

Este documento é um guia de consultas rápidas para a conversão da notação de *Pseudocódigo* apresentada na sala de aula para a Linguagem de Programação *FORTRAN*, permitido assim transformar os algoritmos desenvolvidos no papel em programas que possam ser executados em um computador.

A conversão entre essas duas notações é razoavelmente direta, uma vez que ambas são parecidas. Algumas vezes basta apenas substituir as palavras-chave do Pseudocódigo pelas suas equivalentes em FORTRAN. Nas seções seguintes serão apresentadas as equivalências entre ambas as representações, sempre em duas colunas. A coluna da esquerda mostra a notação em Pseudocódigo e a da direita em FORTRAN.

1 Forma Geral de Representação

A forma geral de representação de um algoritmo em Pseudocódigo possui uma estrutura semelhante em FORTRAN.

| | |
|---|---|
| Algoritmo <nome_do_algoritmo> <declaração_de_variáveis> Início <corpo_do_algoritmo> Fim_Algoritmo <subalgoritmos> | PROGRAM <nome_do_programa> <declaração_de_variáveis> <corpo_do_programa> END <subprogramas> |
|---|---|

OBS.: Em FORTRAN não existe uma palavra reservada para indicar o início do programa, basta colocar os comandos do corpo do programa após a declaração de variáveis. A definição dos subprogramas pode ser feita após o programa principal.

2 Tipos de Dados e Declaração de Variáveis

FORTRAN apresenta muito mais tipos de dados do que os existentes em Pseudocódigo (veja a apostila de FORTRAN para mais detalhes). A declaração de variáveis em ambos os casos possui uma forma parecida, bastando alterar os tipos de dados adequadamente e colocar os tipos das variáveis antes dos seus nomes. Não é necessário indicar o início da declaração de variáveis com a palavra VAR, assume-se que as variáveis seguem após a definição do nome do programa.

IMPORTANTE: FORTRAN possui uma definição implícita dos tipos de dados das variáveis. Isso pode causar muitos erros difíceis de se detectar. Portanto, é muito útil indicar que não se deseja definições implícitas. Isto é feito com o comando **IMPLICIT NONE** após a declaração do nome do programa.

| | |
|--|---|
| VAR <lista_de_var> : <tipo_das_var> | IMPLICIT NONE <tipo_das_var> <lista_de_var> |
|--|---|

Um exemplo de declaração de variáveis.

| | |
|---------------------------------------|---------------------------|
| VAR nome : LITERAL [10] | IMPLICIT NONE |
| idade : INTEIRO | CHARACTER *10 nome |
| salario : REAL | INTEGER idade |
| tem_filhos : LÓGICO | REAL salario |
| sexo : LITERAL | LOGICAL tem_filhos |
| | CHARACTER sexo |

OBS.: Em Pseudocódigo, uma String é delimitada por aspas duplas “ ”. Em FORTRAN, pode-se usar como delimitadores aspas simples ‘ ’ ou duplas. FORTRAN possui um tipo de dado muito interessante, que quase nenhuma outra linguagem possui: **COMPLEX** - que armazena números complexos diretamente.

2.1 Constantes

A definição de constantes em FORTRAN é feita com a palavra reservada **PARAMETER**, no mesmo bloco de definições de variáveis. No entanto, ao contrário da notação de Pseudocódigo, em FORTRAN é necessário declarar o tipo da constante antes de se usá-la. Veja o exemplo abaixo.

| | |
|--------------------------|---|
| CONST pi = 3.1416 | REAL pi PARAMETER (pi = 3.1416) |
|--------------------------|---|

3 Expressões

3.1 Operadores aritméticos

Todos os operadores aritméticos definidos em Pseudocódigo são idênticos em FORTRAN.

3.2 Operadores relacionais

A tabela abaixo mostra a equivalência entre os operadores relacionais de ambas as notações. Compiladores FORTRAN mais antigos não reconhecem a primeira notação.

| Operação | Pseudocódigo | FORTRAN |
|----------------|--------------|-------------|
| Igual a | = | == ou .EQ. |
| Diferente de | ≠ ou <> | \ = ou .NE. |
| Menor | < | < ou .LT. |
| Menor ou igual | ≤ | <= .LE. |
| Maior | > | > ou .GT. |
| Maior ou igual | ≥ | >= ou .GE. |

3.3 Operador de Concatenação de Strings

O operador de concatenação de Strings da notação de Pseudocódigo $\#$, deve ser substituído pelo comando // em FORTRAN. Assim, “ABE” $\#$ “LHA”, deve ser convertido para ‘ABE’ // ‘LHA’.

3.4 Operadores lógicos

A tabela abaixo mostra a equivalência entre os operadores lógicos de ambas as notações.

| Operação | Pseudocódigo | FORTRAN |
|-----------------------|-----------------|---------|
| Disjunção (OU lógico) | \vee ou .OU. | .OR. |
| Conjunção (E lógico) | \wedge ou .E. | .AND. |
| Negação | \neg ou .NÃO. | .NOT. |

4 Instruções Primitivas

4.1 Atribuição

O símbolo de atribuição do Pseudocódigo deve ser substituído por = em FORTRAN.

| |
|---|
| <code><nome_da_variável> <- <expressão></code> |
|---|

| |
|---|
| <code><nome_da_variável> = <expressão></code> |
|---|

4.2 Saída de Dados

Traduzindo a palavra reservada **Escreva** do Pseudocódigo para o inglês, obtemos o comando equivalente em FORTRAN (**WRITE**). Este comando sempre pula uma linha ao fim da sua execução.

| |
|--|
| Escreva <lista_var_ou_literais> |
|--|

| |
|--|
| WRITE (*,*) <lista_var_ou_literais> |
|--|

OBS.: O primeiro asterisco indica que o periférico de saída é o padrão (vídeo). O segundo asterisco fala para se utilizar um formato livre de exibição.

4.3 Entrada de Dados

Traduzindo a palavra reservada **Leia** do Pseudocódigo para o inglês, obtemos o comando equivalente em FORTRAN (**READ**). Este comando sempre pula uma linha ao fim da sua execução.

| |
|-------------------------------|
| Leia <lista_variáveis> |
|-------------------------------|

| |
|-------------------------------------|
| READ (*,*) <lista_variáveis> |
|-------------------------------------|

OBS.: O primeiro asterisco indica que o periférico de entrada é o padrão (teclado). O segundo asterisco fala para se utilizar um formato livre de entrada.

5 Controle do Fluxo de Execução

5.1 Se-Então-Senão

Esta estrutura tem a mesma construção em ambos os casos.

| | |
|--|--|
| Se <condição> Então <Bloco 1> Senão <Bloco 2> Fim_Se | IF (<condição>) THEN <Bloco 1> ELSE <Bloco 2> END IF |
|--|--|

5.2 Escolha

Não existe uma construção equivalente ao Escolha em FORTRAN. Devemos convertê-la em um conjunto de IFs equivalentes.

5.3 Para

| | |
|---|---|
| Para <var> de <inic> até <fim> Faça <Bloco> Fim_Para | DO <var> = <inic>, <fim>, <incr> <Bloco> END DO |
|---|---|

OBS.: Em FORTRAN não é possível indicar um incremento negativo.

5.4 Enquanto

Este comando de repetição possui a mesma estrutura em ambas as notações.

| | |
|--|--|
| Enquanto <condição> Faça <Bloco> Fim_Enquanto | DO WHILE (<condição>) <Bloco> END DO |
|--|--|

5.5 Repita

Não existe uma construção equivalente ao Repita em FORTRAN. Devemos convertê-la em um DO WHILE equivalente.

6 Subalgoritmos

As definições de subalgoritmos (subprogramas) em FORTRAN devem ser colocadas após o programa principal.

6.1 Funções

Em FORTRAN, o tipo de retorno da função deve vir antes da palavra reservada **FUNCTION**. Os tipos de dados dos parâmetros da função são indicados fora do cabeçalho, junto com a declaração de variáveis locais.

| | |
|---|---|
| Função <nome> (<parâm>) : <tipo> VAR <variáveis_locais> Início <Bloco> Retorne <expressão> Fim_Função | <tipo> FUNCTION <nome> (<parâm>) <variáveis_locais> <indicação de tipos dos parâmetros> <Bloco> RETURN END FUNCTION |
|---|---|

OBS.: Em FORTRAN, o valor de uma função é retornado no seu próprio nome. Veja o exemplo abaixo:

| | |
|---|--|
| Função Quad (w : REAL) : REAL Início Retorne w*w Fim_Função | REAL FUNCTION Quad (w) REAL w Quad = w*w RETURN END FUNCTION |
|---|--|

6.2 Procedimentos

A conversão desta estrutura é direta. Os tipos de dados dos parâmetros do procedimento são indicados fora do cabeçalho, junto com a declaração de variáveis locais.

| | |
|---|---|
| Procedimento <nome> (<parâmetros>) VAR <variáveis_locais> Início <Bloco> Fim_Procedimento | SUBROUTINE <nome> (<parâmetros>) <variáveis_locais> <indicação de tipos dos parâmetros> <Bloco> RETURN END SUBROUTINE |
|---|---|

OBS.: Em FORTRAN, mesmo em procedimentos (SUBROUTINE) devemos utilizar o comando **RETURN**. Para executarmos uma SUBROUTINE no programa principal, deve-se usar: **CALL nome**

6.3 Variáveis Globais e Locais

FORTRAN não permite a definição de variáveis globais, todas são locais.

6.4 Passagem de Parâmetros

A linguagem FORTRAN possui apenas um tipo de passagem de parâmetros: por referência. Portanto, todos os parâmetros das FUNCTION e SUBROUTINE são passados por referência.

7 Variáveis Indexadas

FORTRAN permite a declaração de vetores e matrizes da mesma forma que em Pseudocódigo. A única diferença está na apresentação das dimensões dos vetores e/ou matrizes.


```
IdArq <- AbraArq ("CAD.DAT",  
                  'cn', stat)
```

```
IdArq = 30  
OPEN (IdArq, FILE='CAD.DAT',  
      STATUS='NEW', IOSTAT=stat)
```

8.2 FecheArq

O comando para fechamento de arquivos em FORTRAN é o **CLOSE**. Exemplo:

```
FecheArq (IdArq)
```

```
CLOSE (IdArq)
```

8.3 LeiaArq

Em FORTRAN, utiliza-se o mesmo comando para leitura do teclado (READ) para se ler um arquivo. Exemplo:

```
LeiaArq (IdArq, stat) num
```

```
READ (IdArq, *, IOSTAT=stat) num
```

8.4 EscrevaArq

Em FORTRAN, utiliza-se o mesmo comando para escrita na tela (WRITE) para se escreve em um arquivo. Exemplo:

```
EscrevaArq (IdArq, stat) num
```

```
WRITE (IdArq, *, IOSTAT=stat) num
```

8.5 RebobineArq

Mesma estrutura em ambos os casos. Exemplo:

```
RebobineArq (IdArq, stat)
```

```
REWIND (IdArq, IOSTAT=stat)
```

8.6 VolteArq

Mesma estrutura em ambos os casos. Exemplo:

```
VolteArq (IdArq, stat)
```

```
BACKSPACE (IdArq, IOSTAT=stat)
```