *Campo de comandos*, colunas [7,72]: somente nestas colunas pode-se escrever um comando Fortran.

- *Campo de identificação*, colunas [73,80]: este campo é usado somente para identificação do programa. Isto é uma herança muito velha, pouco usada hoje. Geralmente o compilador que ainda se importa com a identificação do programa, escreve os primeiros 8 caracteres do nome do programa nestas colunas.

- *Linhas de comentário*, coluna [1]: qualquer linha que contenha uma letra "c" ou um asterisco "*" na coluna 1 são ignoradas em Fortran. São usadas para inserir comentários dentro de um programa Fortran.

- Para a codificação de arquivos de dados de entrada e saída, estes seguem a estrutura que porventura seja definida no próprio programa que vai ler/escrever neste arquivo.

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

Capítulo 3: Constantes, Variáveis e Conjuntos

3.1 Introdução

- Este capítulo trata de definir corretamente os componentes básicos dos comandos Fortran. Alguns já vimos no capítulo anterior, como "itens de sintaxe".

- "Constantes" são valores fixos, tais como números. Estes valores não podem ser alterados pelos comandos do programa. Exemplos: 3.0, 3, 'palavra'.

- "Variáveis" são nomes simbólicos que representam valores armazenados na memória do computador. Estes valores podem ser alterados pelos comandos do programa.

- "Conjuntos" são grupos de variáveis, cujos valores são armazenados adjacentemente e podem ser referenciados individualmente através de um nome simbólico com um índice. São conhecidos também como "variáveis indexadas" (VI).

- Um "operador" é um símbolo específico para uma determinada operação. Exemplo: + (soma).

- "Expressões" são combinações de constantes, variáveis, elementos de conjuntos e operadores. Exemplo: 3.0*var1+var2.

- No "comando de atribuição" abaixo:

```
fat = a(3) * 4 * (B + 2.5)
```

temos a "expressão" a(3)*4*(B+2.5). Nesta expressão temos duas "constantes" (4 e 2.5), uma "variável" (B), uma variável indexada (a(3)), dois "operadores" (* e +) e parênteses. Vemos também que temos uma constante inteira (4) e uma real (2.5); as variáveis a(3) e B dependem de como tenham sido definidas.

- No trecho de programa abaixo:

```
print *, 'digite a velocidade inicial:'
read *, v0y
g=9.8
dt=0.1
do i =1,np
   t(i)=i*dt
   y(i)=v0y*t(i)-0.5*g*t(i)**2
enddo
...
do i=1,np
   print *,t(i),y(i)
enddo
```

vemos o uso das "variáveis indexadas" t(i) e y(i).

APOSTILA DE FORTRAN

3.2. Constantes

- Uma "constante" é uma quantidade fixa e invariável.

- O Fortran distingue três classes de constantes: numéricas, lógicas e cadeias de caracteres.

- As constantes numéricas que mais nos interessam são:

- Inteiras: para números inteiros decimais (escritos sem o ponto decimal).
- Reais: para números decimais (ou fracionários).
- Dupla precisão: para números reais com mais dígitos significativos que o normal.
- Complexos: para números complexos.

- As constantes lógicas podem ser:

- .true. : representa o valor "verdade".
- .false. : representa o valor "falso".

- As constantes "cadeias de caracteres" são uma seqüência de caracteres alfanuméricos e/ou especiais sempre entre aspas.

- O comando PARAMETER pode ser usado para associar um nome simbólico a uma constante.

3.2.1. Constantes Inteiras

- O intervalo máximo de valores possíveis para constantes inteiras varia de máquina para máquina. O menor intervalo é de [-9999,+9999].

3.2.2. Constantes Reais

- Uma constante real é uma cadeia de dígitos escrita com ponto decimal, com um expoente ou com ambos, expressando um número real. Ex.: 0.123, 123.0E-03, -123.0E-03 .

- O intervalo de valores possíveis para o expoente, bem como a precisão do número (quantidade máxima de dígitos significativos) varia de máquina para máquina. O menor intervalo conhecido é de 10^{-28} a 10^{+28}. A precisão do número é tipicamente no mínimo de 7 dígitos.

3.2.3. Constantes Caracteres

- O comprimento máximo de uma cadeia de caracteres geralmente não é menor que 255.

3.3. Variáveis

- Uma variável possui um nome e um tipo, podendo assumir diversos valores.

APOSTILA DE FORTRAN

- Regras para nomes de variáveis:

1. Os nomes devem começar com uma letra.
2. Os nomes podem conter letras e dígitos.

- Tipos de variáveis: inteiras, reais, dupla precisão, complexas, lógicas e caracteres, que possuem características similares às constantes.

3.3.1. Variáveis Inteiras

- Se não for usado o comando IMPLICIT NONE, por convenção, toda variável numérica cujo nome inicia por uma das letras i, j, k, l, m ou n, é uma variável inteira.

- As variáveis inteiras podem assumir valores na mesma faixa das constantes inteiras.

3.3.2 Variáveis Reais

- Se não for usado o comando IMPLICIT NONE, por convenção, toda variável numérica cujo nome não inicia por uma das letras i, j, k, l, m ou n, é uma variável real.

- As variáveis reais podem assumir valores na mesma faixa que as constantes reais.

3.3.3. Variáveis Caractere

- Uma variável caractere é uma variável que é declarada como caractere através dos comandos de especificação explícita ou do comando IMPLICIT. A quantidade de caracteres numa variável caractere é especificada quando da declaração da variável caractere.

- Veja os seguintes exemplos de comandos de especificação de variáveis caractere:

1. implicit character*28 (y,z)
2. character*36 linha
3. character soma*15,matriz*64

Temos que linha, soma e matriz são especificadas como variáveis caractere contendo 36, 15 e 64 caracteres, respectivamente. Além disso, todas as variáveis cuja primeira letra é y ou z são também declaradas como variáveis caractere contendo 28 caracteres cada uma.

3.4. Conjuntos

- Nesta seção nos ocuparemos dos conjuntos de variáveis, ou variáveis indexadas (VI), já definidos no início deste capítulo.

- Uma VI possui um nome, um tipo e um conjunto de índices (ou um único índice).

- "Vetor" é uma VI que possui um único índice, e por isto é dito uma VI unidimensional.

- A variável t no trecho de programa abaixo é um vetor:

```
dimension t(10)
t(1)=0.1
t(2)=0.2
t(3)=0.3
...
t(10)=1.0
```

- "Matriz" é uma VI que possui dois conjuntos de índices e por isto é dita bidimensional.

- A variável a no trecho de programa abaixo é uma matriz:

```
dimension a(3,3)
do i = 1,3
   do j = 1,3
      a(i,j) = 0.0
   enddo
enddo
```

- Representação Matricial da variável a:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{11} & a_{32} & a_{33} \end{pmatrix}$$

- Forma de armazenamento na memória:

...	a_{11}	a_{21}	a_{31}	a_{12}	a_{22}	a_{32}	a_{13}	a_{23}	a_{33}	...

- Dizemos que uma VI é uma matriz n-dimensional quando esta possui um conjunto de n índices.

- No Fortran 77, uma VI não podia possuir mais do que 7 índices.

3.4.1 Declaradores de Conjuntos

- Comando DIMENSION: permite especificar uma determinada variável como sendo uma variável indexada.

- Exemplos do uso do comando DIMENSION:

APOSTILA DE FORTRAN

1) dimension a(8)
onde 8 é número máximo de diferentes a's.

2) dimension a(8),b(5,6)
sendo que 5 e 6 são os valores máximos do primeiro e segundo índices (que resultam em 30 diferentes b's).

3) integer a
   real b
   dimension a(8),b(5,6)
sendo que o vetor a é um vetor de números inteiros e b uma matriz de números reais.

4) dimension c(0:10),d(1:4,1:3,1:5)

sendo que: c = vetor de 11 elementos, sendo o primeiro c(0) e o último c(10);
d = matriz tridimensional de 60 elementos, sendo que o primeiro elemento é d(1,1,1) e o último é d(4,3,5)

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

Capítulo 4: Expressões

4.1. Introdução

- Uma "expressão" Fortran é definida como uma combinação de itens sintáticos, isto é: uma expressão pode consistir de uma única constante, de uma única variável, de um único elemento de conjunto, ou uma combinação de constantes, variáveis, elementos de conjuntos, unidos com um ou mais operadores e, possivelmente, agrupados com parênteses.

- "Operadores" especificam as computações a serem realizadas sobre os valores dos elementos básicos.

- As expressões, no Fortran, são classificadas em aritméticas, caracteres, relacionais e lógicas.

4.2. Expressões Aritméticas

- Uma "expressão aritmética" é usada para expressar uma computação numérica. O cálculo desta expressão produz um valor numérico cujo tipo é inteiro, real, dupla precisão ou complexo.

- Os "operadores aritméticos" são definidos pela tabela abaixo:

Operador	Definição	Uso	Significado
**	potenciação	a**b	a elevado à potência b
*	multiplicação	a*b	a multiplicado por b
/	divisão	a/b	a dividido por b
+	adição	a+b	a mais b
-	subtração	a-b	a menos b
-	menos unário	-a	a com sinal invertido

- Em expressões contendo um único operando, o tipo do operando define o tipo de expressão.

- Em expressões com mais de um tipo de operando, o tipo da expressão depende dos tipos dos operandos e dos operadores.

- Para todas as operações (com exceção da potenciação):

1. Se os operandos forem do mesmo tipo, este será o tipo da expressão.

APOSTILA DE FORTRAN

2. Se os operandos forem de tipos diferentes, o tipo da expressão será o tipo do operando de maior precedência, dada pela tabela abaixo:

Tipo	Precedência
inteiro	1
real	2
dupla precisão	3
complexo	4

- Regras para ordem de computação:

1. Parênteses podem ser usados para agrupar operandos em uma expressão. Qualquer expressão entre parênteses é calculada antes do cálculo da expressão da qual ela é uma parte. Parênteses mais internos são calculados antes.

2. Com exceção do uso dos parênteses, as operações são realizadas na seguinte ordem de precedência:

Operação	Precedência
+ e -	mais baixa
* e /	intermediária
**	mais alta

3. Para operadores de mesma precedência, o cálculo se desenvolve da esquerda para a direita.
4. Potenciações consecutivas são realizadas da direita para a esquerda.

4.3. Expressões Relacionais

- Uma "expressão relacional" é usada para comparar os valores de duas expressões aritméticas (ou os valores de duas expressões caractere).

- A comparação entre duas expressões aritméticas é feita através dos "operadores relacionais".

- O cálculo de uma expressão relacional produz um resultado lógico, com um valor "verdade" ou "falso". A expressão é interpretada como "verdade" se os valores dos operandos satisfazem a relação matemática especificada pelo operador, e "falso" se eles não satisfazem a relação.

- "Operadores Relacionais" são aqueles definidos na tabela abaixo:

Operador	Definição	Uso	Significado
somente Fortran90 e posteriores			
==	igual a	A==B	A é igual a B ?
/=	diferente de	A/=B	A é diferente de B?

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

<	menor que	A<B	A é menor a B?
<=	menor ou igual	A<=B	A é menor ou igual a B?
>	maior que	A>B	A é maior que B ?
>=	maior ou igual	A>=B	A é maior ou igual a B?
todos			
.eq.	igual a (equal)	A.eq.B	A é igual a B ?
.ne.	diferente de (not equal)	A.ne.B	A é diferente de B?
.lt.	menor que (less than)	A.lt.B	A é menor a B?
.le.	menor ou igual (less or equal)	A.le.B	A é menor ou igual a B?
.gt.	maior que (greater than)	A.gt.B	A é maior que B ?
.ge.	maior ou igual (greater or equal)	A.ge.B	A é maior ou igual a B?

4.4. Expressões Lógicas

- Uma expressão lógica é usada para realizar um cálculo lógico.

- Os "operadores lógicos" são (listamos somente os principais):

Operador	Definição	Uso	Significado
.not.	negação	.not.A	se A é verdade, .not.A é falso
.and.	conjunção	A.and.B	para a expressão ser verdade, A e B precisam ser verdade
.or.	disjunção	A.or.B	para a expressão ser verdade, A ,ou B, precisa ser verdade

- Exemplos:

1) if(x1.eq.x.and.x2.eq.x1) then …

2) fim=.false.
   do while(.not.fim) …
      bloco de comandos
   enddo

- Regras para cálculos de expressões lógicas.

1. Parênteses podem ser usados e são calculados antes.
2. Os operadores aritméticos são calculados em segundo lugar.
3. Os operadores relacionais são calculados em terceiro lugar.
4. As operações lógicas são realizadas na seguinte ordem de precedência:

Operador	Precedência
.not.	mais alta

.and.	intermediária
.or.	mais baixa

Capítulo 5: Comandos de Atribuição

5.1. Introdução

- O "comando de atribuição" é usado para atribuir um valor a (ou definir) uma variável ou a um elemento de conjunto.

- Os comandos de atribuição calculam uma expressão e atribuem o valor resultante a uma variável ou elemento de conjunto.

- A forma de um comando de atribuição é:

variável=expressão

- Os comandos de atribuição são classificados como:

- aritmético;
- caractere;
- lógico.

- Exemplos:

- aritmético: varnum=3.0*a/b
- caractere: varcar='digite o valor de x:'
- lógico: varlog=.true. ou varlog=a<b.and.b==c

- Após o cálculo da expressão, o valor é convertido, se necessário, para o tipo de variável de acordo com a seguinte regra:

Tipo da variável	Valor atribuído
inteiro	a parte inteira da expressão
real	o valor real da expressão
dupla precisão	o valor dupla precisão da expressão
complexo	o valor complexo da expressão

APOSTILA DE FORTRAN

Capítulo 6: Comandos de Especificação: Declaração de Tipos de Variáveis

6.1. Introdução

- Os comandos de declaração de tipo são: IMPLICIT, INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL e CHARACTER.

6.2. Comando IMPLICIT

- A forma geral do comando é:

IMPLICIT tipo (a[,a]...) [,tipo (a[,a])]

- O comando IMPLICIT permite que se defina o tipo de todas as variáveis que começam por uma determinada letra.

- Exemplos:

1) implicit real(i)
- faz com que todas as variáveis que comecem pela letra i sejam reais.

2) implicit real(i,j,k,l)
- faz com que as variáveis com iniciais de i a l sejam reais.

3) implicit real(i-l)
- idem.

4) implicit real(i-l),integer(a,b)
- faz a mesma coisa que o comando anterior mais a definição que as variáveis com iniciais a ou b sejam inteiras.

5) implicit none
- faz com que todas as variáveis do programa tenham que ter o seu tipo obrigatoriamente pré-definidas.

6.3. Comandos de Especificação Explícita

6.3.1. Comando Integer

- O comando INTEGER é usado para declarar variáveis inteiras.

- A forma geral do comando INTEGER é:

```
INTEGER nome [,nome]...
```

ou

```
INTEGER nomevi[(d[,d]...)]
```

sendo que nomevi é o nome de uma variável indexada inteira.

- Exemplos:

```
integer soma,cont
integer a(0:10,0:8)
```

6.3.2. Comando REAL

- O comando REAL é usado para declarar variáveis reais.

- A forma geral do comando REAL é:

```
REAL nome[,nome]...
```

ou

```
REAL nomevi[(d[,d]...)]
```

sendo que nomevi é o nome de uma variável indexada real.

- Exemplos:

```
real a,b,c
real d(1:10,1:8)
```

6.3.3. Comando DOUBLE PRECISION

- O comando DOUBLE PRECISION é usado para duplicar o número de algarismos significativos de uma variável ou de um conjunto.

- A forma geral do comando DOUBLE PRECISION é:

```
DOUBLE PRECISION nome[,nome]...
```

ou

DOUBLE PRECISION nomevi[(d[,d]...)]

sendo que nomevi é o nome de uma variável indexada de dupla precisão.

6.3.4. Comando COMPLEX

- O comando COMPLEX é usado para declarar variáveis do tipo complexo.

6.3.5. Comando LOGICAL

- O comando LOGICAL é usado para declarar variáveis do tipo lógico.

6.3.6. Comando CHARACTER

- O comando CHARACTER é usado para declarar variáveis do tipo caractere.

- A forma geral do comando CHARACTER é:

CHARACTER*comp nome[,nome]...

ou

CHARACTER nome*comp[,nome*comp]...

sendo que comp é o comprimento da(s) variável(eis) caractere.

- Exemplos:

```
character*12 a,b,c
character d*5,e*4,f*10
```

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

Capítulo 7: Comandos de Especificação: Designação de Áreas de Memória

7.1. Introdução

- Neste capítulo, aprenderemos a reservar áreas de memória, com os seguintes objetivos:

1. Determinar a quantidade de memória que deve ser reservada para variáveis indexadas.
2. Indicar áreas de memória que podem ser acessadas por diferentes unidades de programa.

7.2. Comando DIMENSION

- O comando DIMENSION é usado para especificar nomes simbólicos como nomes de variáveis indexadas, define o número de dimensões (índices) de cada conjunto de vi´s e especifica os limites de cada dimensão.

- A forma geral do comando DIMENSION é:

```
DIMENSION c(d[,d]...)[,c(d[,d]...)]
```

sendo que:

- "c" é um nome de conjunto;
- "d" é um declarador de dimensão (que podem ser no máximo 7) e especifica os limites para cada índice do conjunto, em uma das formas: *superior* ou *inferior:superior*
- *superior*: o índice varia de 1 até *superior*.
- *inferior:superior*: o índice varia de *inferior* até *superior*.

- Exemplos:

```
dimension a(0:10),b(3,3),c(3,4,5)
```

7.3. Comando COMMON

- O comando COMMON é usado para especificar áreas de memória que são utilizadas em comum por várias unidades de programa, tal como o programa principal e subprogramas do tipo FUNCTION ou SUBROUTINE.

- Áreas de memória designadas por uma unidade de programa não são "enxergadas" pelas outras unidades. Assim, o que COMMON faz é permitir que diferentes unidades de programa "enxerguem" uma dada área de memória.

- A forma geral do comando COMMON é:

```
COMMON[/r/] lista
```

sendo que:

- "r" é um rótulo do bloco de variáveis que terão suas áreas de memória especificadas.
- "lista" é uma lista com nomes de variáveis que terão suas áreas de memória especificadas.

- Exemplos:

1) common m,a,x

2) dimension a(10),b(5,20)
   common/bloco1/a,b

- O mesmo comando COMMON, escrito do mesmo jeito, deve aparecer em cada unidade de programa que pretende ter acesso às variáveis definidas num dado bloco.

7.4. Unidades de programas do tipo MODULE

- As unidades de programa do tipo MODULE existem a partir do Fortran 90 e podem substituir todas as ocorrências do comando COMMON, do comando INCLUDE (não apresentado nesta apostila) e funções de comando (Capítulo 11).

- Numa unidade de programa do tipo MODULE podem se definir tipos e valores de constantes e variáveis. Ainda é possível incluir subprogramas, mas este tópico não será abordado.

7.5. Comando USE

- O comando USE é utilizado para definir uma unidade de programa do tipo MODULE que deve ser incluída no programa.

- Exemplo de utilização das unidades de programa do tipo MODULE e do comando USE:

```
   module modulo01
     integer:: a=2,b=3
   end module modulo01

   module modulo02
     integer:: a=4,b=5
   end module modulo02

   program exmodls
! Objetivo: exemplifica o uso dos subprogramas do tipo MODULE
   use modulo01
   implicit none
```

```
  integer c,d
! Nas duas linhas a seguir, serão usados os valores de a e b declarados
! no módulo modulo01:
  c=a+b
  d=a*b
  print *,a,b,c,d
! Na subrotina abaixo, ainda serão usados os valores de a e b declarados
! no módulo modulo01, demonstrando a compartilhação de dados entre o
! programa principal e um subprograma:
  call subrotina01
! A subrotina abaixo é idêntica à anterior, com a única diferença que
! utiliza novos valores para a e b declarados no módulo modulo02:
  call subrotina02
  stop
  end

  subroutine subrotina01
  use modulo01
  implicit none
  integer c,d
  c=a+b
  d=a*b
  print *,a,b,c,d
  return
  end

  subroutine subrotina02
  use modulo02
  implicit none
  integer c,d
  c=a+b
  d=a*b
  print *,a,b,c,d
  return
  end
```

7.6. Comando PARAMETER

- O comando PARAMETER fornece um meio de se representar uma constante por um símbolo, como na Matemática, em vez de representá-la por um valor.

- A forma geral do comando PARAMETER é:

```
  PARAMETER(ns=e[,ns=e]...)
```

sendo que:

- "ns" é um nome simbólico.
- "e" é uma constante ou uma expressão constante.

- PARAMETER é um comando de especificação e deve ser escrito entre os comandos de especificação. Porém cada constante simbólica deve ser definida antes de qualquer referência a ela.

- Os novos símbolos das constantes definidas dentro de um comando PARAMETER seguem as mesmas regras para os nomes de variáveis (em relação ao tipo).

- O valor de uma constante definida em PARAMETER jamais é ou pode ser alterado durante a execução do programa.

- Exemplo:

```
?
real pi
parameter(pi=3.1415927)
?
```

7.7. Comando PROGRAM

- O comando PROGRAM é usado para identificar o programa principal.

- Todo programa possui um único programa principal.

- A forma geral de PROGRAM é:

```
program nome
```

sendo que:

- "nome" é um nome simbólico definido para o programa principal.

7.8. Comandos ALLOCATE, DEALLOCATE E ALLOCATABLE

- Os comandos ALLOCATE, DEALLOCATE E ALLOCATABLE têm por função permitir que uma variável indexada possa ser dimensionada durante a execução do programa.

- ALLOCATABLE: indica que a matriz vai ter os valores mínimos e máximos para o seu(s) índice(s) determinados em algum ponto do programa. A área de memória para esta variável

ainda não está reservada.

- ALLOCATE: dimensiona a variável indexada e reserva (aloca) um espaço de memória para ela.

- DEALLOCATE: desaloca, mas não extingue, a variável indexada, liberando a memória antes por ela ocupada.

- No exemplo a seguir temos vários trechos de um programa sendo que as variáveis indexadas A, B, C e D estão sendo
dimensionadas, alocadas e desalocadas usando os comandos ALLOCATE, DEALLOCATE E ALLOCATABLE:

```
...
integer i,j,k,l,m,n
integer,dimension(:,:),allocatable :: a
real,dimension(:),allocatable :: b,c
real,dimension(:,:,:),allocatable :: d
...
print *,'Digite os minimos e maximos das 2 dimensoes de A:'
read *,i,j,k,l
allocate(a(i:j,k:l))
...
print *,'Digite o minimo e o maximo da unica dimensao de B:'
read *,i,j
print *,'Digite o minimo e o maximo da unica dimensao de C:'
read *,k,l
allocate(b(i:j),c(k:l))
...
print *,'Digite os minimos e maximos das 3 dimensoes de D:'
read *,i,j,k,l,m,n
allocate(d(i:j,k:l,m:n))
...
deallocate(a,d)
...
deallocate(b)
...
deallocate(c)
...
```

Capítulo 8: Comandos de Controle de Fluxo e Programação Estruturada

8.1. Introdução

- Normalmente, a execução dos comandos num programa é feita na ordem em que os comandos são escritos: de cima para baixo, linha por linha, de acordo com a estrutura seqüencial. Entretanto você pode usar comandos para transferir o controle de fluxo para outra

parte da mesma unidade de programa ou para outra unidade de programa.

- O controle pode ser transferido somente para um comando executável.

- Por vezes, um comando pode ser identificado por um "número de comando", consistindo de um número inteiro, escrito antes do comando, no início da linha. Cada número de comando deve ser único dentro da unidade de programa. Eles servem principalmente para a especificação de formatos de entrada e/ou saída de dados. Evita-se hoje o uso antigo do número de comando, que servia de referência para uma transferência de controle de fluxo. Isto favorece a criação de códigos de programa "espaguete", sendo que o controle de fluxo pode ser mudado de maneira absurda, dificultando e inviabilizando a manutenção dos programas.

8.2. Estruturas de Controle

- As estruturas básicas de controle são:

1. Estrutura seqüencial.
2. Estrutura de tomada de decisão (if-then-else if-else-endif).
3. Estrutura de laço:
    - faça-enquanto (while-loop).
    - Estrutura de laço repetição (repeat-loop).
4. Estrutura de seleção de caso (select case).

8.3. Comandos GO TO

- Os comandos GO TO transferem o controle de fluxo para um comando executável na mesma unidade de programa. Os três tipos de GO TO são:

1. GO TO incondicional.
2. GO TO computado.
3. GO TO assinalado.

8.3.1. Comando GO TO incondicional

- O comando GO TO incondicional transfere o controle para o comando identificado pelo número de comando especificado no GO TO.

- A forma geral do comando GO TO incondicional é:

```
go to n
```

sendo que "n" é um número de comando executável.

APOSTILA DE FORTRAN

8.4. Comandos IF

- Os comandos IF transferem o controle do fluxo ou executam outro comando (ou um bloco de comandos) dependendo do resultado verdadeiro ou falso de uma expressão lógica contida no particular comando IF. Os três tipos de comandos IF são:

1. IF aritmético.
2. IF lógico.
3. IF bloco.

- Os comandos ELSE, ELSE IF e END IF são também apresentados nesta seção, pois eles são usados somente em conjunto com um comando IF bloco.

8.4.1. Comando IF lógico

- O comando IF lógico calcula uma expressão lógica/relacional e executa ou ignora um comando executável contido no próprio IF, dependendo do valor (falso ou verdadeiro) dessa expressão.

- A forma geral do comando IF lógico é:

```
if (e) c
```

sendo que:

- "e" é uma expressão lógica ou relacional.

- "c" é o comando que será executado caso a expressão seja verdade; "c" não pode ser um comando DO.

- Vejamos como fica o fluxo do programa quando este chega a estas três linhas:

```
x=a-b
if(a.eq.b) x=a+b
y=x**2
```

Observe que nas últimas 2 linhas temos três comandos:
1) if(a.eq.b)
2) x=a+b
3) y=x**2

Então:
a) Se de fato a=b, teremos a seguinte ordem de execução: 1 >> 2 >> 3.
b) Se, por outro lado, a é diferente de b, teremos a ordem de execução: 1 >> 3.

APOSTILA DE FORTRAN

8.4.2. Comandos IF bloco

- O comando IF bloco permite a execução de um determinado bloco de comandos, dependendo do valor da(s) expressões lógica(s)/relacional(is) nele contido.

- A forma geral do comando IF bloco é:

```
if (e1) then
   bloco1
else if (e2) then
   bloco2
else if (e3) then
   bloco3
...
else
   bloco n
endif
```

sendo que e1, e2,e3, ... são expressões lógicas/relacionais. Deve ficar claro que somente um dos blocos de comandos será executado.

- O menor IF bloco é a estrutura:

```
if (e) then
   bloco
endif
```

Vemos que os comandos ELSE IF e ELSE são opcionais.

- O comando IF THEN permite a execução condicional de um bloco de comandos executáveis. É obrigatório usar em conjunção o comando ENDIF para fechar o comando.

- A forma geral do comando IF THEN é:

```
if (e) then
```

sendo que "e" é uma expressão lógica/relacional (portanto produzindo um resultado verdadeiro ou falso).

- Se a expressão "e" for verdadeira, a execução passa para o próximo comando executável e daí para o comando ENDIF. Se for falsa, o controle é transferido para o próximo comando: ELSE IF THEN, ELSE ou ENDIF.

- O comando ELSE fornece uma rota alternativa para um comando IF THEN ou ELSE IF THEN. Sua forma geral é simplesmente:

```
else
```

- O comando ELSE IF THEN combina as funções dos comandos ELSE e IF THEN, pois fornece uma rota alternativa e torna possível uma estrutura IF bloco com mais de uma alternativa.

- A forma geral do comando ELSE IF THEN é:

```
else if (e) then
```

sendo que "e" é uma expressão lógica/relacional.

- Se a expressão for verdadeira, o controle passa para o próximo comando executável e daí para o comando ENDIF. Se for falsa, passa para o próximo comando ELSE IF THEN, ELSE ou ENDIF.

- O comando ENDIF ou END IF finaliza uma estrutura IF bloco. Sua forma geral é simplesmente:

```
end if
```

- Exemplo:

```
...
print *,'Digite a nota do aluno: '
read *,nota
if(nota>=5.0) then
   print *,'O aluno foi aprovado.'
elseif(nota>=3.5) then
   print *,'O aluno foi reprovado mas podera cursar o RE.'
else
   print *,'O aluno foi reprovado e nao podera cursar o RE.'
endif
?
```

8.4.3. Estruturas de IF bloco encaixados

- Um bloco de comandos executáveis dentro de um comando IF bloco pode conter outra estrutura IF bloco. E isto pode se repetir indefinidamente.

```
...
print *,'Digite a nota do aluno: '
read *,nota
if(nota>=5.0) then
   if(nota>=9.5) then
      print *,'O aluno foi aprovado com louvor.'
   elseif(nota>=8.0) then
      print *,'O aluno foi aprovado com distincao.'
   else
      print *,'O aluno foi aprovado.'
   endif
elseif(nota>=3.5) then
   print *,'O aluno foi reprovado mas podera cursar o RE.'
else
```

```
    print *,'O aluno foi reprovado e nao podera cursar o RE.'
  endif
  ?
```

8.5. Comando DO

- O comando DO é um comando de controle que permite que um bloco de comandos seja repetitivamente executado. O número de execuções depende da variável de controle.

- A forma geral do comando DO é:

```
do v = vi , vf [, incr]
```

sendo que:

- "v" é a variável do controle do laço DO.
- "vi" é o valor inicial de v.
- "vf" é o valor final ou máximo de v.
- "incr" é o incremento de v. Se for omitido, é suposto como de valor 1.

- O comando ENDDO ou END DO finaliza uma estrutura DO. Sua forma geral é simplesmente:

```
enddo
```

- Exemplo:

```
do i=0,100,5
   bloco de comandos
enddo
```

- Existem duas estruturas DO:

1) A estrutura DO - END DO (repeat loop) repete um bloco de comandos condicionado ao valor de uma variável de controle que deve variar de modo fixo e pré-determinado:

```
do v=vi,vf,incr
   bloco de comandos
 end do
```

2) A estrutura DO WHILE - END DO (while loop) executa o laço DO enquanto uma expressão lógica/relacional for verdadeira:

```
do while (var.ge.0)
   bloco de comandos
end do
```

- O comando DO implícito é uma terceira forma do comando DO e somente pode ser usando em comandos de entrada ou saída de dados.

- Exemplos de uso do comando DO implícito:

```
print *,(var(i),i=1,10,1)
print *,(var(i,j),j=10,20,2)
read(1,100)(var(i),i=2,8,2)
write((3,200)(var(i,j),j=1,5,1)
```

8.5.1. Laços de DO encaixados

- É possível encaixar dois ou mais comandos DO.

- Para as estruturas DO encaixadas, o primeiro comando END DO após o comando DO, fecha o laço.

8.6. Comando EXIT

- O comando EXIT termina abruptamente um laço DO, direcionando o fluxo do programa para a primeira linha de comando após o ENDDO daquele respectivo comando DO envolvido.

8.7. Comando CYCLE

- O comando CYCLE reinicia abruptamente um laço DO, direcionando o fluxo do programa para a linha do comando DO dentro do qual ele está inserido.

8.8. Comando CONTINUE

- O comando CONTINUE é um comando executável que somente passa o controle para o próximo comando executável. A forma geral é:

```
continue
```

8.9. Comando STOP

- O comando STOP termina a execução do programa. Apesar de sua localização dentro do programa em geral ser a penúltima linha de comando, ele pode ser colocado em qualquer posição dentro do programa.

- A forma geral do comando STOP é:

```
stop [varcar]
```

sendo que "varcar" é uma variável caracter que será exibida na saída padrão (em geral, a tela do monitor do operador).

APOSTILA DE FORTRAN

8.10. Comando END

- O comando END indica o final de uma unidade de programa para o compilador. Sua forma geral é:

```
end
```

e deve obrigatoriamente ser colocado na última linha do programa.

8.11. Comando SELECT CASE

- A estrutura SELECT CASE permite a execução de apenas uma entre muitas opções possíveis.

- A estrutura SELECT CASE possui a seguinte forma:

```
select case (var)
   case (um_valor_de_var)
      bloco de comandos 1
   case (outro_valor_de_var)
      bloco de comandos 2
   case (outro_ainda_valor_de_var)
      bloco de comandos 3
   ...
end select
```

- Exemplo:

```
...
print *,'Digite um numero de 1 a 10: '
read *,nu
select case (nu)
   case (1:4)
      print *,'A opcao foi um numero no intervalo [1,4]'
   case (5)
      print *,'A opcao foi pelo numero 5'
   case (6:10)
      print *,'A opcao foi um numero no intervalo [6,10]'
   end select
   ...
```

Capítulo 9: Comandos de Entrada/Saída

9.1. Introdução

- Os "Comandos de Entrada" fornecem um método de transferir dados de um dispositivo periférico (como um teclado) ou de um arquivo interno para a memória principal. Este processo é chamado de "leitura de dados".

APOSTILA DE FORTRAN

- Os "Comandos de Saída" fornecem um meio de transferir dados da memória principal para um dispositivo periférico (como um monitor de vídeo ou impressora) ou um arquivo interno. Este processo é chamado de "escrita" ou "gravação" de dados.

- Alguns comandos de E/S permitem que se edite os dados durante a sua transferência.

- Neste capítulo somente serão apresentados os comandos de E/S que transferem dados (READ, PRINT, WRITE). Comandos auxiliares não serão discutidos.

9.2. Registros, Arquivos e Unidades

- Um "caractere" é um único símbolo, tal como a letra *z*, o dígito *4* ou um asterisco *.

- Um "campo" é uma sucessão de caracteres, que representa alguma informação, tal como *ANTONIO* ou *128.47*.

- Um "registro" é um grupo de campos que reúne informações sobre um único item, tal como:

ANTONIO 26-01-1937 6543.21
(nome) (data nasc.) (salário)

- Um "arquivo" é um grupo de registros, tal como:

A	*N*	*T*	*O*	*N*	*I*	*O*				*2*	*6*	*-*	*0*	*1*	*-*	*1*	*9*	*3*	*7*	*6*	*5*	*4*	*3*	*.*	*2*	*1*
A	*R*	*N*	*A*	*L*	*D*	*O*				*2*	*2*	*-*	*1*	*0*	*-*	*1*	*9*	*4*	*5*		*9*	*8*	*7*	*.*	*6*	*5*
M	*.*	*C*	*R*	*I*	*S*	*T*	*I*	*N*	*A*	*0*	*7*	*-*	*0*	*9*	*-*	*1*	*9*	*4*	*2*	*1*	*2*	*3*	*4*	*.*	*5*	*6*

Temos acima um arquivo, com três registros, cada registro com três campos e comprimento total de 27 caracteres.

- Cada comando de E/S opera sobre um único registro.

- Uma unidade de E/S é um meio de se referir a um arquivo. Usa-se números de unidades de E/S para distinguir um arquivo de outro.

- Certos números de unidades podem ser pré-definidos. Assim, por exemplo, o número 5 pode indicar o teclado e 6 pode indicar o monitor de vídeo.

- O comando OPEN estabelece a conexão entre o número da unidade e um particular arquivo.

APOSTILA DE FORTRAN

9.3. Componentes dos Comandos de E/S

- Os comandos de E/S são constituídos de 3 partes:

comando (especificadores) [lista]

sendo que:

- *especificadores* definem apropriadamente a execução do comando;
- *lista* define a(s) variável(eis) sobre a(s) qual(is) opera o comando.

9.3.1. Lista de Especificadores de Controle de E/S

- Existem muitos especificadores de controle de entrada e saída. Veremos somente os mais importantes para o iniciante.

- UNIT: o especificador de unidade designa a unidade externa de E/S ou o arquivo interno a ser usado para o comando de E/S no qual ele aparece. Seu uso é:

[unit=]u ou [unit=]*

sendo que:

- "u" é uma expressão inteira que designa o número de E/S.
- "*" designa a unidade de entrada padrão em READ e saída padrão em PRINT ou WRITE.

- Os caracteres "unit=" somente podem ser omitidos no caso do especificador de unidade ser o primeiro na lista de especificadores.

- FMT: o especificador de formato designa o formato a ser usado para o comando de E/S formatado ou o comando de E/S com lista direta. Seu uso é

[fmt=]f ou [fmt=]*

sendo que:

- "f" é um identificador de formato;
- "*" designa uma formatação em lista direta.

- Os caracteres "fmt=" somente podem ser omitidos se o especificador de formato for o segundo especificador e ainda o primeiro especificador for o de unidade sem os caracteres "unit=" .

- O identificador de formato "f" pode ser um número de um comando FORMAT. Ou uma expressão inteira (que acaba por indicar um comando FORMAT). Ou especificações de formato (em forma de variável caractere).

- Exemplos. Todos eles fazem a mesma coisa e ainda o comando READ poderia ser substituído pelo comando WRITE.

Exemplo 1:
```
  read(unit=1,fmt=100) a,b
  100 format(i5,f10.3)
```
Exemplo 2:
```
  read(1,100) a,b
  100 format(i5,f10.3)
```
Exemplo 3:
```
  read(1,*) a,b
```

9.3.2. Lista de E/S

- Uma lista de E/S tem a forma:

  s[,s]…

sendo que "s" é uma lista de itens simples ou uma lista de DO implícito.

- Um item da lista pode ser um nome de variável, um nome de conjunto, um nome de elemento de conjunto, um nome de subcadeia caractere ou uma lista de DO implícito. No caso de lista de saída, ainda pode ser uma expressão.

- Exemplos de listas de E/S (sem DO implícito):
```
  read(5,100) A,B,C,D
  read(3,20) X,Y,C(J),D(3,4),E(I,K,7),T
  read(4,121)I,A(I),J,B(I,J)
  write(2,302) ALFA,BETA
  write(6,122) GAMA(4*J+2,5*I+2,4*K),C,D(L)
```

- Registro de entrada contendo os dados L, M e N, que são definidos como números inteiros:

100223456789

que é lido pelos comandos:

```
      read(5,60) L,M,N
60    format(I3,I2,I7)
```

apresentará os seguintes valores:

L=100
M=22
N=3456789 .

- Uma lista de DO implícito tem a seguinte forma:

```
(d , i = e1, e2 [,e3] )
```

sendo que:

- "d" é uma lista de E/S.
- i , e1, e2 e e3 têm as mesmas formas, funções e efeitos que num comando DO.

- Ao invés de usar:

```
print *, A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9)
```

use

```
print *, (A(i), i = 1,9) .
```

9.4. Comandos READ

- Os comandos READ transferem dados de entrada para a memória de trabalho.

- IMPORTANTE: cada novo comando READ passa a ler dados em uma nova linha.

9.4.1. Comandos READ seqüenciais

- As formas aceitas dos comandos READ seqüencial são:

```
read([unit=]u,[fmt=]f) [lista]
```

ou

```
read f [, lista]
```

sendo que "f" pode ser substituído por "*" para designar uma lista direta. Se "u" for substituído por "*" , a leitura será feita na unidade de entrada padrão.

- Exemplos:

```
      read(unit=3,fmt=100) a,b,c
100  format(3f5.2)
```

é igual a:

```
      read(3,100) a,b,c
100  format(3f5.2)
```

que ainda podem ser simplificados para:

```
   read(3,*) a,b,c
   (lista direta)
```

ou

```
   read(*,*) a,b,c
   (aqui a leitura passa a ser feita a partir da entrada padrão)
```

9.5. Comandos WRITE (PRINT)

- Os comandos WRITE (PRINT) transferem dados armazenados na memória do computador para dispositivos de saída (vídeo, arquivos ou impressora).

- IMPORTANTE: cada novo comando WRITE (PRINT) passa a escrever dados em uma nova linha.

9.5.1. Comandos WRITE (PRINT) seqüenciais

- As formas aceitas dos comandos WRITE seqüencial são:

```
   write([unit=]u,[fmt=]f) [lista]
```

sendo que "f" pode ser substituído por "*" para lista direta.

- Exemplos:

```
      write(unit=7,fmt=200) a,b,c
200  format(3f5.2)
```

ou

```
      write(7,200) a,b,c
200  format(3f5.2)
```

(igual ao comando anterior)

ou

```
      write(7,200) a,b,c
200  format(3f5.2)
```
(praticamente igual aos comandos anteriores)

ou

```
      write(*,*) a,b,c
```
(escreve na saída padrão)

- A diferença entre WRITE e PRINT é que este último somente escreve na saída padrão (o vídeo, no nosso caso). Por isso PRINT dispensa o especificador de unidade.

- A forma aceita do comando PRINT é:

```
   print f [, lista]
```

sendo que "f" pode ser substituído por "*" para lista direta.

- Exemplos:

```
      print 200, a,b,c
200  format(3f5.2)
```

ou

```
   print *, a,b,c
```
   (lista direta)

9.6. Comandos de Condição de Arquivos de E/S

9.6.1. Comando OPEN

- O comando OPEN serve para inicializar um arquivo.

- A forma geral simplificada do comando OPEN é:

```
   open([unit=]u[,file='a'][,status='b'])
```

sendo que:

- "u" é o número da unidade.
- 'a' é uma expressão caractere que especifica o nome do arquivo;
- 'b' especifica a condição do arquivo, que pode ser:
  -'NEW', para arquivos novos, que vão ser criados pelo programa;
  -'OLD', para arquivos já existentes (em geral, arquivos de entrada de dados);
  -'SCRATCH', para arquivos que são criados pelo programa e que devem ser apagados no final

da execução deste.
- 'UNKNOWN', para arquivos para os quais não se aplica nenhuma das condições acima.

9.6.2. Comando CLOSE

- O comando CLOSE serve para finalizar um arquivo. É o oposto de OPEN.

- A forma geral simplificada do comando CLOSE é:

```
close([unit=]u)
```

sendo que "u" é o número da unidade.

Capítulo 10: Comando FORMAT e Especificações de Formato

10.1. Introdução

- FORMAT é o comando que contém uma lista de especificações de formato que permite ao programador ter controle sobre os dados de entrada e saída.

- Exemplo do uso de um comando de formatação:

```
      PRINT 10 , N, X, Y
10    FORMAT ( I3, F8.3, F5.1)
```

sendo que:
- PRINT e FORMAT são comandos (ou palavras-chave);
- "10" é número de comando;
- "N,X,Y" é lista de saída;
- "I3,F8.3,F5.1" são especificações de formato.

10.2. Comando FORMAT

- A forma geral do comando FORMAT é:

```
n format (ef [,ef [,…,ef]])
```

sendo que:

- "n" é um número de comando;
- "ef" é uma especificação de formato (conversão ou edição).

- Pode-se usar o mesmo comando FORMAT em diversos READ, WRITE ou PRINT.

UNIVERSIDADE ESTADUAL DO CEARÁ – UECE - CURSO DE FÍSICA
INTRODUÇÃO A CIÊNCIA DOS COMPUTADORES
PROFESSOR DR. CARLOS SANTANA –dr.acss@gmail.com

APOSTILA DE FORTRAN

10.3. Especificações de Formato (EF) de Conversão

- As EF de conversão mais usadas são:

Especificação	Converte dados
- I w	inteiros decimais
- F w.d	reais decimais, sem expoente
- E w.d	reais decimais, com expoente
- D w.d	reais decimais dupla precisão, com expoente
- G w.d	reais decimais, com ou sem expoente
- L w	lógicos
- A w	caracteres

sendo que:

- "a" é uma constante inteira, sem sinal e não-nula, usada para indicar o número de vezes que a mesma EF é seguidamente usada.

- "w" é uma constante inteira, sem sinal e não-nula, usada para indicar a largura total do campo externo (incluindo dígitos, espaços, sinais algébricos + ou - , ponto decimal e expoente).

- "d" é uma constante inteira, sem sinal e que indica a quantidade de dígitos à direita do ponto decimal dentro do campo de largura w. Na saída, todos os valores são arredondados.

- Exemplos:

Exemplo 1. Para escrever na tela do monitor as variáveis numéricas
var1=111 var2=2222.22 var3=3 var4=44.44
pode-se usar a seguinte combinação de comandos:

```
    print 5,var1,var2,var3,var4
5   format(i3,f7.2,i1,f5.2)
```

Exemplo 2: para escrever na tela do monitor a seguinte lista de variáveis
var1=.true. var2='america' var3=111 var4=2222.22 var5=333.33e+05
pode-se usar a seguinte combinação de comandos:

```
      print 6,var1,var2,var3,var4,var5
6     format(l1,a7,i3,f7.2,e10.2)
```

10.4. Especificações de Formato de Edição

- As especificações de edição mais usadas no Fortran são:

Especificação	Função
w x	espaçamento
/	nova linha

sendo que "w" representa o número de espaços desejados.

10.5. Especificações de Formato em Grupos Repetidos

- O comando:

```
520 format( i5, i5, f8.3, e16.7, f8.3, e16.7, f8.3, e16.7)
```

pode ser escrito de maneira mais reduzida:

```
520 format( 2i5, 3(f8.3, e16.7))
```

Fonte: http://wwwp.fc.unesp.br/~lavarda/ por Francisco C. Lavarda

Referências

LINGUAGEM DE PROGRAMAÇÃO ESTRUTURADA FORTRAN, (Editora McGraw-Hill, 1986) de M. E. Hehl