XIII CONGRESSO NACIONAL DE PROCESSAMENTO DE DADOS

OUTUBRO 1980

SUCESU

RIO DE JANEIRO

004.06
C749

XIII CONGRESSO NACIONAL DE PROCESSAMENTO DE DADOS

ANAIS DO XIII CNPD

Hotel Nacional-Rio
Outubro de 1980

COMISSÃO ORGANIZADORA

Raulino Carvalho de Oliveira
Salvador Perrotti
Sylvio de Carvalho S. Mattos
Carlos Eduardo C. Fonseca
Raul Isiris
Fernando Vianna
Miguel Stábile
Eduardo Jorge Caldas Pereira
Gabriel de Almeida
Ronaldo Latsch
Carlos Alberto Pontes
Maria Elzira C. Isiris
Marlene Caldeira

COMISSÃO TÉCNICA

Fernando Vianna
Artur Edson Dias Pereira
Francisco Antônio Dantas Monteiro
José Eduardo Thyrso de Lara
Paulo Frota Simas de Oliveira
Jorge Peñaranda Coimbra
Carlos Jorge Zimmermann
Luiz Carlos Pinna

SUCESU-Sociedade dos Usuários de Computadores e Equipamentos Subsidiários

PROJETOS DE PROGRAMAS ASSISTIDOS POR MICROCOMPUTADORES

CARLOS J. P. LUCENA
RAUL C. B. MARTINS
MARIA DE LOURDES KRAHE

SUCESU–SOCIEDADE DOS USUÁRIOS DE COMPUTADORES
E EQUIPAMENTOS SUBSIDIÁRIOS – RIO DE JANEIRO (1980)

PROJETOS DE PROGRAMAS ASSISTIDO POR MICROCOMPUTADORES

Carlos J. P. Lucena, Raul C. B. Martins, Mª de Lourdes Krahe

Pontifícia Universidade Católica do Rio de Janeiro

RESUMO:

Descreve-se resumidamente o Sistema PREPROG para a assistência automatizada ao desenvolvimento de programas (documentação da especificação de um programa). Trata-se de sistema interativo para uso em microcomputador. A implementação do sistema preserva, pelo menos, os mesmos recursos de sistemas semelhantes projetados para uso em computadores de grande porte.

1 – INTRODUÇÃO

A área de Projeto de Engenharia Assistido por Computador ("Computer Assisted Design") começa a ter uma crescente aceitação no país, em particular nas áreas de eletricidade, mecânica e civil. Recentemente, a área de projeto de Software começou a merecer maior atenção face à necessidade de se dispor de produtos Software de melhor qualidade. Nos últimos anos, o entendimento do processo de desenvolvimento de programas cresceu consideravelmente e uma nova área começa a ganhar expressão.

Trata-se da área de Projeto de Software Assistido por Computador. A maior parte do trabalho nesta área pode ser classificado em duas categorias:

Sistemas automatizados para dar assistência ao projeto de sistemas de programação (programação em ponto grande) e sistemas automatizados para dar assistência ao projeto de módulos de programas (programação em ponto pequeno). A motivação para o uso de sistemas automatizados para projetos nos dois níveis mencionados é, básicamente a mesma:
Proporcionar o desenvolvimento *Sistemático e bem documentado* de produtos Software.

O presente trabalho descreve resumidamente um sistema automatizado para o apoio ao projeto de módulos de programas denominado PREPROG. Do ponto de vista metodológico a maior parte das técnicas de implementação usadas em PREPROG podem ter aplicabilidade, em sistemas para desenvolvimento de software que utilizem níveis mais alto de abstração (ex.: expressão da interligação entre módulos, com a abstração de seus detalhes internos).

PREPROG utiliza como linguagem base um amplo conjunto de pseudo-instruções que permite a adoção da técnica de programação estruturada (mecanismos de controle estruturado). A programação estruturada também requer a estruturação de dados. com esta finalidade PREPROG prevê em sua linguagem base mecanismos para a definição de tipos abstratos de dados. A saída final de um projeto desenvolvido com o auxílio do Sistema PREPROG é uma documentação da descrição, a nível de especificação, dos módulos de programa, que resolvem um determinado problema de programação, e suas interconexões. Esta documentação se destina a dirigir a implementação final de um programa que poderá ser feita em qualquer linguagem de programação escolhida para a aplicação.

PREPROG é um sistema interativo que foi dimensionado para funcionar em um microcomputador. A capacidade de análise do PREPROG é pelo menos igual a sistema semelhantes que foram desenvolvidos para operar em grandes computadores.

2 – DESCRIÇÃO DA LINGUAGEM PREPROG

A linguagem base do PREPROG destina-se ao projeto e documentação de programas estruturados e modulares. A linguagem está projetada para ser utilizada de modo interativo. Ela faz uso de mecanismos de controle usuais constantes de linguagens como o PASCAL, ALGOL, PL–I. A liguagem permite o entrelaçamento de comentários com construções lógicas de programação, permitindo o desenvolvimento do programa sem preocupações de ordem sintática. A linguagem tem ainda mecanismos para a especificação de tipos abstratos de dados e, como complemento, oferece ao programador além da edição dos módulos programados uma série de relatórios : índice de módulos, índice de dados, referência cruzada de módulos, referência cruzada de dados e árvore de chamada de módulos.

O sistema PREPROG requer do usuário a especificação de uma série de módulos:

a. Título

b. Texto

c. Interligação

d. Código

e. Tipo de dado

f. Externo

O programa consiste de, ao menos, um módulo título, um de interligação e um de código. Os únicos módulos que não podem ser repetidos são: O módulo título e o externo.

O módulo Título determina o nome do programa em desenvolvimento, bem como permite a apresentação de breves comentários (12 linhas) sobre o funcionamento e requisitos do programa.

Módulos texto (um ou vários) precedem os módulos de interligação e código. Eles permitem a especificação inicial do programa (que será discutida com o usuário ou analista) e a introdução dos comentários necessários ao entendimento das interligações e das funções dos módulos que se seguem.

Os módulos de interligação definem as entradas, as saídas e a função dos módulos de código que os seguem. Com esta finalidade, três palavras chaves, são fornecidas pelo sistema ao usuário; INPUT, OUTPUT, FUNCITON.

O módulo raiz principal, necessariamente será precedido de um módulo de interligação com o meio externo.

propriamente dito dos programas. Eles são compostos de pseudo-códigos variáveis e comentários. Os pseudo-códigos são construções do tipo:

```
IF. . .  THEN. . .  FI
IF. . .  THEN. . .  ELSE. . .  FI
IF. . .  THEN. . .  ELIF. . .  THEN. . .  FI
IF. . .  THEN. . .  ELIF. . .  THEN. . .  ELSE. . .  FI
DO. . .  OD
WHILE. . .  DO. . .  OD
UNTIL. . .  DO. . .  OD
DO CASE OF              OD
```

que possuem o significado semântico usual.

As variáveis são declaradas em:

Declare nome-de-variável *AS* tipo-de-dados.

As chamadas a outros módulos são feitas através do uso do identificador % nome-do-módulo e são permitidos comentários em qualquer ponto de texto.

As variáveis são distinguidas, por serem precedidas pelo carater *.

Os módulos de tipos de dados definem um tipo de dado e as operações a eles associados. As operações podem ser descritas através do uso do pseudo-código, constituindo-se cada uma em um módulo de código especial. Neste módulo existe a possibilidade de serem definidas variáveis internas (da representação do tipo de dados) que não podem ser exportadas para outros programas.

O módulo externo coleta os nomes de procedimentos, rotinas e tipo de dados, que não serão definidos no programa ou por já terem sido anteriormente definidos ou por serem tipos primitivos do sistema.

3 – OPERAÇÃO DO SISTEMA

O sistema prevê a utilização dos recursos definidos anteriormente em quatro etapas:

Aceitação, Execução, Alteração e Relatórios.

Na fase de aceitação o usuário é convidado a definir os módulos, na ordem que desejar, porém seguindo as restrições de tamanho de módulos (uma tela), de variáveis (seis letras), de nome de módulos (seis letras), e outras restrições comunicadas pelo sistema através de comentários apropriados.

Esta primeira fase corresponde a aquisição de dados, acompanhada de alguma análise de consistência. Por exemplo, é impossível nesta fase o uso errado de mecanismos de controle no módulo de código.

as possibilidades de execução, alteração ou relatório lhe são oferecidas.

Na fase de alteração, o usuário pode alterar qualquer elemento de qualquer módulo, bem como acrescentar outros que julgar necessário para a complementação da especificação do programa.

A alteração é realizada com o mesmo tipo de assistência dada na fase de aceitação, isto é, procurando-se examinar automaticamente a consistência dos dados fornecidos pelo usuário. Encerrada esta fase oferece-se ao usuário o mesmo leque de opções inicial.

Na fase de execução, são preparados os arquivos para os relatórios de índice de módulos, índice de dados, referência cruzada de módulos, referência cruzada de dados, e arvore de chamadas de módulos bem como testes de consistência adicionais do sistema. É, também, fornecida ao usuário, a listagem de erros detetados pelos testes de consistência aplicados. No final da fase o sistema volta a oferecer o mesmo leque de opções inicial.

Na fase de relatórios, são produzidas as listagens de todos os módulos e arquivos do sistema.

Passando uma vez pela fase de aceitação, e pela aplicação repetida das demais fases, o usuário deve conseguir a especificação do programa desejado, que estará estruturado e apresentado de forma modular.

4 – EXEMPLO DE PROJETO DE PROGRAMA ATRAVÉS DO SISTEMA PREPROG

TÍTULO

VERPLV

O PROGRAMA SE DESTINA A ATUALIZAÇÃO DA PILHA DE PALAVRA CHAVES DE UM PROGRAMA E A VERIFICAÇÃO DA VALIDADE DO USO DESTAS PALAVRAS (ESTE MÓDULO DE PROGRAMA FAZ PARTE DA DOCUMENTAÇÃO DO PRÓPRIO SISTEMA PREPROG).

INDICE

TEXTO

VERPLV

O MÓDULO SE DESTINA A VERIFICAÇÃO DE CADA PALAVRA DA LINHA DIGITADA.

SE A PALAVRA SE REFERE À CONTRÔLE, UMA PILHA DE PALAVRAS É ATUALIZADA. A PILHA, FOI INICIALIZADA QUANDO DA ANÁLISE SINTÁTICA DO PROGRAMA.

SE A PALAVRA É PRECEDIDA POR %, O ARQUIVO DE MÓDULOS É ATUALIZADO.

CASO CONTRÁRIO, A PALAVRA É TRATADA COMO COMENTÁRIO.

INTERLIGAÇÃO

VERPLV

@ INPUT : QUALQUER PALAVRA DA LINHA QUE ACABOU DE SER SOLICITADA.

@ OUTPUT: SE A PALAVRA FOR PALAVRA DE CONTRÔLE, PILHA ATUALIZADA,
SE A PALAVRA FOR NOME DE VARIÁVEL OU MÓDULO,
ATUALIZAR OS ARQUIVOS CORRESPONDENTES.

@ FUNCTION: DESCRITO NA SAÍDA.

CÓDIGO

VERPLV

```
DECLARE          * PALAVR          AS % STRING
DECLARE          * ERROXX          AS % STRING
BEGIN
    VERIFICA   * PALAVR
    * ERROXX = "Ø"
    DO CASE    * PALAVR
        A) "IF":
            CALL % PRRRIF
        B) "THEN":
            CALL % PRTHEN
        C) "ELSE":
            CALL % PRELSE
        D) "FI":
            CALL % PRRRFI
        E) "ELIF":
            CALL % PRELIF
        F) "DO":
            CALL % PRRRDO
        G) "OD":
            CALL % PRRROD
        H) "UNTIL":
            CALL % PRUNTL:
        I) "WHILE":
            CALL % PRWHIL
        J) "CASE":
```

```
                CALL % VARIAV
                CALL % MODULO
                CALL % COMENT

        OD

          IF   * ERROXX   <>  Ø   THEN   CALL   %   PRERRO

    END
```

INTERLIGAÇÃO

```
PRRRIF

        @ INPUT:      A PALAVRA CHAVE "IF" PARA SER ANALISADA

        @ OUTPUT:     A PILHA ATUALIZADA OU ERRO

        @ FUNCTION:  VERIFICAR SE O USO DA PALAVRA CHAVE"IF" É ADEQUADO.
```

CÓDIGO

```
PRRRIF

        DECLARE   * CONTRL  AS %  PILHAX

        BEGIN

            IF    * CONTRL   =  @  PVAZIA OU
                  * CONTRL   =     "DO"   OU
                  * CONTRL   =     "UNTIL" OU
                  * CONTRL   =     "WHILE" OU
                  * CONTRL   =     " : "
              THEN @EMPILH  * PALAVR  EM  * CONTRL

              ELSE   * ERROXX  = "IF"

      END
```

TIPO DE DADOS

```
PILHAX

        REP  = * PILHAS  AS  %  VETORX  (1:100) DE  %  STRING,
               * INDICE  AS  %  INTEIR
               * PPPPPP  AS  %  STRING

        OPS  = @TOPOXX,
               @EMPILH,
               @DESEMP,
               @VAZIA

        OP   = @TOPOXX
               BEGIN
                  * PPPPPP  = PILHAS  (   * INDICE)

      END

        OP     @ EMPILH
               BEGIN
                  * INDICE  =  * INDICE  +  1
```

```
            * PILHAS  (  * INDICE)  = * PPPPPP

        END

         OP      @DESEMP
                  BEGIN
                    IF  * INDICE  <> Ø  THEN  * INDICE  =  * INDICE  -  1
                                        ELSE  * INDICE  =  -  1

        END

         OP       @PVAZIA
                   BEGIN
                       * INDICE  =  - 1

        END
```

EXTERN

```
EXTERN

        %  PRTHEN   MODULO NÃO IMPLANTADO ANALISA A PALAVRA THEN

        %  PRELSE   MODULO NÃO IMPLANTADO ANALISA A PALAVRA ELSE

        %  PRRRFI   MODULO NÃO IMPLANTADO ANALISA A PALAVRA FI

        %  PRELIF   MODULO NÃO IMPLANTADO ANALISA A PALAVRA ELIF

        %  PRRRDO   MODULO NÃO IMPLANTADO ANALISA A PALAVRA DO

        %  PRRRDO   MODULO NÃO IMPLANTADO ANALISA A PALAVRA OD

        %  PRUNTL   MODULO NÃO IMPLANTADO ANALISA A PALAVRA UNTIL

        %  PRWHIL   MODULO NÃO IMPLANTADO ANALISA A PALAVRA WHILE

        %  PRCASE   MODULO NÃO IMPLANTADO ANALISA A PALAVRA CASE

        %  VARIAV   MODULO NÃO IMPLANTADO ANALISA NOME DA VARIÁVEL
                    E ATUALIZA O ARQUIVO DE VARIÁVEIS

        %  MODULO   MODULO NÃO IMPLANTADO ANALISA NOME DO MÓDULO E
                    ATUALIZA O ARQUIVO DE MODULOS

        %  COMENT   MODULO NÃO IMPLANTADO TRATA A PALAVRA COMO COMENTÁRIO

        %  STRING   TIPO DE DADO PRIMITIVO DO SISTEMA

        %  PREPRO   TRATA OS ERROS RESULTANTES DO USO INDEVIDO DE PALAVRA CHAVE.

ARVORE

        VERPLV

              PRRRIF
              PRTHEN
              PRELSE
              PRRRFI
              PRELIF
```

PRRROD
PRUNTL
PRWHIL
PRCASE
VARIAV
MODULO
COMENT
PREPRO

INDICE DE MODULOS

INDICE DE VARIÁVEIS

REFERÊNCIA CRUZADA

COMENT

PAG8 – PAG4

MODULO

PAG8 – PAG4

PILHAX

PAG7 – PAG6

PRCASE

PAG8 – PAG4

PRELIF

PAG8 – PAG4

PRELSE

PAG8 – PAG4

PRERRO

PAG8 – PAG4

PRRRDO

PAG8 – PAG4

PRRRFI

PAG8 – PAG4

PRRRIF

PAG6 – PAG4

PRRROD

PAG8 – PAG4

PRTHEN

PAG8 – PAG4

PRUNTL

PAG8 – PAG4

PRWHILE

PAG8 – PAG4

STRING

PAG8 – PAG4 – PAG7

VARIAV

VERPLV

PAG4

5 – MÓDULOS DE ANÁLISE

Os testes de consistência do projeto do programa são realizados em duas etapas:

A primeira tem lugar durante a fase de aceitação e produz as entradas para a segunda, que se passa durante a fase de execução.

A primeira etapa de análise da consistência é, em princípio, uma análise da sintaxe dos módulos. São verificadas as validades dos nomes de variáveis e módulos, a existência de certos comandos obrigatórios, e, principalmente, se a ordem das palavras é a correta. O princípio básico, que norteia a fase é o de que todos os erros serão determinados no momento de escrita, e o usuário é convidado a repetir o ítem errado até que seja proposto um ítem válido. Parte do objetivo é conseguido pelo sistema, através da orientação do usuário sobre qual fase e quais palavras chaves deverão ser digitadas nas diferentes etapas do desenvolvimento do programa. Durante esta etapa uma série de arquivos, com informações sobre nome-de-módulos, nome-de-variáveis, instante em que os módulos são declarados, instante em que os módulos são referenciados, declaração de variáveis e referência a variáveis, são atualizados.

A segunda etapa da análise de consistência é realizada durante a construção dos relatórios. Ao se preparar o relatório de referências cruzadas verifica-se se cada módulo referenciado foi definido. Entende-se por definição de módulo, a sua criação através de um módulo de codificação ou sua declaração no módulo externo. Um módulo em preparação pode ser, incialmente declarado num módulo externo para o teste de consistência, e, mais tarde, quando desenvolvido, a entrada no módulo de codificação com aquele nome passa a substituir a declaração anterior. Pode-se inicialmente, utilizar-se o relatório para verificar se faltam módulos no programa inicial, e, também, para, durante a fase de alterações, saber em que implica a alteração ou destruição de um determinado módulo.

Função semelhante possue o relatório de dicionário de variáveis. Pode-se detetar a falta de declaração de tipo de dados, uso indevido de tipo de dados e parâmetros, bem como declarações redundantes e incoerentes de tipo de dados e nomes de variáveis. Tem-se, ainda, dicionário de ocorrência de variáveis. Tem-se, ainda, dicionário de ocorrência de variáveis em módulos, permitindo, assim fácil localização de alterações provocadas por mudanças em tipo de dados. O relatório árvore de chamada, fornece a estrutura hierárquica dos módulos que constituem o programa, e, com isso, pode-se detetar e analisar falhas na estrutura.

Tem-se, ainda, relatórios do tipo índice de módulos com finalidades gerenciais, visando o auxílio à identificação de alterações e correções.

6 – MÉTODOS DE IMPLEMENTAÇÃO

O Sistema PREPROG está implementado num micro-computador da Societé Occitane d'Eletronique com microprocessador da Motorola.

Como unidade auxiliar dispõe-se de duas unidades de mini-diskettes com 70 kbytes cada, e de uma impressora em série.

A linguagem BASIC possui variáveis dos tipos inteiros, reais e cadeia de caracteres com as operações usuais de soma, subtração, multiplicação, divisão para números e concatenação, retirada de subcadeias para cadeia de caracteres bem como algumas funções de mudanças de tipo.

Os comandos de controle são do tipo

GO TO, IF. . . THEN. . . ELSE, FOR. . . NEXT, GOSUB.

O sistema de controle (Monitor) permite dois tipos de arquivo: sequencial e acesso direto. O segundo tipo possui bastante limitações quanto ao tamanho do arquivo e dos registros.

O Sistema PREPROG foi programado em quatro fases:

Aceitação, Execução, Alteração e Edição. As fases se alternam na memória, e, durante a operação, apenas, parte de cada fase está residente.

É definida uma série de arquivos nos minidiskettes:

Uns sequenciais, os de texto, interligação, codificação e tipo abstratos de dados e outros de acesso direto, os que irão permitir os relatórios finais.

7 – CONCLUSÕES

A aplicação de uma metodologia (programação estruturada) com a assistência de um Sistema automatizado facilita o uso generalizado e correto desta metodologia. A área de mercado reservada a produtos nacionais é a dos micro e minicomputadores.

Fase à necessidade de se melhorar no país os padrões técnicos do desenvolvimento de software, os Sistemas Nacionais poderão vir equipados com software de apoio para o desenvolvimento de software, promovendo

desta forma, uma nova dimensão na utilização de pequenos sistemas. O projeto PREPROG desenvolvido na PUC/RJ procurou fazer um estudo de viabilidade da possibilidade de se abrigar um sistema de documentação útil em um micro-computador. O Sistema PREPROG tem sido destinado a fins educacionais mas não é inadimissível o seu aproveitamento dentro da rotina de um ambiente de produção.

REFERENCES

1 – CAINE, S. H.; FARBER, D. J.; GORDON, K.E.; "Program Design Language"; CAINE, Farber and Gordon Inc., mar. 1975.

2 – LINDEN, N. M.;"Software Development Processor: A Tool for Program Design", University of California, 1976.