MESTRA: UMA METODOLOGIA DE ESCRITA DE DOCUMENTAÇÃO INTERNA TRANSPORTÁVEL

CAMILO DE LELIS GONDIM MEDEIROS

UNIVERSIDADE FEDERAL DA PARAÍBA
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
AV. APRÍGIO VELOSO, S/N - BODOCONGÓ
58.100 - CAMPINA GRANDE - PB - BRASIL

RESUMO: Este trabalho apresenta uma Metodologia de Programação voltada para a escrita de comentários transportáveis a serem incluídos em programas que serão submetidos a um Sistema Transformacional para a derivação de suas várias versões. Aqui são também apresentados os conceitos de comentários constantes e comentários variáveis. Um pré-processador para conversão automática dos comentários variáveis é delineado.

1. INTRODUÇÃO

Embora o desenvolvimento de Software de Análise Numérica seja feito tradicionalmente utilizando a Linguagem de Programação FORTRAN, a sua transportabilidade nem sempre é assegurada. Para solucionar os problemas de portabilidade, uma solução frequentemente adotada consiste em efetuar manualmente todas as adaptações até que o software produza os resultados desejados. Uma outra solução, modernamente adotada, para o mesmo problema, é a utilização dos Sistemas Transformacionais que, em geral, transformam programas FORTRAN previamente padronizados.

A redução dos custos de desenvolvimento e o grau de confiabilidade do software assim produzido são os dois aspectos que mais encorajam a utilização dos Sistemas Transformacionais. Para que um Sistema Transformacional tenha ampla aceitação é aconselhável que ele exija o mínimo possível de padronização na escrita dos programas que serão transformados. Alterar automaticamente comentários, fornecer os programas transformados com uma boa formatação, mudar a precisão da aritmética utilizada e efetuar otimizações em programas fonte, são outras características importantes dos Sistemas Transformacionais.

Enquanto que os Sistemas Transformacionais foram amplamente abordados por [1], [3], [4], [9], [10], [12] e [22]; Transformações e Transportabilidade de programas FORTRAN foram discutidos por [2], [5]-[8], [11], [13], [14], [16]-[19], [23], [24], [26] e [27]; a padronização e a documentação de programas foram abordados por [15], [25] e [28], entre outros: a portabilidade da documentação interna foi superficialmente discutida em [20] e será tratada, de maneira mais detalhada, no presente trabalho. A seguir apresentamos uma disciplina de programação, que chamamos de MESTRA (Metodologia de Escrita de Documentação Interna Transportável), que visa facilitar a obtenção de programas transportáveis inclusive quanto à documentação interna.

As proposições aqui efetuadas levam em consideração que a linguagem FORTRAN é utilizada no desenvolvimento dos programas, podendo, no entanto, ser adaptadas para outras linguagens de programação de alto nível sem maiores problemas, e supõem a existência de um pré-processador para alterar os comentários existentes nas unidades de programa.

A utilização dos comentários na forma em que estamos sugerindo é justificável quando várias versões devem ser produzidas a partir de uma mesma unidade de programa, quer por razões de mudança da precisão aritmética utilizada, quer por razões de mudança de ambiente de computação ou até mesmo pela introdução de otimizações. Esta forma de comentários está sendo utilizada atualmente na Biblioteca Transportável de Análise Numérica (BITAN) da Universidade Federal da Paraíba, em Campina Grande, onde existe um pré-processador chamado Sistema Transformador de Programas FORTRAN (TRAPO) que, além de desempenhar outras funções, altera automaticamente os comentários das unidades de programa, quando necessário (veja [20] e [21] para maiores detalhes).

2. UMA PADRONIZAÇÃO PARA ESCRITA DOS COMENTÁRIOS.

Na MESTRA são utilizados dois tipos e duas classes de comentários os quais descrevemos a seguir. Quando possível todos estes comentários devem ser incluídos, na forma descrita, nas unidades de programa.

2.1. CLASSIFICAÇÃO DOS COMENTÁRIOS

Há duas classes de comentários em programas: os Comentários de Prólogo e os Comentários Explicativos (veja [28]). O conteúdo destes comentários fica a cargo da livre escolha do programador. Diferente dos comentários tradicionalmente utilizados, cujas informações são irrelevantes em tempo de compilação (e por isso são desprezados pelos compiladores), sugerimos a utilização de comentários de dois tipos: os comentários constantes e os comentários variáveis (veja [20]).

As Classes de Comentários.

Cada unidade de programa da biblioteca contém alguns comentários no início para explicar o que ele faz. Estes comentários são chamados de Comentários de Prólogo. A segunda classe de comentários, os Comentários Explicativos, são inseridos na unidade de programa para explicar qualquer código fonte cujo entendimento não seja possível com a sua simples leitura.

Os Tipos de Comentários.

Os comentários constantes são escritos da maneira convencional, começando pelo caráter 'C' e contendo informações que não mudam para as várias versões de uma mesma unidade de programa. Os comentários variáveis são comentários especiais cujas informações variam de versão para versão de uma mesma unidade de programa e, por isso, devem ser considerados pelo Sistema ao efetuar qualquer tipo de transformação.

2.2. CONSIDERAÇÕES SOBRE O CONTEÚDO E A FORMA DOS COMENTÁRIOS

Visto que a documentação interna das unidades de programa deve ser escrita levando em consideração sua transportabilidade, propomos algumas padronizações sobre a forma de escrevê-la para que a mesma possa ser transportada com o mínimo esforço. Os programadores que estiverem escrevendo software transportável deverão ter sempre em mente o princípio básico de que deverão ser evitadas quaisquer referências a nomes de variáveis, de equipamentos, de precisão, ou de quaisquer outras entidades que sirvam, de alguma forma, para associar a unidade de programa a uma versão específica de unidade de programa, à exceção dos lugares destinados a tal finalidade.

Comentários Explicativos.

A quantidade e o conteúdo dos comentários são dois

aspectos bastante importantes. Recomenda-se adotar a sugestão de Tassel [28] e utilizar pelo menos um Comentário Explicativo para cada dez (10) linhas de código fonte.

Quanto ao conteúdo, recomenda-se ao programador que assuma a familiaridade do leitor com a linguagem FORTRAN e, portanto, os comentários devem explicar a finalidade de um grupo de comandos do programa, e não, descrever a operação dos comandos. Por exemplo, antes do comando:

```
      IF ( Q.NE.0 ) GO TO 320,
```

um comentário como:

```
C     DESVIO SE Q DIFERENTE DE ZERO
```

não é um bom comentário pois tenta explicar a sintaxe do comando em vez de informar por que um desvio deve ser usado.

Por outro lado, o comentário:

```
C
C        QUANDO O PARÂMETRO DE MARQUARDT É ZERO ATRIBUI-LHE
C        O MENOR DOS VALORES DO TRAÇO E NORMA INFINITA ( MÁ
C        XIMA  SOMA DOS ELEMENTOS DA LINHA) DA MATRIZ  NOR-
C        MAL.
C
```

tirado de uma unidade de programa da BITAN, onde aparece antes do comando acima, é um bom comentário pois informa por que um desvio deve ser usado.

Comentários de Prólogo

O programador deve incluir nos Comentários de Prólogo, no mínimo, como sugerido por Tassel [28], as seguintes informações, observadas as convenções de padronização que incluímos:

(1) Uma descrição do que a unidade de programa faz, escrita em comentários convencionais da linguagem FORTRAN;

(2) Versão: o programador deve incluir três linhas de comentários variáveis para identificar a versão da unidade de programa: a primeira linha contém "C& VERSÃO:", a segunda contém "C& EQUIPAMENTO:" seguido da identificação do equipamento utilizado, e a última contém "C& PRECISÃO: " seguido da precisão usada na versão. Por questões de eficiência de execução decidiu-se pela utilização de comentários variáveis, iniciados por "C&", o que dispensa o pré-processador de analisar um grande número de comentários (os comentários convencionais e que são maioria, quase sempre). Quando a unidade de programa for submetida ao TRAPO para a geração de novas versões, o Sistema se encarrega de preencher os valores referentes a " EQUIPAMENTO" e "PRECISÃO" com informações obtidas do "menu " preenchido pelo usuário;

(3) Utilização: como chamar ou utilizar a unidade de programa. Deve-se ser incluída uma linha contendo "C& UTILIZAÇÃO:", seguida do exemplo de utilização, que deve ser colocado em uma nova linha de comentário variável começando pelos caracteres "C&". Como esta é uma linha especial de comentário, caso necessite ser continuada, a linha de continuação também deverá começar pelos caracteres"C&";

(4) Uma lista e descrição de todos os parâmetros. A lista deve ser iniciada por uma linha contendo "C& PARÂMETROS:" e deve conter, para cada parâmetro: uma linha começando pelos caracteres "C&" seguidos do parâmetro e do seu tipo(acompanhado da dimensão ou dimensões, se aplicável) separados por pelo menos um espaço em branco um do outro, seguida de uma ou mais linha(s) de comentários descrevendo o parâmetro. A lista descreve inicialmente os parâmetros que são passados para o subprograma e a seguir, os parâmetros que são retornados do subprograma. Na chamada, os parâmetros obedecem também a este sequenciamento. Su

gere-se que a descrição de cada parâmetro seja iniciada na mesma posição de tabulação que aquela usada para iniciar o tipo do parâmetro;

(5) Uma lista dos subprogramas utilizados. A lista deve ser iniciada por uma linha contando " C& SUBPROGRAMAS:". Este comentário deve ser iniciado pelos caracteres "C& e deve conter o nome e a classe de todos os subprogramas utilizados na unidade de programa (um por linha). Adicionalmente, cada nome de suprogramas pode vir acompanhado,além da classe, por uma breve descrição do que faz o subprograma, nesta ordem. Primeiro descrever os subprogramas externos e depois os intrínsecos que foram utilizados;

(6) O nome de qualquer método científico utilizado, acompanhado de referência sobre onde mais informações possam ser encontradas;

(7) Alguma indicação do tempo requerido para execução;

(8) Tamanho da unidade de programa:
.número de comandos
.memória requerida;

(9) Requisitos especiais de operação;

(10) Autor;

(11) Data em que a unidade de programa foi escrita.

Todas estas informações devem ser colocadas diretamente no corpo da unidade de programa para que possam ser encontradas facilmente. Assim, a unidade de programa conterá a sua própria documentação.

3. PRÉ-PROCESSADOR PARA CONVERSÃO DA DOCUMENTAÇÃO INTERNA.

Os comentários de prólogo de tipos 2, 3, 4 e 5 são comentários variáveis (começam com "C& ") e são analisados pelo pré-processador ao efetuar a conversão da documentação interna. Os demais comentários, inclusive os explicativos, são escritos da maneira convencional (começando com o caráter C) com a pequena restrição de não ter o C inicial seguido do caráter " & ", e portanto, não são levados em consideração ao se efetuar a conversão.

Se por alguma razão o programador sentir a necessidade de escrever nos comentários o nome de entidades que estejam relacionadas com uma versão específica de unidade de programa , isto só será permitido para os comentários de prólogo de tipos (3), (4) e (5), que são comentários variáveis, com a imposição de que as entidades sejam analisadas quanto à possibilidade de conversão. Em última instância, se o programador tiver que efetuar referência, em um comentário explicativo, a nomes de entidades que estejam relacionadas com uma versão específica de unidade de programa, este comentário deverá ser um comentário variável (portanto, será iniciado por "C &") e o(s) nome(s) da(s) referida(s) entidade(s) dever(á) (ão) vir antecedido(s) pelo caráter " & " em cada referência, para que o Sistema possa identificar as transformações a serem efetuadas.

Em todos estes comentários variáveis o Sistema tenta identificar a entidade a ser convertida (vem antecedida por " & ") e atua de acordo com as ações especificadas na tabela 1. Mas, qualquer que seja o caso, o Sistema deixa o comentário iniciando por "C&", e deixa o " & " antes das entidades, para que novas versões da unidade de programa possam ser produzidas a partir da que está sendo preparada, se vierem a ser requeridas. Isto possibilita a reversibilidade de conversão da documentação interna para uma forma "equivalente", sempre que necessário. Um módulo do Sistema elimina todas as ocorrências de " & ", do texto fonte, quando requerido.

4. CONCLUSÃO

Quando várias versões são automaticamente produzidas para uma mesma unidade de programa, a partir da mesma versão mestre, por um programa de computador, torna-se particularmente importante que a sua documentação seja também automaticamente modificada. A conversão automática da documentação interna implica, além de confiabilidade, em redução nos custos de produção de programas desenvolvidos utilizando Programação Transformacional (definida em [22]).Unidades de Programa com documentação interna transportável podem ser obtidas seguindo a nova Metodologia de Programação ora proposta, desde que se tenha um pré-processador para efetuar as devidas atualizações.

5. BIBLIOGRAFIA

[1] AIRD, Thomas J. - The IMSL FORTRAN Converter:An Aproach to Solving Portability Problems, **Portability of Numerical Software**, W. Cowell (ed.), Oak Brook, Illinois, Jun/1976, pp. 369-388.

[2] AIRD, Thomas J.; Battiste, Edward L.; Gregory,Walton C. - Portability of Mathematical Software Coded in FORTRAN, ACM **Trans. Math. Software**, 3 (1977),pp.133-127.

[3] BENTLEY, J. & Ford, Brian - On the Enhancement of Portability Within the NAG Project - A Statistical Survey, **Portability of Numerical Software**, W. Cowell (ed.), Oak Brook, Illinois, Jun/1976,pp.505-528.

[4] BOYLE, James M. - Mathematical Software Transportability Systems - Have the Variations a Theme?, **Portability of Numerical Software**, W. Cowell (ed.), Oak Brook , Illinois, Jun/1976, pp. 305-360.

[5] BOYLE, J. & Matz, M. - Automating Multiple Program Realizations, **Symposium on Computer Software Engineering**, Polytechnic Institute of New York - Apr/1976,pp. 421-456.

[6] BROWN, W.S. & Hall. A.D. - FORTRAN Portability via Models and Tools, **Portability of Numerical Software**,W.Cowell (ed.), Oak Brook, Illinois, Jun/1976, pp.158-164.

[7] CODY, W. J. - Machine Parameters for Numerical Analysis, **Portability of Numerical Software**, W. Cowell (ed.),Oak Brook, Illinois, Jun/1976, pp. 49-67.

[8] DEKKER, T.J. - Machine Requirements for Reliable, Portable Software, **Portability of Numerical Software**, W. Cowell (ed.) Oak Brook, Illinois, Jun/1976, pp.22 - 36.

[9] DRITZ, Kenneth W. - Multiple Program Realizations Using the TAMPR System, **Portability of Numerical Software**, W. Cowell (ed.)O.Brook, Illinois, Jun/1976, pp. 405 - 423.

[10] DU CROZ, J. J.; Hague, S.J.; Siemieniuch, J. L.-Aids to Portability within The NAG Project, **Portability of Numerical Software**, W. Cowell (ed.), Oak Brook,Illinois, Jun/1976, pp. 389-404.

[11] FORD,Brian - Preparing Conventions for Parameters for Transportable Numerical Software, **Portability of Numerical Software**, W. Cowell (ed.), Oak Brook, Illinois, Jun/1976, pp. 68-91.

[12] FORD, Brian; Bentley, J.; Du Croz, J.J.; Hague, S. J. - The NAG Library 'Machine', **Software - Practice and Experience**, 9 (1979), pp. 65-72.

[13] FORD, Brian & Sayers, D.K. - Developing a Single Numerical Algorithms Libray for Different Machine Ranges , ACM Trans. Math. **Software**, 2 (1976),pp. 115-131.

[14] HAGUE, Stephen J. & Ford, Brian - Portability - Prediction and Correction, **Software - Practice and Experience**, 6 (1976), pp. 61-69.

[15] HATTORI, Mário T. - Uma Disciplina de Programação em FORTRAN-66. DSC/UFPb, Relatório Técnico, Maio/1984.

[16] KROGH, Fred T. - Features for FORTRAN Portability, **Portability of Numerical Software**, W. Cowell (ed.), Oak Brook, Illinois, Jun/1976, pp. 361-367.

[17] LARMOUTH, J. - FORTRAN 77 Portability, **Software- Practice** & Experience, 11 (1981), pp. 1017-1117.

[18] LOVEMAN, David B. - Program Improvement by Source to Source Transformation, **3rd ACM Symposium on Principles of Programming Languages**, Atlanta, Georgia, Jan/1976, pp. 140-152.

[19] MAHER, B. & Sleedman, D. H. - Automatic Program Improvement: Variable Usage Transformations, **ACM Trans. Program. Lang. Syst.**, 5 (1983), pp. 236-264.

[20] MEDEIROS, Camilo de L. G. - Projeto e Implementação de um Sistema Transformador de Programas FORTRAN, **Dissertação de Mestrado**, UFPb (1986).

[21] MEDEIROS, Camilo de L. G.; Mongiovi, Giuseppe; Hattori, Mário Toyotaro - Sistemas Transformacionais de Programas FORTRAN: Um Estudo de Caso, **Anais do III ERMAC**, Natal-RN (1986).

[22] PARTSCH, H. & Steinbrueggen, R. - Program Transformation Systems, **Computing Surveys**, 15 (1983), pp. 199 - 236.

[23] POOLE, P. C. & Waite, W.M. - **Portability and Adaptability - Software Engineering - An Advanced Course**. F. L. Bauer (ed.), Springer-Verlag, pp. 183-277.

[24] RYDER, B. G. - The PFORT Verifier, **Software- Practice and Experience**, 4 (1974), pp. 359-377.

[25] SCOWEN, R. S. - Some Aids for Program Documentation, **Software - Pratice and Experience**, 7 (1977), pp.779-792.

[26] STANDISH, Tomas A.; Kibler, Dennis F.; Neighbors, James M.- Improving and Refining Programs by Program Manipulation, **Proceedings of ACM Annual Conf** - Houston, Texas, 1976, pp.509-516.

[27] TANENBAUM, A. S.; Klint, P.; Bohm, W. - Guidelines for Software Portability, **Software - Practice and Experience, 8** (1978), pp.681-698.

[28] TASSEL, Dennis Van - **Program Style, Design, Efficiency, Debugging and Testing.** Prentice-Hall, Englewood Cliffs, N.J., 1974.

TIPO DE ENTIDADE	AÇÃO DO SISTEMA
Identificador	Se for possível, converte o nome do identificador de acordo com as "Convenções de Nomenclatura"(veja seção 3.3.1 de [20] da versão anterior para a versão atual.
Constante	Converte a sintaxe das constantes tipo REAL ou COMPLEX, p. ex.,0.5E0 para 0.5D0.
Nome de Precisão	Substitui o nome da precisão anterior pelo nome da precisão atual.
Todos os demais	Nenhuma (O texto fica inalterado)

Tabela I : Ações de Módulo de Conversão da Documentação Interna.