PUC

SÉRIE: MONOGRAFIAS EM CIÊNCIA DA COMPUTAÇÃO

Nº 19/78

FORTS: UM PREPROCESSADOR PARA FORTRAN-IV ESTRUTURADO

por

RUBENS N. MELO
P.L. STEINBRUCH

DEPARTAMENTO DE INFORMÁTICA

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 – CEP-22453
RIO DE JANEIRO – BRASIL

SÉRIE: MONOGRAFIAS EM CIÊNCIA DA COMPUTAÇÃO

Nº 19/78

EDITOR DA SÉRIE: Michael F. Challis Dezembro, 1978

FORTS: UM PREPROCESSADOR PARA FORTRAN-IV ESTRUTURADO*

por

RUBENS N. MELO

P.L. STEINBRUCH

* Este relatório faz parte do projeto "Ferramentas para o projeto e construção de sistemas de informação apoiados em bancos de dados" coordenado pelo Prof. Rubens N. Melo e parcialmente financiado pelo contrato TC 22.22.0060/77 do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

RESUMO

O FORTRAN Estruturado (FORTS) foi desenvolvido na PUC-RJ [1] a partir de um processador de macros de proposito geral [2].

A finalidade precípua do desenvolvimento desta linguagem foi a de prover novas estruturas de controle ao FORTRAN IV a fim de permitir maior facilidade no desenvolvimento de programas usando conceitos de programação estruturada.

O presente relatório tem por objetivo apresentar uma visão geral das facilidades oferecidas pelo FORTS e servir de um pequeno "guia" prático de utilização deste preprocessador.

PALAVRAS-CHAVE

FORTS, FORTRAN, PROGRAMAÇÃO ESTRUTURADA, MACROS

SUMÁRIO

ABSTRACT

FORTS (Structured FORTRAN) was developed at PUCRJ [1] from a general purpose macro-generator [2].

This extention to FORTRAN-IV permits the development of programs according to the concepts of structured programming.

This report overviews the FORTS facilities and may be used as a "User's Guide" of the system.

KEYWORDS

FORTS, FORTRAN, STRUCTURED PROGRAMMING, MACRO-GENERATOR.

I. INTRODUÇÃO

Provavelmente uma das linguagens mais utilizadas, ainda hoje, é o FORTRAN [4,5]. Isto sem mencionar os diversos programas existentes codificados em FORTRAN e em seus diversos dialetos. Isso se deve principalmente as características do FORTRAN, que além de ser uma linguagem relativamente simples, é extremamente eficiente e tem um alto índice de portabilidade. No entanto, o FORTRAN não apresenta estruturas de controle adequadas para implementar programas segundo os conceitos de programação estruturada.

Destes pontos, surgiu a idéia de implementar certas estruturas de controle em FORTRAN; e, desta idéia, surgiu o FORTS.

O FORTS é essencialmente um processador de macros [2] de propósito geral onde a sintaxe das chamadas de macros estão orientadas para o FORTRAN. Os novos comandos do FORTRAN estruturado são chamadas de macros que geram comandos em FORTRAN padrão.

Para evitar o ineficiente processo de pesquisar todo o texto de entrada, o FORTS apenas analisa os registros que contêm os novos comandos que são identificados rapidamente pela marca "underscore" que precede seus nomes. A única rotina do sistema que é orientada para a linguagem base FORTRAN, processa então este registro transformando-o numa chamada de macro padrão e visto do processador de macros tudo se passa como ele estivesse processando um texto qualquer entremeado com chamadas de macro.

Do exposto acima todas as aplicações típicas do processador de macros [3] são mantidas em FORTS. Orientando estas aplicações para o contexto do programa podemos por exemplo:

a. definir novos comandos locais ao programa ou publicos, isto é, podemos catalogá-los numa biblioteca de comandos (macros) de uso público,

b. alterar o código gerado por certos comandos novos o que nos permite procurar experimentalmente as melhores estruturas,

c. Repetir o mesmo código em vários pontos do programa. Por exemplo declarações de variáveis que devem ser gerais em vários subprogramas ou trechos de programa semelhantes exceto em algumas partes que podem ser passados como parâmetros, etc.

Exemplos destas aplicações são apresentadas na seção III deste relatório.

II. ESTRUTURAS DE CONTROLE EM PROGRAMAÇÃO

"A programação estruturada é uma disciplina de programação que visa permitir uma boa documentação, maior facilidade para provas de correção, clareza de codificação e facilidade para depuração e manutenção de programas." [6]

Para atingir os objetivos a que se propõe, a programação estruturada utiliza algumas estruturas de controle, e, dentre elas, as que estão implementadas no FORTS podem ser classificadas em tres grupos:

GRUPO A) As que expressam alternativas para o fluxo de processamento.

- Estrutura IF para decisões entre duas alternativas.
- Estrutura DOCASE para decisões entre várias alternativas.

GRUPO B) As que expressam repetições da execução de blocos de comandos.

- Estrutura DOWHILE que expressa a repetição de um bloco de comandos enquanto certa condição continuar verdadeira.
- Estrutura REPEAT que expressa a repetição de um bloco de comandos até que certa condição seja satisfeita.
- Estrutura FOR que expressa a repetição de um bloco de comandos controlada por uma seqüência indicada de valores.

GRUPO C) As que permitem a organização do programa em blocos que executam tarefas delimitadas.

- Estrutura PERFORM que permite a execução de um bloco de comandos correspondente a uma seção do programa, de maneira remota.

A seguir apresentaremos estas estruturas de controle e a maneira como estão implementadas no FORTS. Supomos que o leitor está familiarizado com o uso de diagramas de blocos para descrever a estrutura de programa.

II.1 - Os comandos do FORTS

A) Comandos que Implementam as Estruturas do Grupo A

A.1. O comando IF

Proposito: O comando IF expressa uma decisão entre duas alternativas para o processamento baseado no valor de uma expressão logica.

Forma: Há duas formas para o comando IF

A) IF-THEN-ELSE

```
_IF(C)
  _THEN
    B1
  _ELSE
    B2
  _FI
```

B) IF-THEN

```
_IF(C)
  _THEN
    B1
  _FI
```

onde C é uma expressão lógica, B1 e B2 são blocos de comandos.

Significado: A Fig. 1 ilustra o processamento do comando IF e mostra o significado das duas cláusulas.

No caso mais geral (fig. 1a) se a expressão C é verdadeira, então o processamento segue pelo bloco B1 senão pelo bloco B2. Após a execução de um dos dois blocos o processamento continua no comando logo após o FI.

Caso mais simples (fig. 1b) se a expressão C é verdadeira, o bloco B1 é executado senão o processamento apenas continua no próximo comando após o FI.

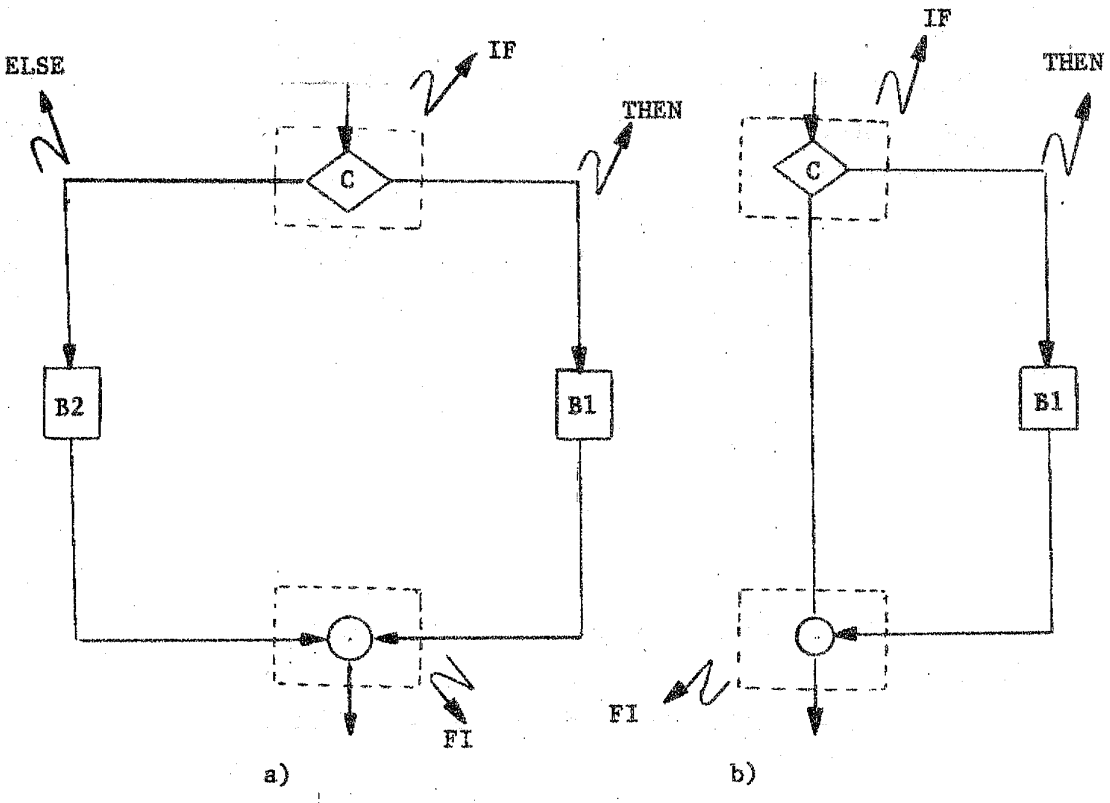

Figura 1. Ilustração do comando IF

Exemplo: Se SALARY > 10000 calcular TAXA = 10% de SALARY e DESC = 12% de SALARY do contrário calcular TAXA = 8% de SALARY e definir DESC = 200. Neste último caso se TAXA > DESC corrigir TAXA e DESC para cada um.

```
_IF(SALARY.GT.10000.0)
 _THEN
  TAXA = 0.10*SALARY
  DESC = 0.12*SALARY
 _ELSE
  TAXA = 0.08*SALARY
  DESC = 200.0
 _IF(TAXA.LT.DESC)
  _THEN
   TAXA = 100.0
   DESC = 100.0
  _FI
_FI
```

Observações:

1. Não são permitidos GO TOs para as cláusulas THEN, ELSE e FI.

2. As cláusulas THEN, ELSE e FI devem vir sozinhas na linha.

3. A cada FI corresponde obrigatoriamente um único IF.

4. A cláusula THEN pode ser omitida.

   Exemplo:

```
_IF(A.GT.0)
 B = A + 1
 C = A - 2
_FI
```

5. A cláusula ELSE é executável, portanto se for procedida por um comando de desvio provocará reclamações ("WARNINGS") do compilador.

A.2. O Comando DOCASE

Propósito: O comando DOCASE expressa uma seleção entre várias alternativas para o processamento.

Forma:

```
_DOCASE
 _CASE C1
      B1
 _ESAC
 _CASE C2
      B2
 _ESAC
    ...
 _CASE Cn
      Bn
 _ESAC
_ESACOD
```

onde C1, C2, ... Cn são expressões lógicas e B1, B2, ... Bn são blocos de comandos. Apenas um desses blocos de comando poderá ser executado. O bloco selecionado será o primeiro cuja expressão lógica é verdadeira.

Significado: A FIG. 2 ilustra o processamento do comando DOCASE e mostra o significado de suas cláusulas.

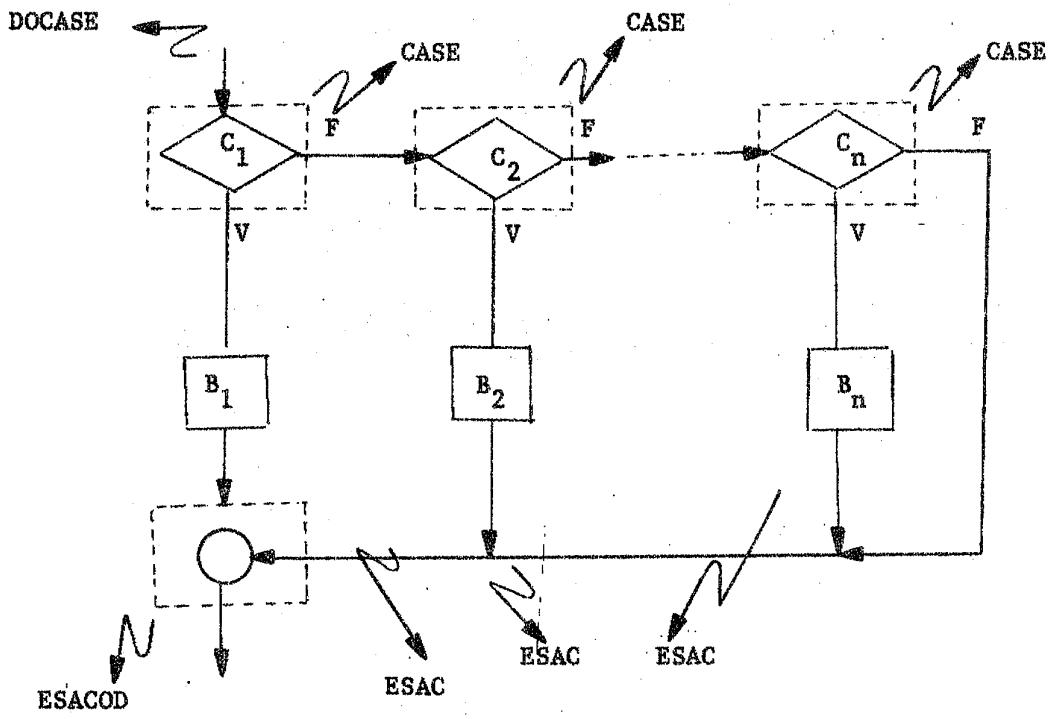

Fig. 2. Ilustração do Comando DOCASE

Cada condição C1, C2, ...Cn é testada em sequência até que se encontre a primeira verdadeira. O bloco de comandos correspondente a expressão é executado e em seguida o processamento continua no primeiro comando após a cláusula ESACOD.

Caso nenhuma condição seja satisfeita, o processamento apenas continua no primeiro comando após a cláusula ESACOD. Note que a condição "nenhum dos casos acima" pode ser simulada de maneira simples definindo o último caso como CASE(.TRUE.).

Exemplos: Ler um par de números reais (X,Y) representando um ponto num plano e dependendo do quadrante a que pertence, efetuar procedimentos diferentes. No caso do ponto cair num dos eixos, imprimir mensagem 'ERRO'.

```
      READ(5,10)X,Y
10    FORMAT(2F10.5)
      _DOCASE
       _CASE(X.GT.0.AND.Y.GT.0)

          procedimento 1

       _ESAC
        _CASE(X.GT.0.AND.Y.LT.0)

          procedimento 2
          .
       _ESAC
      _CASE(X.LT.0.AND.Y.GT.0)

          procedimento 3

       _ESAC
        _CASE(X.LT.0.AND.Y.LT.0)

          procedimento 4

       _ESAC
      _CASE(.TRUE.)
        WRITE(6,11)
11    FORMAT('ERRO')
      _ESAC
     _ESACOD
```

Observações:

1. Não são permitidos GO TOs para as cláusulas CASE, ESAC e ESACOD.

2. A cláusula EXITC pode ser usada dentro do bloco DOCASE como um comando de desvio para o fim do CASE (ESAC).

```
Exemplo:  _DOCASE
          _CASE(A.GT.0)
           --------
           IF(B.LT.A)_EXITC
          _ESAC
           ========
         _ESACOD
```

3. A cláusula DOCASE sempre será seguida de uma cláusula CASE e a cláusula ESACOD será sempre procedida de uma cláusula ESAC.

4. Vide observações gerais.

B) Comandos que Implementam as Estruturas do Grupo B

B.1. O Comando DOWHILE

Propósito: O comando DOWHILE expressa a repetição de um bloco de comandos, enquanto certa condição continuar verdadeira.

```
Forma:  _DOWHILE(C)
                  B
        _OD
```

onde C é uma expressão lógica e B é um bloco de comandos.

Significado: A Fig. 3 ilustra o processamento do comando DOWHILE e mostra o significado de suas cláusulas.

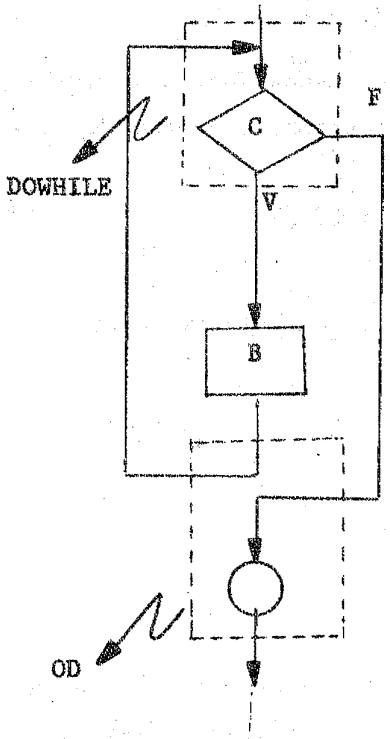

Fig. 3. Ilustração do Comando DOWHILE

Enquanto a condição C for verdadeira, o bloco de instruções B, será executado. Quando a condição C for falsa, o processamento continuará no próximo comando após a cláusula OD.

Exemplo: Leia uma sequência de números um a um, enquanto o valor lido for positivo, calcule a sua raiz quadrada, imprima o número lido e a raiz calculada e leia o próximo número.

```
      READ(5,20)X
20    FORMAT(F4.2)
      DOWHILE(X.GT.0)
        R = SQRT(X)
        WRITE(5,30)X,R
30      FORMAT(1H-,2F10.2)
        READ(5,20)X
      OD
```

Observações:

1. Não são permitidos GO TOs para a cláusula DOWHILE.

2. A cláusula EXITW pode ser usada dentro do bloco B de instruções como um comando de desvio para o primeiro comando após a cláusula OD.

```
Exemplo: _DOWHILE(Y.GT.X)
         ---------
         IF(X.LT.0)_EXITW(desvia para o comando READ, se X < 0)
        _OD
         READ(5,10)W
```

3. Vide observações gerais.

B.2. O comando REPEAT

Propósito: O comando REPEAT expressa a repetição de um bloco de comandos até que certa condição seja satisfeita.

```
Forma: _REPEAT
            B
       _UNTIL(C)
```

onde B é um bloco de comandos e C é uma expressão lógica.

Significado: A Fig. 4 ilustra o processamento do comando REPEAT e mostra o significado de suas cláusulas.

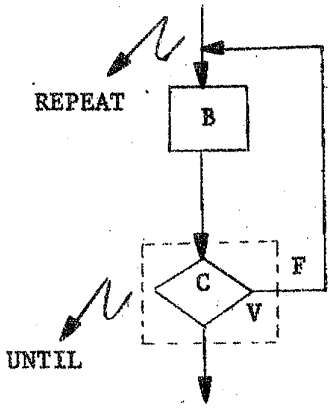

Fig. 4. Ilustração do comando REPEAT

O bloco B de instruções é executado até que a condição C seja verdadeira. Quando isto ocorre, o processamento continua no primeiro comando após a cláusula UNTIL.

Imprima o quadrado e o cubo dos números inteiros positivos até que o valor do cubo ultrapasse 10000.

```
      N=0
      QD=0
      CB=0
      REPEAT
      N=N+1
      QD=N**2
      CB=N**3
      WRITE(6,10)QD,CB
10    FORMAT(1H-,2I5)
      UNTIL(CB.GT.10000)
```

Observações:

1. Não são permitidos GO TOs para as cláusulas REPEAT ou UNTIL.

2. A cláusula EXITR pode ser usada dentro do bloco B de instruções, como um desvio para o primeiro comando após a cláusula UNTIL.

Exemplo:

```
      REPEAT
      ------
      IF(X.GT.0) EXITR (desvio para o comando READ se X > 0)
      ------
      UNTIL(Y.LT.X)
      READ(NRDR,10)V
      ----
```

3. Vide observações gerais.

B.3. O comando FOR

Propósito: O comando FOR expressa a repetição de um bloco de comandos controlada por uma sequência indicada de valores.

Forma: _FOR V _FROM R _TO S _BY T
B
_ROF

onde V é uma variável, R, S e T são expressões aritméticas e B é um bloco de instruções.

Significado: A Fig. 5 ilustra o processamento do comando FOR e mostra o significado de suas cláusulas.

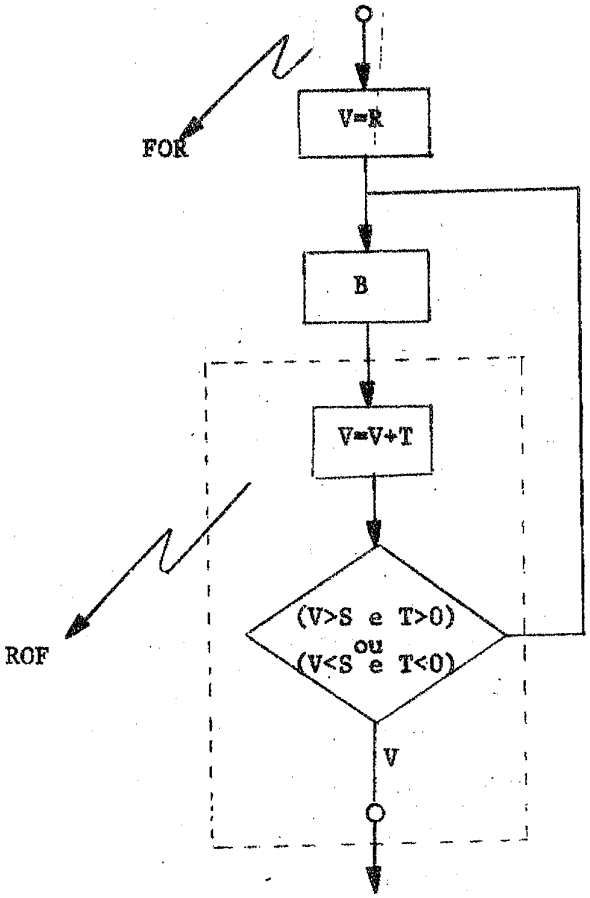

Fig. 5. Ilustração do Comando FOR

Inicialmente V é inicializado com o valor de R, o bloco B de instruções é executado, V é incrementado do valor de T e comparado com o valor de S.

Se o valor de V for maior do que o valor de S e o valor de T for maior do que zero (V > S e T > 0), ou, se o valor de V for menor do que o valor de S e o valor de T for menor do que zero (V < S e T < 0), o processamento continua no primeiro comando após o ROF.

Se o valor de V for menor ou igual ao valor de S ou o valor de T for menor ou igual a zero (V ≤ S ou T ≤ 0), e, se o valor de V for maior ou igual ao valor de S ou o valor de T for maior ou igual a zero (V ≥ S ou T ≥ 0), o bloco B de instruções é executado novamente, V é incrementado do valor de T e novamente é comparado com S. O número de vezes que este procedimento é repetido é:

$$\left\lfloor \frac{|S| - |R|}{T} \right\rfloor + 1$$

onde $|x|$ é o valor absoluto de x e $\lfloor x \rfloor$ é a parte inteira do valor de x.

Exemplo: Calcule o fatorial dos dez primeiros números inteiros positivos.

```
    FAT = 1
_FOR K _FROM 1 _TO 10 _BY 1
    FAT = FAT*K
    WRITE(6,10)K,FAT
    FORMAT(1H,T18,I2,T22,I10)
    _ROF
```

Observações:

1. Convêm lembrar que o incremento (T) pode ser negativo.

2. Não são permitidos GO TOs para as cláusulas FOR ou ROF.

3. A cláusula EXITF pode ser usada dentro do bloco B de instruções como um comando de desvio para o primeiro comando após a cláusula ROF.

Exemplo:

```
_FOR K _FROM 110 _TO X*2-1 _BY -5
  ------------
  IF (Y.GT.(2*X))_EXITF (desvia para o comando READ, se y > 2x)
  ------------
_ROF
 READ(5,100)Z
```

4. Vide observações gerais.

C. Os comandos que implementam as estruturas do Grupo C

C.1. O comando PERFORM

Propósito: O comando PERFORM juntamente com a definição de seções do programa através dos comandos SEC e ENDSEC permite a organização do programa em blocos que executam tarefas específicas. Com o uso adequado destes comandos, podemos estruturar o programa de maneira clara e simples.

Forma:

```
_PERFORM S          _SEC    S
-------               B
                    _ENDSEC S
                    --------
```

onde S deve ser uma variável INTEGER do FORTRAN e B é um bloco de comandos correspondente a seção de nome S.

Significado: A Fig. 6 ilustra o processamento do comando PERFORM e mostra o significado das outras cláusulas utilizadas.

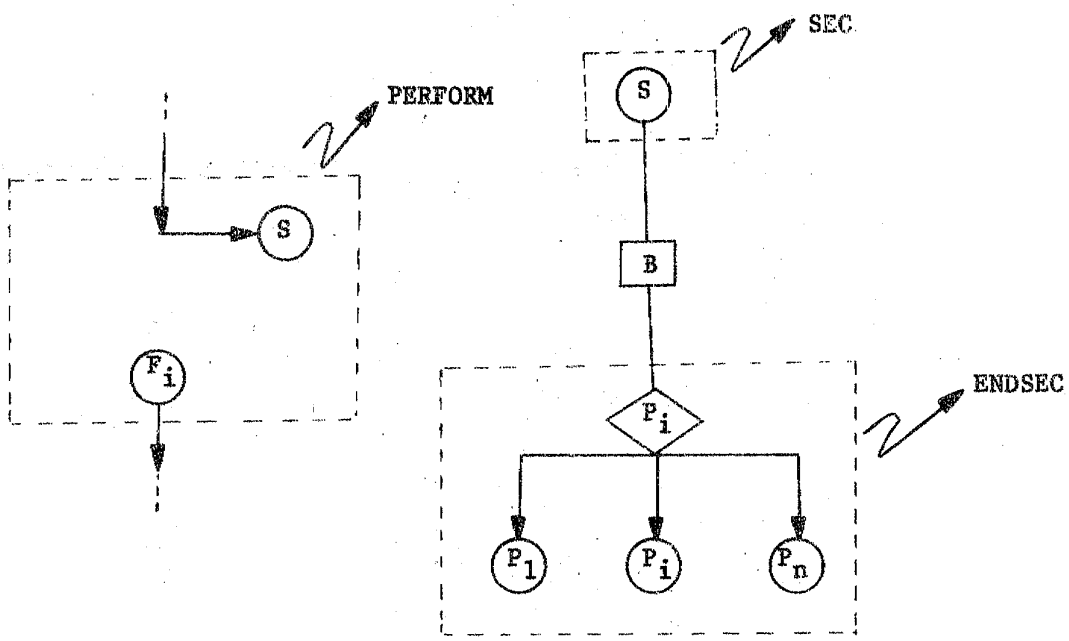

Fig. 6. Ilustração do Comando PERFORM

O comando PERFORM implica num desvio para o início da seção referenciada (S) e na definição de um rótulo (P_i) correspondente a posição de retorno após a execução do bloco B da seção. Note que a cláusula ENDSEC corresponde a um desvio para a posição de retorno correspondente ao último PERFORM executado.

Exemplo:

Fazer um bloco de programa (uma seção) que lê um número inteiro e caso ocorra fim de arquivo de entrada, define o número como zero e liga uma chave CH.

```
----------                  ----------
PERFORM LER                 _SEC'    LER
----------          10         READ(5,10,END=1000)NUM
                               FORMAT(I10)
                               _EXITS
                    1000           NUM=0
                                   CH=.TRUE.
                            _ENDSEC
```

Observações:

1. A cláusula EXITS pode ser usada dentro do bloco B da seção como um desvio para seu fim (ENDSEC).

```
Exemplo: _SEC     S2
         -------
         IF(X.GT.0)_EXITS
         --------
         --------
         _ENDSEC S2
```

2. A definição de uma seção deve ser precedida (no texto) de pelo menos uma referência a ela através de comando PERFORM.

3. Seções encaixantes não são permitidas.

4. O nome de seção deve ser uma variável INTEGER simples do FORTRAN.

5. Os comandos PERFORM, SEC e ENDSEC devem vir sozinhos na linha onde são usados.

6. As seções devem de preferência serem definidas logo antes do comando END do programa.

7. Vide observações gerais.

II.2. Observações Gerais

A seguir faremos algumas observações de ordem geral a respeito do FORTS.

1. O FORTS deteta alguns erros na sintaxe de suas instruções. Os erros detetados se referem a formação da chamada de macro correspondente ao comando. Deste modo erros da sintaxe do FORTRAN serão detetados somente na fase de compilação.

2. Os números de comandos de usuário não deverão ultrapassar 80000. Os números de comando acima de 80000 são reservados para o pre-processador.

3. O usuário pode opcionalmente obter listagem do programa fonte em FORTS bem como do programa FORTRAN gerado. As opções são escolhidas pelos parâmetros do pre-processador. (Vide informações sobre parâmetros no Capítulo IV).

4. No programa fonte em FORTS pode-se usar um caracter '?' na primeira coluna de um cartão como um comando para "passar para a próxima folha" na listagem, do programa fonte.

III. OUTRAS FACILIDADES DO FORTS

III.1. Definição de Macros

Em FORTS pode-se definir macros que correspondam a novos comandos ou a textos auxiliares que são usados com pequenas mudanças repetidas vezes no programa fonte.

Como definir uma nova macro

Para definir uma macro de nome M e cujo texto de definição seja D usa-se o seguinte comando:

```
$DEFINE!M!<D>;
```

Em geral, o texto que define o macro é grande e não cabe num só registro de entrada. Precisamos saber então como escrever um texto que continue em vários registros. Para isto basta saber o seguinte:

Um registro contendo chamadas de macro deve começar por um sinal '%' na coluna 1.

O texto deste registro que vai da coluna 2 até a coluna 72 ou até um sinal '#' é examinado pelo FORTS. Por isso pode-se usar o resto do registro após o '#' para comentários.

Exemplo:

```
%$DEFINE!MAC1!<# ...comentários...
%...texto de definição...>;# ...comentários...
```

. O texto que define a macro deve vir entre os sinais '<' e '>'. Normalmente este texto consistirá de comandos FORTRAN cujas partes que serão passadas como parâmetros são indicadas por ' a'. Para indicar o fim de cada comando FORTRAN no texto usamos o sinal '/'.

Exemplo 1

```
%$DEFINE!ARQ!<# ..comando para definir arquivo..
%      INTEGER V /#
%      DEFINE FILE  a  ( a 2,  a 3, U, V')/#
%>;#
```

No programa fonte a chamada

```
_ARQ!10!20!500
```

geraria

```
INTEGER V
DEFINE FILE 10(20,500,U,V001)
```

Esta mesma definição poderia ser feita num só registro da seguinte maneira:

```
%$DEFINE!ARQ!<  INTEGER V /    DEFINE FILE  a ( a 2,  a 3, U,V)/>;#
```

Exemplo 2

Vamos definir uma macro que gera um texto usado em vários pontos do mesmo programa por exemplo a definição de uma área de COMMON.

```
%$DEFINE!AREA!</#
%      COMMON A1(1000)  B1(200)/#
%      INTEGER A1,B1/#
%>;#
```

No programa esta macro seria usada por exemplo, da seguinte maneira:

```
-----
_AREA
-----
-----
END
SUBROUTINE SUB1(-----)
_AREA

END
SUBROUTINE SUB2(-----)
_AREA

END
```

Maiores detalhes necessários para a definição de macros mais significativas podem ser obtidos em [3].

III.2. Atualização da Definição de Macro

A definição de uma macro pode ser alterada desde que não aumente o texto de definição. O comando para a atualização de macro M é:

```
$UPDADE!M!<D>;
```

onde M é o nome da macro já definido e D é o novo texto de definição.

III.3. Exclusão de Macro

Uma macro pode ser definida pelo comando DEFINE e redefinida várias vezes pelo mesmo comando DEFINE. A referência a uma macro é sempre a sua última versão. Esta última versão pode ser excluída do sistema através do seguinte comando:

```
$DELETE!M;
```

ou no caso de exclusão de várias macros diferentes de uma vez só:

$DELETE!M_1!M_2!...M_n;

Para cada nome dado apenas a última versão (que pode ser a única) é excluída.

III.4. A Biblioteca de Macros Permanentes

Certas macros podem ser definidas de maneira bem geral para servirem a vários programas e neste caso devemos guardá-la numa biblioteca em disco para uso posterior. Esta biblioteca é gerenciada por utilitário externo e por dois comandos especiais do FORTS.

Como catalogar uma macro na biblioteca

Após definida pelo comando DEFINE a macro M é catalogada na biblioteca pelo comando:

$SYST!M;

ou $SYST!M_1!M_2!...!M_n;

no caso de catalogação de várias macros de uma vez.

Como excluir uma macro da Biblioteca

Para excluir a macro M previamente catalogado por um $SYST usamos o comando:

$DSYST!M;

ou $DSYST!M_1!M_2!...!M_n;

no caso da exclusão de vários macros de uma vez.

Como modificar uma macro na Biblioteca

A modificação de uma macro não é feita diretamente na Biblioteca. O usuário tem que atualizar a definição de macro na memória (através do comando UPDATE), excluir a macro da Biblioteca e catalogá-la novamente. Por exemplo para modificar a macro M catalogada na Biblioteca devemos fazer os seguintes passos:

```
$UPDATE!M! novo texto; modifica M
$DSYST!M;  exclui a versão de M da Biblioteca
$SYST!M;   insere M na Biblioteca
```

Como usar as macros de uma certa Biblioteca

O sistema permite usar apenas uma Biblioteca de macros de cada vez. Para usar uma certa Biblioteca gravada no data-set X do sistema basta asssociar o arquivo FT04F001 do programa FORTS ao data-set X. O usuário poderá então utilizar as macros catalogadas nesta Biblioteca.

IV. PROCEDIMENTOS PARA USO DO FORTS

IV.1. Caracteres de Controles

Após a transformação dos comandos estruturados para chamadas de macros, o processador analisa caracter por caracter do texto e toma certas decisões quando certos caracteres de controle são encontrados. Estes caracteres de controles são passados ao processador em um único registro de dados no arquivo FT08F001.

A explicação detalhada de cada caracter de controle está fora do escopo deste relatório. Entretanto alguns deles podem ser entendidos neste contexto.

Os caracteres de controle nos exemplos deste manual corresponderiam ao arquivo FT08F001 com um único registro contendo

```
< $ ! ;  a ' > #  % / _ 10000
```

< /e >	delimitadores das definições de MACRO
$	inicia uma chamada de MACRO
!	separa parâmetros das chamadas de MACRO
;	indica o fim de uma chamada de MACRO
#	fim lógico do REGISTRO DE ENTRADA
@	indica um parâmetro formal na definição de MACRO
/	fim lógico do REGISTRO DE SAÍDA
%	usado na coluna 1 para ativar o processador de MACROS
_	identifica um comando próprio do FORTS

Os últimos digitos (10000), em seguida aos caracteres de controle significam:

. Digito que indica se o texto expandido (FORTRAN) deve ser armazenado (1) ou não (0) no arquivo FT10F001.

. Digito que indica as opções Listagem

0 não lista nem FORTS nem FORTRAN

1 lista a chamada de MACRO gerada e o texto FORTRAN correspondente a sua expansão

3 só lista texto em FORTS.

. Os 3 últimos digitos indicam o valor inicial do gerador de RÓTULOS dos COMANDOS FORTRAN gerados pelo FORTS.

Os caracteres de controle podem ser mudados, porém devem ser escolhidos com cuidado e devem estar fora do conjunto de caracteres da linguagem base; entretanto, isto é muito difícil conseguir se considerarmos os comentários, strings, etc. Alguns cuidados portanto devem ser tomados nos registros do programa fonte contendo comandos estruturados ou chamadas de MACRO para que caracteres de controles não sejam usados sem intenção.

Por exemplo no comando

```
_IF (X.GT.0) CALL SUB ('STRING')
```

O caracter ' seria interpretado como controle e resultaria num parâmetro inválido na chamada da rotina SUB.

IV.2. USO EM MODO BATCH

PROCEDURES DE JCL

Para o processamento BATCH existem 4 procedures:

1. FORTSC - Pré-processa, compila o programa fonte (FIG.7)
2. FORTSCL - Pré-processa, compila e linkedita (FIG.8)
3. FORTSCLG - Pré-processa, compila, linkedita e executa (FIG.9)
4. FORTSJ - Pré-processa, compila, linkedita-executa (FIG.10)

EXEMPLO DE USO DO FORTS

```
//..... JOB .....
//....EXEC   FORTSCLG
//SYSIN DD   *
     PROGRAMA FONTE
//
```

```
//FOETSC   .PROC
//*************************************************
//***** MACRO, PREPARA O PGM P/ O COMPILADOR ****
//*************************************************
//FORTS EXEC PGM=MACF1,REGION=150K
//STEPLIB DD DSN=PBTX.CCE.SCHEMA,DISP=SHR
//FT08F001 DD DSN=SY S2.PARMLIB(FORTSPUC),DISP=SHR,LABEL=(,,,IN)
//FT06F001 DD SYSOUT=A
//FT04F001 DD DSN=PBTX.DIGBENH.MACRO,DISP=SHR
//FT10F001 DD DSN=&&TEMPP,DISP=(NEW,PASS),UNIT=SYSDA,DCB=BLKSIZE=80,
// SPACE=(80,(500.100))
//FT05F001 DD DDNAME=SYSIN
//FORT     EXEC PGM=IEYFORT,PARM='NOSOURCE',REGION=150K
//*************************************************
//*****   COMPILADOR FORTRAN  *********************
//*************************************************
//SYSIN DD DSN=&&TEMPP,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSLIN DD DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,
//           SPACE=(80,(200,100),RLSE),DCB=BLKSIZE=80
```

FIGURA 7 - PROCEDURE FORTSC

```
//FORTSCL   PROC
//*************************************************
//***** MACRO, PREPARA O PGM P/ O COMPILADOR ****
//*************************************************
//FORTS EXEC PGM=MACF1,REGION=150K
//STEPLIB DD DSN=PBTX.CCE.SCHEMA,DISP=SHR
//FT08F001 DD DSN=SYS2.PARMLIB(FORTSPUC),DISP=SHR,LABEL=(,,,IN)
//FT06F001 DD SYSOUT=A
//FT04F001 DD DSN=PBTX.DIGBENH.MACRO,DISP=SHR
//FT10F001 DD DSN=&&TEMPP,DISP=(NEW,PASS),UNIT=SYSDA,DCB=BLKSIZE=80,
// SPACE=(80,(500,100))
//FT05F001 DD DDNAME=SYSIN
//FORT      EXEC PGM=IEYFORT,PARM='NOSOURCE',REGION=150K
//*************************************************
//*****   COMPILADOR FORTRAN  ********************
//*************************************************
//SYSIN DD DSN=&&TEMPP,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSLIN  DD DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,
//           SPACE=(80,(200,100),RLSE),DCB=BLKSIZE=80
//LKED EXEC PGM=IEWL,REGION=96K,PARM=(NOXREF),COND=(4,LT,FORT)
//*************************************************
//*****   LINK-EDITOR          ********************
//*************************************************
//SYSLIB  DD DSN=SYS1.FORTLIB,DISP=SHR
//         DD DSN=PBTX.DIGBENH.DBLIB,DISP=SHR
//         DD DSN=PBTX.DIGBENH.SLIB,DISP=SHR
//SYSLMOD  DD DSNAME=&GOSET(MAIN),DISP=(MOD,PASS),UNIT=SYSDA,
//        SPACE=(1024,(20,10,1),RLSE),DCB=BLKSIZE=1024
//SYSPRINT DD SYSOUT=A
//SYSUTI DD UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),DCB=BLKSIZE=1024,
//        DSNAME=&SYSUTI
//SYSLIN  DD DSNAME=&LOADSET,DISP=(OLD,DELETE)
//        DD DDNAME=SYSIN
```

FIGURA 8 - PROCEDURE FORTSCL

```
//FORTSCLG   PROC
//*************************************************************
//***** MACRO, PREPARA O PGM P/ O COMPILADOR ****
//*************************************************************
//FORTS EXEC PGM=MACF1,REGION=150K
//STEPLIB DD DSN=PBTX.CCE.SCHEMA,DISP=SHR
//FT08F001 DD DSN=SYS2,PARMLIB(FORTSPUC),DISP=SHR,LABEL=(,,,IN)
//FT06F001 DD SYSOUT=A
//FT04F001 DD DSN=PBTX.DIGBENH.MACRO,DISP=SHR
//FT10F001 DD DSN=&&TEMPP,DISP=(NEW,PASS),UNIT=SYSDA,DCB=BLKSIZE=80,
//  SPACE=(80,(500,100))
//FT05F001 DD DDNAME=SYSIN
//FORT       EXEC PGM=IEYFORT,PARM='NOSOURCE',REGION=150K
//*************************************************************
//*****   COMPILADOR FORTRAN  ********************
//*************************************************************
//SYSIN DD DSN=&&TEMPP,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSLIN  DD DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,
//            SPACE=(80,(200,100),RLSE),DCB=BLKSIZE=80
//LKED EXEC PGM=IEWL,REGION=96K,PARM=(NOXREF),COND=(4,LT,FORT)
//*************************************************************
//*****   LINK-EDITOR          ********************
//*************************************************************
//SYSLIB  DD DSN=SYS1,FORTLIB,DISP=SHR
//         DD DSN=PBTX.DIGBENH.DBLIB,DISP=SHR
//         DD DSN=PBTX.DIGBENH.SLIB,DISP=SHR
//SYSLMOD  DD DSNAME=&GOSET(MAIN),DISP=(MOD,PASS),UNIT=SYSDA,
//         SPACE=(1024,(20,10,1),RLSE),DCB=BLKSIZE=1024
//SYSPRINT  DD SYSOUT=A
//SYSUTI DD UNIT=SYSDA,SPACE=(1024,(100,10),RLSE),DCB=BLKSIZE=1024,
//        DSNAME=&SYSUTI
//SYSLIN  DD DSNAME=&LOADSET,DISP=(OLD,DELETE)
//        DD DDNAME=SYSIN
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((4,LT,FORT),(4,LT,LKED)),REGION=150K
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD SYSOUT=A
//FT07F001 DD SYSOUT=B
```

FIGURA 9 - PROCEDURE FORTSCLG

```
//FORTSJ   PROC IMSL=S
//*************************************************
//***** MACRO, PREPARA O PGM P/ O COMPILADOR ****
//*************************************************
//FORTS EXEC PGM=MACF1,REGION=150K
//STEPLIB DD DSN=PBTX.CCE.SCHEMA,DISP=SHR
//FT08F001 DD DSN=SYS2,PARMLIB(FORTSPUC),DISP=SHR,LABEL=(,,,IN)
//FT06F001 DD SYSOUT=A
//FT04F001 DD DSN=PBTX.DIGBENH.MACRO,DISP=SHR
//FT10F001 DD DSN=&&TEMPP,DISP=(NEW,PASS),UNIT=SYSDA,DCB=BLKSIZE=80,
// SPACE=(80,(500,100))
//FT05F001 DD DDNAME=SYSIN
//FORT      EXEC PGM=IEYFORT,PARM='NOSOURCE',REGION=150K
//*************************************************
//*****   COMPILADOR FORTRAN  ********************
//*************************************************
//SYSIN DD DSN=&&TEMPP,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD SYSOUT=B
//SYSLIN  DD DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,
//           SPACE=(80,(200,100),RLSE),DCB=BLKSIZE=80
//GO  EXEC  PGM=LOADER,COND=(4,LT,FORT),REGION=100K
//*************************************************
//*****     LINK-EDITOR E EXECUCAO            ****
//*************************************************
//SYSLIB  DD  DSNAME=SYS1,FORTLIB,DISP=SHR
//         DD DSNAME=PROG,N036,FORTLIB,DISP=SHR
//         DD DSNAME=PROG.N110.IMSL&IMSL,DISP=SHR
//         DD DSNAME=PROG.N111.HARWELL,DISP=SHR
//        DD DSNAME=PROG.N110.IMSLS,DISP=SHR
//        DD DSNAME=PBTX.DIGBENH.DBLIB,DISP=SHR
//        DD DSNAME=PBTX.DIGBENH.SLIB,DISP=SHR
//SYSLOUT  DD  SYSOUT=A
//SYSLIN  DD  DSNAME=*.FORT.SYSLIN,DISP=(OLD,DELETE)
//FT05F001  DD  DDNAME=SYSIN
//FT06F001  DD  SYSOUT=A
//FT07F001  DD  SYSOUT=B
```

FIGURA 10 - PROCEDURE FORTSJ

IV.3. USO VIA TSO/IBM

COMANDO: TFORTS

TFORTS - Pré-processa, compila, linkedita e executa

FORMATO:

```
                                                          new
   TFORTS nome-do-arquivo IN (DADOS) MOD(CARGA) DSP  old   MEMBER(MB1)
                                                          shr
```

NOME-DO-ARQUIVO - arquivo que contém o programa fonte

DADOS - arquivo de dados para o programa

DEFAULT: * (terminal)

CARGA - nome do arquivo que conterá o módulo de carga (saída linkeditor).

DEFAULT: xmod.load

DSP - disposição do arquivo

NEW - deverá ser criado

OLD - já existe

SHR - já existe, e poderá ser compartilhado

DEFAULT: SHR

MB1 - nome do membro do arquivo definido por MOD, onde está o módulo de carga

DEFAULT: GO

EXEMPLO:

```
TFORTS 'PBTX.M818000.CNTL(FORTS)' MOD(CARGA) DSP(NEW)
```

O programa Fonte é o membro FORTS do arquivo 'PBTX.M818000.CNTL'. Na linkedição será criado um arquivo 'Id-do-usuário.carga.load' com o membro 'GO'. O programa na fase de execução lerá dados pelo terminal.

OBS.: TFORTS - Tem como restrição o uso do arquivo FT10F001.

REFERÊNCIAS

[1] Melo, Rubens N. & Schwabe, Daniel - Extending the control structures of FORTRAN via a macro-generator - TR 1/76 - PUC/RJ - 1976.

[2] Melo, Rubens N. - "PM - Um processador de macros" - Tese de Mestrado ITA - 1971.

[3] Melo, Rubens N. & Steinbruch, P.L. - Algumas aplicações para um macro-expansor - TR/79 - PUC/RJ - A ser publicado.

[4] IBM - FORTRAN IV Language - GC28-6515.

[5] IBM - FORTRAN IV Programer's guide - GC28-6817.

[6] Steinbruch, P.L. & Fresneda, P.S.V - Novos aspectos em modularização de programas - Revista Brasileira de Tecnologia - Vol. 7 - 1976.

[7] Melo, Rubens N. - Implementing character string in FORTRAN - TR/76 - PUC"RJ - 1976.

[8] IBM - TSO Command Language - GC28-6732.

[9] Stanton, Michael A. - Manual de uso do IBM 370/165, DI-PUC/RJ, 1978.

[10] IBM - JCL Reference - GC28-6704.

[11] IBM - JCL User's Guide - GC28-6703.