UM GERADOR DE DIALOGOS

Francisco Montalvo Acosta

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS GRADUAÇÃO DE ENGENHARIA DE SISTEMAS DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DO MESTRE EM CIÊNCIAS (M.Sc.).

Aprovada por:

MIGUEL JONATHAN
(Presidente)

SUELI MENDES DOS SANTOS

PAULO MARIO BIANCHI FRANÇA

Rio de Janeiro - RJ - BRASIL

DEZEMBRO DE 1983

MONTALVO, ACOSTA FRANCISCO

Um Gerador de Diálogos (Rio de Janeiro) 1983.

X, 268, 29,7 cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas de Computação, 1983).

Tese - Universidade Federal do Rio de Janeiro - Faculdade de Engenharia.

1. Interface Usuário - Computador I. COPPE/UFRJ II. Título (série).

Ã minha esposa Martha

e

Minha terna filha

Martha Isabel

A G R A D E C I M E N T O S

Ao prof. Miguel Jonathan, pelas idéias, conhecimentos e paciente orientação ministrada durante o desenvolvimento deste trabalho.

À COPPE-SISTEMAS pelos conhecimentos recebidos durante meus estudos de Mestrado.

Ao INECEL (Instituto Ecuatoriano de Electrificación) e em especial aos Engs. Germán Saltos e Nelson Medina pelo apoio e incentivo constante por eles dados.

Ao prof. Sergio M. Schneider e ao Eng. Carlos Tejada, pela comprovação do sistema proposto, na criação da interface usuário de seus respectivos trabalhos de Tese de Doutorado e Mestrado.

Aos professores Sueli Mendes dos Santos e Paulo Mario Bianchi França por terem acedido a participar da banca examinadora.

À Dulce Maria Vilela pela boa vontade no trabalho de datilografia.

R E S U M O

Este trabalho apresenta uma ferramenta para geração automática de software para suportar diálogos entre usuários e sistemas de aplicação computarizados, que foi implementado no computador Burroughs B-6700.

O modelo proposto utiliza o conceito de menus para a interface usuário-computador.

O projetista da interface fornece as descrições dos diálogos de maneira interativa e o sistema gera código fonte ALGOL, que após compilado, constitue a interface desejada.

São apresentados finalmente, exemplos do uso do sistema proposto, assim como, as conclusões e recomendações para futuras pesquisas.

A B S T R A C T

This work presents a tool for the automatic generation of software to support dialogues between users and application systems and its implementation on a Burroughs B-6700 computer.

The proposed model uses the concept of menus to describe the user-computer interface.

The designer of the interface supplies the descriptions of the dialogues in an interactive way,the system then generates source code in ALGOL which is ready for use after compilation.

Finally, the conclusions of this research are presented and recommendations for future developments are suggested.

Í N D I C E

CAPÍTULO I - INTRODUÇÃO 1

1.1- Motivação 1

1.2- Descrição do tema 2

CAPÍTULO II - A INTERFACE USUÁRIO-COMPUTADOR (IUC) 5

2.1- Introdução 5

2.2- Descrição dos componentes da IUC 5

2.2.1- O usuário da IUC 6

2.2.2- O hardware utilizado pela IUC . 10

2.2.3- O software utilizado no projeto da IUC 14

2.3- Os modelos de interface usuário-computador 17

2.3.1- Linguagem natural 18

2.3.2- As linguagens de programação .. 19

2.3.3- Linguagem de consultas 20

2.3.4- Linguagem de comandos 21

2.3.5- Diálogos dirigidos pelo computador 23

2.4- Os fatores que caracterizam o bom desempenho da IUC 27

2.4.1- Facilidade de uso da IUC 28

2.4.2- Versatilidade da IUC 30

2.4.3- Capacidade de auxílio 31

2.4.4- A Prevenção e o tratamento de erros 32

CAPÍTULO III - DIVERSAS ABORDAGENS NA IMPLEMENTAÇÃO DA IUC 35

3.1- Instruções da IUC misturadas com o software de aplicação 35

3.2- A IUC implementada em um Sistema Estruturado 37

3.3- Utilização de geradores de software .. 40

3.3.1- DIALOG 42

3.3.2- DIAGEN: A Dialogue Generator .. 46

3.3.3- MCIS: Menu Creation And Interpretation System 50

3.4- "G D": Um gerador de diálogos 53

CAPÍTULO IV - MODELAGEM E ESPECIFICAÇÃO DO GERADOR DE DIÁLOGOS "G D" 56

4.1- Introdução 56

4.2- A descrição do sistema Gerador de Diálogos "G D" 58

4.3- As descrições de diálogos 59

4.3.1- Menus 60

4.3.2- Telas para coleta de parâmetros (TCP) 65

4.3.3- Mensagens de auxílio ao usuário e recuperação de erros 70

4.4- A interpretação das descrições de diálogos 70

4.4.1- A Entrada das descrições de diálogos 70

4.4.2- A interface Projetista - "G D" (A Procedure CRIAR) 74

4.4.3- A geração de código fonte (procedure GERADOR) 82
4.4.4- A interface da procedure GERADOR 83
4.4.5- Os programas fonte gerados pela procedure GERADOR 87
4.5- O uso do sistema Gerador de Diálogos "G D" 104
4.5.1- O uso previsto para os diálogos produzidos pelo "G D" 106
4.5.2- A utilização da procedure "criar" 107
4.5.3- Atualizações feitas sobre Diálogos já existentes 110
4.5.4- A utilização da procedure GERADOR 119
4.5.5- O uso do código fonte gerado .. 121

CAPÍTULO V - A IMPLEMENTAÇÃO DO GERADOR DE DIÁLOGOS 123
5.1- Características gerais da implementação 123
5.2- A implementação da interface com o projetista 124
5.2.1- Definição dos módulos principais da procedure CRIAR 125
5.2.2- O funcionamento da procedure CRIAR 126
5.2.3- Descrição dos módulos utilizados na implementação da procedure CRIAR 127

5.2.4- Descrição do módulo MENUDESC 136
5.2.5- Descrição do módulo TCPDESC 148
5.2.6- Descrição dos módulos que a procedure CRIAR utiliza como interface 152
5.3 - A Implementação do GERADOR 159
5.3.1- Definição dos módulos principais. 159
5.3.2- O funcionamento básico do GERADOR 161
5.3.3- Verificação de erros 163
5.3.4- Descrição dos módulos de GERADOR. 165
5.3.5- Módulos utilizados repetidamente na execução do GERADOR 191

CAPÍTULO VI - CONCLUSÕES 207
6.1- Resumo e Discussão 207
6.2- Recomendações 214

APÊNDICE I - ILUSTRAÇÃO DO USO DO "G D" 217

REFERÊNCIAS BIBLIOGRÁFICAS 262

I - INTRODUÇÃO

1.1 - Motivação

Um sistema de informação que utilize o computador é composto de três elementos básicos: software, hardware e usuário.

A interface usuário-computador, constituída pela interelação desses três elementos, é sem dúvida, peça fundamental para o sucesso desse sistema.

No projeto dessa interface, além do dimensionamento dos recursos do hardware, devem ser levados em conta os fatores humanos que delineam o comportamento do usuário, no seu relacionamento com a máquina.

O maior ou menor grau de dificuldade que o usuário encontra na utilização de determinado sistema depende, em grande parte, da avaliação correta desses fatores humanos por parte dos projetistas.

Infelizmente, somente nestes últimos tempos é que fatores como facilidade de uso, facilidade de aprendizagem, e outros, começam a ser utilizados pelos projetistas de sistemas - Simpson[53]; Ketil Bo[8]; Zloof[36].

Essa falta de atenção ao projeto da interface com o usuário pode ser explicada pelos seguintes aspectos:

- o uso do computador ficou restrito, por muito tempo, a programadores ou pessoas treinadas na utilização de determinado sistema;
- o projetista das linguagens ou da interface com o usuário colocava maior ênfase no aproveitamento dos recursos do hardware, que eram onerosos e

mais limitados;

- a complexidade do software da interface aumenta, à medida em que se pretenda atingir uma classe maior de usuários.

Contudo, o rápido crescimento do número de usuários não especializados, sustentado na constante baixa dos custos do hardware, tem influenciado para que grande parte dos esforços da pesquisa em computação, atualmente dedique-se também,ao desenvolvimento de técnicas e ferramentas (software-tools),que facilitem a construção da interface com o usuário.

No conjunto dessas ferramentas, existe particular interesse pela criação de programas auxiliares, que automatizem o processo de produção do software da interface - Heffler[19]; Pfaff[9];Kaiser[16]; Negus[18] e outros.

A utilização desses programas auxiliares traz, consigo, as seguintes vantagens:

- criação de uma metodologia para o projeto de sistemas interativos - nesta metodologia estaria garantida a atenção aos fatores humanos mais relevantes.
- identificação e possível padronização de um conjunto de técnicas da interação homem-máquina;
- redução dos custos da produção do software da interface com o usuário.

1.2 - Descrição do Tema

O interesse do autor no presente trabalho está diri

gido para a elaboração de uma ferramenta de software, que permita a geração de programas fonte. Esses programas, quando compilados e executados, constituirão a interface usuário-computador, de algum sistema de aplicação.

A ferramenta em questão foi denominada "G D" - Gerador de Diálogos - e assim será referenciada ao longo do texto.

O uso esperado para o código fonte, produzido pelo "G D", poderá ser dos seguintes tipos:

a) criação de interfaces que permitam a execução de programas através de uma estrutura de apresentação de menus (menu driven systems);

b) criação de rotinas ou procedimentos que realizem a interação com o usuário, no contexto de algum programa de aplicação;

c) uma combinação dos casos anteriores.

Visando facilitar a criação da interface com o usuário, o "G D" foi projetado de acordo com as seguintes características:

- o relacionamento projetista - "G D", durante a entrada da descrição de uma interface, é totalmente interativo;

- após a recepção dessa descrição, a geração do código fonte correspondente é automática;

- descrições de interfaces previamente recebidas, podem ser facilmente expandidas ou modificadas;

- os programas de aplicação, que devam ser utilizados durante a execução dessa interface, poderão estar escritos em FORTRAN, COBOL ou ALGOL.

No capítulo II deste trabalho, apresentamos um extrato da literatura que fora pesquisada em relação ao estudo da interface usuário-computador.

No capítulo III é feita uma comparação entre as características mais significativas do "G D" e as de outros modelos de geradores de software de interface que foram pesquisados.

No capítulo IV são apresentadas a modelagem e especificação do "G D", assim como os aspectos mais relevantes de sua utilização. Exemplos com aplicações práticas do uso do"GD" encontram-se no Apêndice I.

No capítulo V são descritos os detalhes da implementação do "G D", no computador Burroughs-6700 da UFRJ.

Finalmente, no capítulo VI, é feita uma avaliação crítica do trabalho, com as conclusões e recomendações para futuras pesquisas.

II - A INTERFACE USUÁRIO - COMPUTADOR (IUC)

2.1 - Introdução

Neste capítulo apresentaremos um extrato da matéria que foi pesquisada, com relação ao estudo da interface usuário computador.

Esta apresentação, tem em vista colocar em evidência os fatores que caracterizam o bom desempenho da IUC de algum sistema de aplicação e ao mesmo tempo, ser utilizada como base da análise das abordagens existentes para produção do software da IUC.

As matérias que serão tratadas compreendem os seguintes aspectos:

- descrição dos componentes da IUC;
- apresentação dos modelos existentes;
- resumo dos fatores que caracterizam o bom desempenho da IUC.

2.2 - Descrição dos Componentes da IUC

Em um determinado sistema de informação, que utilize o computador, a IUC está constituída pela interelação que existe entre o software, o hardware e o usuário.

Apresentaremos, a seguir, os detalhes de cada desses componentes.

2.2.1 - O Usuário da IUC

Os usuários que interagem com o computador são dos mais diferentes tipos. As maiores diferenças podem ser encontradas no seu grau de treinamento, no seu interesse por determinado tipo de aplicação e no seu grau de utilização da IUC.

Apesar de todas essas diferenças, existem certos fatores humanos, que são em grande parte, os responsáveis pelo comportamento do usuário que interage com o computador.

Trabalhos relacionados com o estudo desses fatores tem sido apresentados por Martin[39], Shneiderman[34],[37], Davis[41], Tracz[35], Simpson[53].

O sistema humano de processamento da informação

De acordo com Tracz[35], os componentes básicos do sistema humano para o processamento da informação são: a memória e os processos que controlam o fluxo das informações. Em tais processos estão incluídos:

- a atenção, a percepção, a aprendizagem, a recordação e o processo de repetição.

Este sistema de processamento da informação utilizado pelo homem está constituído por três níveis de memória:

- memória de muito curto prazo
- memória de curto prazo
- memória a longo prazo

A memória de muito curto prazo

Neste nível de memória, a informação é retida por período muito curtos (0.5 - 1 segundo). Essas informações são

trazidas do nível de memória de curto prazo através do processo da atenção.

A memória a curto prazo

Neste nível de memória, são retidas e processadas pequenas quantidades de informação denominadas "chunks". Elas poderão ser: um dado ou um nome que represente a uma coleção de informações relacionadas. O número de "chunks" manejados pela memória de curto prazo é de (7 ± 2); Martin[39], Tracz[35]. Essa quantidade limitada de informações, é compensada pela facilidade do homem em colocar, em um "chunk", o nome que resulta da abstração de um ou mais níveis de informações relacionadas.

O período de retenção dessas informações na memória a curto prazo é também pequeno (20 - 30 segundos).

No entanto, este período pode ser aumentado mediante sucessivas repetições que, finalmente, ajudem a informação a ficar retida na memória a longo prazo, o que constitui o processo de aprendizagem.

O processamento da informação retida na memória de curto prazo é totalmente sequencial.

O processo humano da análise e solução de problemas

As limitações que aparecem no sistema humano do processamento da informação são amplamente compensadas pelos modelos e características que o homem possui, para conseguir soluções aos problemas que lhe são apresentados. Tais modelos incluem:

- testes e comprovação de erros, planificação e abstração, comparação com experiências anteriores, análise top-down e bottom-up, processos de busca ou revisão, etc

Essas técnicas de solução são apoiadas por uma incrível memória associativa, junto a outras características próprias do homem, como a intuição e a habilidade criativa.

Fatores psicológicos

Ligados ao comportamento do usuário na interação com o computador, existem alguns aspectos do tipo psicológico, que podem afetar positiva ou negativamente o desempenho da IUC.

- o desejo de controle
  esse desejo é uma força diretriz do comportamento humano - à medida em que a experiência dos usuários aumenta, eles preferem utilizar o computador como uma ferramenta e não como um instrumento que os controle; Shneiderman[34].

- atitudes e preconceitos
  os usuários que apresentam atitudes negativas ao uso do computador deverão, muito provavelmente, apresentar menor desempenho na utilização da IUC;

- ansiedade
  usuários submetidos à utilização do computador de maneira apressada, ou que estejam sujeitos a algum tipo de ansiedade ou restrição, são mais

suscetíveis de cometer erros

Os tipos de usuário

O conhecimento do tipo de usuário é fundamental para o projeto de qualquer IUC.

Esses tipos de usuários podem ser definidos em função dos seguintes aspectos:

- o grau da utilização da interface;
- a quantidade de treinamento do usuário;
- o nível de conhecimentos dos assuntos abordados pela IUC.

Levando em conta os aspectos acima apontados, os usuários da IUC podem ser colocados nos seguintes grupos:

- usuários casuais

  serão considerados todos aqueles cujo grau de utilização da IUC seja bastante irregular, não constituindo sua atividade principal.
  Uma característica fundamental desse grupo é sua falta de conhecimento de computação, programação e da própria IUC. Codd[51].

- usuários dedicados:

  são aqueles cuja atividade principal dedica-se ao uso do computador. Estes usuários são altamente treinados e sua experiência na interação com o computador permite-lhes a aprendizagem de linguagens e outras características da máquina.

Os usuários casuais e os dedicados constituem os limites do grau de utilização da IUC. Entretanto, existe uma grande quantidade de usuários que podem ser colocados dentro destes limites.

Um outro fator a ser destacado é a possibilidade de um usuário se transformar de usuário casual em no mais experiênte, à proporção que aumenta seu grau de utilização da IUC.

2.2.2 - O Hardware utilizado pela IUC

Dificilmente algum setor da tecnologia tem crescido tanto nos últimos anos, como o setor de processamento de dados por computador.

O crescimento dessa tecnologia parte da época na qual os usuários utilizavam o computador de maneira indireta, (processamento em "batch").

O aparecimento dos sistemas de multi-usuário renovou, inteiramente, os métodos e modelos do processamento em batch, substituindo-os por sistemas on-line, utilizados através de terminais remotos.

Nos últimos tempos, os sistemas distribuídos de mini e microcomputadores levam, até o ambiente do trabalho do usuário, todos os recursos dos sistemas de grande porte, aos quais encontram-se ligados mediante redes de computadores. Estes sistemas distribuídos permitem que muitas das barreiras impostas pelos sistemas centralizados, possam ser diminuidas.

O rítmo de avanço da ciência da microeletrônica é tão elevado, que constitui tarefa difícil, comentar sobre os últimos desenvolvimentos da computação que ainda não tenham si

do substituídos por outros mais aperfeiçoados. No entanto,este avanço da tecnologia apresenta uma tendência que tem-se manti do ao longo do seu desenvolvimento. Esta tendência é associada à baixa do custo de equipamentos, cada vez mais sofisticados. Branscomb(38).

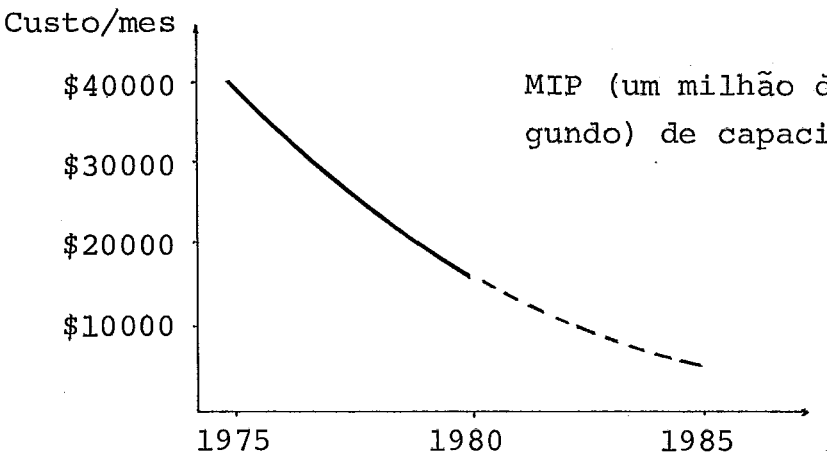


Fig. (2.1) - Os preços do Hardware

Fonte: (Lewis M. Branscomb, IBM, "Bringing Computing to people", COMPUTER, IEEE, Julho/82).

Esta tendência muito particular do desenvolvimento do hardware pode ser, perfeitamente, aproveitada no projeto de interfaces mais sofisticadas,que necessitam de equipamentos poderosos para equilibrar os "over-head" produzidos e, ao mesmo tempo, exequíveis a um maior número de usuários.

Apresentaremos, abaixo, algumas das facilidades do hardware, utilizadas para melhorar a comunicação usuário-computador:

- teclados especiais:

a utilização do teclado padrão, ao qual estamos todos acostumados, é o resultado de várias décadas de estudo dos fatores humanos, relacionados com seu uso.

No entanto, é possível a utilização de certo tipo de teclados, que ajudem na entrada de dados ou de outras informações específicas a algum tipo de aplicação. Martin(39).

- "light-pen":

utilizado para que o usuário possa apontar uma determinada área da tela do vídeo - desta maneira, ele poderia, por exemplo: escolher diretamente um item de um menu ou interagir de maneira gráfica com o computador.

O uso do "light-pen" pode ser ainda sofisticado através de uma tela de tipo "touch-screen", que permite, ao usuário, apontar, com seu dedo, uma determinada área da tela, porém sem a mesma resolução do "light-pen".

- "joystick":

o "joystick" permite que o usuário possa mover o cursor de maneira contínua; a utilização deste dispositivo tem sido orientada, principalmente, à interação com gráficos e seu sucesso, nos vídeo-jogos, é definitivo.

Entretanto, o princípio do "joystick", de converter sinais analógicos em digitais, tem sido

aproveitado em muitas outras aplicações - Barden[54].

- "digitizer":

este dispositivo, que aproveita as vantagens do "light-pen" e do "joystick", permite a transformação de informações gráficas em valores digitais; o "digitizer" utiliza uma tela quadriculada, na qual o usuário define a origem e os valores de um sistema cartesiano.
Após essa definição, o usuário pode apontar um lugar na tela, fato que será entendido pelo computador, como o valor de um par ordenado desse sistema cartesiano.

- Facilidades do vídeo:

As mais significativas são as seguintes:

- possibilidade para destacar o brilho de certos setores da tela.
- formatação da tela; que permite ao usuário editar uma ou mais linhas de dados antes de serem lidos pelo computador.
- vídeo colorido; especialmente útil na interação com gráficos.
- endereçamento aleatório da tela, carateres gráficos, etc.

2.2.3 - O Software utilizado no projeto da IUC

As linguagens de programação

Nas primeiras aplicações do computador, o tratamento da informação era realizado a seu nível mais baixo, passo a passo ou a nível de máquina.

Desde esses primeiros tempos, muitos desenvolvimentos na área do software, têm procurado uma saída para este tipo de manipulação, permitindo formas mais naturais de expressão dos códigos, necessários para a realização de uma determinada tarefa por parte do computador.

A história das linguagens de alto nível identifica plenamente esta tendência. Tais linguagens permitem que os usuários do computador, no caso, os programadores, possam manipular informações ou comandar a máquina, mediante um programa predeterminado.

No entanto, a construção desses programas requer o conhecimento de regras sintáticas e semânticas correspondentes à especificação formal de alguma dessas linguagens. Tal requisito exclui aos usuários não especializados.

Para esse tipo de usuários, uma solução alternativa consiste na utilização de diálogos.

Em princípio, esses diálogos são constituídos por uma série de mensagens que o usuário e o computador trocam através da interface. Entretanto, o estilo destas mensagens podem ser modelado de acordo com o tipo de usuário ou tipo de aplicação.

Encontram-se, na utilização de diálogos, duas tendências que dependem fortemente de quem começa o diálogo; par

tindo deste ponto de vista, os diálogos podem ser divididos em diálogos dirigidos pelo usuário ou diálogos dirigidos pelo computador.

Diálogos dirigidos pelo usuário

Nesta modalidade, a iniciativa na execução da IUC é tomada pelo usuário.

Os modelos mais comuns deste tipo de diálogos são os seguintes:

- Linguagem Natural:

  Esta forma de interação, embora se mostre como a mais atraente, ainda não se encontra bastante desenvolvida para garantir seu lugar em aplicações comerciais.

- Linguagens de Consultas:

  Neste tipo de interação, o usuário especifica as tarefas que deverão ser realizadas pelo computador, sem se preocupar pela forma como deverão ser resolvidas. Tal modelo de IUC tem sido muito bem sucedida na operação de sistemas de banco de dados.

- Linguagens de Comandos:

  Neste tipo de IUC o usuário entrega, ao computador, uma série de instruções, que determinam a execução de determinadas tarefas. Tais linguagens de comandos são muito utilizadas por usuários que tenham um determinado grau de treinamento.

Diálogos dirigidos pelo computador

Nesta modalidade, o propósito fundamental do diálogo é a extração de determinada informação que, fornecida pelo usuário, permita o funcionamento de todo o sistema. A maneira como tal informação será utilizada pelo computador depende de cada aplicação.

Outros detalhes relacionados com todos esses tipos de diálogos serão tratados na secção 2.3 deste capítulo.

Os custos do software

Ao contrário do que acontece com o hardware, os custos do software não tem decrescido; figura (2.2); isto se deve aos seguintes motivos:

- o número de programadores especializados não tem crescido na mesma proporção do desenvolvimento dos equipamentos e a tendência é diminuir. Brascomb[38].

- A complexidade do software tampouco tem diminuido, em uma proporção que acompanhe o progresso do hardware;

- A quantidade de software produzido por um programador, (número de linhas de código numa determinada linguagem) é constante;

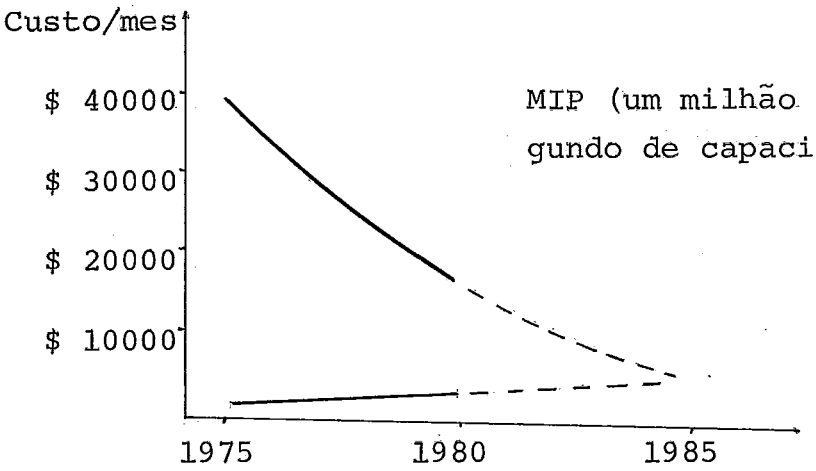


Fig. (2.2)
Relação de custos hardware vg. software nos USA

Fonte: (Lewis M. Branscomb, IBM, "Bringing Computing to People", COMPUTER, IEEE, Julho/82).

Os custos e a complexidade do software tem sido uma constante barreira para o desenvolvimento de melhores IUC. Martin[39], Zloof[36].

2.3 - Os Modelos de Interface Usuário - Computador

Tal como ficou demonstrado na seção anterior, existe uma variedade de modelos de IUC, projetados para acompanhar os diversos tipos de usuários e as várias formas de aplicações. Martin[39] apresenta uma lista que destaca as vantagens e des

vantagens de dezoito possíveis categorias de interface.

A seguir, apresentaremos os detalhes dos modelos que consideramos mais significativos.

2.3.1 - Linguagem Natural

A utilização de linguagens tais como o português, o inglês, etc.,seria certamente a maneira mais fácil do usuá interagir com o computador. No uso destas linguagens, porém,en contram-se numerosas inconsistências e ambiguidades que resul tam por exemplo: do uso de palavras que têm mais de um signifi cado.

O maior número de trabalhos envolvendo o uso da linguagem natual tem-se orientado na linha de banco de dados - Schneiderman[37], Codd[33], Coelho[6].

No entanto, apesar do grande número de trabalhos reportados, utilização da linguagem natural nesse tipo de sis temas, está ainda na fase de pesquisa - Schneiderman[37].

Outros sistemas, tais como o ELIZA - Weizenbaum[47] utilizam apenas a análise sintática para manter a conversação com o usuário - neste caso, as entradas que o usuário fornece não são entendidas pelo sistema, mas apenas aproveitadas algu mas palavras para que seja gerada uma resposta.

As maiores dificuldades que podem ser apontadas no uso da linguagem natural, na interação usuário-computador, são as seguintes:

- programação extremamente complexa da linguísti ca de uma linguagem natural - Martin[39], Saba ni[22];

- criação de um "overhead" muito forte;certas respostas do sistema ELIZA levam alguns minutos para aparecer - Martin[39];

- na utilização da linguagem natural em sistema relacionados com banco de dados, faz-se necessário o uso de diálogos que esclareçam as consultas do usuário - Codd[33], Coelho[6], porém o uso desses diálogos aumenta o "overhead" da IUC Schneiderman[37];

- nesses mesmos sistemas de banco de dados, o uso da linguagem natural pode criar, no usuário, a ilusão de uma máquina com inteligência ilimitada - Schneiderman[37], Sousa[56].

Devido a estes e outros motivos, a utilização da linguagem natural em aplicações comerciais é ainda pouco provável, porém constitui um desafio constante para a pesquisa.

2.3.2 - As Linguagens de Programação

Ao contrário do que acontece com a linguagem natural, o uso das linguagens de programação - COBOL, FORTRAN, PASCAL por exemplo - está baseado numa especificação formal das regras sintáticas e semânticas dessas linguagens.

Essa formalidade das linguagens de programação permite que os textos nelas escritos possam, facilmente,ser interpretados pelo computador.

Entretanto, essa mesma formalidade constitui a

maior barreira para a grande maioria dos possíveis usuários do computador.

Desse fato parte a necessidade de se ter diálogos que permitam, a esse grande número de usuários, o acesso à execução das aplicações desenvolvidas em tais linguagens de programação.

Existem, no entanto, algumas linguagens de programaçao interativas com o usuário (no caso, o programador).Algumas dessas linguagens constituem novas versões de linguagem já conhecidas. Assim, existem um BASIC, um FORTRAN, um PL/1, interativos.

Dentre as linguagens que foram criadas especialmente para serem interativas com o programador, destaca-se o APL.

2.3.3 - Linguagem de Consultas

Uma solução para o problema da ambiguidade da linguagem natural, assim como para a formalismo das linguagens de programação, é obtida através da definição de certas linguagens, cuja sintaxe utiliza um subconjunto da linguagem natural, porém restrito às regras de uma definição formal.

O objetivo destas linguagens é permitir, ao usuário, a especificação de tarefas que deverão ser executadas pelo computador.

Nessa especificação o usuário não se preocupa pela forma como tais tarefas deverão ser resolvidas.

Atualmente existem muitas linguagens de especificação ou consulta a banco de dados que têm sido bem sucedidas

Date[52], Zloof[36], Reisner[30]. Entre elas estão o SEQUEL, QUEL, QUERY BY EXAMPLE e outras.

Na figura (2.3)pode ser visto um exemplo que apresenta a similaridade de uma especificação feita em SEQUEL uma linguagem natural.

Consulta : obter o número de fornecedores, cujo saldo seja maior que 20.000 e que estejam localizados no Rio.

Linguagem SEQUEL :
```
SELEQ S
FROM FORNECEDORES
WHERE CIDADE = "RIO"
AND
SALDO > 20.000
```

Fig. (2.3) - Especificação de uma tarefa em SEQUEL

Uma limitação o uso dessas linguagens está em seu carater restrito a um tipo de aplicação.

Vale a pena colocar, também, que o usuário deste tipo de linguagem deverá ter um grau de conhecimento do sistema, assim como da IUC que esteja sendo utilizada.

2.3.4 - Linguagens de Comandos

Nesse tipo de diálogo, o usuário utiliza um conjunto de comandos ou instruções que ordenam ao computador a realização de determinadas tarefas.

Estas instruções são geralmente, mnemônicos ou abreviações dos nomes das tarefas que deverão ser executadas. fig. (2.4)

U: FIND/TEXTO,/100-END C: WORKFILE TESTE 100, 300, 700	Programa Editor do Burroughs-6700 comando para encontrar um determinado texto.

Fig. (2.4) - Exemplo da utilização de um comando, constituido por uma sequência de instruções.

Numa linguagem de comandos podem ser distinguidos os seguintes atributos: O estilo, a estrutura e o nível de abstração - Hardy[5].

O estilo

Este atributo das linguagens de comandos, está relacionado com o tipo dos textos utilizados. O estilo poderia ser tão claro quanto a linguagem natural ou resumido a umas poucas letras.

A estrutura

As instruções de uma linguagem de comandos poderão ser utilizadas de maneira separada ou organizadas numa certa estrutura sintática.

Os comandos poucos estruturados, tendem a expressar as tarefas em poucas palavras.

Os comandos mais estruturados, estão preparados

para serem usados em diversas opções, permitindo inclusive que algumas delas possam ser do tipo "default".

O nível de abstração

Esse atributo das linguagens de comandos, refere-se à quantidade de tarefas que podem ser realizadas pelo computador, em função de um comando recebido.

Esses três atributos apresentados, deverão ser equacionados com o grau de utilização da interface. Neste sentido as linguagens de comandos que sejam mais estruturadas, de maior nível de abstração e cujos textos lembrem a linguagem natural, terão tempos de aprendizagem menores e sua utilização será mais livre de erros semânticos.

Por outro lado, as linguagens de comandos de baixo nível de abstração e pouco estruturadas serão muito mais flexíveis e, ao mesmo tempo, difíceis de serem utilizadas por usuários que não tenham um certo grau de treinamento.

2.3.5 - Diálogos Dirigidos pelo Computador

Nesse modelo de interface, o computador apresenta ao usuário as facilidades disponíveis do sistema utilizado.

Tal apresentação realiza-se através de uma sequência de consultas formuladas pelo computador e respostas do usuário que, a medida que vão sendo realizadas, determinam a execução de determinadas ações.

A organização dos textos apresentados pelo computador e das respostas esperadas do usuário, dependem de cada aplicação.

Uma característica que deve ser destacada neste modelo de IUC é a facilidade de poder ser utilizada por diferentes tipos de usuário. Isto pode ser obtido através do dimensionamento do número, extensão e complexidade dos textos que deverão ser apresentados.

As formas mais comuns de diálogos dirigidos pelo computador são as seguintes:

- Instruções aos usuário

  Exemplo: C: ENTRE COM SEU NÚMERO DE CONTA

  U: cos99201/abc

  C: CONTA NÃO REGISTRADA,
  ENTRE COM SEU NÚMERO DE CONTA

  U:

- Seleção em "menu"

  Esse diálogo é apropriado para os casos em que exista um conjunto limitado de alternativas,que possam ser escolhidas pelo usuário - Fig.(2.5).

C: ALTERNATIVAS OFERECIDAS PELO SISTEMA

1 INFORMAÇÕES SOBRE A BASE DE DADOS
2 TROCAS NA SUA ATUAL CONSULTA
3 ENTRADA PARA UMA NOVA CONSULTA
4 RESUMO DAS CONSULTAS REALIZADAS
5 FIM DA SESSÃO
/__ / espaço para resposta

Fig. (2.5) - Exemplo de "menu".

Como pode ser visto, a resposta do usuário a uma tela tipo menu não vai além de uns poucos caracteres, que identificam o item escolhido ou o desejo de ajuda por parte do usuário.

Interessantes sugestões relacionadas à preparação de telas de tipo menu, são colocadas por Simpson[53].

O benefício da utilização de menus pode ser resumido nos seguintes aspectos:

- o usuário recebe uma visão estruturada do sistema que estiver sendo executado;

- o usuário não precisa lembrar-se de uma determinada sintaxe, tal como acontece com as linguagens de comandos;

- a probabilidade de erros diminui consideravelmente, pois os menus podem ser preparados para interagir com diferentes tipos de usuários;

- o usuário poderá responder a um menu solicitando um novo menu de auxílio.

Esta facilidade de uso dos menus na IUC, constitui ao mesmo tempo, uma forte limitação para aplicações, nas quais a IUC deva interagir com usuários mais treinados. Nestes casos, a alternativa de uma concisa linguagem de comandos sempre constituirá uma melhor escolha, Hardy[5].

A apresentação de menus requer terminais de vídeo cujas velocidades de transmissão sejam relativamente rápidas. Desta maneira, o tempo da apresentação das telas não influirá negativamente no desempenho da interface, Schneiderman[37].

Uma forma de incrementar a flexibilidade da IUC que utilize menus, é permitindo que o usuário possa alterar a sequência da apresentação das telas, à medida em que cresce o seu conhecimento da interface, Heffler[19].

Apresentação de telas para a coleta de parâmetros

Nesse modelo de interface, o computador apresenta textos que explicam ao usuário, os detalhes dos parâmetros que deverão ser fornecidos. Fig. (2.6).

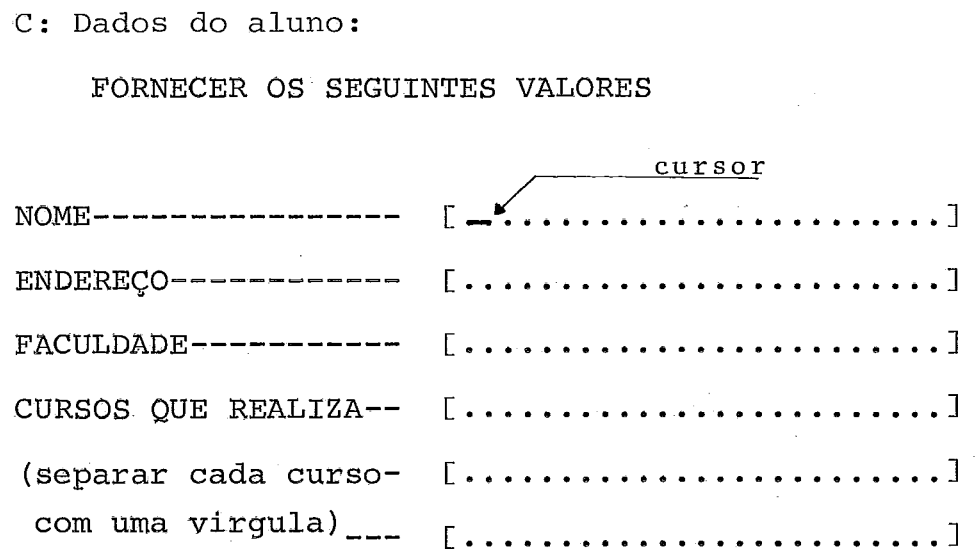


Fig. (216) - Esquema de uma tela para coleta de parâmetros.

Tal esquema apresenta várias vantagens:

- permite que possam ser utilizadas uma ou mais variáveis, cujos valores poderão ser utilizados, dependendo de cada aplicação;

- a coleta de parâmetros alivia a restrição de escolha simples, colocada pela apresentação de menus;

- a coleta de parâmetros pode adquirir a flexibilidade de uma linguagem de comandos, à proporção que sejam retirados os textos explicativos.

A utilização de telas para coletar parâmetros não está livre de erros, pois, contrariamente ao que acontece no caso dos menus, os valores que deverão ser fornecidos pelo usuário não podem estar contidos dentro de um conjunto restrito de escolha. Por este motivo, a utilização destas telas deverá estar preparada para fornecer, ao usuário, mensagens de auxílio, que facilitem o conserto de valores errados.Martin[39], Heffler[19], Negus[18].

É válido esclarecer que os projetos de IUC podem não utilizar, exclusivamente, um tipo de tais formas de diálogos, mas sim uma combinação delas, que venha a prestar uma maior flexibilidade para cada aplicação. Assim, é possível que sejam utilizados menus para apresentar as ações que podem ser executadas pela IUC, tela para coleta de parâmetros para obter os valores necessários em um determinado instante da execução e apresentar instruções ao usuário, nos casos em que tenha cometido algum tipo de erro.

2.4 - Os Fatores que caracterizam o bom desempenho da IUC

O sucesso do desempenho da IUC está relacionado a um conjunto de fatores que facilitam o relacionamento usuário-computador. De maneira geral, tais fatores podem ser analisados através dos seguintes conjuntos:

- facilidade do uso da IUC;

- versatilidade da IUC;

- capacidade de auxílio, para o usuário;

- tratamento de erros.

É possível que existam outros fatores relacionados ao desempenho da IUC que não estejam contidos nos conjuntos acima mencionados, porém este agrupamento mostra uma linha de análise que pode servir de base para a construção de sistemas de interação atraentes para o usuário.

2.4.1 - Facilidade de uso da IUC

O conhecimento do tipo de usuário:

A facilidade de uso de qualquer IUC está ligada ao conhecimento do tipo de usuário para qual a IUC deverá ser projetada.

Esse conhecimento é fundamental, pois serve de base para a obtenção do restante dos parâmetros necessários, durante o projeto da interface. Classificações dos tipos de usuários, associadas suas possíveis aplicações, são apresentadas por Martin[39].

Minimizar a memorização: Schneiderman[34]

As limitações da memória a curto prazo do usuário podem ser compensadas com as seguintes técnicas:

a) utilização de menus; Ketil Bo[8].

b) comportamento consistente da IUC, Simpson[53].

c) acesso a informações anteriores;

d) tempos de resposta curtos e pouco variáveis.

e) mantimento de uma constante "feedback" a cada uma das ações do usuário.

Uso de textos compatíveis com os conhecimentos do usuário; Simpson[53].

É importante que as mensagens apresentadas pelo computador sejam compatíveis com os conhecimentos sintáticos e semânticos do usuário - caso contrário, só aumentam as probabilidades de erro ou pioram o desempenho da interface, Schneiderman[34].

Algumas recomendações podem ser colocadas nesse sentido:

a) clareza na apresentação das telas;

b) não confundir os conhecimentos sintáticos e semânticos do projetista, com aqueles que o usuário possui;

c) utilizar textos que sejam autoexplicativos.

A motivação do usuário

A motivação ou desmotivação do usuário na utilização de uma IUC são fatores determinantes do sucesso dessa interface. Assim, uma posição negativa do usuário pode, simplesmente, conduzir a situações nas quais aquela IUC deixará de ser utilizada na procura de melhores alternativas.

Na preparação dos textos que o computador terá de apresentar, existem maneiras que, além de explicar ao usuário o que está acontecendo, podem, também, motivá-lo a continuar interagindo.

Por exemplo:

a) "ILLEGAL PASSWORD" poderia ser substituido por:

"SUA SENHA NÃO COINCIDE COM A SENHA REGISTRADA"

b) "INVALLID COMMAND" poderia substituir-se por:

"OS COMANDOS PERMITIDOS PELO SISTEMA SÃO:
LOAD, SAVE OR EXPLAIN"

No caso de interfaces a serem usadas por usuários casuais, mensagens do tipo FATAL ERRO, RUN ABORTED, deverão provocar,muito provavelmente, uma grande desmotivação Brans comb[38].

2.4.2 - Versatilidade da IUC

Uma das características do homem é tornar-se cada vez mais esperto, à medida em que aumenta sua experiência e o grau de conhecimento sobre determinado aspecto.

Por esse motivo, a IUC deve preparar-se para reagir de acordo com o grau de treinamento que o usuário apresenta.

Uma outra razão para a versatilidade da interface é a possibilidade de ser usada por uma gama de usuários, com difirentes conhecimentos e atitudes. Algumas formas de tornar uma IUC mais flexível são as seguintes:

a) colocar opções para que os usuários mais experimentados possam utilizar caminhos mais curtos, dentro de uma sequência de ações preparadas para usuários com um menor grau de utiliza

ção; Heffler[19].

b) colocar alternativas para que o usuário possa decidir quanto ao tamanho das mensagens de auxílio ou de crítica e recuperação de erros; Schneiderman[34].

2.4.3 - Capacidade de auxílio para o usuário

Em qualquer instante da interação com o computador, o usuário pode precisar de informações de auxílio que o ajudam na tomada de alguma decisão.

De acordo com Waston[4], essas informações podem ser dos seguintes tipos:

a) Informações sobre ações executadas:
É sempre provável o interesse do usuário por recordar a sequência de ações que podem ter sido executadas durante sua interação com a IUC. Estas informações poderiam-lhe permitir retornar a algum estado de execução anterior.

b) Informações sobre o estado atual de execução:
Nesse caso, a IUC pode fornecer, ao usuário, orientação sobre as alternativas que poderiam ser executadas a partir do estado em que se encontra;

c) Informações sobre os detalhes de como continuar execução:

Estas informações ajudariam ao usuário a encontrar as maneiras ou a sintaxe que a IUC precisa para continuar as ações, sobre uma determinada alternativa.

Uma característica particular das informações de auxílio será sua apresentação por níveis, para desta forma, permitir uma maior flexibilidade à IUC.

2.4.4 - A prevenção e o tratamento de erros

Uma das melhores maneiras de tratar os erros que podem acontecer durante a execução da IUC é impedir que eles aconteçam. Neste sentido, o projeto da IUC deverá ser feito de forma a minimizar o acontecimento das circunstâncias que tendem a provocar erros nas respostas do usuário.

De acordo com Norman[46], a maior parte dos erros cometidos pelo usuário da interface, podem ser dos seguintes tipos:

a) Erros de modo:

Neste tipo de erros, o usuário geralmente oferece uma resposta "certa", porém dentro de um contexto errado. Estes tipos de erro são comuns nas respostas do usuário. Exemplo classico deste tipo de erro é a utilização do editor de textos, quando, na verdade, o sistema está no estado de linguagem de comandos;

b) Erros de descrição:

Este tipo de erros comete-se quando o usuário responde da maneira que ele considera correta, de acordo com a descrição percebida. Exemplo deste tipo de erro são as respostas que o usuário pode dar corretamente, porém no item errado;

c) Erros por falta de consistência:

Neste caso o usuário pode responder corretamente de acordo com situações que ele encontra similares, mas que não são, entretanto, "estritamente similares". Exemplos destes tipos de erros podem acontecer, quando o usuário responde de acordo com uma sintaxe que costuma ser padrão do sistema, porém, nesse caso em particular, devia ser diferente.

As recomendações que podem ser colocadas para este tipo de erros, são decorrentes das próprias descrições que foram anotadas.

O tratamento de erros:

No tratamento dos erros por parte da IUC podem ser colocadas as seguintes considerações:

- detecção oportuna de erros cometidos pelo usuário e apresentação de mensagens que identifiquem, claramente, o erro e a viabilidade do conserto;

- verificação da consistência dos dados fornecidos pelo usuário, pois o fornecimento desses dados constitui ponto crítico para o correto desempenho da IUC;

- a utilização de formatos é, a miúdo, fonte de erros na entrada de dados, uma maneira mais segura pode ser obtida, permitindo a entrada de dados em formato livre, no entanto, a entrada dos dados, neste tipo de formato, precisa de uma clara descrição dos espaços onde deverão ser preenchidos os valores de cada dado;

- a IUC deverá impedir, enquanto for possível, que o usuário possa cometer erros difíceis de serem retificados.
  Exemplo: a perda de arquivos.

Desta maneira, encerramos o estudo dos fatores relacionados ao projeto da interface usuário-computador.

A maior parte das matérias tratadas, pode ser encontradas na literatura que tem sido referenciada. Contudo, certos aspectos que tem sido abordados, respondem ao critério do autor ou são o resultado de simplificações e condensações realizadas sobre materias pesquisadas em vários artigos.

III - DIVERSAS ABORDAGENS NA IMPLEMENTAÇÃO DA IUC

A implementação de uma interface usuário-computador, que satisfaça as características de bom desempenho, que foram colocadas na secção 2.4 do capítulo II, requer de um esforço considerável na produção de software, cujo grau de complexidade e número de instruções aumenta, na medida em que essa interface consegue atingir um maior número de usuários não especializados em computação. Branscomb[38].

Nos modelos de implementação que apresentaremos a seguir, constam as diferentes tendências que existem, com relação à maneira como deve ser construída a IUC de um determinado sistema de aplicação.

Nestas tendências, destacam-se as seguintes:

- Instruções da IUC misturadas com o software de aplicação;
- A IUC implementada num sistema estruturado;
- A IUC produzida através do uso de geradores de software.

Para encerrar, apresentaremos a nossa proposta para a implementação da interface de um sistema de aplicação.

3.1 - Instruções da IUC misturadas com o Software de Aplicação

Nesta forma de implementação da IUC, as instruções correspondentes à comunicação com o usuário encontram-se intercaladas com as que correspondem ao software de aplicação. Tal método de solução é direto, pois permite que o projetista dos

programas de aplicação possa, conforme torne-se necessário, colocar as instruções que permitem que o usuário forneça as informações requeridas para a execução do sistema. Neste modelo de implementação devem ser considerados os seguintes aspectos:

- A complexidade da produção de todo o software do sistema é incrementada, pois devem ser resolvidos ao mesmo tempo, os problemas referentes à aplicação com aqueles que correspondem à interface com o usuário.

- Será dificil conseguir algum tipo de padronização nos projetos de interface, pois cada solução fica demasiado ligada aos problemas específicos de uma determinada aplicação.

- Pela mesma razão anterior, o software de uma determinada interface, dificilmente poderá ser utilizado em novas aplicações.

- A realização de modificações tanto no projeto da interface como no software de aplicação, fica prejudicada por causa dessa interrelação existente.
  Apesar das desvantagens apresentadas, este tipo de implementação é amplamente utilizado, talvez por permitir uma construção mais direta do software de aplicação.
  Sua utilização pode ser perfeitamente aceitável em sistemas pequenos nos quais o uso de outros modelos, mais complicados de implementação da IUC não seriam total

mente justificáveis.

3.2 - A IUC implementada em um Sistema Estruturado

Nesta forma de implementação, tanto o software da IUC como aquele dos programas de aplicação, é dividido em blocos, cujas funções encontram-se claramente determinadas. Yourdon, 1979.

A implementação de um sistema de aplicação, quando feita de maneira estruturada, elimina a maioria das desvantagens que foram apontadas para o modelo de implementação anterior e, ao mesmo tempo, consegue produzir os seguintes benefícios:

a) a execução paralela da programação do software de aplicação e do software da interface com o usuário implica em uma diminuição do tempo e do custo total de implementação;

b) a formação de equipes especializadas no desenvolvimento da interface com o usuário;

c) a diminuição do grau de complexidade da implementação, pois as duas tarefas seriam tratadas separadamente;

d) a facilidade para a utilização dos modelos da programação estruturada, cuja aplicação significa o aproveitamento de todas as vantagens do projeto estruturado. Stevens (40).

A implementação da IUC em um sistema estruturado apresenta duas formas de solução:

a) no contexto das instruções do software de aplicação, produzem-se chamadas para determinados blocos da IUC, que após interagir com o usuário, devolvem, ao programa de aplicação certas informações necessárias para continuar sua execução; Neste tipo de solução, a implementação da IUC depende das necessidades do software de aplicação e, por tal motivo, os blocos de software que compõem a IUC podem ser considerados como unidades independentes. Esta independência, entretanto, requer que cada bloco da interface apresente todas as características que foram apontadas para garantir o bom relacionamento usuário-computador.

b) no contexto das instruções do software da IUC, produzem-se chamadas para a execução dos programas de aplicação. Neste modelo de implementação, o software da IUC constitui, de alguma maneira, o programa principal de todo o sistema de aplicação.

   Esta segunda solução pode ser conseguida considerando-se a IUC o módulo de controle do sistema de aplicação. A execução deste módulo pode ser feita em face a aplicação do modelo de estados finitos. Esta aplicação foi proposta primeiramente por Arnas, 1969. Aplicações mais recentes são apresentadas em Dwyer[14], Jacob[43], Bass[10], Casey[17].

De acordo com esse modelo, a execução da IUC começaria em um estado inicial, no qual o usuário deveria fornecer a identificação e os dados de entrada do primeiro comando que deva ser executado.

Para satisfazer o pedido do usuário, a IUC deveria, então, cuidar dos seguintes aspectos:

- verificar se aquele comando pode ser aceito;
- comprovar a validade dos dados de entrada;
- caso tais verificações tenham sido bem sucedidas, chamar o programa de aplicação que executa o comando; caso contrário, interagir com o usuário, na procura de novas informações;
- colocar-se num estado seguinte da execução, a partir do qual o usuário ordenará a execução de novos comandos.

O software necessário para a implementação desse tipo de solução pode ser resumido nos seguintes pontos:

a) construção das tabelas de estados

b) preparação das rotinas de avaliação dos dados de entrada e auxílio ao usuário;

c) inclusão das rotinas de aplicação.

Uma limitação que deve ser apontada a esse modelo de solução é a dificuldade em se conseguir a execução de comandos aninhados, ou comandos condicionais encadeados, Dwyer[14].

3.3 - Utilização de Geradores de Software

Uma outra alternativa na criação de IUC consiste na utilização de geradores de software, que automatizam o processo de implementação.

O objetivo básico destes geradores é receber, do projetista, uma descrição relativamente simples do desenho da interface e traduzir esta descrição para alguma linguagem de programação.

A utilização mais comum para estes geradores, tem sido a área da instrução assistida pelo computador, visando talves, liberar aos instrutores das tarefas da programação Martin[39]. No entanto, é possível que esses geradores sejam utilizados, na produção de uma grande parte do software da IUC de qualquer sistema de aplicação. Heffler[19].

Uma estrutura genérica para este tipo de geradores poderia ser composta pelos seguintes módulos:

a) um módulo editor:

neste módulo seria feita a recepção da descrição da interface; durante a entrega dessa descrição, o editor poderia detectar pequenos erros sintáticos, como por exemplo o uso indevido de palavras reservadas - o resultado final do uso deste editor seria a criação de um arquivo, contendo a descrição recebida, que serviria de entrada para um módulo gerador, ou para a realização de futuras modificações;

b) um módulo gerador:

Este módulo verificaria a validade da descrição da IUC recebida e a traduziria, de forma au

tomática, para programas escritos na linguagem de programação escolhida.

c) uma estrutura de arquivos, contendo dados de entrada e/ou software de suporte para o código produzido.

Tal como ficou colocado na secção 1.1 do capítulo I, a utilização deste tipo de geradores de software, traria as seguintes vantagens:

- a utilização de um mecanismo gerador dispensaria, o projetista, de grande parte do trabalho de produção do software da IUC;

- este gerador serviria de ponte entre as necessidades finais do usuário e a maneira de projetá-las no computador;

- o tempo de produção do software da IUC seria diminuido;

- facilitar-se-ia a criação de IUC fáceis de serem modificadas;

- a utilização deste tipo de instrumento permitiria que o tipo de IUC por ele produzido, conseguisse de maneira espontânea, muitos dos fatores que caracterizam o bom relacionamento usuário-computador.

Contudo, a utilização deste tipo de geradores poderia significar a criação de modelos de interfaces demasiado pa

dronizadas ou que não possam ser utilizadas para resolver qualquer tipo de aplicação. Esta limitação pode ser resolvida permitindo que as interfaces produzidas, possam aceitar segmentos de software especialmente projetados para interagir com o usuário nos casos mais complicados, tais como na verificação dos dados de entrada ou na preparação de procedimentos de auxílio. Heffler[19], Negus[18].

A seguir, apresentaremos as características mais relevantes de três exemplos de geradores de diálogos que foram pesquisados, e que tem servido de base, para a formulação do modelo de gerador de diálogos, apresentado neste trabalho.

3.3.1 - DIALOG: A scheme for the Quick and Effective Production of Interactive Application Software. Negus, B. et al.[18].

O DIALOG consiste em um conjunto de rotinas que, quando utilizadas juntamente com os programas de aplicação de um determinado sistema, podem constituir a IUC deste sistema.

O DIALOG foi desenvolvido pelo Computer Center da University of Technology, Loughborough U.K. em 1979.

Os objetivos procurados pelo DIALOG são os seguintes:

- criação de interface para ser utilizada por três diferentes tipos de usuários: casuais, ocasionais ou regulares;

- dispensar o projetista dos detalhes da implementação desse tipo de IUC;

- outros objetivos mencionados são: a comunicação

mais robusta, a padronicação e a possibilidade de modificações no software produzido através do DIALOG.

A comunicação usuário-computador, obtida a partir do uso do DIALOG, baseia-se nos seguintes pontos:

- apresentação, ao usuário, das tarefas que podem ser realizadas pelo sistema;
- a escolha do usuário de uma determinada tarefa para ser executada;
- apresentação, ao usuário, dos textos explicativos, correspondentes aos valores que servirão de entrada para a execução dessa tarefa;
- execução da tarefa escolhida;
- possibilidade do usuário escolher novas tarefas.

Os programas produzidos pelo DIALOG permitem, além disso, que o usuário possa responder com comandos de HELP, STOP, CANCEL e outros, durante a escolha de qualquer tarefa.

A Interface do DIALOG com o projetista

A estrutura de um sistema de aplicação produzido através do DIALOG está dividido em dois módulos:

O primeiro, que é criado pelo DIALOG constitui a IUC e fica disponível para ser utilizado na forma de código objeto;

O segundo módulo está composto pelo software de aplicação preparado pelo projetista.

O sistema DIALOG requer que este software de apli

cação esteja dividido em segmentos que realizem determinadas tarefas e que façam a coleta dos parâmetros para a execução de cada tarefa.

Tanto os segmentos que executam as tarefas, quanto os que coletam os parâmetros, deverão ter nomes colocados pelo projetista.

É necessário, também, que cada um dos parâmetros utilizados tenha um nome.

Uma última condição imposta pelo DIALOG é a necessidade de fornecer todo o software de aplicação escrito na linguagem FORTRAN.

A forma como todas essas informações são fornecidas ao DIALOG, pode ser descrita nos seguintes termos:

a) criação de uma subrotina SETUP, que defina os nomes de todos os segmentos de software que executem tarefas e os associe aos nomes dos segmentos que realizarão a coleta dos parâmetros necessários para a execução dessas tarefas;

b) fornecimento de todos os segmentos de aplicação;

c) fornecimento de uma subrotina que apresente ao usuário, todos os textos explicativos das descrições de cada tarefa;

d) fornecimento das subrotinas de HELP, que usualmente serão em número de uma para cada tarefa.

Vantagens que podem ser apontadas ao uso do sistema DIALOG

- criação de uma interface flexível, para três tipos de usuário, partindo de um só tipo de descrição;

- facilidade do usuário em responder, com comandos de HELP e outros previstos pelo DIALOG, durante qualquer escolha de tarefa a ser executada pelo computador;

- facilidade do usuário em controlar o cumprimento das mensagens recebidas;

Aspectos negativos do uso do DIALOG

- embora a linguagem FORTRAN seja muito popular na produção de software, seu uso exclusivo no DIALOG pode constituir uma limitação;

- uma outra limitação constitui a necessidade de se definir, no DIALOG, todas as variáveis a ser utilizadas - isto impede a leitura de valores que estejam gravados em um arquivo em disco;

- o fornecimento da descrição da interface é feito na base da preparação de subrotinas escritas de maneira FORTRAN-like, isto pode constituir uma limitação na execução de futuras modificações.

3.3.2 - DIAGEN: A DIALOGUE GERATOR

Daiser, P. et al.[16].

O DIAGEN é uma ferramenta para a criação de software da IUC. O projetista que utiliza este sistema deverá fornecer os textos que serão apresentados ao usuário, por outro lado, o DIAGEN prove ao projetista, dos meios necessários para a definição da validade das respostas.

O DIAGEN foi desenvolvido no Centro de pesquisa de Computação da Bratislava, Czchoslovakia, em 1977.

As funções previstas para o sistema DIAGEN são as seguintes:

- geração de diálogos a partir das descrições recebidas;

- colocação de funções para o auxílio da comunicação ou apresentação de possíveis alternativas;

- coleta de parâmetros e execução de programas de aplicação.

A estrutura do DIAGEN apresenta os seguintes componentes:

a) um tradutor para uma linguagem DIAGEN, que gera o diálogo;

b) um interpretador dos diálogos, que interage com o usuário;

c) um sistema de arquivos que armazena os programas de aplicação e os diálogos produzidos.

A linguagem DIAGEN

Os diálogos aceitos pelo DIAGEN são compostos de "passos".

Um passo consiste em uma pergunta formulada pelo computador e de uma resposta do usuário.

O projetista dos diálogos deverá, então, fornecer, ao DIAGEN, as seguintes informações:

- os textos das consultas que serão apresentadas ao usuário;

- a especificação do conjunto de possíveis respostas; e

- a definição das ações que o computador deverá realizar, a partir de cada resposta.

O DIAGEN recebe todas estas informações, estruturadas em um conjunto de sequências, de acordo com a sintaxe apresentada na Fig. (3.1).

```
      labelp      DISP          panelid
                  SELECT
test expression:  action, . . .  GO TO labelp
      .           .                .
      .           .                .
                  END helpid
```

labelp = nome da sequência
panelid = identificador para o texto a ser apresentado
helpid = identificador do procedimento de auxílio à sequência.

Fig. (3.1) - Sintaxe para a entrada do diálogo no sistema DIAGEN

O interpretador de diálogos do DIAGEN

As funções previstas para o interpretador são as seguintes:

- apresentação, ao usuário, da lista de diálogos disponíveis;
- apresentação da descrição de algum diálogo em particular;
- interpretação e execução do diálogo escolhido pelo usuário;
- apresentação de mensagens explicativas sobre o status da execução de alguma tarefa realizada pelo diálogo.

A execução do interpretador pode ser resumida nos seguintes termos:

- o usuário pede ao DIAGEN, através de um nome, que um determinado diálogo seja executado;
- o interpretador coloca, no vídeo do usuário, a primeira consulta desse diálogo;
- a resposta que o usuário fornecer será comparada com as expressões de teste dessa sequência, o que determinará quais ações deverão ser executadas;

Uma característica importante do interpretador é a possibilidade de apresentar, ao usuário, uma história das interações realizadas, permitindo-lhe a realização de modificações para respostas previamente fornecidas.

A comunicação com o usuário termina no momento em que o interpretador encontra, no diálogo, uma chamada para a

execução de algum programa de aplicação.

Vantagens que podem ser apontadas no uso do DIAGEN

- as descrições de diálogos recebidas são facilmente modificáveis;

- facilidade para o agrupamento de vários programas de aplicação em uma biblioteca de programas;

- criação, a baixo custo, de diálogos especialmente úteis para usuários casuais;

- apresentação da história da interação do usuário com o computador, durante a execução de um diálogo.

Aspectos negativos no uso do DIAGEN

- este sistema só permite a realização de uma tarefa por diálogo;

- a IUC criada através do DIAGEN não permite que o usuário possa utilizar alternativas mais compactas, na medida em que se aumenta o conhecimento do sistema;

- a definição dos testes de comparação, assim como das ações que deverão ser realizadas pelo computador, deve ser feita em uma linguagem que não tem sido claramente especificada.

3.3.3 - MCIS: MENU CREATION AND INTERPRETATION SYSTEM

Heffler,[19]

O sistema MCIS é uma ferramenta que estrutura o acesso do usuário a programas de aplicação ou facilidades do computador, através de uma interface baseada em um sistema de menus.

O MCIS foi desenvolvido nos laboratórios da BELL, New Jersey, USA.

A utilização do MCIS pode ser feita em três fases: criação e definições dos menus, utilização do sistema criado e realização de modificações.

A criação de menus

O MCIS fornece ao projetista um programa interativo que recebe a descrição do sistema a ser criado. Tal sistema está constituído de menus ou listas de ítens a serem escolhi pelo usuário e de menus para coleção de parâmetros.

O projetista do MCIS poderá associar, a cada item de um menu, uma das seguintes ações: apresentar um novo menu, executar uma rotina de aplicação ou apresentar um menu que faça uma coleta de parâmetros e, logo após, executar uma rotina de aplicação que receba, como entradas, tais parâmetros assim coletados.

A utilização do sistema de menus

A utilização de um sistema criado pelo MCIS pode ser descrita nos seguintes termos:

- o usuário chama, através de um nome, o sistema de menus;

- esta chamada recebe, como resposta, a apresentação de um menu inicial, ao qual ele pode responder das seguintes maneiras:

  a) escolhendo um determinado item;

  b) pedindo uma lista dos menus do sistema;

  c) solicitando a apresentação de um determinado menu

Realização de modificações

O MCIS fornece ao projetista um sistema editor de menus, que permite que sejam realizadas, facilmente, qualquer tipo de modificações em um sistema de menus já criado.

A utilização do MCIS produz os seguintes resultados:

- dois arquivos contendo a descrição dos menus recebidos;

- uma função da escrita em "C", que será usada para as chamadas das rotinas de aplicação;

- um arquivo de comandos (UNIX shell script), que realizará a compilação do programa que interpretará os menus, da função escrita em "C" e das rotinas de aplicação. O resultado desta compilação será o novo sistema de menus.

Uma condição imposta pelo MCIS requer que todo o software de aplicação esteja descrito na linguagem "C".

Vantagens que podem ser apontadas no uso do MCIS

- a utilização de menus fornece, ao usuário, uma visão estruturada das possibilidades do sistema;

- o sistema interativo do MCIS, para a criação dos menus, dispensa o projetista do conhecimento de uma determinada sintaxe para a entrada das descrições destes menus;

- o programa editor do MCIS permite um fácil método de modificação ou atualização dos menus;

- a apresentação das listas dos menus do sistema criado e a possibilidade do usuário escolher a execução de um determinado menu, constituem características da flexibilidade da IUC criada através do MCIS;

- a utilização de menus para coletar parâmetros incrementa as possibilidades de um sistema, baseado unicamente na escolha de ítens.

Aspectos desfavoráveis ao uso do MCIS

- a IUC que pode ser criada através do MCIS só pode ser utilizada para a execução de programas de aplicação.

- uma outra limitação da utilização do MCIS é a restrição do uso de programas de aplicação escritos na linguagem "C" que ainda não é muito utilizada.

- o MCIS não permite que os programas de aplicação possam utilizar a interface criada;

- o MCIS só permite a execução de um programa de aplicação por item escolhido de um menu.

3.4 - "G D" : Um Gerador de Diálogos

A proposta do presente trabalho para a implementação da interface usuário-computador, de um determinado sistema de aplicação, consiste de um gerador de diálogos, que recebe, como entrada, a descrição dessa interface e produz, como resultado, código fonte.

Esse código fonte gerado, unido ao software de rotinas de auxílio ao usuário e recuperação de erros, constituem a IUC desse sistema de aplicação.

Os tipos de diálogos que podem servir de entrada para esse gerador também estão baseados em menus e telas para coletar parâmetros que logo serão utilizados pelos programas de aplicação.

O sistema de menus e telas para coleta de parâmetros foi adotado tendo em consideração os seguintes motivos:

- tal como foi colocado na secção 2.3.5 do capítulo II, telas do tipo menu, podem ser utilizadas por uma grande quantidade de usuários.

- as limitações das telas de tipo menu são quase totalmente absorvidas pelas telas de tipo coleta de parâmetros.

- para os casos nos quais tanto as telas de tipo menu como as que coletam parâmetros, sejam inadequa

das, o sistema proposto permitirá a utilização de segmentos de programas, que constituem solução definitiva para aquele tipo de circunstâncias.

Neste sentido, o sistema proposto aproveita o esquema colocado no MCIS[19] que já foi descrito.

No entanto, o tipo de utilização prevista para o software, produzido através do uso desse gerador, pode ser de dois tipos:

a) acesso estruturado do usuário à programação de aplicação, através de um sistema de menus; (+)

b) coleta de parâmetros que logo poderão ser utilizados no contexto de um determinado programa de aplicação. (*)

Estas duas formas de utilização permitem que o sistema proposto seja mais flexível e, ao mesmo tempo, possa ser utilizado em um maior número de aplicações.

Outras características do software produzido através do uso desse sistema são as seguintes:

- facilidade do usuário em executar, através da escolha de um determinado item de menu, uma sequência de programas de aplicação, com ou sem parâmetros; (*)

- facilidade do projetista da interface em associar a apresentação de um menu à prévia execução de uma sequência de comandos; (*)

(+) Características aproveitadas do sistema MCIS

(*) Características únicas do sistema "G D" proposto neste trabalho

- o usuário poderá responder a um "prompt" de menu com o nome de outro menu, o que torna a IUC mais flexível; (+)

- o fornecimento das descrições da IUC é feito de maneira interativa; (+)

- as descrições recebidas podem ser modificadas, através do uso do editor do computador; (*)

- o software gerado através desse sistema poderá ser utilizado tanto por um usuário final,como por programa de aplicação; (*)

- os programas de aplicação, relacionados ao uso da IUC produzida pelo gerador, poderão estar escritos nas seguintes linguagens: COBOL, FORTRAN e ALGOL; (*)

A especificação e modelagem do GERADOR DE DIALOGOS "G D" é apresentada no Capítulo seguinte.

(+) Características aproveitadas do sistema MCIS

(*) Características únicas do sistema "G D" proposto neste trabalho

IV - MODELAGEM E ESPECIFICAÇÃO DO GERADOR DE DIALOGOS "G D"

4.1 - Introdução

Neste capítulo realizaremos a apresentação do "G D", cuja proposta fora colocada na secção 3.4 do capítulo anterior.

Esta apresentação compreende os seguintes aspectos:

- Em primeiro lugar será colocada a definição do "G D" e das condicionantes que resultam da sua implementação física.

- A seguir, serão apresentadas as características físicas e semânticas dos tipos de diálogos aceitos pelo "G D". A justificativa para a escolha desses tipos de diálogos pode ser vista nas seções 2.3.5 e 2.4 do capítulo II e 3.4 do capítulo III.

- O processo de interpretação dos diálogos foi dividido em duas etapas:
  Na primeira o projetista fornece ao "G D" as descrições desses diálogos.
  Na descrição dessa etapa, serão apresentadas a interface do "G D" com o projetista e a estrutura dos arquivos em disco que salvarão as descrições desses diálogos.
  Numa segunda etapa o "G D" interpreta as descrições recebidas e com elas geram código fonte.
  Na descrição dessa segunda fase, serão apresentadas a estrutura dos programas fonte gerados.

- Para encerrar a apresentação do "G D"; serão tratados os aspectos mais relevantes da sua utilização, nesses aspectos estão incluídos:

  - Os detalhes da utilização da interface do "G D" com o projetista da IUC,

    A realização de atualizações,

    A utilização do instrumento gerador de código fonte, e

    O uso previsto para o código fonte gerado.

A especificação e modelagem desse "G D" teve como fonte de consulta o modelo MCIS[19], e os geradores de diálogos DIAGEN[16] e DIALOG[18] que foram apresentados na secção 3.3 do capítulo anterior.

O código fonte produzido pelo "G D" está escrito na linguagem ALGOL (*).

Os detalhes que correspondem ao uso dessa linguagem foram consultados no manual do ALGOL da Burroughs[55].

A implementação do "G D" será discutida no capítulo V.

Exemplos do uso do "G D" podem ser encontrados no Apêndice I.

(*) A escolha da máquina, assim como da linguagem ALGOL, são discutidas no início do capítulo V.

4.2 - Descrição do Gerador de Diálogos "G D"

O "G D" é um sistema interativo, que recebe como entrada descrições de diálogos e as traduz para código fonte. Este código produzido pelo "G D", unido ao software dos programas de aplicação e à procedures de auxílio ao usuário e recuperação de erros, constituirão a interface com o usuário de um determinado sistema de aplicação.

A esssa definição devem ser acrescentadas algumas restrições de implementação.

Tais restrições podem ser divididas em dois grupos:

1 - aquelas que provêm do hardware utilizado: Computador Burroughs B-6700.(*)

- A implementação no B6700 implica em que o sistema "on-line", obtido no uso do "G D", seja aquele que pode ser sustentado pela rede de terminais remotos TD-110 e TS-800, atualmente ligados ao B-6700 através de uma linha, cuja taxa de transmissão oscila em torno dos 240 caracteres por segundo (2400 Bauds).

(*) A escolha da máquina, assim como da linguagem ALGOL, são discutidas no início do capítulo V.

- o tamanho físico dos diálogos está, também, sujeito à limitações impostas pela configuração destes terminais remotos;

2- restrições que provêm do software utilizado na implementação do "G D";

- sendo o "G D" de diálogos uma ferramenta cuja utilização produz, como resultado, código fonte - ALGOL, a especificação e modelagem do "G D" acompanha os requerimentos da linguagem ALGOL.

4.3 - As Descrições de Diálogos

Chamaremos descrição de diálogo ao conjunto de instruções, mediante as quais, o projetista da interface-usuário de um determinado software de aplicação, fornece ao "G D" a descrição do relacionamento que existirá entre o usuário final e aquele programa de aplicação.

Para o "G D", os diálogos entre o usuário e o computador estarão baseados nas seguintes possibilidades:

a) O computador apresenta ao usuário 3 tipos de telas:

- Menus - lista de itens que apresentam possibilidades de escolha para o usuário do sistema;
- Telas para Coleta de Parâmetros (TCP's) - lista de textos, que descrevem valores a serem preenchidos pelo usuário e que logo poderão ser usados como parâmetros;

- Mensagens de explicação de erros ou de auxílio ao usuário;

b) O usuário interage com a máquina através de:

- nomes de menus, que o usuário deseja sejam apresentados;
- números de ítens, que representarão, à vontade do usuário, na escolha de uma determinada ação do computador;
- a entrega de valores, a serem passados como parâmetros, na execução de determinado software de aplicação.
- respostas do tipo Sim ou Não, durante a execução das procedures de auxílio ao usuário.

4.3.1 - Menus

Características Físicas:

No vídeo do terminal do usuário, um menu estará constituído por um conjunto de linhas. (72 carateres EBCDIC/linha)

A distribuição destas linhas pode ser vista no esquema da figura 4.1.

O número de linhas utilizadas para cada setor da tela do menu é opcional, sendo que o total de linhas por menu não ultrapassará 24 (limitação imposta pela configuração dois terminais). Em qualquer linha da tela de um menu poderá existir um espaço para receber a resposta do usuário, constituído por um máximo de 70 caracteres brancos, contidos entre chaves.

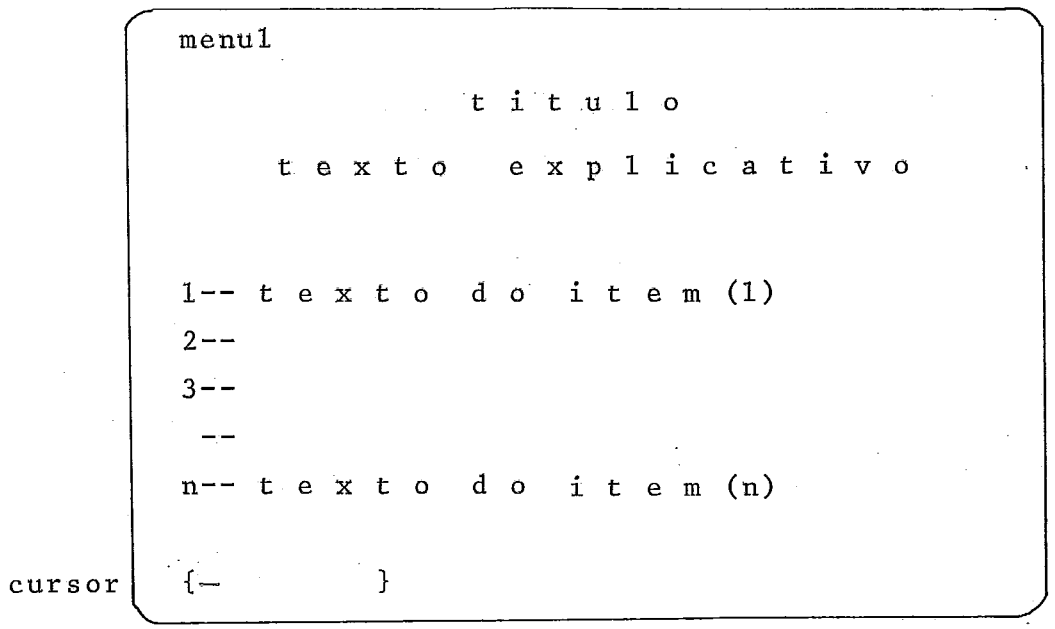


Fig. (4.1) - Esquema da tela de um "menu"

Características Sintáticas e Semânticas:

a) O "G D" requer que cada menu seja associado a um nome. Este nome será um identificador do tipo ALGOL , porém limitado a 12 (doze) caracteres. (limite imposto durante a implementação).
A necessidade do identificador para cada menu provém do fato do "G D" transformar cada descrição de diálogo em uma procedure ALGOL , que deverá ser declarada e chamada através daquele identificador.

b) As linhas reservadas para títulos ou textos explicativos do menu, são utilizadas para orientação do usuário.

c) Os Ítens do Menu:

Os textos que compõem os diferentes ítens daquele menu também são utilizados como orientação; porém, durante a apresentação do menu, o usuário poderá escolher um dos números associados aos textos de cada item. Quando isto acontecer, o projetista dos diálogos poderá esperar que o computador execute, para o usuário, as seguintes ações:

1- apresente um novo menu;

2- execute um programa de aplicação e, em seguida, inicie a execução de uma sequência (que poderia ser vazia) de outros programas de aplicação, com ou sem parâmetros;

3- apresente uma tela para coleta de parâmetros e, em seguida, inicie a execução de uma sequência (que poderia ser vazia) de outros programas de aplicação, com ou sem parâmetros;

4- O "G D" permite também, que as ações correspondentes ao primeiro item do menu, possam ser executadas, antes da apresentação ao usuário dos textos desse menu.

Esta alternativa permite por exemplo: que o projetista possa associar à execução de um ou mais comandos à apresentação de um determinado menu. Este tipo de item será chamado "item-ficticio".

d) O usuário, quando diante de uma tela do menu, poderá responder, no espeço em branco envolvido entre { }; o resto da tela do menu será ignorado pelo computador.

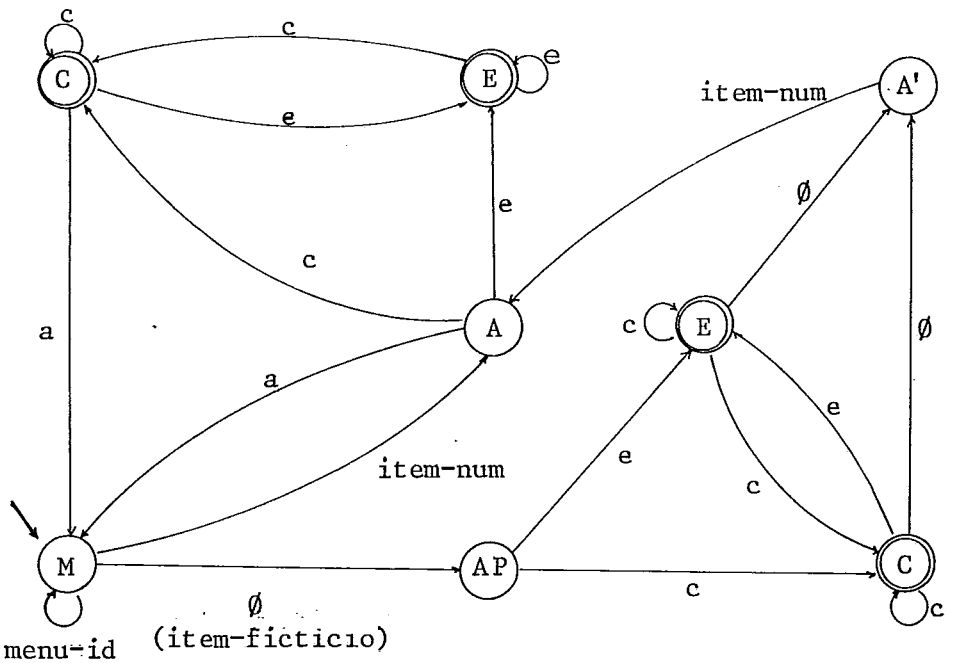
A resposta do usuário poderá ser:

1- um número igual a algum dos números que identificam os ítens daquele menu. Neste caso a ação correspondente aquele item será executada.

2- o nome de algum menu diferente daquele que está sendo apresentado. Para informação do usuário, os nomes dos menus poderão ser incluídos ou texto explicativo.

3- uma cadeia não prevista de caracteres. Neste caso, o mesmo menu será apresentado novamente ao usuário. Vale a pena destacar, que algum dos ítens integrantes do menu, pode ser utilizado para a execução de alguma procedure de auxílio ou a apresentação de algum outro menu que oriente o usuário.

Na figura (4.2) apresentamos um diagrama de estados finitos, que mostra, para o caso dos "menus", as ações do computador frente às possíveis respostas do usuário.



1) Estados

- inicial
- final
- M Menu
- E Executar (uma procedure de aplicação)
- C Coletar (uma tela para coleta de parâmetros)
- A Ações
- AP Ações Prévias

2) Alternativas disponíveis para o projeto de um menu

- a apresentar um novo menu
- e executar uma procedure de aplicação
- c coletar parâmetros
- Ø vazio

3) Respostas do usuário

item-num	número de um item
menu-id	nome de um menu (ação: apresentar um novo menu de nome id)
cadeia não prevista	ação: apresentação do mesmo menu

Fig. (4.2) - Diagrama de estados para um menu

4.3.2 - Telas para coleta de Parâmetros (TCP)

Uma TCP está constituída, basicamente, por uma lista de textos que descrevem valores a serem preenchidos pelo usuário. Estes valores logo serão utilizados como parâmetros a serem passados para a execução de procedures ou programas de aplicação.

Além desses textos explicativos, o preenchimento dos valores, por parte do usuário, poderá ser auxiliado mediante a execução de procedures de auxílio e avaliação, associados a cada TCP.

Características Físicas:

A forma da apresentação de uma tela correspondente a uma TCP é similar à de um menu porém, na tela de uma TCP, pode-se ter 1 ou mais espaços contidos entre chaves, para o posicionamento do cursor. Em tempo de execução, cada um destes espaços representa o lugar onde o usuário deve colocar os valores correspondentes a cada parâmetro - figura (4.3).

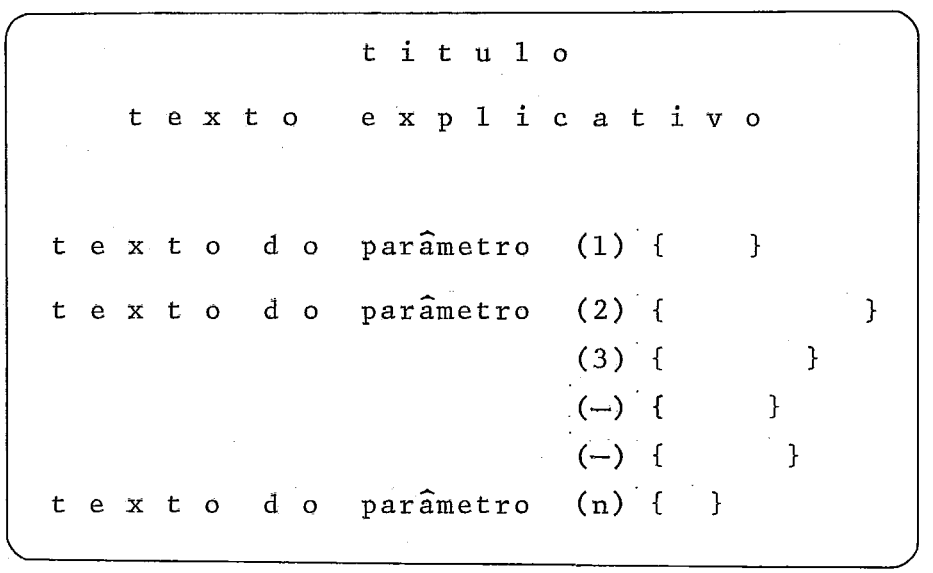


Fig. (4.3) - Esquema da tela de uma TCP

Para o caso dos terminais TD-110, que não aceitam a formatação da tela, os delimitadores para as respostas, deverão aparecer também ao final da tela de uma TCP, único lugar onde esse tipo de terminais aceita as respostas do usuário.

Características Sintáticas e Semânticas

a) A descrição correspondente a uma TCP será traduzida pelo "G D" para uma procedure ALGOL com parâmetros. Por esta razão, o "G D" requer que cada TCP seja identificada através de um nome e dos nomes de uma lista de parâmetros.Tais nomes, que também deverão ser identificadores do tipo ALGOL, serão utilizados na declaração dessa procedure e na especificação dos seus parâmetros;

b) Cada espaço entre chaves, utilizado para receber a resposta do usuário, deverá ter um número suficiente de brancos, para o preenchimento total do parâmetro;

c) Considerando o limite de 24 (vinte e quatro) linhas, imposta pela configuração dos terminais para o tamanho físico das telas de MENUS ou TCP, o número de parâmetros de uma TCP também teve que ser limitado.
O limite para o número de parâmetros de uma TCP foi colocado em 12 (doze), valor que consideramos suficiente para as aplicações mais co

muns.

Para as aplicações nas quais for necessário um maior número de parâmetros, poderão ser utilizadas uma ou mais TCP's;

d) Os tipos de variáveis, que poderão ser utilizados na transmissão de parâmetros através do uso de uma TCP, são os seguintes:

- Variáveis Inteiras: INTEGER ....id.....
  contendo valores de números,compreendidos no intervalo de $(-2^{39}-1)$ a $+(2^{39}-1)$.

- Variáveis Reais: REAL .....id.....
  contendo valores de números fracionários,que podem variar, em valor, absoluto, aproximadamente de $8.758*10^{-47}$ a $4.314*10^{68}$. Estas variáveis reais poderão também conter até 6 (seis) caracteres alfanuméricos.

- Variáveis Booleanas: BOOLEAN .....id.....
  contendo valores de verdadeiro ou falso.

- Arranjos de caracteres: EBCDIC ARRAY .....id.....
  contendo cadeias de até 70 (setenta) caracteres alfanuméricos.
  A limitação de 70 (setenta) caracteres foi colocada para sispensar o projetista do fornecimento dos limites do arranjo, realizar a leitura de caracteres em um único registro

e, ao mesmo tempo, evitar um sobredimenciona mento do uso da memória.

No caso de serem necessárias cadeias com um maior número de caracteres, poderão ser usa dos vários parâmetros.

O "G D" não considera o uso de outros tipos de arranjos, na transmissão de parâmetros a través de uma TCP. Isto deve-se aos seguin tes motivos:

- o uso de outros tipos de arranjos signifi caria o incremento da descrição de uma TCP, dos valores correspondentes aos limites su perior do array, assim como daqueles que especificam o número de dimensões;

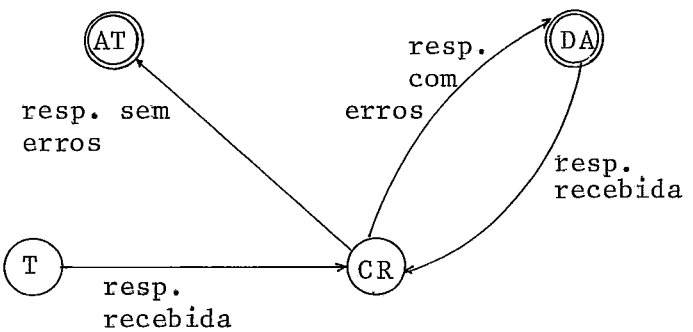
- o fornecimento de uma grande quantidade de valores, correspondentes a algum arranjo, quando feito de uma vez só, traz consigo a utilização de formatos de leitura e, com eles, um aumento significativo das possibi lidades de erro, no preenchimento desses parâmetros;

- uma maneira mais prática para a realização da leitura ou gravação de grandes quantida des de valores é obtida através do uso de arquivos em disco - para isto é suficiente conhecer o nome do arquivo e a posição re

lativa do registro de leitura ou gravação; estas informações são perfeitamente tratadas através de uma cadeia de caracteres e de uma variável inteira;

- é possível também que uma TCP possa ser utilizada na coleta de uma quantidade razoável de valores, porém esta coleta deverá ser realizada em ciclos de até 12 (doze) valores cada;

e) Como a coleta de parâmetros é suscetível a erros, o "G D" requer a existência, para cada parâmetro da TCP, de uma procedure de auxílio (HELP) ao usuário e outra de crítica e recuperação de erros.



Estados

T - Apresentação de uma TCP
CR - Crítica de erros
DA - Apresentação de uma procedure de auxílio
AT - Atribuição dos parâmetros

Fig. (4.4) Diagrama de estados finitos para uma TCP.

4.3.3 - Mensagens de Auxílio ao Usuário e Recuperação de Erros

Os valores fornecidos pelo usuário para a atribuição de parâmetros, quando verificados nos respectivos procedimentos de avaliação, podem não ser considerados válidos. Tais valores, então, serão devolvidos as respectivas procedures de auxílio ao usuário.

Tanto as procedures de crítica e recuperação de erros como as de auxílio, poderão apresentar mensagens explicativas ao usuário, referentes às possíveis falhas ou alternativas que a interface oferece.

As citadas mensagens serão preparadas durante a construção destas procedures e portanto, não terão uma estrutura predeterminada pelo "G D".

4.4 - A Interpretação das Descrições de Diálogos

A tradução de uma descrição de diálogo para uma procedure ALGOL será feita pelo "G D", basicamente em duas etapas:

a) entrada da descrição de diálogo;

b) geração de código fonte ALGOL.

4.4.1 - A Entrada das Descrições dos Diálogos

Para o "G D" a unidade básica do relacionamento usuário-computador será o "diálogo". Um diálogo poderá estar constituído por um ou mais menus e por zero ou mais TCP. Estes

menus e Telas de Coleta de Parâmetros, por sua vez, acionarão:

- procedures ou programas de aplicação;
- prodedures de auxílio ao usuário
- procedures de avaliação de parâmetros;
- novos menus

Na figura (4.5) apresentamos um grafo de nº finito de estados que expõe, de maneira compacta, o relacionamento usuário-computador, através de um diálogo. Abaixo apresentamos as alternativas oferecidas pelo "G D", na configuração do dito DIÁLOGO.

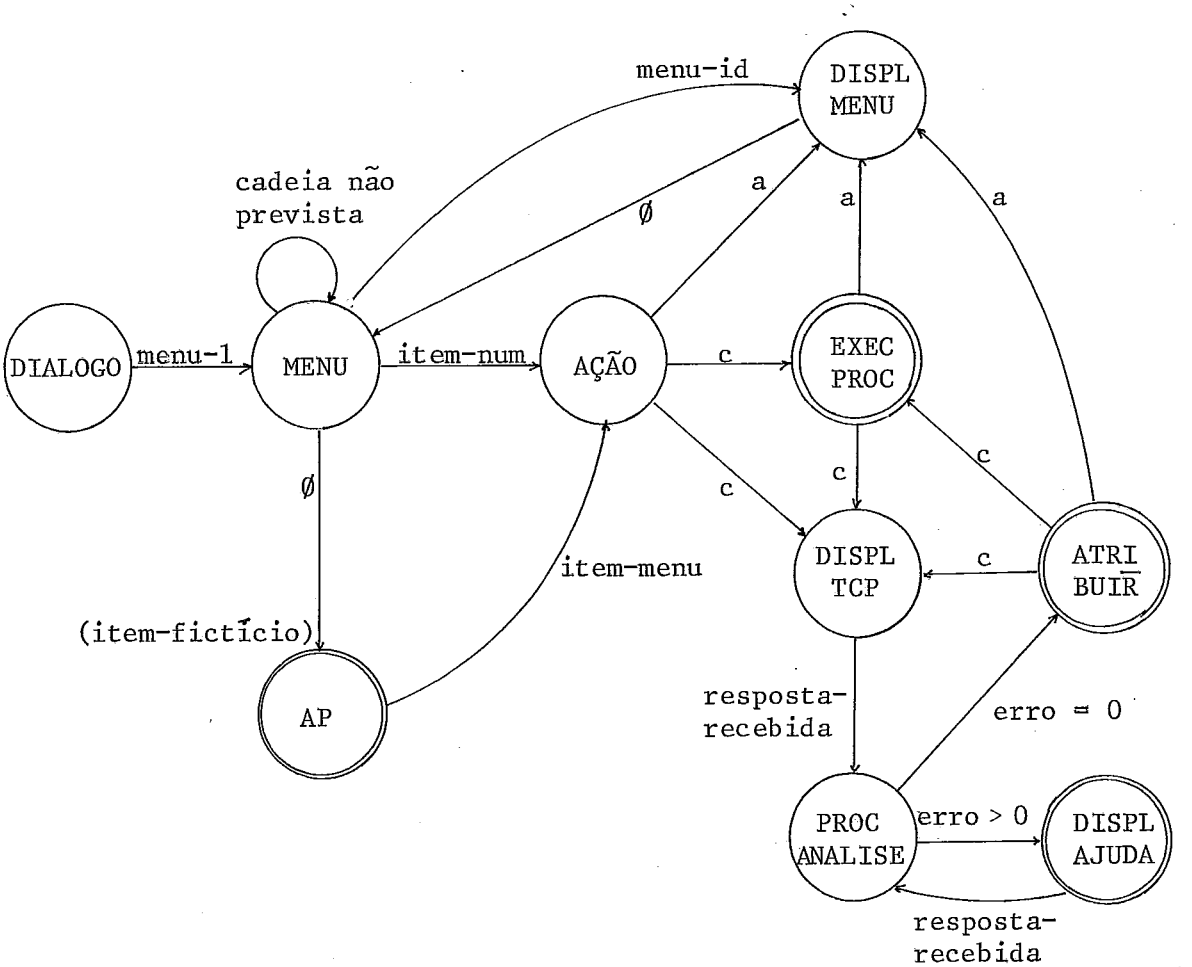


Fig. (4.5) - Diagrama de estados finitos para um "diálogo" (ver fig. 4.5').

- estado inicial

- estado final

Fig. (4.5).

disp menu – apresentação da tela de um menu

proc analise – exucução de proc. de avaliação

disp TCP – apresentação da tela de uma TCP

displ ajuda – execução de proc. de ajuda

Ø – vazio

exec proc – execução de software de aplicação

atribuir - atribuição de parâmetros

AP - Ações Prévias

Respostas do Usuário:

item-num - número de um item

menu-id - nome de um menu

resposta recebida – valores para o preenchimento de parâmetros

cadeua não prevista – cadeia de caracteres não prevista como resposta à tela de um menu.

Alternativas oferecidas pelo "G D"

a - apresentar menu

e - executar software de aplic.

c - coleção de parâmetros

menu-1 - chamada do primeiro menu

erro = 0 – preenchimento certo de parâmetros

erro > 0 preenchimento errado de 1 ou mais parâmetros

De maneira geral, a ordem que o projetista deverá manter, na entrada das descrições de um diálogo para o "G D", é a seguinte:

1- o nome do diálogo que vai ser interpretado pelo "G D".

2- uma lista de parâmetros (possivelmente nula).

O uso esperado para o código fonte produzido pelo "G D" pode ser: (ver seção 4.5.1).

a) Direto

Neste caso o "G D" gera uma procedure-ALGOL que, quando compilada, pode ser utilizada diretamente pelo usuário, através de algum comando:

RUN $ .....id.....

ou

b) Como subrotina:

Neste caso o "G D" gera uma procedure-ALGOL que, quando compilada pode ser utilizada no contexto de algum programa de aplicação:

ex.: ======= instruções do programa de aplicação
=======
=======
CALL .....id.....
=======

Neste segundo caso, a procedure produzida pelo "G D" poderia ser chamada mediante o uso de parâmetros:

ex.: CALL .....id..... (lista de parâmetros)

Por motivos de implementação, o número máximo de parâmetros permitidos para um diálogo é 12 e os tipos de parâmetros que o "G D" reconhece são: Integer, Real, array ebcdic e booleano;

3- o nome do primeiro menu que deverá ser apresentado ao usuário:

- o nome deste primeiro menu é importante para o "G D", pois é através dele que o usuário iniciará o comando das ações na execução do diálogo;

4- a seguir, o projetista continuará com a entrada das descrições de menus ou TCP's, sem preocupação pela ordem de entrada destas descrições.

4.4.2 - A Interface Projetista - "G D" (A Procedure CRIAR)

A fim de que as descrições dos diálogos possam efetivamente ser entregues ao "G D", o projetista deverá utilizar uma das procedures do "G D", especificamente a procedure "CRIAR". Esta procedure, cujo esquema funcional apresentamos na figura (4.6) interagirá com o projetista, até que todas as descrições dos diálogos tenham sido entregues, ou até que o projetista decida parar com a sua execução, coisa que poderá ser realizada somente no momento em que a descrição de algum menu ou TCP tenha sido totalmente entregue ao "G D".

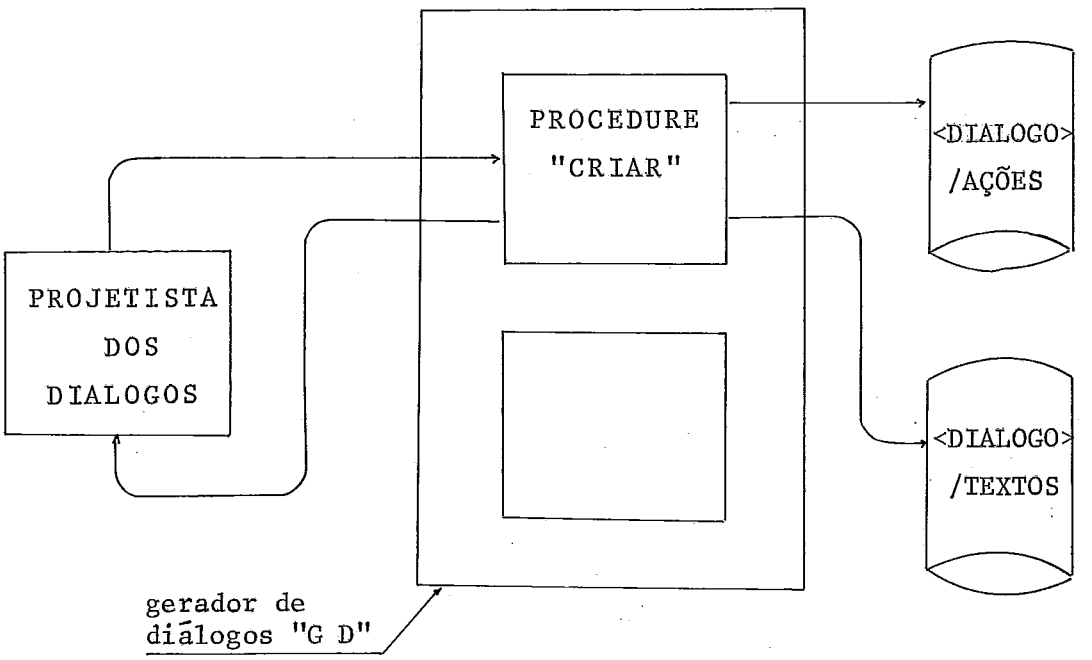


Fig. (4.6) - Esquema funcional da procedure "CRIAR".

A comunicação entre o projetista e o computador, realizada através da interface CRIAR é totalmente interativa, refletindo, assim, as características que o diálogo, a ser interpretado, apresentará ao usuário.

Um esquema que expõe o relacionamento entre o projetista e a procedure CRIAR é apresentado na figura (4.7).

Como resultado de sua execução, a procedure CRIAR armazenará todas as descrições recebidas em dois arquivos, no espaço em disco do uso do projetista.

Esses dois arquivos, ao quais chamaremos "TEXTOS" e "AÇÕES", passarão a fazer parte de um mesmo diretório, que constituirá o produto final da execução do "G D". Tal diretório será identificado com o nome do diálogo que está sendo gerado. Para facilitar, chamaremos a este diretório de <DIÁLOGO>.

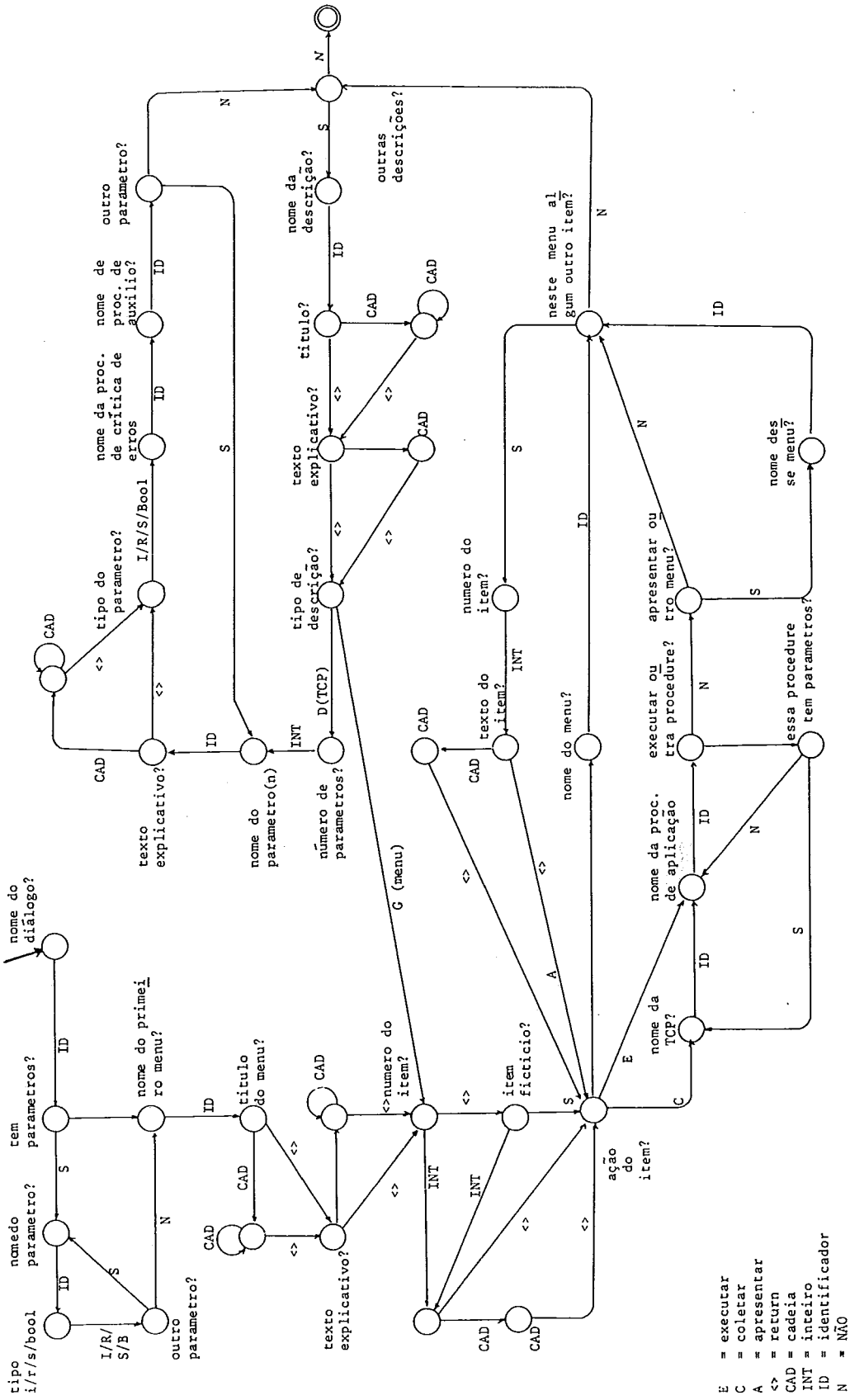


Fig. (4.7) - Procedure "CRIAR" (interface com o projetista).

No arquivo <DIALOGO>/AÇÕES, será conservada a parte semântica das descrições dos diálogos - por exemplo: o nome do diálogo que está sendo gerado, dos seus parâmetros, ..., nomes dos menus, ações dos seus ítens etc. Figs. (4.8) e (4.9)

No arquivo <DIAGOLO>/TEXTOS, ficarão conservados todos os textos dos menus e TCP, que serão os que aparecerão no vídeo do terminal do usuário, quando o diálogo que estiver sendo entregue for executado. Figs. (4.10) e (4.11).

Explicações com maiores detalhes sobre o uso da procedure CRIAR poderão ser encontradas mais adiante, no subcapítulo 4.5, correspondente ao uso do "G D".

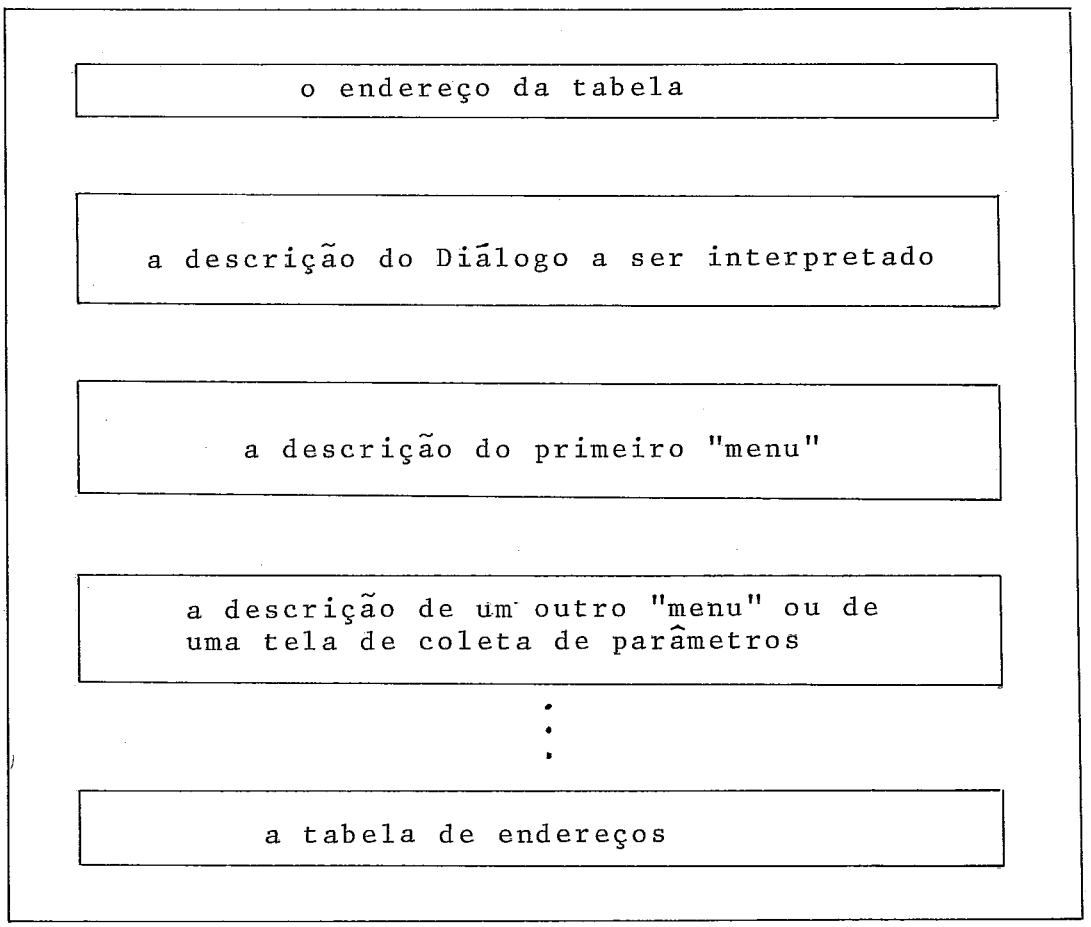


Fig. (4.8) - Esquema do arquivo <DIALOGO> /AÇÕES

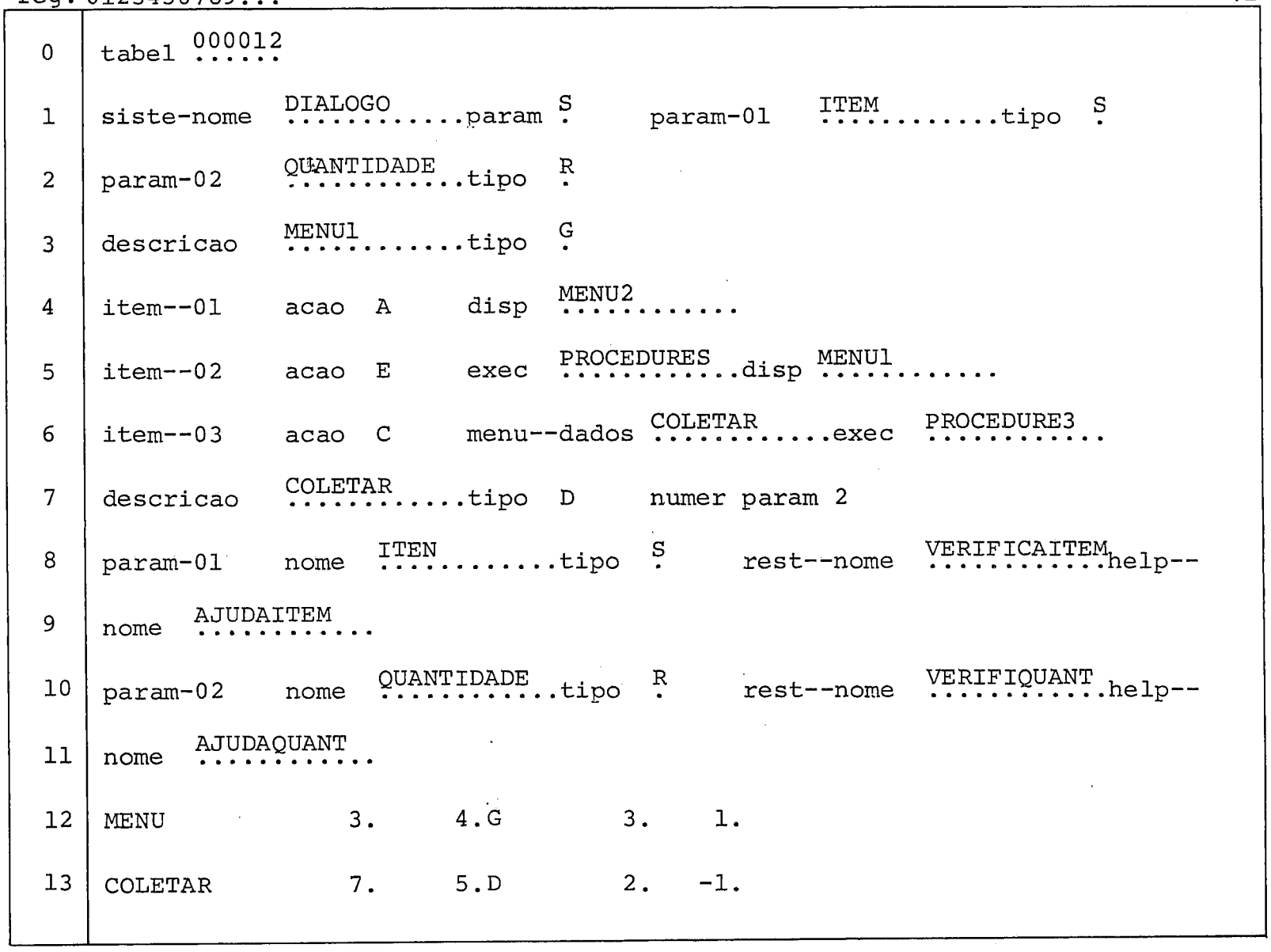


Fig. (4.9) - Exemplo do arquivo <DIALOGO> /AÇÕES (ver fig. (4.9')).

reg 0 - : endereço da tebela = 12

reg 1 - reg 2 : descrição do diálogo; nome = DIALOGO; parâmetros = Sim; parâmetro 1 de nome = ITEM e de tipo String

reg 3 - reg 6 : primeira descrição, de nome = MENU1 e de tipo G = menu
ação do item 1 será Apresentar o MENU2
ação do item 2 será Executar a PROCEDURE2 e após essa execução apresentar o MENU1
ação do item 3 será Coletar os parâmetros descritos na TCP de nome = COLETAR, após essa coleta será executada a PROCEDURE3

reg 7 - reg 11: descrição de nome = COLETAR e de tipo D = TCP contendo 2 parâmetros
parâmetro 1 de nome = ITEM de tipo String, cujo valor será analisado através da procedure VERIFICAITEM e da procedure AJUDAITEM que auxiliaria ao usuário, caso ele for preenchido erradamente.

reg 12 - reg 13: tabela de endereços; primeira descrição de nome MENU1 descrita a partir do registro 3, que ocupa 4 registros, que é de tipo G ou seja menu, que contém 3 ítens e que tem sido o menu número 1 em aparecer no arquivo.

Fig. (4.9').

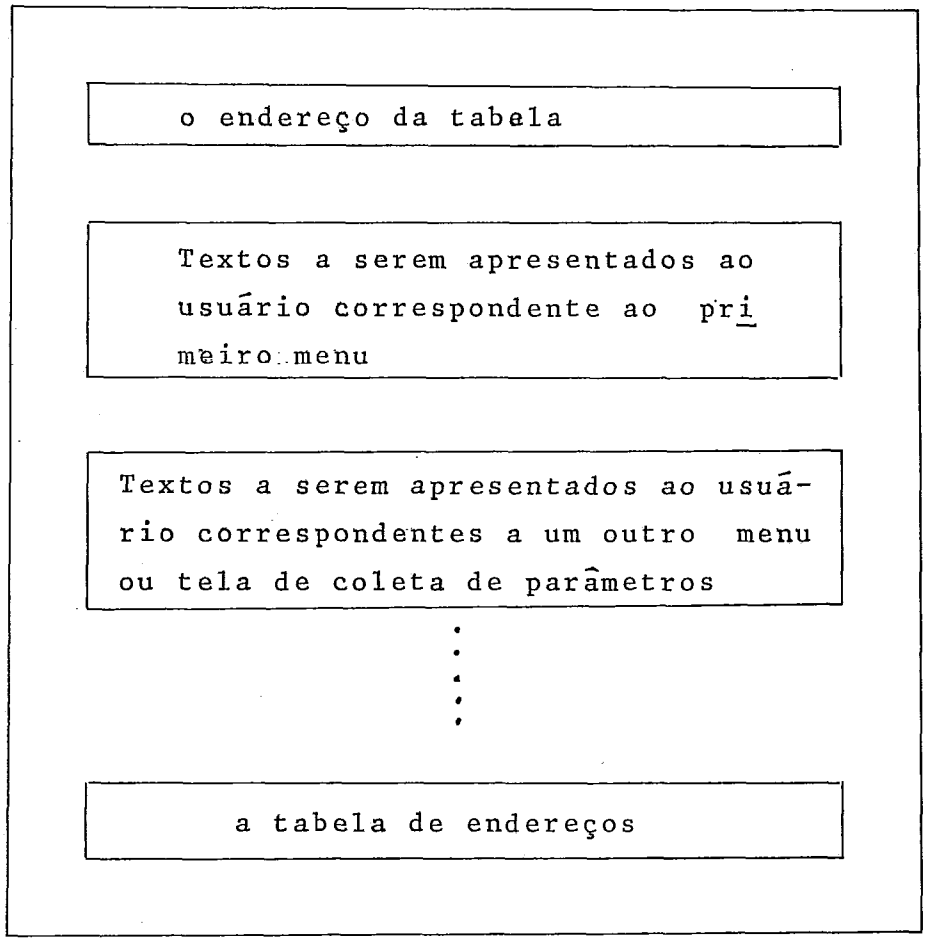


Fig. (4.10) - Esquema de arquivo <DIALOGO> /TEXTOS

```
0  tabel 000022
1                                                          menu1
2
3                TELA  PARA  A  INCLUSÃO  DE  ITENS
4                ************************************
5  1-- INSTRUÇÕES
6  2-- RELATÓRIO DE ITENS INCLUIDOS
7  3-- INCLUSÃO DE UM ITEM
8   --
9
10
11               I N C L U S Ã O  D E  U M  I T E M
12               ***************  ***  ***  *******
13
14            NOME DO ITEM A INCLUIR ?
15            (nomes de itens já incluidos serão
16             considerados inválidos) .................
17
18            QUANTIDADE DESTE ITEM ?
19            (só serão aceitas quantidades
20             maiores que zero)  ......................
21
22 MENU1            1.    9.G           1.
23 COLETAR         10.   12.D          -1.
```

reg 0 - endereço da tabela = 22

reg 1 - reg 9: texto da primeira descrição

reg 10 - reg 21: texto da segunda descrição

reg 22 - reg 23: tabela de endereços;
primeira descrição de nome MENU1; ocupa 9 registros; de tipo G = menu, sendo o menu numero 1 do arquivo

Fig. (4.11) - Exemplo do arquivo <DIALOGO> /TEXTOS

4.4.3 - A Geração do Código Fonte (PROCEDURE GERADOR)

A Segunda etapa no processo de interpretação de um diálogo, através do "G D", é a geração de código fonte.

Para que esse cógigo fonte, produto final da utilização do "G D", possa ser utilizado por um computador,ele deve pertencer a alguma das linguagens de programação aceitas por aquele determinado computador.

Por motivos de implementação, discutidos no Capítulo 5, a linguagem escolhida para a geração do código fonte é o ALGOL-Extended do B-6700.

A geração desse código fonte, feita pelo "G D", está confiada a outra das procedures do "G D", especificamente a chamada GERADOR.

A seguir, apresentaremos algumas das características mais relevantes, impostas pela linguagem ALGOL ao código fonte, gerado através do uso do "G D".

1- a procedure GERADOR deverá traduzir cada descrição de menu ou TCP para código fonte ALGOL. Para facilidade de implementação, esta tradução foi feita associando uma procedure ALGOL a cada menu ou TCP;

2- de acordo com a nossa especificação de diálogo uma das formas do usuário responder a uma tela de menu é através do nome de outro menu.
Tal possibilidade, que representa uma ótima alternativa para o usuário que já conheça uma determinada sequência de apresentações de menus deve, no entanto, ser produzida pela execu

ção de uma outra procedure ALGOL, que deverá ser gerada junto ao restante de procedures, durante a interpretação de cada diálogo.

3- Deve ser observado, pelo acima exposto, que todas as procedures ALGOL, geradas a partir da interpretação das "descrições de menus",mantêm a mesma hierarquia no contexto do diálogo que está sendo gerado.
Isto trás consigo a necessidade da criação de declarações do tipo "forward" para cada uma dessa procedures.
Essas declarações "forward garantirão a possibilidade do usuário chamar um menu sem preocupação da posição relativa no diálogo do menu originário da chamada.

4- por razões do uso do código fonte, produzido mediante a utilização do "G D", cada diálogo será uma procedure ALGOL, cuja execução incluirá chamadas para cada uma das procedures que foram traduzidas a partir das descrições do dito diálogo.

4.4.4 - A Interface da Procedure Gerador Fig. (4.12)

Tal como foi apresentada, esta procedure é a que se encarrega da tradução da descrição de um diálogo para código fonte ALGOL.

Para sua execução, esta procedure deverá contar com as seguintes entradas:

1- os arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS, contendo a descrição do diálogo que vai ser traduzido. Estes dois arquivos devem ter sido produzidos pela execução da procedure CRIAR.

2- um arquivo que deverá conter todas as procedures de auxílio e avaliação, que possam ter sido referidas durante a entrega da descrição do diálogo.

Devemos ressaltar que a atual versão da procedure GERADOR só aceita procedures escritas na linguagem ALGOL. Entretanto, as procedures geradas pelo uso do "G D" podem ser utilizadas por programas de aplicação escritos em FORTRAN, COBOL, ou ALGOL. Esta limitação obedece às restrições impostas pelo programa "BINDER" do B-6700, que permite a ligação de código objeto, obtido a partir do uso dos compiladores das linguagens mencionadas em um único programa fonte.

Ao arquivo que contém as procedures de auxílio e avaliação, chamaremos de <DIÁLOGO> /AJUDA.

3- um arquivo que deverá conter todas as procedures de aplicação referidas durante a descrição do diálogo.

Essas procedures de aplicação devem estar colocadas em um arquivo separado, para desta maneira, permitir que elas possam ser desenvolvidas paralelamente à interface usuário-computador. Ao arquivo que contém essas procedures de aplicação, chamaremos de <DIÁLOGO> /APLICAÇÃO;

4- no início de sua execução, a procedure GERADOR precisará, também, do nome do diálogo que vai ser interpretado - este nome permitirá que a procedure GERADOR identifique o diretório de arquivos, contendo a descrição do diálogo.

Após sua execução, a procedure GERADOR apresentará os seguintes resultados:

1- um arquivo contendo o código fonte gerado para a procedure que será utilizada na execução do diálogo. Este arquivo, assim como a procedure gerada será identificado com o nome do diálogo que foi interpretado; Fig. (4.14).

2- um arquivo contendo o código fonte gerado para as procedures que foram traduzidas a partir de cada descrição de menu ou TCP. Este arquivo será identificado como <DIÁLOGO> /PROC; Fig. (4.16) e (4.18).

3- um arquivo contendo o código fonte gerado para a procedure que permitirá ao usuário, no momento da execução do diálogo, responder à apresen

tação da tela de um menu com o nome de outro menu. Este arquivo é identificado como <DIÁLOGO> /EXEC e a procedure nele contido é identificado como PROCEDURE EXEC; Fig. (4.19).

4- um arquivo contendo o código fonte correspondente a todas as declarações ALGOL de tipo FORWARD, que servirão para manter o mesmo nível de hierarquia nas procedures. Este arquivo é identificado como <DIÁLOGO> /FOWAR Fig.(4.20).

Um esquema que apresenta a interface de entrada e saída da procedure GERADOR consta na figura (4.12).

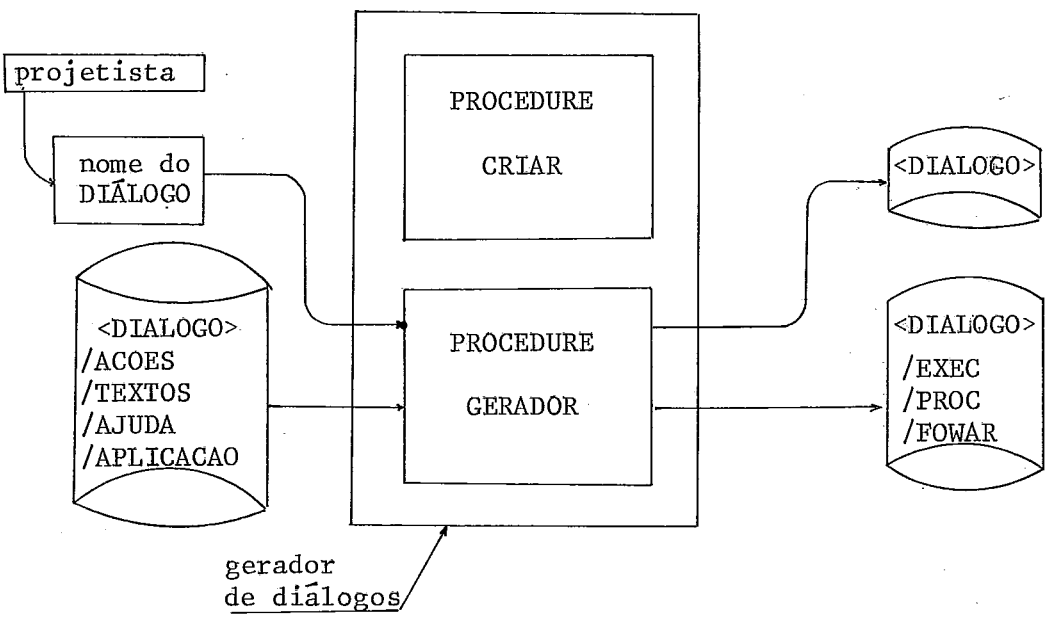


Fig. (4.12) - A interface da procedure GERADOR.

4.4.5 - Os Programas Fontes Gerados pela Procedure GERADOR

Em seguida apresentaremos o detalhe dos programas fontes, gerados pela procedure GERADOR.

Neste detalhe estarão incluídos o objetivo e a interface de entrada e saída de cada programa gerado, assim como um modelo que apresente as características mais relevantes de cada um destes programas.

1- A procedure <DIALOGO>, que será utilizada para execução do diálogo.

A função desta procedure é ligar todas as procedures geradas através do "G D" em uma única procedure ALGOL, para que, desta maneira, a descrição do diálogo possa ser transformada em um só programa. Tal programa poderá, então, ser utilizado diretamente pelo usuário ou incluído nas instruções de algum software de aplicação.

- Valores recebidos como parâmetros de entrada: não tem.
- Valores fornecidos como parâmetros de saída:
  - aqueles valores que, fornecidos pelo usuário, podem ter sido atribuídos à lista de parâmetros especificada durante a entrega da descrição do diálogo.

A execução desta procedure começa com uma chamada da procedure CARREGA - fig. (4.13).

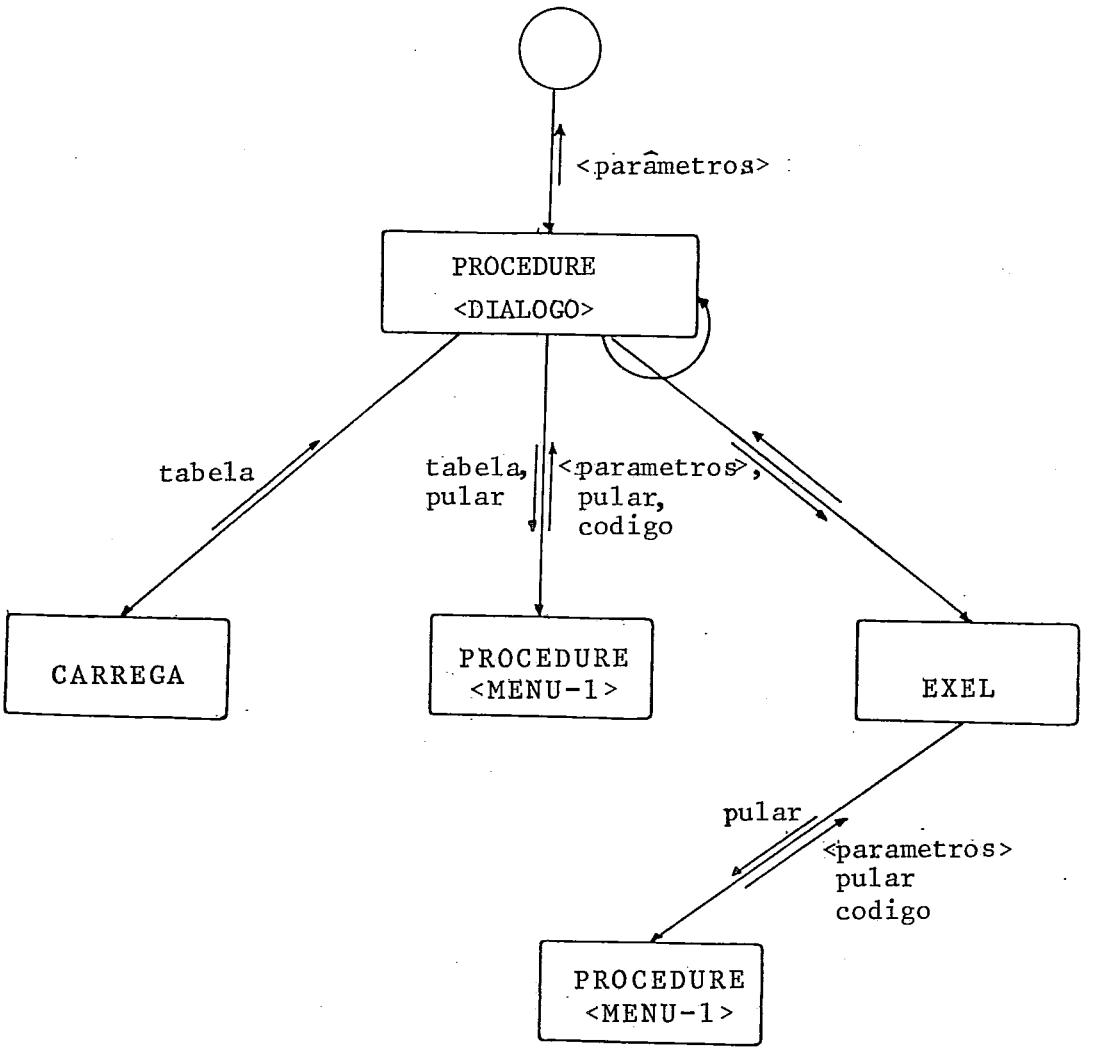


Fig. (4.13) - Estrutura da procedure <DIALOGO> que será utilizada para a execução do dialogo.

a) a procedure CARREGA tem, como função, colocar em um array tabela os nomes de todos os menus que possam ser encontrados no arquivo <DIÁLOGO> /TEXTOS. No array tabela,os nomes destes menus ficarão associados a um CODIGO numérico. (ver Fig. 4.26).

A procedure CARREGA deveria ser outra das procedures geradas pelo "G D", porém como sua execução só depende do arquivo <DIALOGO> /TEXTOS que esteja sendo lido, ela pode ser perfeitamente incluída como uma procedure de avaliação no arquivo <DIÁLOGO> /AJUDA.

Após sua execução, a procedure CARREGA devolve o array tabela atualizado, com os nomes e códigos de cada menu.

b) em seguida, será executada a chamada procedure ALGOL, que foi traduzida a partir da descrição do primeiro menu do diálogo

Esta procedure receberá, como entrada, a variável booleana pulár, colocada com valor falso e o array tabela.

Dependendo da resposta do usuário, a execução desta procedure apresenta duas alternativas:

1- a resposta do usuário implica na escolha de um dos ítens do menu:

- neste caso serão executadas as ações previstas naquele item; os resultados

da execução destas ações poderão incluir os valores que tenham sido recebidos como parâmetros , durante a chamada de alguma procedure traduzida a partir da descrição de uma TCP;

2- a resposta do usuário é o nome de um novo menu:

- neste caso a procedure encerra sua execução e devolve a variável booleana pular colocada como verdadeira, assim como um valor inteiro código, que identifica no array tabela o menu que o usuário deseja que lhe seja apresentado;

c) se o valor booleano recebido na variável pular for verdadeiro, a execução desta procedure continuará com uma chamada da procedure EXEC.

A procedure EXEC recebe como parâmetro de entrada um código que identifica o nome do menu que deverá ser apresentado. A procedure ALGOL, traduzida a partir da descrição deste novo menu, recebe como entrada a variável pular, colocada, novamente, com valor = falso e sua execução produz resultados similares aos que foram escritos para o caso do primeiro menu.

A procedure EXEC encerra sua execução no mo

mento em que o valor recebido de volta na variável pular seja = falso.

Na figura (4.14) apresentamos um modelo do código fonte, que deve ser gerado para a procedure que realizará a execução do diálogo.

```
PROCEDURE.....id.....(.....lista de parâmetros.....);

  declaração dos parâmetros;
BEGIN
  declaração das variáveis;
  declaração dos arquivos de entrada e saída;
  $ INCLUDE " DIALOGO /APLICACAO."      % inclusão em tempo de compila
  $ INCLUDE " DIALOGO /AJUDA."            ção, de todo o código fonte
  $ INCLUDE " DIALOGO /FOWAR."            gerado na interpretação do
  $ INCLUDE " DIALOGO /EXEC."             "diálogo"
  $ INCLUDE " DIALOGO /PROC."

  BEGIN
    substituição dos nomes dos arquivos de entrada;
    CARREGA;
    chamada da procedure que foi gerada pela descrição do 1º
    menu;
    WHILE  "o usuário deseja chamar algum menu"   DO
      EXEC(codigo);    % chamar aquele menu
  END;                                  % --- comentarios
END;
```

Fig. (4.14) - Esquema do código fonte que deve ser gerado para a procedure que executará a descrição do "dialogo".

2- A procedure <MENU>, traduzida a partir da descrição de um menu.

Parâmetros de entrada:

- array tabela, contendo os nomes e os códigos dos menus encontrados no arquivo <DIALOGO> /TEXTOS;

- variável pular, colocada com valor = falso

a) a execução desta procedure começa com a execução de uma leitura ao arquivo <DIALOGO> /TEXTOS. A operação de leitura recebe, como parâmetros, a posição pos e o número de registros tam do texto correspondente à descrição deste menu. Os valores de pos e tam serão atribuídos pela procedure GERADOR, no momento da geração do código fonte. Como resultado desta leitura, um buffer de caracteres EBCDIC é carregado com o conteúdo do texto;

b) a seguir, o conteúdo deste buffer é escrito na tela do usuário, através de uma operação de escrita;

c) depois da tela do usuário receber o texto do menu, a execução desta procedure espera pela resposta do usuário. Tal resposta será recebida em formato livre em um outro array de caracteres cadeia;

d) após ter sido recebida a resposta do usuário, a execução desta procedure continua com uma chamada da procedure ANÁLISE. Esta última recebe como parâmetros de entrada:

- a cadeia de caracteres recebida como resposta; e
- o array tabela, contendo os nomes e códigos dos menus integrantes do diálogo;

A função da procedure ANALISE é analisar a cadeia recebida e devolver o resultado desta análise nos seguintes parâmetros:

- array real resposta, contendo a cadeia analisada. Esta cadeia pode ter sido transformada em um valor inteiro ou real, dependendo da configuração dos caracteres recebidos.
- variável real tipo, contendo um código para diferenciar a resposta devolvida. Este código especificará se a resposta devolvida foi considerada um valor inteiro,real, array EBCDIC ou booleano;

Na figura (4.15) apresentamos um esquema da estrutura desta procedure.

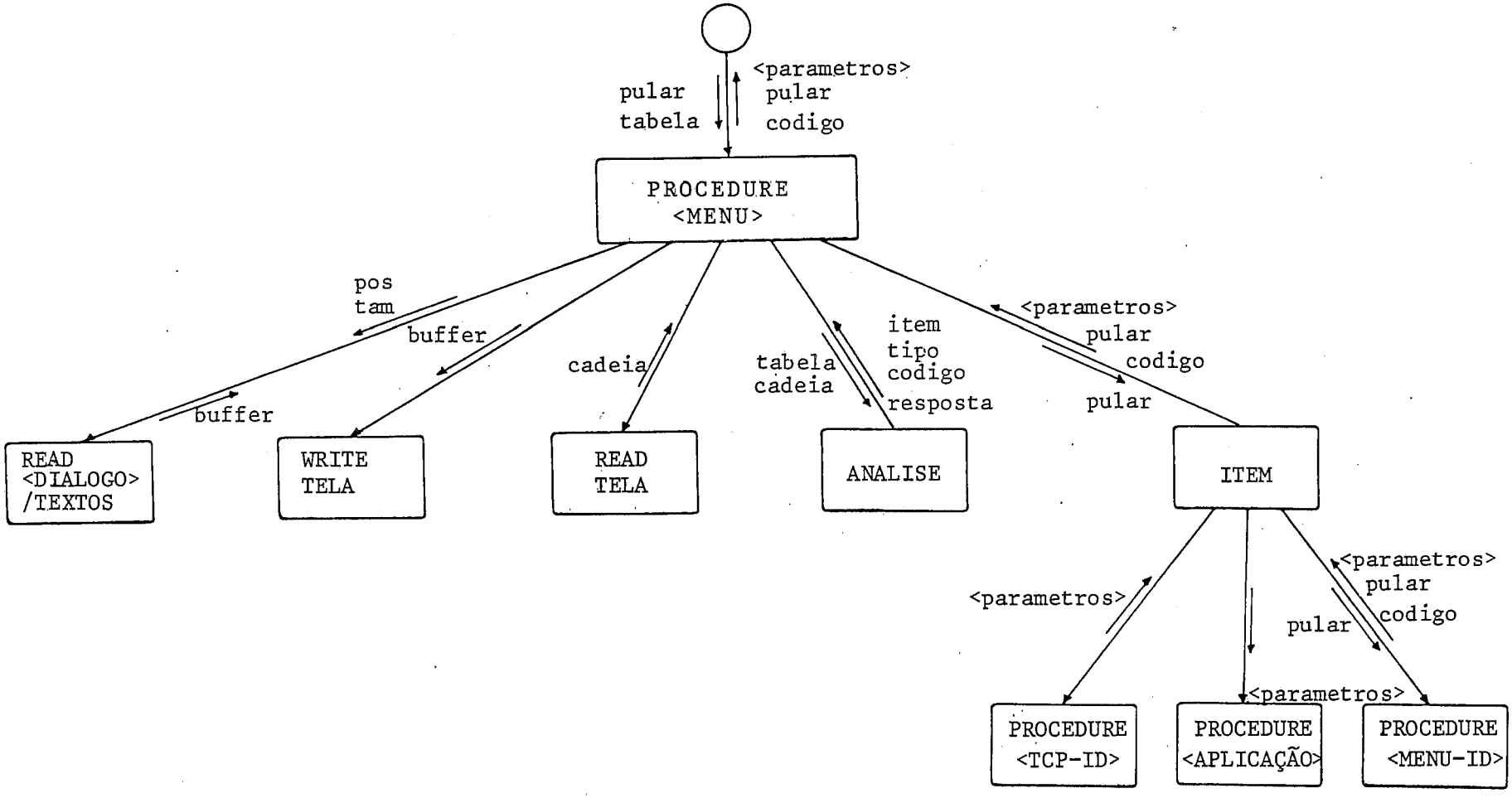


Fig. (4.15) - Estrutura de procedure <MENU> traduzida a partir da descrição de um "menu".

- item, variável inteira, contendo o número do item do menu, no caso da resposta analisada coincidir com algum;

- codigo, variável inteira, contendo o código correspondente ao nome do menu encontrado, igual ao possível identificador fornecido pelo usuário na cadeia de caracteres.

Da mesma maneira que para o caso da procedure EXEC, esta procedure ANÁLISE deveria ser outra das procedures geradas pelo "G D", porém, devido a que sua função depende exclusivamente da resposta do usuário e sua codificação é sempre a mesma, ela pode ser perfeitamente incluída como uma procedure de avaliação no arquivo <DIÁLOGO> /AJUDA.

e) Dependendo dos resultados recebidos pela execução da procedure ANÁLISE, a execução da procedure <MENU> tem três opções:

- caso a resposta recebida seja uma cadeia não prevista, a execução continuará com uma chamada recursiva da mesma procedure e, desta maneira, o usuário teria uma nova oportunidade de sucesso;

- caso a resposta recebida for o nome de um menu, fato que pode ser detetado pelo valor recebido em código, a execução da procedure será encerrada, devolvendo-se, em

código, o valor que identificará aquele menu.

- caso a resposta recebida seja o número de algum dos ítens do menu, fato que será detetado pelo valor recebido no item, a execução da procedure continuará, com a execução das chamadas das possíveis procedures de aplicação, de TCP's ou de outros menus, que possam ter sido especificados como ações daquele item, durante a entrega da descrição do diálogo. Neste caso, a execução da procedure terminará devolvendo como resultados:

  - os parâmetros recolhidos a partir da execução de procedures que representam TCP's;

    e

  - os valores designados para as variáveis pular e código, durante a execução de alguma procedure que represente um menu.

Na figura (4.16) é apresentado um modelo do código fonte que deve ser gerado na tradução da descrição de um menu em uma procedure ALGOL.

```
PROCEDURE  .....id.....;
BEGIN
  declarações das variaveis;          % nestas declarações, não aparecerão
                                      % incluídas, as declarações daquelas
  BEGIN                               % variáveis que já foram declaradas
                                      % na procedure "diálogo".

  inicialização das variáveis booleanas de control;
  leitura do texto da tela deste menu no arquivo <DIALOGO>/TEXTOS;
  gravação da tela deste menu no arquivo  TELA  ;  % o video do usuário
  leitura da resposta do usuário;
  ANALISE (RESPOSTA,CADEIA,ITEM,CODIGO); %análise da resposta do usuário

  IF   "  a resposta do usuário for um numero inteiro   THEN

  CASE "  o numero inteiro for


          01:  ações a serem executadas correspondentes ao previsto pelo
               projetista, caso o item 01 seja "o escolhido pelo usuário

          02:  ações a serem executadas...  correspondentes ao item  02

            :  .........      item  03

            :

          ELSE .....id.....;     % chamada recursiva desta procedure,
                                 % que apresenta uma nova oportunidade
        END;                     % ao usuário, pois o número inteiro dado
                                 % resposta, não foi igual a algum dos
                                 % numeros dos itens propostos no menu

  Se a resposta do usuário não foi
  um inteiro, então ela é uma cadeia de caracteres.


  IF  "  esta cadeia de caracteres, for o nome de algum menu integrante
         do diálogo"

      THEN  EXEC (codigo);           % chamada daquele menu

      ELSE  .....id.....;            % chamada recursiva desta procedure,

                                     % .......

  END;

END;
```

Fig. (4.16) - O esquema da procedure, que interpreta a descrição de um "menu".

3- A procedure <TCP>, tracuzida a partir da descri ção de uma TCP

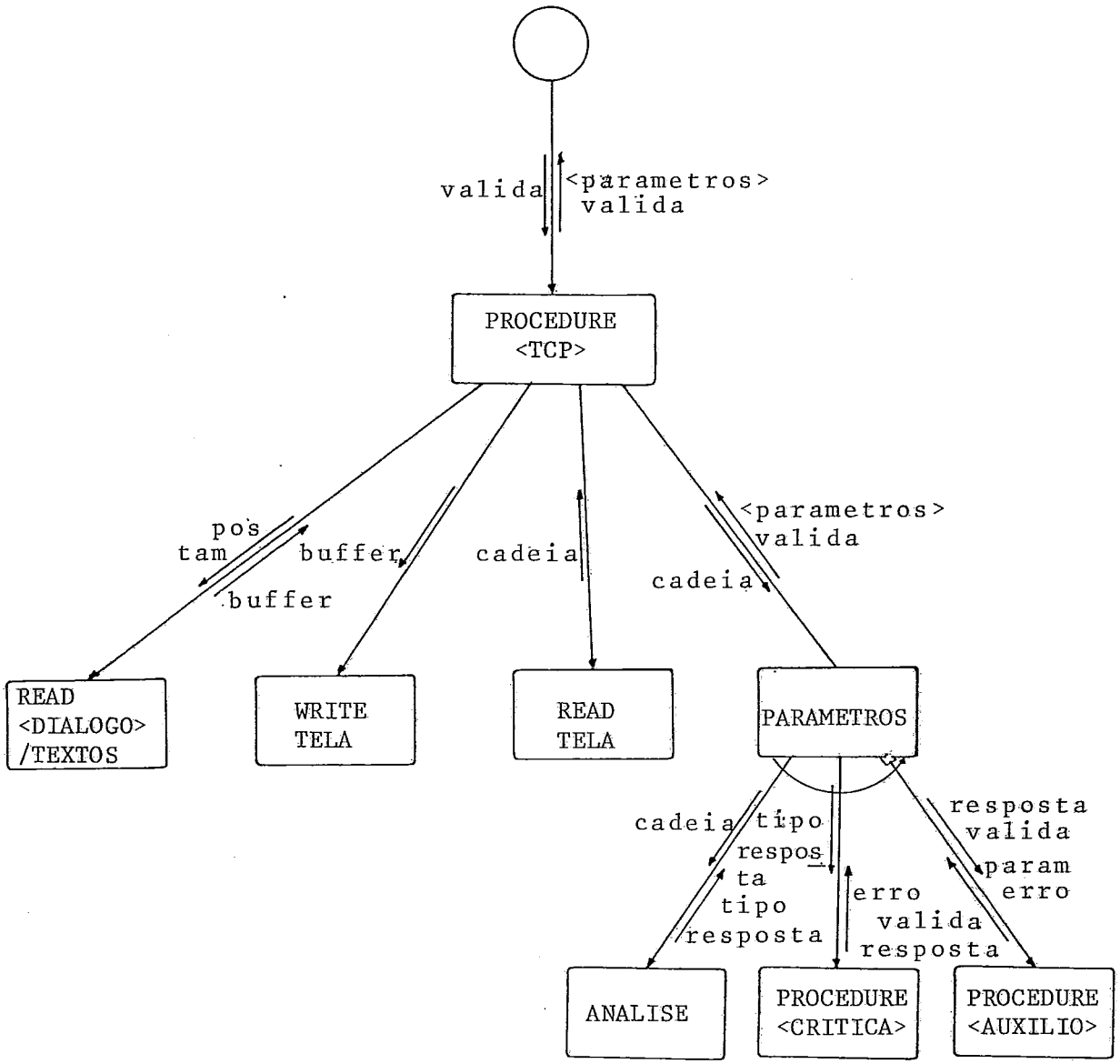


Fig. (4.17) - Estrutura da procedure ALGOL traduzida a partir da descrição de uma TCP.

Parâmetros de entrada:

Variável booleana valida, colocada como verdadeira.

Parâmetros de saída:

- a lista de parâmetros entregues durante a descrição da TCP;

- uma variável booleana valida, que se tiver valor "verdadeiro" indicará que a atribuição dos parâmetros foi realizada sem erros.

A execução desta procedure começa da mesma forma que no caso daquela que representa um menu:

a) leitura do texto da TCP do arquivo <DIALOGO>/TEXTOS:

   Parâmetros de entrada: pos e tam

   Parâmetros de saída : buffer contendo o texto da TCP.

b) gravação do conteúdo do buffer na TELA do usuário;

c) leitura das respostas fornecidas pelo usuário - estas respostas serão recebidas com formato livre, em um array de caracteres cadeia;

d) em seguida, a execução desta procedure con

tinua, com a verificação das respostas fornecidas pelo usuário - tal verificação, que deverá ser realizada para cada parâmetro, é feita da seguinte maneira:

- Uma chamada da procedure ANÁLISE, que recebe como entrada a cadeia de caracteres fornecida pelo usuário - a execução desta procedure devolve, como resultados, o valor e o tipo da resposta que deverá ser utilizada no preenchimento do parâmetro; estes resultados são devolvidos no array real resposta e na variável real tipo.

- Com os valores recebidos no resposta e tipo, é executada a chamada da procedure CRITICA; que deverá conferir a validade da resposta do usuário para esse parâmetro - o resultado desta avaliação será recebido em uma variável inteira erro.

- No caso do valor recebido em erro ser igual a zero, isto significará que o valor contido em resposta é válido - caso contrário (ERRO $\neq$ 0), será executada a chamada da respectiva procedure de AUXÍLIO, que receberá, como parâmetros de entrada:

  o valor contido em resposta;

  a codificação do erro em erro;

a identificação da resposta que estiver sendo analisada em param (esta identificação corresponde à posição relativa dessa resposta, na cadeia de caracteres fornecida pelo usuário).

a variável booleana valida com valor falso

A execução da procedure de AUXÍLIO deverá apresentar ao usuário, o erro cometido e receber uma nova resposta para o preenchimento - caso a execução da procedure AUXÍLIO seja bem sucedida, ela devolverá como resultados: o array resposta atualizado e a variável booleana valida, colocada como verdadeira; no caso contrário, o resultado devolvido, será a variável valida, com valor falso;

- após a verificação das respostas, a execução das procedure: <TCP>, apresenta duas opções:

  - o valor contido em valida é verdadeiro: neste caso, a atribuição das respostas recebidas na lista de parâmetros da TCP será realizada - e o resultado final da execução desta procedure, será a entrega destes parâmetros com seus valores atualizados, e valida com valor - "true";

```
PROCEDURE ..... id ..... (..... lista de parâmetros ..... , ..... , .....);
  declarações dos parâmetros
BEGIN
  declarações das variaveis

  BEGIN
    leitura do texto de tela desta T.C.P. no arquivo "TEXTOS";
    gravação da tela da T.C.P. no arquivo "TELA"; % o vídeo do usuário
    leitura das respostas do usuário;

       - chamada da procedure que verifique a resposta do usuário  corres
         pondendo ao primeiro parâmetro;

       - chamada da procedure .....     do segundo parâmetro;

       -       .....                    terceiro parâmetro;

       -
    IF "alguma (s) das respostas do usuario não foram válidas"
       THEN chamada da procedure que ajude ao usuario no preenchimento  da
            quele parâmetro;

    IF " após a execução destas procedures, todas as respostas  apresenta
         das pelo usuário são válidas"

    THEN atribuir cada uma destas respostas, aos correspondentes  parâme
         tros com os quais foi entregue a descrição desta TCP;

  END;

END;
```

Fig. (4.18) - O esquema da procedure que interpreta a descrição de uma TCP.

- o valor contido em valida é falso:
  neste caso, a atribuição dos parâmetros não será realizada e o resultado da execução desta procedure será a devolução da variável valida com valor falso.

Na figura (4.18) apresentamos um esquema que apresenta um modelo do código fonte, que deve ser gerado a partir da descrição de uma TCP.

Nas figuras (4.19) e (4.20), apresentamos os esquemas do código fonte gerado para a procedure EXEC salvo no arquivo <DIALOGO> /EXEC e para as declarações FORWARD salvo no arquivo <DIALOGO> /FOWAR.

```
PROCEDURE EXEC (CODIGO);
INTEGER CODIGO;                 % código é um valor numérico associado
CASE CODIGO OF                  % a cada menu do dialogo
BEGIN                           % esse código é salvo no último campo
   01 : MENU1                   % das tabelas de endereços (secção 4.5.3)
   02 : MENU2
   03 :
END;
```

Fig. (4.19) - Modelo do código fonte gerado para a procedure EXEC.

```
PROCEDURE MENU1;  FORWARD;
PROCEDURE TCP1 (ITEM, VALOR);
INTEGER ITEM; REAL VALOR; FORWARD;
PROCEDURE MENU2; FORWARD;
```

Fig. (4.20) - Modelo do código fonte gerado para as declarações FORWARD.

4.5 - O Uso do Gerador de Diálogos "G D"

A seguir, serão apresentadas algumas considerações que estão relacionadas com o uso do "G D". Nestas considerações serão tratados os seguintes aspectos:

- o uso previsto para os diálogos produzidos pelo "G D".
- a utilização da procedure CRIAR
- as atualizações que podem ser feitas em diálogos já existentes
- a utilização da procedure GERADOR
- o uso do código fonte gerado

Na figura (4.21) está demonstrado o esquema da utilização do "G D" na criação da interface usuário-computador.

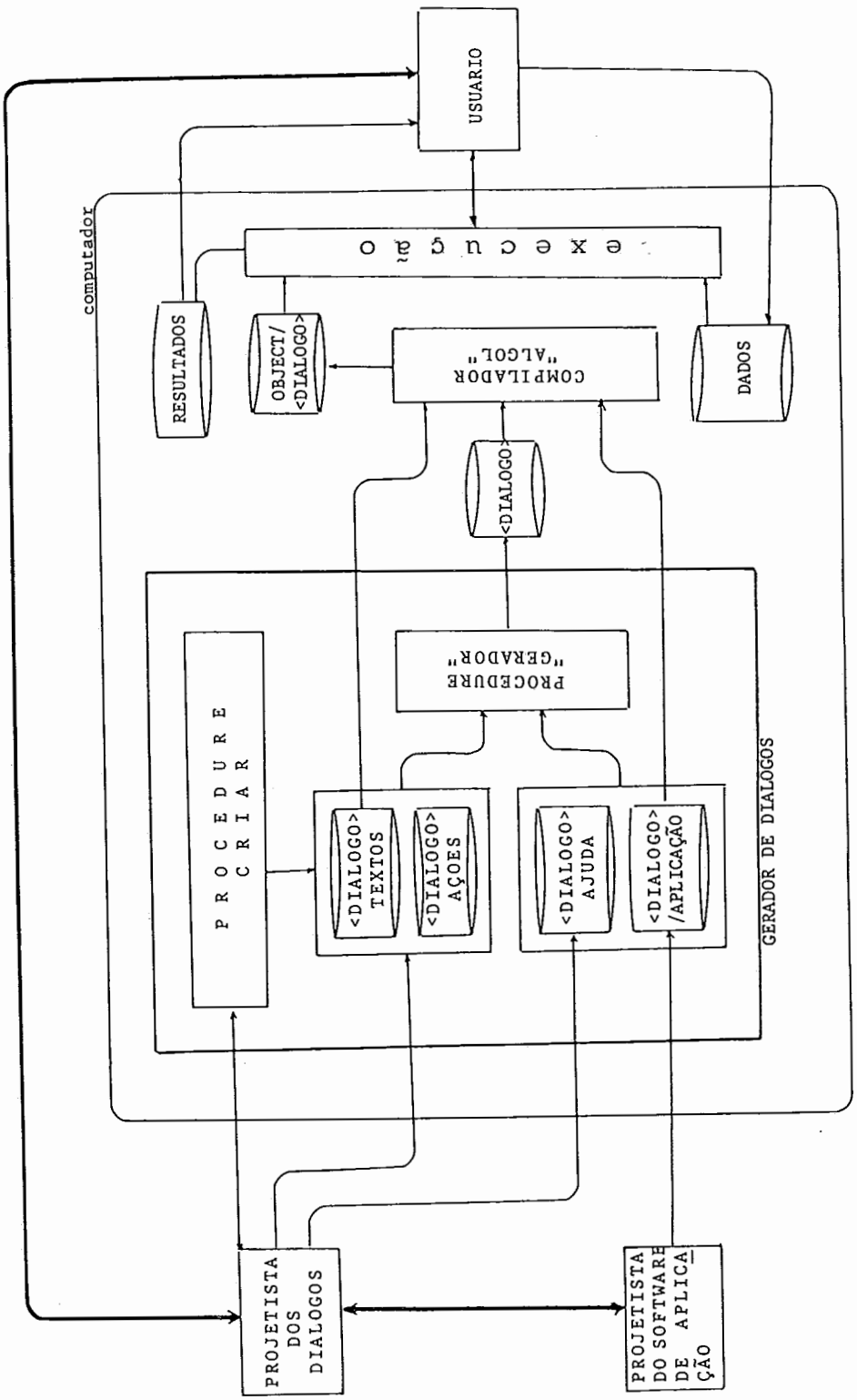


Fig. (4.21) - Esquema da utilização do "G D" na criação da interface usuário-computador

4.5.1 - O Uso Previsto para os Diálogos Produzidos pelo "G D"

a) No agrupamento de vários programas de aplicação:

Nesta primeira forma,o "G D" será utilizado para gerar a interface com o usuário, que possibilite o uso unificado de vários programas de aplicação com ou sem parâmetros. A esta forma de utilização chamaremos de execução de programas de aplicação.

b) No contexto de um programa de aplicação:

Nesta forma de aplicação, o "G D" será utilizado para gerar a interface com o usuário, necessária para a coleção de parâmetros que devem ser utilizados por um programa de aplicação em algum instante da sua execução.

A esta forma de utilização chamaremos de coleta de parâmetros.

c) Combinação dos casos anteriores.

Exemplos desses tipos de utilização do "G D" são apresentados no APÊNDICE 1.

4.5.2 - A Utilização da Procedure "CRIAR"

O programa CRIAR está preparado para interagir totalmente com o projetista. Quando executada, esta procedure colocará, no terminal de vídeo, mensagens e textos que auto-explicarão o seu uso.

Apresentamos, em seguida, um resumo que sugere a maneira como o projetista e o computador relacionam-se através dessa procedure.

- INÍCIO:
  - NOME DO DIALOGO A SER CRIADO? id
  - O DIALOGO TEM PARÂMETROS? S/N s/n
  - NOME DO PARÂMETRO 01? id
  - TIPO DESTE PARÂMETRO? I, R, S, B i, r, s, b
  - OUTRO PARÂMETRO? S/N s/n

- PRIMEIRO MENU
  - NOME DO PRIMEIRO MENU? id
  - TÍTULO DESTE MENU? até 72 carac
    até 72 carac
    ⋮
  - TEXTO ADICIONAL? até 72 carac
    até 72 carac
    ⋮

- NÚMERO DO PRIMEIRO ITEM?
  (SE O NÚMERO FOR INVÁLIDO, O ITEM SERÁ CONSIDERADO FICTÍCIO) ..... num

- TIPO DE AÇÃO DESTE ITEM?
  APRESENTAR, EXECUTAR OU COLETAR ..... a, e ou c

- AÇÃO DE TIPO "APRESENTAR"

  - NOME DO MENU QUE SERÁ APRESENTADO PELA ESCOLHA DESTE ITEM? ..... id

- AÇÃO DE TIPO "EXECUTAR

  - NOME DA PROCEDURE A SER EXECUTADA? ..... id
  - ALGUMA OUTRA PROCEDURE? S/N ..... s/n
  - NOME DESSA PROCEDURE? ..... id
  - TEM PARÂMETROS? S/N ..... s/n
  - NOME DA TELA QUE OS COLETA? ..... id
  - ALGUMA OUTRA PROCEDURE ..... s/n

- AÇÃO DO TIPO "COLETAR"

  - NOME DA TELA PARA A COLEÇÃO DOS PARÂMETROS? ..... id
  - NOME DA PROCEDURE A SER EXECUTADA? ..... id
  - ALGUMA OUTRA PROCEDURE? S/N ..... s/n

- TEM PARÂMETROS? S/N ..... s/n
-
-

- OUTROS ÍTENS
  - NESTE MENU - ALGUM OUTRO ITEM? S/N ..... s/n
  - NUMERO DO ITEM? ..... num
  -
  -

- OUTROS MENUS
  - DESEJA DESCREVER OUTRA DESCRIÇÃO? S/N ..... s/n
  - MENU OU TELA DE COLETA DE PARÂMETROS? S/N ..... m/t
  - NOME DO NOVO MENU? ..... id
  -
  -

- TELAS PARA COLETA DE PARÂMETROS
  - NOME DA TELA DE COLETA DE PARÂMETROS? ..... id
  - NÚMERO DE PARÂMETROS > 0 e ≤ 12 ..... num
  - NOME DO PARÂMETRO 01? ..... id

- TEXTO DESCRITIVO DO PARÂMETRO? _____
_____ { }
até 72 carac

- TIPO DO PARÂMETRO? I, R, S, B i, r, s, b

- NOME DA PROCEDURE DE AVALIAÇÃO? id

- NOME DA PROCEDURE DE AUXÍLIO id

- NOME DO PARÂMETRO 02? id

-

-

As mensagens até aqui apresentadas não são totalidade das que CRIAR possui. O restante delas pode ser deduzido a partir do esquema apresentado na Fig. (4.7) que apresenta o relacionamento Projetista - "G D" através da interface CRIAR.

4.5.3 - Atualizações feitas em Diálogos já existentes

A necessidade de atualizações dos diálogos interpretados pelo "G D", na criação de alguma interface com o usuário, pode provir, entre outros, dos seguintes aspectos:

- a natureza mutável do relacionamento usuário-computador requer que interfaces com o usuário, preparadas para agir de uma determinada forma, devam ser modificadas para satisfazer novas necessidades e condições impostas pelos usuários;

- situações do desempenho da interface, não pre

vistas durante o projeto e somente detetadas na etapa de utilização;

- erros cometidos durante o fornecimento das descrições dos diálogos.

Pelas razões apontadas, faz-se necessário que as descrições de diálogos possam ser agilmente atualizadas ou modificadas. Tais modificações podem estar incluídas nos seguintes grupos:

a) alterações que não envolvam a inclusão de novas descrições de menus ou TCP's ou a destruição de descrições já existentes.
Exemplos:
- atualizações dos identificadores das descrições ou procedimentos;
- atualização, na ordem de execução, das ações de algum item do menu;
- alterações nos textos apresentados ao usuário;
- alterações dos nomes dos parâmetros de uma TCP;
- em geral, alterações que não impliquem na retirada ou inclusão de registros nos arquivos <DIALOGO> /AÇÕES ou <DIALOGO> /TEXTOS.

Estas atualizações, podem ser realizadas através do uso do Editor de Textos do computador. O uso do Editor é perfeitamente aceitável pelo

"G D", pois os arquivos <DIALOGO> /AÇÕES e <DIALOGO> /TEXTOS, criados a partir da execução da procedure CRIAR são do mesmo tipo daqueles arquivos tratados normalmente pelo programa Editor do B-6700.

b) atualizações que impliquem a inclusão de novas descrições de menus ou TCP's ou a destruição de descrições já existentes.
Este tipo de atualizações é também possível de ser realizado através do uso do editor do computador, porém, após estas modificações serem realizadas, será necessário manter a coerência das informações contidas na tabela de endereços dos arquivos <DIALOGO> /AÇÕES e <DIALOGO> /TEXTOS;

c) atualizações que impliquem na inclusão de numerosas descrições de menus ou TCP's:

- neste caso o projetista poderá utilizar novamente a procedure CRIAR para a entrada dessas novas descrições. Os resultados da nova utilização do CRIAR poderão ser adicionados aos arquivos <DIÁLOGO> /AÇÕES e <DIALOGO> /TEXTOS. Tal como nos casos anteriores, é necessário manter a coerência das informações contidas nas respectivas tabelas de endereços destes arquivos.
Para que qualquer uma destas atualizações

possa ser realizada da maneira prevista pelo "G D", é preciso que o projetista dos diálogos, não altere a sintaxe utilizada pelo "G D", para a codificação das descrições de menus ou TCP's, nos arquivos <DIALOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.

Apresentamos, a seguir, o detalhe da estruturação das informações contidas nestes arquivos:

1) O Arquivo <DIÁLOGO> /AÇÕES

Na figura (4.22) apresentamos um esquema que expõe os diferentes "setores" deste arquivo, utilizados na codificação das ações de um diálogo.

1	endereço da tabela de endereços
2	codificação das informações correspondentes à características do diálogo.
	codificação das informaçoes correspondentes ao primeiro menu do diálogo.
3	codificação das informações correspondentes a outros menus ou TCP integrantes do diálogo.
5	tabela de endereços de arquivo

Fig. (4.22) - Os setores da codificação de um diálogo no arquivo <DIALOGO> /ACOES

O setor (1) — Endereço da Tabela

Exemplo: tabel 000020

O valor 20 indicaria que o endereço do primeiro registro da tabela do arquivo estará na posição 20

O setor (2) — Informações correspondentes ao diálogo interpretado:

Sintaxe utilizada: figura (4.23)

```
    DIALOGO ⟶ dial--nome--      ID     PARAMETROS
PARAMETROS  ⟶ sim---NOMES
              | nao---

     NOMES  ⟶ param---INT       ID     TIPO--IRSB  NOMES
              | null

       INT  ⟶ string                    (valor inteiro)
        ID  ⟶ string                    (identificador)
      IRSB  ⟶ 'I'                       (integer      )
              | 'R'                     (real         )
              | 'S'                     (abcdic array )
              | 'B'                     (boolean      )
```

nomenclatura: Não terminais --- maiusculas
terminais --- minusculas ou entre apóstrofes

Fig. (4.23) Sintaxe utilizada na codificação das informações sobre o diálogo interpretado.

O setor (3) Informações correspondentes ao primeiro menu do diálogo interpretado.

Na figura (4.24) é apresentada a sintaxe utilizada na codificação das informações correspondentes ao primeiro menu e que é válida para o restante dos menus que pudessem ser parte do diálogo.

```
     MENU ⟶ descricao___   NOME     tipo__'G'
     ITEM ⟶ item__  INT ACOES
    ACOES ⟶ acao__'A'   APRESENTAR
            | acao__'E'   EXECUTAR
            | acao__'C'   COLETAR
APRESENTAR ⟶ displ_      ID
  EXECUTAR ⟶ exec__     NOME    SEQUÊNCIA
   COLETAR ⟶ tela__colec_      NOME   exec__   ID   SEQUÊNCIA
 SEQUENCIA ⟶ EXECUTAR
            |exec__      ID    tela__colec_    ID   SEQUÊNCIA
            |displ_      ID
       ID ⟶ string                    (identificador)
            |null
     NOME ⟶ string                    (identificador)
      INT ⟶ string                    (valor inteiro)

          nomenclatura: Nao terminais --- maiúsculas
                        terminais --- minúsculas,  ou
                                      entre apóstrofes
```

Fig. (4.24) - Sintaxe utilizada na codificação da descrição de um menu.

O setor (4) Informações correspondentes ao resto dos menus ou TCP's presentes no diálogo.

Na figura (4.25) mostramos a sintaxe utilizada para codificação das informações correspondentes a uma TCP - a sintaxe apresentada na figura (4.24) é válida para o caso dos menus.

```
      TCP ⟶ descricao---  NOME--  tipo--'D'  numer-param-  INT
PARAMETRO ⟶ param-  INT   NOME    tipo--IRSB rest--nome--   NOME
            help--nome--  NOME
     NOME ⟶ string                (identificador)
      INT ⟶ string                (valor inteiro)
     IRSB ⟶ 'I'                   (integer      )
          | 'R'                   (real         )
          | 'S'                   ( ebcdic array)
          | 'B'                   (boolean      )
```

nomenclatura: Terminais --- minúsculas ou entre apóstrofes
Nao terminais - maiúsculas

Fig. (4.25) - Sintaxe utilizada na codificação na codificação da descrição de uma TCP.

O setor (5) Informações contidas na tabela de endereços.

No final do arquivo <DIÁLOGO> /AÇÕES, encon

tra-se uma tabela que registra,para cada descrição do diálogo, certas informações que logo poderão ser utilizadas na localização ou identificação de cada menu ou TCP, codificados no arquivo. Nesta tabela, cada tupla corresponderá a uma descrição.

Na figura (4.26) apresentamos o detalhe das informações contidas em cada um dos campos desta tabela.

①	②	③	④	⑤	⑥
MENU3	7.	4.	G	2.	3.

① - Nome da descrição

② - Endereço do primeriro registro da descrição no arquivo

③ - Número de registros utilizados da codificação desta descrição

④ - Tipo de descrição, G - Menus D - TCP's

⑤ - Número de ítens para o caso dos menus ou de parâmetros para o caso das TCP's.

⑥ - Contador das descrições de menus. Este contador será utilizado como código numérico para a identificação de cada menu.

Fig. (4.26) - Informações contidas na tabela do arquivo <DIALOGO> /ACOES

2) O Arquivo <DIÁLOGO> /TEXTOS

Neste arquivo ficam armazenados os textos correspondentes a cada descrição de menu ou TCP. Eles são armazenados sem sofrer nenhuma alteração e, por este motivo, qualquer observação deverá ser colocada unicamente na construção da tabela de endereços do arquivo.

Na figura (4.27) mostramos o detalhe das informações contidas nessa tabela de endereços.

①	②	③	④	⑤
MENU3	15.	6.	G	3.

① - Nome da descrição

② - Endereço do primeiro registro do texto, correspondente a esta descrição

③ - Número de total de registros utilizados naquele texto

④ - Tipo da descrição: G - menus D - TCP

⑤ - Contador de menus

Fig. (4.27) - Informações contidas na tabela do arquivo <DIÁLOGO> /TEXTOS

O primeiro registro do arquivo <DIÁLOGO> /TEXTOS é também utilizado para salvar o endereço da tabela de endereços.

4.5.4 - A Utilização da Procedure "GERADOR"

As descrições dos diálogos recebidos através da procedure CRIAR estão livres de erros sintáticos e, por tal motivo, os arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS, produzidos pela execução do CRIAR são perfeitamente aptos para ser utilizados como entrada da execução da procedure "GERADOR".

No entanto, durante a execução da procedure CRIAR, não é realizado nenhum tipo de análise semântica sobre as descrições de diálogos recebidas.

É possível também que as descrições de diálogos, submetidas ao GERADOR através dos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS, estejam contendo algum tipo de erro sintático, produzido por um manuseio não previsto, destes arquivos.

Por esses motivos, a procedure GERADOR está preparada para realizar a análise sintática e semântica das descrições de diálogos que vão ser traduzidas para código fonte ALGOL.

A execução da procedure GERADOR que começa perguntando ao projetista pelo:

NOME DO DIÁLOGO QUE ESTÁ SENDO GERADO?

continuará até o final da geração de todo o cógigo fonte correspondente aquele diálogo, ou será suspensa no caso de serem encontrados erros sintáticos ou semânticos.

Na presença de erros, a execução do gerador apresenta as respectivas mensagens de erro, que identificam o número do registro no arquivo <DIÁLOGO> /AÇÕES ou <DIÁLOGO>/TEXTOS que contêm o erro, assim como explicam, ao projetista, o tipo de erro detetado.

Embora as mensagens de erro apresentadas pelo GERADOR correspondam à etapa de implementação, apresentaremos,no entanto, as circunstâncias mais comuns nas quais aqueles erros acontecem.

- identificadores encontrados inválidos:
  poderiam ser palavras reservadas da linguagem ALGOL ou nomes repetidos de menus ou TCP's, assim como poderiam aparecer repetidamente em uma mesma lista de parâmetros etc.;

- nomes de descrições que não fazem parte do diálogo;

- nomes de procedures que não foram encontradas nos arquivos <DIÁLOGO> /AJUDA ou <DIÁLOGO> /APLICAÇÃO;

- alteração da sintaxe estabelecida para a codificação das descrições;

- falta de coerência das informações colocadas nas tabelas de endereços;

- inconsistências entre as informações fornecidas pelas tabelas de endereços e as informações encontradas em cada uma das descrições;

- valores numéricos que ultrapassam os limites estabelecidos.

4.5.5 - O uso do Código Fonte gerado

O código fonte produzido pela execução da procedure GERADOR é colocado no espaço em disco do projetista,através dos seguintes arquivos:

<DIÁLOGO>

<DIÁLOGO>/FOWAR

<DIÁLOGO>/EXEC

<DIÁLOGO>/PROC

Para que este código fonte possa ser executado, é necessário que o arquivo <DIÁLOGO> seja compilado. Isto poderá ser feito pelo projetista através de algum comando do computador do tipo:

COMPILE <DIÁLOGO>

Durante a compilação, devem estar presentes,no espaço em disco do projetista, todos os arquivos acima mencionados, assim como o arquivo <DIÁLOGO> /TEXTOS.

Após a compilação ter sido realizada, o usuário do diálogo poderá fazer uso do código objeto, produzido pelo compilador ALGOL.

O uso deste código objeto será aquele que foi previsto para os diálogos interpretados através do "G D".

- Para o caso do agrupamento de programas de aplicação, o uso deste código objeto é direto.

  Exemplo: RUN<DIÁLOGO>

- No caso da coleta de parâmetros, devem ser observados os seguintes aspectos:

  a) a coleta de parâmetros produz-se pelo mapeamento dos nomes da lista de parâmetros do diálogo, com os nomes das listas de parâmetros de uma ou mais TCP's, integrantes daquele diálogo. Portanto, a coleta de parâmetros não se produzirá naqueles parâmetros cujos nomes não se encontrem em nenhuma das listas de parâmetros das TCP's do diálogo;

  b) a coleta de parâmetros não será realizada, para os casos em que algum dos valores fornecidos pelo usuário não cumpra com as condições impostas durante a descrição do diálogo;

  c) como esta coleta de parâmetros será feita no contexto de um novo programa de aplicação, é necessário que, neste programa, sejam satisfeitas as condições necessárias para ligação do código objeto do diálogo.

V - A IMPLEMENTAÇÃO DO GERADOR DE DIÁLOGOS

5.1 - Características Gerais da Implementação.

A versão atual do "GERADOR DE DIÁLOGOS" foi escrita na linguagem de programação "ALGOL EXTENDED" do computador Burroughs 6700.

Na escolha da máquina, para a implementação do "G D", foram levados em conta os seguintes fatores:

a) quando implementado no B6700, o "G D" poderia ser utilizado diretamente, por uma grande quantidade de projetistas de software, que atualmente trabalham nesse tipo de computador na universidade Federal do Rio de Janeiro.

b) a quantidade de recursos, assim como a versatilidade da interface de entrada e saída do B-6700, prognosticavam a implementação do "G D", sem maiores dificuldades;

c) o programa "BINDER" do B-6700 permitiria que o "CODIGO FONTE ALGOL", gerado pelo "G D", pudesse ser utilizado por programas de aplicação, escritos em qualquer uma das seguintes linguagens de programação: ALGOL, COBOL e FORTRAN;

d) o aproveitamento das funções de formação da tela, dos terminais TS-800, atualmente ligados ao B-6700.

Na escolha da linguagem de programação, foram levados em conta os seguintes fatores:

a) a linguagem de programação "ALGOL EXTENDED" é a que melhor aproveita todos os recursos do B-6700;

b) sendo o ALGOL uma linguagem "estruturada", o trabalho de implementação seria, facilmente, separado em módulos;

c) o poder e a versatilidade do ALGOL EXTENDED, na manipulação de cadeias de caracteres, facilitariam, de maneira definitiva, a tarefa de geração de código fonte da interface.

Nas seções seguintes serão descritas a forma de implementação dos componentes da especificação apresentada no capítulo anterior.

5.2 - A Implementação da Interface com o Projetista

O "G D" realiza a interpretação de uma descrição de diálogo para código fonte em duas etapas:

- Recepção de descrição do diálogo;

- Geração do código fonte ALGOL

A procedure CRIAR, cuja implementação discutiremos a seguir, permite que o projetista possa fornecer de maneira interativa as descrições desses diálogos.

5.2.1 - Definição dos Módulos Principais da Procedure CRIAR

1) O Módulo "DIALCAR" (Fig. 5.1)

Este módulo é o responsável pela entrada das características do Diálogo que vai ser interpretado.

Tais características compreendem:

- nome do diálogo;

- nome e tipo dos (possíveis) parâmetros que façam parte do diálogo.

2) O Módulo "DESCRIÇOES"

Através deste módulo, o projetista fornece as informações correspondentes a cada uma das descrições do diálogo.

Estas descrições poderão ser menus ou telas para coleta de parâmetros (TCP).

3) O Módulo "ENDEREÇOS"

A função deste módulo é a gravação das tabelas de endereços ao final dos arquivos <DIALOGO> / AÇOES e <DIALOGO> /TEXTOS.

A função destes arquivos assim como de suas respectivas tabelas de endereços foi especificada no capítulo IV, secção (4.4.2).

5.2.2 - O Funcionamento da Procedure CRIAR

De uma maneira muito simplificada, o funcionamento do CRIAR pode ser descrito nos seguintes termos:

1) Entrada das características do diálogo, através da execução do módulo DIALCAR;

2) Entrada de cada um dos menus ou TCPs do diálogo, através da execução do módulo DESCRIÇÕES;

3) Gravação das Tabelas de endereços dos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS, através da execução do módulo ENDEREÇOS.

No entanto, uma das características de maior destaque do funcionamento do CRIAR é a forma como ele interage com o projetista, no ingresso das descrições.

Essa maneira de interagir do CRIAR reflete as facilidades que o sistema "G D" pretende oferecer ao usuário do diálogo. Algumas dessas facilidades são as seguintes:

- o projetista fornece as informações em um esquema on-line de perguntas e respostas;

- todas as informações fornecidas pelo projetista são entregues com formato livre;

- na presença de erros, o CRIAR oferece, ao projetista, diálogos explicativos, que facilitam a recuperação do fluxo das informações, após terem acontecido tais erros.

5.2.3 - Descrição dos Módulos Utilizados na Implementação da Procedure CRIAR

O Módulo "DIALCAR" (Fig. 5.1)

Finalidade: A função deste módulo é receber, do projetista, o nome do diálogo e os nomes dos possíveis parâmetros que façam parte desse diálogo.

O módulo DIALCAR realiza seu objetivo, face às chamadas dos módulos DIALNOME e DIALPARAM.

As informações recolhidas pelo módulo DIALNOME são devolvidas ao módulo DIALCAR em um array de caracteres bufdial.

No caso do Diálogo ter parâmetros, o DIALCAR continuará sua execução chamando o módulo "DIALPARAM". Se não, o "DIALNOME" devolverá o buffer bufdial ao CRIAR.

Caso o módulo "DIALPARAM" for executado, ele recebe, como parâmetro de entrada, o buffer bufdial e devolve, como parâmetro de saída, o mesmo buffer bufdial, porém acrescido da informação correspondente ao tipo de cada parâmetro de Diálogo (nesta implementação o número máximo de parâmetros é 12 - doze).

Os módulos DIALNOME E DIALPARAM interagem com o projetista, através dos módulos OBTEMNOME e PEGALETRA.

A finalidade do módulo OBTEMNOME é apresentar uma mensagem ao projetista e receber dele um identificador ALGOL.

Esse módulo é utilizado, repetidamente, ao longo dos diferentes módulos da procedure CRIAR e encontra-se descrito na Seção 5.2.6, deste Capítulo.

A finalidade do módulo PEGALETRA é expor uma men

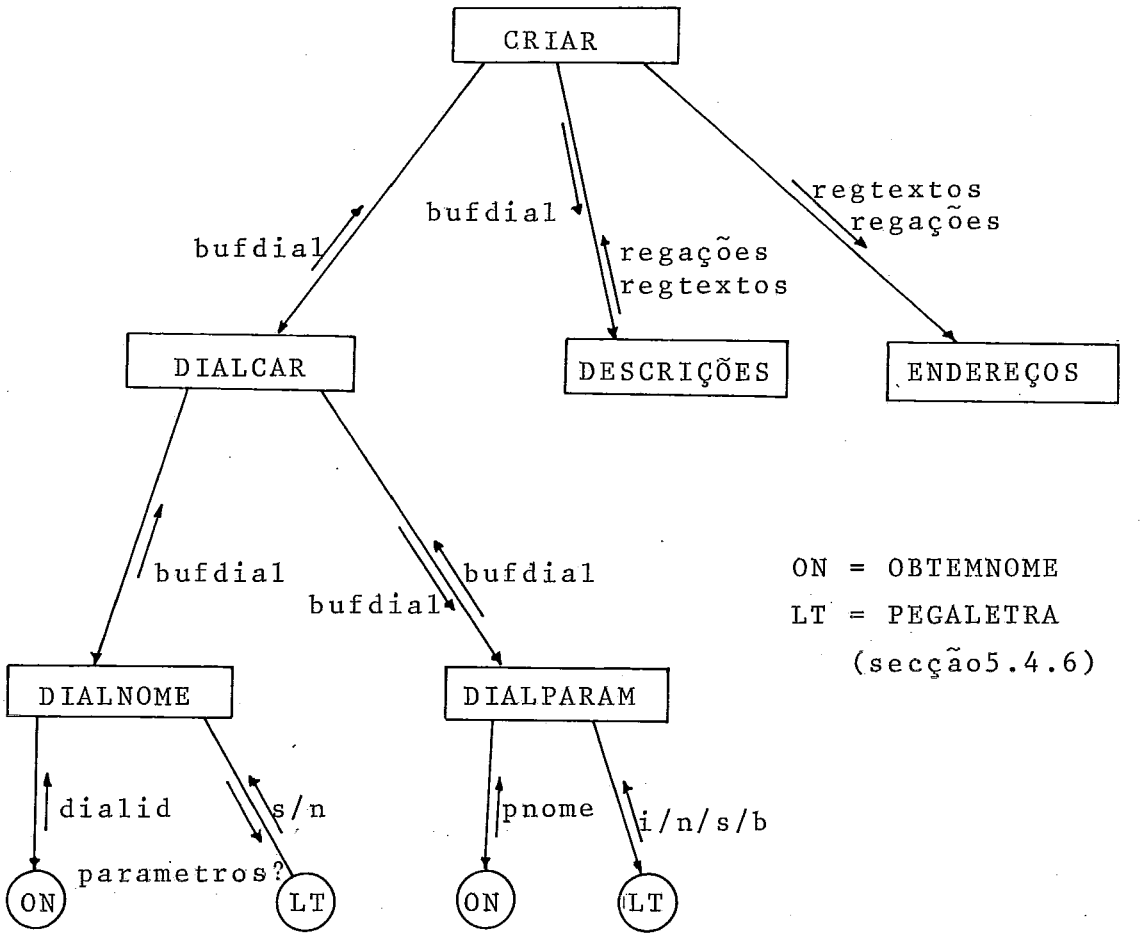


Fig. (5.1)

Procedure CRIAR; módulo DIALCAR

sagem ao projetista e receber dele um carater.

O módulo PEGALETRA, que também é utilizado repetidamente ao longo dos diferentes módulos da procedure CRIAR, encontra-se descrito na Seção 5.2.6, deste capítulo.

O Módulo "DESCRIÇÕES" (Fig. 5.2)

Finalidade: A função deste módulo é receber, do projetista, as informações correspondentes a cada uma das descrições componentes do diálogo.

Parâmetros de Entrada:

O módulo DESCRIÇÕES recebe, como entrada, o buffer bufdial, contendo as informações do diálogo, recolhidas pelo módulo DIALCAR.

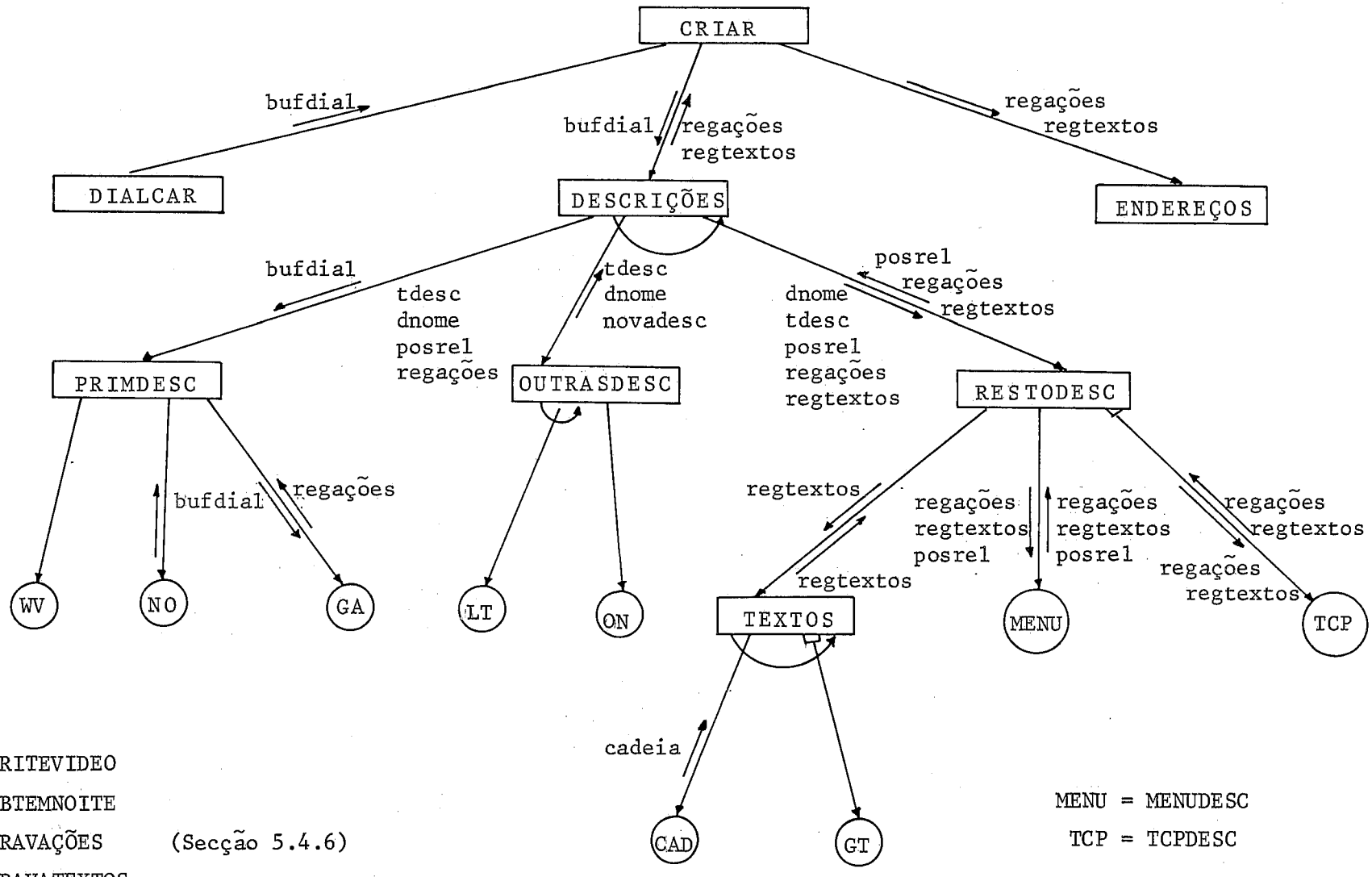
A execução do DESCRIÇÕES é feita através da execução dos módulos PRIMDESC, OUTRASDESC e RESTODESC.

O objetivo do módulo PRIMDESC é receber, do projetista, o "nome" da primeira descrição do diálogo e gravar, no arquivo <DIALOGO> /AÇÕES, as informações contidas no buffer bufdial.

O objetivo do módulo OUTRASDESC é permitir ao projetista continuar com o fornecimento de novas descrições ou,encerrar a execução do CRIAR.

No caso do projetista desejar fornecer as informações de uma nova descrição, o OUTRASDESC receberá o nome e o tipo dessa descrição.

O objetivo do módulo RESTODESC é receber, do projetista, as informações correspondentes a cada descrição do diálo



WV = WRITEVIDEO

ON = OBTEMNOITE

GA = GRAVAÇÕES (Secção 5.4.6)

GT = GRAVATEXTOS

CAD = PEGACADEIA

MENU = MENUDESC

TCP = TCPDESC

Fig. (5.2)

Procedure CRIAR; Módulo DESCRIÇÕES

go e gravar estas informações nos arquivos <DIALOGO> /AÇÕES ou <DIALOGO> /TEXTOS.

A execução do módulo DESCRICOES podem ser descrita nos seguintes termos:

1) Chamada ao módulo PRIMDESC

2) Após a execução do PRIMDESC, é chamado o módulo RESTODESC que recebe do projetista o resto das informações da primeira descrição do DIÁLOGO.

3) Após a execução do módulo RESTODESC o módulo DESCRICOES consultará ao projetista através da execução do módulo OUTRASDESC se existem ou não novas descrições.
No caso da resposta do projetista ser afirmativa, o módulo DESCRICOES executará novamente o RESTODESC que receberá o resto das informações correspondentes a esta nova descrição. Desta maneira,o processo continua, até que os resultados obtidos pela execução do OUTRADESC demonstrem o desejo do projetista, de encerrar o ingresso de novas descrições.

Como pode ser visto, o nome e o tipo de cada descrição são obtidos através de módulos diferentes, isto se deve aos seguintes motivos:

1) após o projetista ter fornecido as informações correspondentes ao diálogo, ele precisa saber

que, a seguir, deverá começar o fornecimento das descrições desse diálogo. Além disso ele precisa saber, também, que essas descrições devem iniciar-se por uma do tipo "menu", tal como foi exposto no Capítulo IV (Seção 4.4.1). Estes fatos têm sido destacados na implementação do módulo PRIM DESC;

2) a utilização do módulo OUTRASDESC foi necessária, para permitir que o projetista possa, em um determinado momento, encerrar o fornecimento de novas descrições.

A execução do módulo RESTODESC é feita pela execução dos módulos TEXTOS, MENUDESC e TCPDESC.

O objetivo do módulo TEXTOS é receber, do projetista, os títulos e textos explicativos correspondentes a cada descrição e gravá-los no arquivo <DIÁLOGO> /TEXTOS.

Esses títulos e textos explicativos são recebidos através da execução do módulo PEGACADEIA e sua gravação é feita com o módulo GRAVATEXTOS. Os dois módulos citados encontram-se descritos na Seção 5.2.6, deste Capítulo.

As gravações nos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS, que estes módulos realizam, começam a partir do primeiro registro disponível para gravação, em cada um destes arquivos. Tais endereços ficam disponíveis nos parâmetros:regações e regtextos. Após as gravações terem sido realizadas,estes endereços são atualizados e devolvidos ao módulo RESTODESC.

O objetivo do módulo MENUDESC é receber, do proje

tista, todas as informações correspondentes a uma descrição do tipo menu e gravá-las nos arquivos <DIALOGOS> /AÇÕES e <DIÁLOGO> /TEXTOS.

O objetivo do módulo TCPDESC é o mesmo que o do anterior, porém para uma descrição do tipo tela de coleta de parâmetros TCP.

A execução dos módulos MENUDESC ou TCPDESC dependerá do tipo de descrição tdesc que estiver sendo fornecida pelo projetista. A descrição de cada um destes módulos encontra-se nas secções 5.2.4 e 5.2.5 deste capítulo.

Cada execução do módulo MENUDESC incrementa de 1 o contador de menus posrel, cuja utilização foi descrita na Seção 4.4.2, do capítulo IV.

Quando a execução do módulo MENUDESC ou TCPDESC (segundo o caso) é terminada, o módulo RESTODESC também encerra sua execução, devolvendo ao módulo DESCRIÇÕES os parâmetros regações, regtextos e posrel, atualizados.

Após sua execução, o módulo DESCRIÇÕES devolve, à procedure CRIAR, os seguintes parâmetros:

Regações: contendo o endereço do primeiro registro disponível para gravação no arquivo <DIÁLOGO> /AÇÕES.

Regtextos: contendo o endereço do primeiro registro disponível para gravação no arquivo <DIÁLOGO> /TEXTOS.

O Módulo "ENDEREÇOS" (Fig. 5.3)

Finalidade: A função deste módulo é gravar nos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> / TEXTOS as respectivas tabelas de endereços.

As informações contidas nestas tabelas de endereços foram sendo gravadas durante a execução dos módulos MENUDESC e TCPDESC, em dois arquivos auxiliares - AUXAÇÕES e AUXTEXTOS.

A descrição desses módulos encontra-se na Seção 5.2.4 e 5.2.5 deste capítulo.

Os módulos MENUDESC e TCPDESC tiveram que realizar a gravação destas tabelas de endereços nesses arquivos auxiliares, por não serem conhecidas, por antecipação, as posições nas quais elas poderiam ser gravadas nos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.

Tais posições, que correspondem aos endereços regações e regtextos, só foram conhecidas após o módulo DESCRIÇÕES ter sido executado.

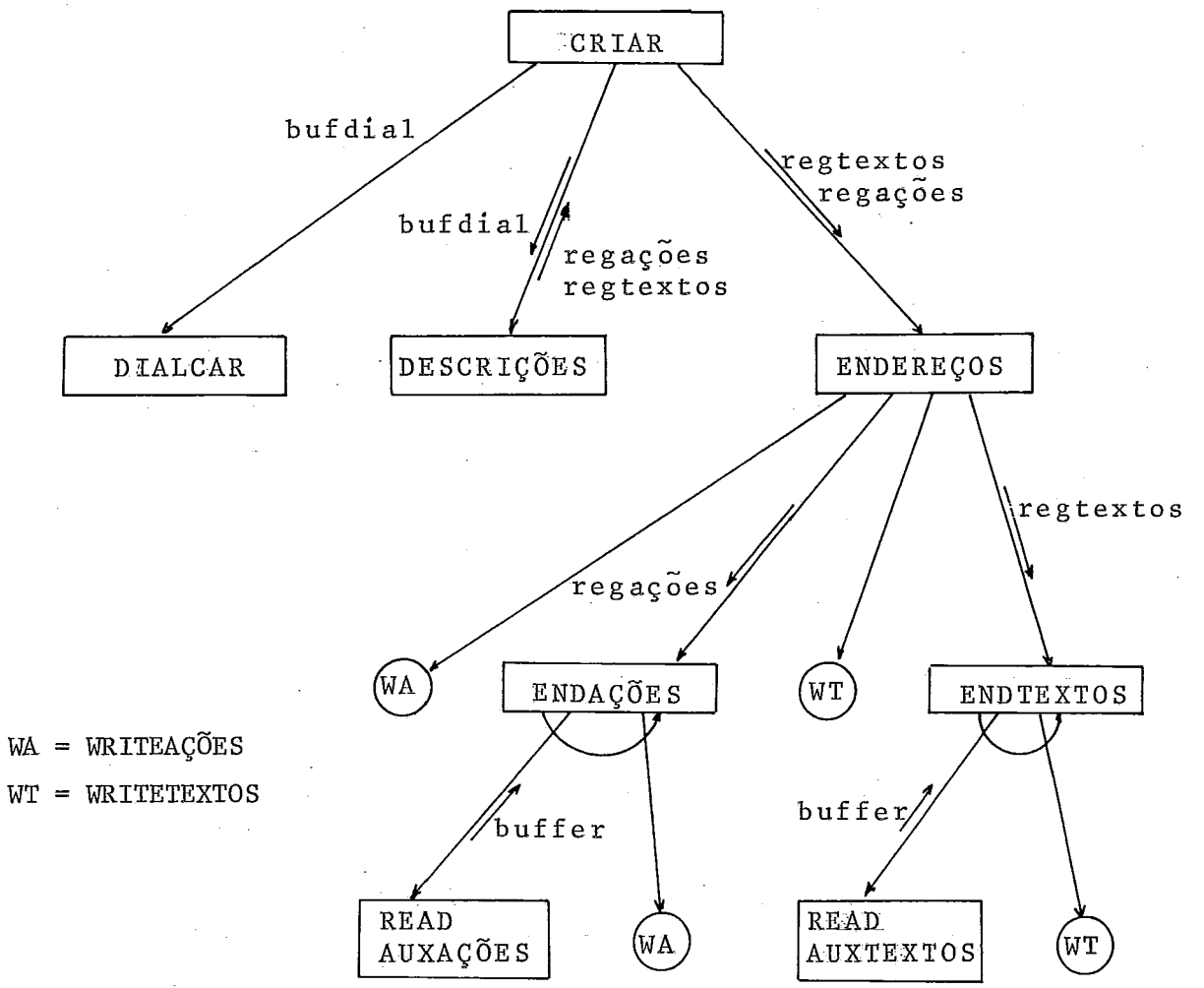
Essa forma de solução, no entanto, garante a otimização do espaço, em disco, determinado para os arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.

A execução do ENDEREÇOS inicia-se com a gravação,no primeiro registro do arquivo <DIÁLOGO> /AÇÕES do endereço regações, onde começará a gravação da tabela do arquivo <DIÁLOGO> /AÇÕES.

A seguir, o módulo ENDEREÇOS chama o módulo ENDAÇÕES.

A função do módulo ENDAÇÕES é passar o conteúdo do

Fig. (5.3)
Procedure CRIAR; Módulo Endereços



arquivo AUXAÇÕES (a tabela de endereços) para o arquivo <DIALOGO> /AÇÕES.

No momento em que todos os registros do arquivo AUXAÇÕES tenham sido lidos e logo gravados no arquivo <DIALOGO> /AÇÕES, o módulo ENDAÇÕES removerá, do disco, o arquivo AUXAÇÕES.

Quando o ENDAÇÕES termina sua execução, o módulo ENDEREÇOS repete todos os passos descritos para o arquivo <DIALOGO> /AÇÕES no arquivo <DIALOGO> /TEXTOS. Desta forma, o módulo ENDEREÇOS encerra sua execução e, com ela, a execução da procedure "CRIAR".

5.2.4 - Descrição do Módulo "MENUDESC" Fig. (5.4)

Finalidade: a função deste módulo é receber todas informações correspondentes a uma descrição de tipo menu e gravá-las nos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> / TEXTOS.

Esse módulo recebe os seguintes parâmetros de entrada:

dnome : contendo o nome dessa descrição;

regações e regtextos : contendo o endereço do primeiro registro disponível para gravação no arquivo <DIÁLOGO> /AÇÕES e <DIALOGO> /TEXTOS.

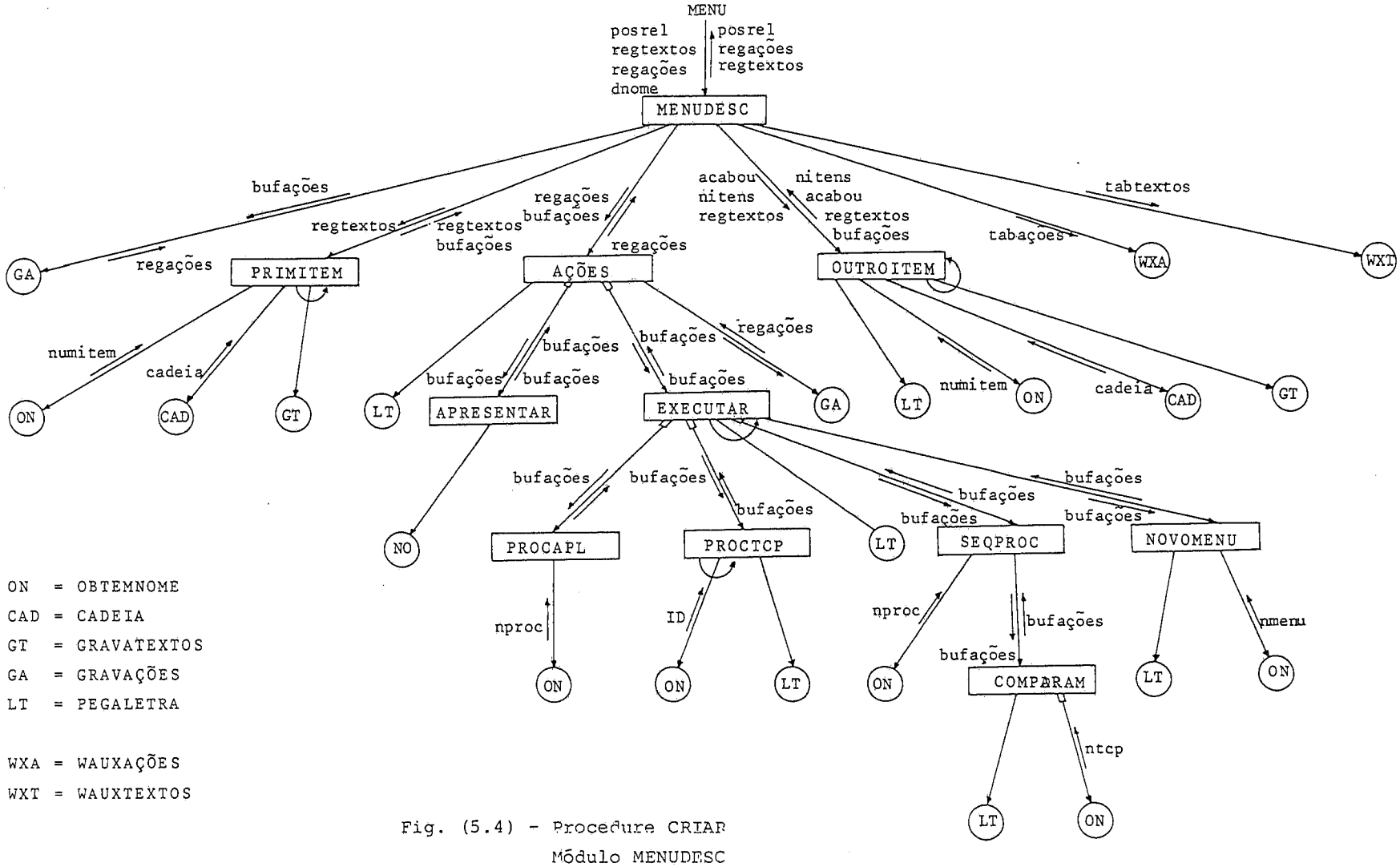
posrel : contendo o contador de descrições de tipo "menu" - o uso deste contador foi descrito na Seção 4.4.2; do capítulo IV.

A execução do MENUDESC é feita face às chamadas dos módulos PRIMITEM, AÇÕES e OUTROITEM.

A função do PRIMITEM é receber, do projetista, o número e o texto explicativo correspondente ao primeiro item desta descrição de tipo menu.

A função do AÇÕES é receber, do projetista, as informações correspondentes às ações que o sistema deverá realizar, no caso do item que está sendo descrito pelo projetista ser aquele escolhido pelo usuário, durante a apresentação do menu.

A função do OUTROITEM é consultar o projetista so



ON = OBTEMNOME
CAD = CADEIA
GT = GRAVATEXTOS
GA = GRAVAÇÕES
LT = PEGALETRA

WXA = WAUXAÇÕES
WXT = WAUXTEXTOS

Fig. (5.4) - Procedure CRIAR
Módulo MENUDESC

bre a existência de novos ítens, que pertençam ao menu cuja descrição está sendo recebida. Caso a resposta do projetista for afirmativa, o módulo OUTROITEM receberá o número e o texto explicativo correspondentes a este novo item.

A maneira como se realiza a execução do MENUDESC pode ser descrita nos seguintes termos:

1) gravação no arquivo <DIÁLOGO> /AÇÕES do nome e do tipo dessa descrição - estas informações serão recebidas em um buffer de caracteres bufações, que torna possível o uso do módulo GRAVAÇÕES;

2) chamada ao módulo PRIMITEM - após a execução deste módulo, O MENUDESC realiza o passo (3);

3) chamada ao módulo AÇÕES - após a execução deste módulo, o MENUDESC realiza o passo (4);

4) chamada ao módulo OUTROITEM - dependendo da resposta que este módulo receba do projetista,a execução do MENUDESC apresenta duas opções:

   a) resposta afirmativa - neste caso, o MENUDESC realizará o passo (3);

   b) resposta negativa - a execução do MENUDESC realizará o passo (5);

5) para encerrar, o módulo MENUDESC grava, em dois arquivos auxiliares, AUXAÇÕES e AUXTEXTOS, os va

lores correspondentes aos endereços de gravação e outras informações que ficarão no final dos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.

O conteúdo dessas tabelas de endereços foi descrito na Seção 4.4.2, do capítulo IV.

A justificativa para a utilização desses arquivos auxiliares foi descrita na Seção 5.2.3, deste capítulo, na parte que corresponde à descrição do módulo ENDEREÇOS.

Como pode ser visto, a recepção do número e do texto do primeiro item de uma descrição de tipo menu tem sido tratada separadamente, através do módulo PRIMITEM. Isto se deve aos seguintes motivos:

a) tal como foi apresentado na Seção 4.3.1, do capítulo IV, o projetista tem a possibilidade de incluir, no conjunto dos ítens de um menu, um "item fictício", cujas ações serão executadas pelo sistema, antes da apresentação, ao usuário, do restante dos ítens do menu;

b) entretanto, o número de ítens fictícios permitidos na descrição de um menu é um só e as informações correspondentes às ações deste item deverão ser prévias às informações correspondentes ao restante dos ítens do menu.

O número de ítens fictícios foi limitado a um (1) para facilidade de implementação. No caso de novos ítens fictícios virem a ser necessários, sem

pre poder-se-ia colocá-los nas descrições de outros menus.

A necessidade do item fictício ser o primeiro a ser descrito, também foi colocada para facilitar a implementação;

c) a possibilidade de que o primeiro item de um menu possa ser um item fictício é destacada no módulo PRIMITEM.

A seguir, apresentaremos alguns detalhes de cada um dos módulos chamados pelo MENUDESC:

O Módulo "PRIMITEM"

Tal como foi descrito, a finalidade deste módulo é receber, do projetista, o número e o texto explicativo correspondentes ao primeiro item do menu. Caso o projetista deseje que aquele primeiro item seja considerado item fictício, o PRIMITEM substituirá o número do item por uma marca, que identificará que aquele item foi considerado fictício.

Os textos que o PRIMITEM recebe são gravados diretamente no arquivo <DIÁLOGO> /AÇÕES; as informações correspondentes ao número do item são devolvidas em um buffer de caracteres bufações.

O Módulo "AÇÕES"

A função deste módulo é receber, do projetista, as ações especificadas para cada um dos ítens do menu.

Esse módulo recebe, como entrada, os seguintes parâ

metros:

regações : contendo o endereço do primeiro registro disponível para gravação no arquivo <DIÁLOGO> /AÇÕES;

bufações : contendo o número do item.

A execução do AÇÕES pode ser descrita nos seguintes termos:

1) consulta ao projetista sobre o tipo de ação preparada para o item; as ações possíveis são: Apresentar, Executar e Coletar (Seção 4.3.1, capítulo IV);

2) no caso da resposta do projetista ser Apresentar, o AÇÕES realizará o passo (3) - se a resposta do projetista for Executar ou Coletar, o AÇÕES realizará o passo (4);

3) chamada ao módulo APRESENTAR - este módulo recebe, do projetista, o nome do menu que deverá ser apresentado caso o item que está sendo descrito for escolhido pelo usuário; a resposta do projetista será adicionada ao buffer bufações (de acordo com a sintaxe estabelecida na Seção 4.5.3, do capítulo IV) e nele devolvida ao módulo AÇÕES; após a realização deste passo (3),o módulo AÇÕES realizará o passo (5);

4) chamada ao módulo EXECUTAR - este módulo receberá, do projetista, os nomes das procedures de aplicação ou TCP's (telas de coleta de parâmetros), que deverão ser executadas como ações desse item - as informações recolhidas pelo Executar serão adicionadas no buffer bufações e nele devolvidas ao módulo AÇÕES; após a realização deste passo (4), o módulo AÇÕES realizará o passo (5);

5) o módulo AÇÕES grava, no arquivo <DIÁLOGO> / AÇÕES o conteúdo do buffer bufações. Depois desta gravação ter sido realizada, o AÇÕES enviará de volta o endereço de gravação regações, atualizado.

Visando uma maior clareza na descrição deste módulo MENUDESC, a descrição da implementação do módulo EXECUTAR, anteriormente mencionado, encontra-se no final desta Seção 5.2.4.

O Módulo "OUTROITEM"

Finalidade: a função deste módulo é consultar o projetista se ainda existem outros ítens que pertençam ao menu, cuja descrição está sendo recebida.

O módulo OUTROITEM começa sua execução, perguntando ao projetista, se deseja ou não fornecer a descrição de um novo item.

Se a resposta do projetista for negativa, o OUTRO ITEM encerra sua execução e devolve, ao módulo MENUDESC,a variável booleana acabou, como verdadeira. Em caso contrário, o módulo OUTROITEM continuará sua execução, devolvendo ao final dela, os seguintes parâmetros;

acabou : com valor verdadeiro;

itens : aumentado de 1;

bufações: contendo o número de novo item;

regtextos: endereço do último registro gravado no arquivo <DIÁLOGO> /TEXTOS.

A maneira pela qual o OUTROITEM recebe, do projetista, o número e o texto do novo item, é semelhante à que foi descrita para o módulo PRIMITEM.

Porém, os itens recebidos neste módulo, não poderão ser ítens fictícios, de acordo com o que dito durante a descrição do PRIMITEM.

O Módulo "WAUXAÇÕES"

Finalidade: a função deste módulo é a gravação, no arquivo auxiliar AUXAÇÕES, dos valores correspondentes aos endereços de gravação e outras informações do menu que está sendo descrito e que deverão ser salvas na tabela de endereços, ao final do arquivo <DIÁLOGO> /AÇÕES.

(A definição desta tabela de endereços consta na Secção 4.4.2, do capítulo IV; a justificativa para a utilização de um arquivo auxiliar foi dada na Seção 5.5.2, deste capítulo, na parte correspondente à descrição do módulo ENDEREÇOS).

Os valores que o módulo WAUXAÇÕES deverá gravar são recebidos em um buffer de caracteres tabações.

O Módulo "WAUXTEXTOS"

A descrição dada para o módulo WAUXAÇÕES é válida para este módulo, porém suas ações deverão ser referidas aos arquivos AUXTEXTOS e <DIÁLOGOS> /TEXTOS.

A seguir apresentaremos a descrição da implementação do módulo EXECUTAR, que não foi apresentada durante a descrição do módulo AÇÕES.

O Módulo "EXECUTAR"

Finalidade: a função deste módulo é receber, do projetista, os nomes das procedures de aplicação ou TCP's que deverão ser executadas, no caso das ações determinadas para um certo tipo de item fossem do tipo Executar ou Coletar.

O Executar recebe, como entrada, um buffer de caracteres bufações, contendo o número e o tipo de ação a ser executada pelo item que está sendo descrito.

Na sua execução, o EXECUTAR utiliza os módulos PROCAPL, PROCTCP, SEQPROC e NOVOMENU.

As funções de cada um destes módulos são as seguintes:

a) PROCAPL - receber do projetista o nome de uma procedure de aplicação;

b) PROCTCP - recebe do projetista o nome de uma TCP (tela de coleta de parâmetros) e, se for o caso, o nome da procedure de aplicação, que receberá os valores dos parâmetros coletados pela TCP e será executada logo a seguir (esta possibilidade foi descrita na Seção 4.3.4, do capítulo IV);

c) SEQPROC - receber do projetista os nomes de uma sequência de procedures de aplicação, com ou sem parâmetros;

d) NOVOMENU - receber do projetista o nome de um novo menu, que poderá ser apresentado após a execução de todas as procedures de aplicação ou TCP's.

O modo de execução do módulo EXECUTAR pode ser definido nos seguintes termos:

1) dependendo do tipo de ação que será executada pelo item em questão, a execução do módulo EXECUTAR apresenta duas opções:

   a) a ação do item é do tipo Executar - neste caso, será realizado o passo (2);

b) a ação do item é do tipo Coletar - neste caso, a execução deste módulo continuará com o passo (3);

2) chamada ao módulo PROCAPL - este módulo recebe como entrada o buffer bufações, contendo o número e o tipo da ação do item; após sua execução, o PROCAPL devolverá o mesmo buffer bufações, porém acrescido do nome da procedure de aplicação que deverá ser executada - em seguida, o módulo EXECUTAR realizará o passo (4);

3) chamada ao módulo PROCTCP - este módulo também recebe no buffer bufações, o número e o tipo de ação do item; após sua execução, o PROCTCP devolverá o bufações acrescido do nome da tela de coleta de parâmetros TCP ou, se for o caso, do nome dessa TCP, seguido do nome da procedure de aplicação que deverá ser executada, após a apresentação dessa TCP - esta possibilidade será esclarecida através de uma consulta ao projetista; em seguida o módulo EXECUTAR realizará o passo (4);

4) consulta ao projetista sobre a possibilidade de se ter outras procedures de aplicação que devam ser executadas como continuação das ações do item - se a resposta do projetista for afirmativa, o módulo EXECUTAR realizará o passo (5), caso contrário, será realizado o passo (6);

5) chamada ao módulo SEQPROC - este módulo recebe como entrada o buffer bufações, contendo todas as informações fornecidas até o momento pelo pro projetista, referentes as ações do item.

A execução do SEQPROC permitirá que o projetista possa fornecer os nomes da sequência de procedu res de aplicação e TCP's que deverão ser executa das como continuação das ações do item.

A descrição da utilização dessa sequência consta no capítulo IV, Seção 4.3.1.

Após a recepção do nome da nova procedure de a plicação, o SEQPROC consultará o projetista,atra vés da execução do módulo COMPARAM, sobre a ne cessidade de apresentar uma TCP que faça a cole ta de parâmetros para essa nova procedure de aplicação; se a resposta do projetista for afir mativa, o módulo COMPARAM receberá o nome dessa TCP.

As informações recebidas pelo SEQPROC serão adi cionadas ao buffer bufações e nele devolvidas ao módulo EXECUTAR; em seguida, o módulo EXECUTAR realizará o passo (4);

6) chamada ao módulo NOVOMENU - este módulo recebe rá, como entrada, o buffer bufações, contendo to das as informações recolhidas sobre o item que está sendo descrito; a execução deste módulo con sultará o projetista sobre o nome do menu do diá logo, que deve ser apresentado ao usuário após

terem sido executadas todas as ações deste item.

A resposta do projetista poderá ser de dois tipos:

a) o nome de um menu componente do diálogo - neste caso, o módulo NOVOMENU adicionará este nome ao buffer bufações.

b) uma resposta vazia (return) - neste caso o NOVOMENU entenderá que, após a execução das ações desse item não deverá ser apresentado nenhum outro menu - esta possibilidade significa que, depois da execução das ações correspondentes a este item, o menu ao qual elas pertencem, e com ele, o diálogo que estiver interagindo com o usuário serão encerrados.

Desta forma, o módulo NOVOMENU encerra sua execução devolvendo, ao módulo EXECUTAR, o buffer bufações atualizado.

5.2.5 - O Módulo "TCPDESC" Fig. (5.5)

Finalidade: a função deste módulo é receber do projetista as informações correspondentes a uma descrição do tipo TCP (tela para coleta de parâmetros) componente do diálogo.

Este módulo recebe os seguintes parâmetros de entra

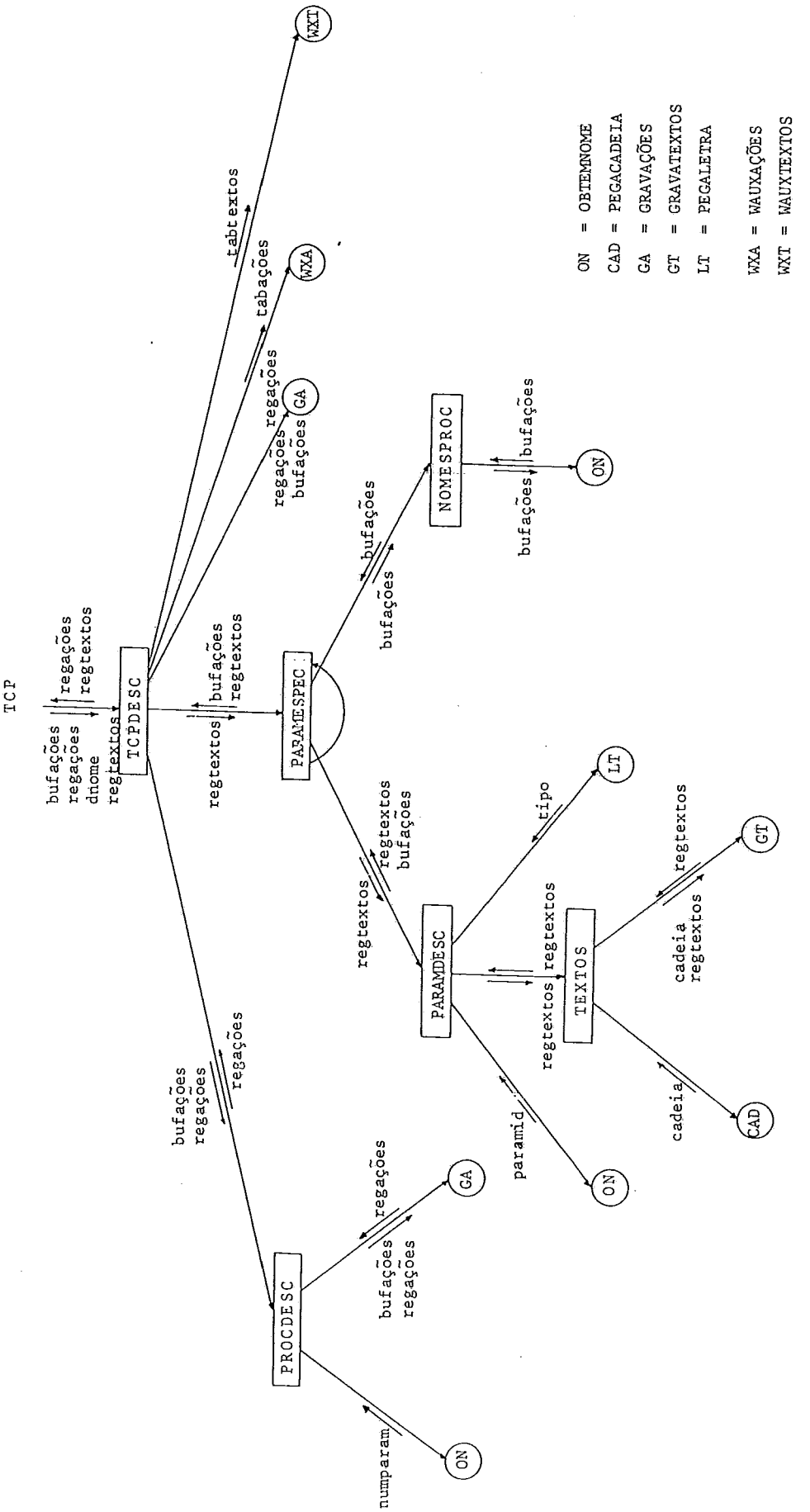


Fig. (5.5) Procedure CRIAR; Módulo TCPDESC

da:

regações : endereço do primeiro registro para gravação do arquivo <DIÁLOGO> /AÇÕES

dnome : o nome desta descrição

regtextos: endereço do primeiro registro disponível para gravação do arquivo <DIÁLOGO> /TEXTOS

Na execução do TCPDESC são chamados os módulos PROCDESC e PARAMESPC.

O módulo PROCDESC tem como função receber, do projetista, o número de parâmetros que integrarão a TCP. De acordo com o que foi colocado na Seção 4.3.2, do capítulo IV, este número de parâmetros será, no máximo, igual a 12.

O módulo PARAMESPC tem como função receber, do projetista, todas as informações correspondentes a cada parâmetro da TCP. Estas incluem: o nome e o tipo do parâmetro e os nomes das procedures de auxílio e crítica de erros para cada parâmetro (ver Seção 4.3.2, capítulo IV).

O modo de execução do módulo TCPDESC pode ser descrito nos seguintes termos;

1) chamada ao módulo PROCDESC - este módulo recebe, como entrada, o endereço de gravação regações e o buffer bufações, contendo o nome e o tipo da descrição; após ter recebido do projetista o número de parâmetros que integrarão a TCP, o PROCDESC procede à gravação destas informações no arquivo <DIÁLOGO> /AÇÕES.

2) chamada ao módulo PARAMESPC - este módulo recebe como entrada, o endereço de gravação para o arquivo <DIÁLOGO> /TEXTOS; Na execução deste módulo, serão gravados no arquivo <DIÁLOGO> /TEXTOS, os textos explicativos correspondentes a cada um dos parâmetros da TCP, e adicionados ao buffer bufações, o restante das informações correspondentes a esses parâmetros - após sua execução, o PARAMESPC devolverá os parâmetros bufações e reg textos, atualizados.

3) gravação das informações correspondentes a cada parâmetro da TCP, recebidas no buffer bufações. A seguir, o módulo TCPDESC gravará nos arquivos AUXAÇÕES, e AUXTEXTOS os valores correspondentes aos endereços de gravação e outras informações da TCP que está sendo descrita e que deverão ser armazenados na tabela de endereços, ao final dos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGOS> /TEXTOS - a maneira como estas gravações são realizadas, nos arquivos AUXAÇÕES e AUXTEXTOS, é a mesma que já foi descrita para o caso do módulo MENUDESC, na Seção 5.2.5, deste capítulo.

O Módulo "PARAMESPC"

Tal como foi dito, a função deste módulo é receber, do projetista, o nome e o tipo de cada parâmetro da TCP; os tipos de parâmetros aceitos pelo "G D" foram descritos na Seção 4.3.2, do capítulo IV.

Além dessas informações, o projetista deverá fornecer, também, os textos explicativos que deverão ser apresentados ao usuário para o preenchimento dos valores de cada parâmetro e os nomes das procedures de auxílio e crítica de erros.

No cumprimento desses objetivos, o módulo PARAMESPEC utiliza os módulos PARAMDESC e NOMESPROC.

O módulo PARAMDESC tem como função receber do projetista, o nome e o tipo de cada parâmetro da TCP, assim como os textos explicativos que deverão ser apresentados ao usuário para o preenchimento desses parâmetros.

O módulo NOMESPROC tem como função receber do projetista os nomes das procedures de crítica de erros e auxílio ao usuário para cada um desses parâmetros.

5.2.6 - Descrição dos Módulos que a Procedure CRIAR Utiliza como Interface

O Módulo "OBTEMNOME" Fig. (5.6)

Finalidade: este módulo tem como função apresentar ao projetista, uma mensagem e receber dele uma resposta. As respostas recebidas por este módulo podem ser de dois tipos: identificadores ALGOL ou números inteiros.

Parâmetros de entrada:

Mensagem: array de caracteres, contendo uma mensa

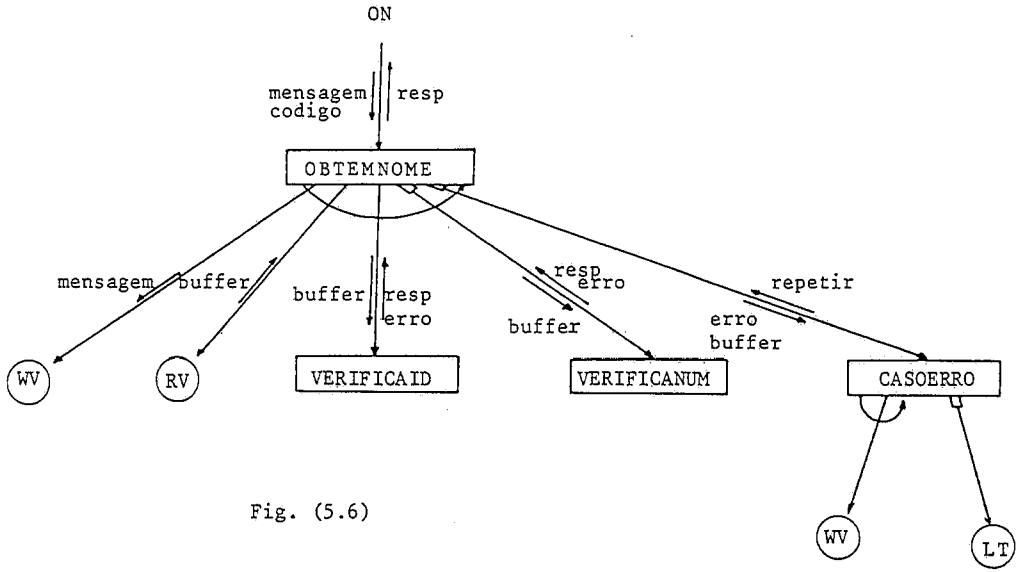


Fig. (5.6)

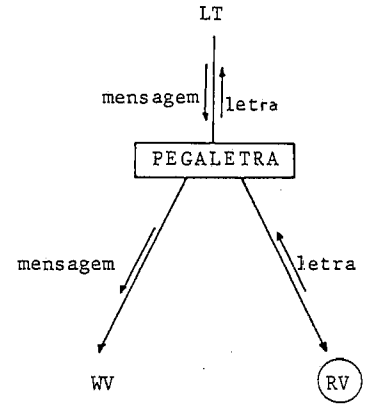


Fig. (5.7)

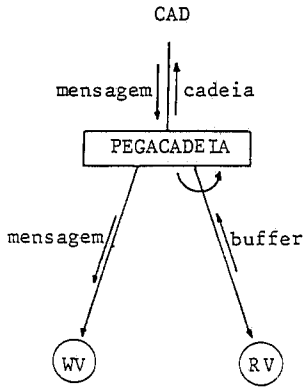


Fig. (5.8)

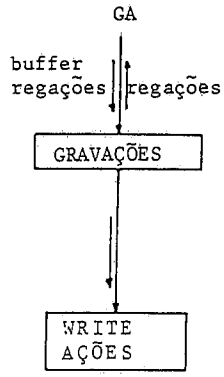


Fig. (5.9)

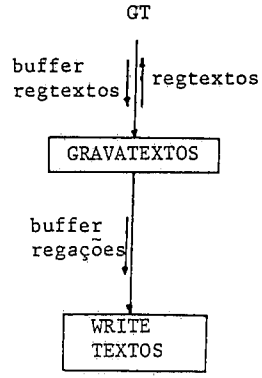


Fig. (5.10)

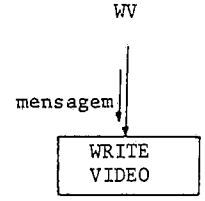


Fig. (5.11)

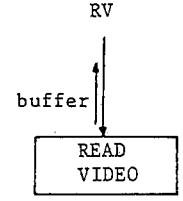


Fig. (5.12)

gem a ser apresentada ao projetista.

código: variável inteira cujo valor será utilizado para especificar o uso do módulo OBTEMNOME, que poderá ser para receber um identificador ALGOL ou um número inteiro.

Esta dualidade de uso do módulo OBTEMNOME foi utilizada visando um melhor aproveitamento da estrutura do software, utilizada na obtenção destes dois tipos de respostas do projetista.

O módulo OBTEMNOME inicia sua execução apresentando, no vídeo do projetista, um texto explicativo (o conteúdo do array mensagem).

A seguir, recebe, no array buffer, a resposta do projetista.

Dependendo do valor do parâmetro código o OBTEMNOME chamará o módulo VERIFICAID ou VERIFICANUM, segundo o caso.

O Módulo "VERIFICAID"

Finalidade: analisar a resposta do projetista em face da sintaxe de um identificador ALGOL, porém limitado a 12 caracteres.

A limitação de 12 caracteres para os identificadores aceitos pelo sistema "G D" foi colocada pelos seguintes motivos:

1) os identificadores fornecidos pelo projetista, juntamente com o resto de informações correspon

dentes às ações do diálogo, são gravadas no arquivo <DIÁLOGO> /AÇÕES, em forma modular.Cada um destes módulos tem 6 caracteres, que correspondem a uma palavra da memória do B-6700. Tal gravação modular permite que este arquivo possa ser facilmente tratado pelo editor de textos do B-6700, nas possíveis futuras atualizações do diálogo;

2) uma alternativa para o comprimento dos identificadores aceitos pelo sistema "G D" poderia ter sido 6 caracteres, porém a consideramos demasiado restrita, ou 18 ou mais caracteres, a que consideramos demasiado custosa.

A interação do módulo VERIFICAID realiza-se através dos seguintes parâmetros:

Parâmetros de entrada:

buffer : cadeia de caracteres fornecida pelo projetista.

Parâmetros de saída:

resp : array de caracteres, contendo o primeiro subconjunto de cadeia de caracteres, que satisfaça a sintaxe estabelecida para um identificador.

erro : variável inteira, necessária à codificação de algum tipo de erro encontrado.

O Módulo "VERIFICANUM"

Tudo o que foi descrito acima, para o módulo VERIFICAID, é válido para o VERIFICANUM, diferenciando-se, somente, a análise realizada com a cadeia de caracteres, que no caso do VERIFICANUM, acompanha a sintaxe estabelecida para um número inteiro.

Após a execução dos módulos VERIFICAID ou VERIFICANUM, conforme o caso, o módulo "OBTEMNOME" apresenta duas opções na sua execução:

a) Não aconteceram erros (ERRO = 0)

Nesta opção, o OBTEMNOME encerra sua execução, devolvendo, no resp, a cadeia analisada.

b) Aconteram erros (ERRO > 0)

Nesta opção, o OBTEMNOME executa o módulo CASO ERRO.

O Módulo "CASOERRO"

A função do módulo CASEERRO é apresentar, ao projetista, a cadeia de caracteres analisada, acompanhada de uma mensagem explicativa, que corresponda ao tipo de erro detectado.

Este módulo recebe como entrada:

buffer : array de caracteres, contendo a resposta incorreta fornecida previamente pelo projetista;

erro : variável inteira que codifica o tipo de erro detetado.

Dependendo do tipo de erro recebido em erro, a execução do CASOERRO apresenta duas opções:

a) o tipo de erro indica que, na resposta do projetista, existe uma parte que poderia ser uma resposta válida.
Nesse caso, o CASOERRO apresenta, ao projetista, o subconjunto da resposta que poderia ser considerado válido. Diante dessa opção, o projetista poderá responder aceitando ou rejeitando a proposta do sistema.
Tal designação do projetista será registrada em uma variável booleana repetir, que será colocada como verdadeira, no caso de uma resposta negativa do projetista.

B) o tipo de erro indica que a resposta do projetista foi considerada totalmente inválida.
Neste caso, o CASOERRO simplesmente apresentará, ao projetista, a resposta incorreta e devolverá ao OBTEMNOME a variável booleana repetir, com valor verdadeiro.

Dependendo do conteúdo da variável repetir o módulo OBTEMNOME repetirá todo o processo, até que não aconteçam erros, ou até que repetir seja devolvida com valor "falso".

O Módulo "PEGALETRA" Fig. (5.7)

A função do PEGALETRA é apresentar, no vídeo do projetista, uma mensagem explicativa e receber, como resposta, um

caracter alfanumérico.

Ex.: S, N (SIM ou NÃO)

M, T (MENU ou TCP)

I, R, S, B (INTEIRO, REAL, STRING ou BOOLEANA)

Parâmetro de entrada:

mensagem : array de caracteres

Parâmetro de saída:

letra : um caracter alfanumérico

O Módulo "PEGACADEIA" Fig. (5.8)

A função do PEGACADEIA é apresentar no vídeo uma mensagem explicativa, que permite ao projetista fornecer uma cadeia de caracteres, que será utilizada como texto.

Esta cadeia é recebida através de repetidas leituras, que utilizam um buffer de 72 (setenta e dois) caracteres, e é devolvida no parâmetro cadeia.

Parâmetro de entrada:

mensagem : array de caracteres

Parâmetro de saída:

cadeia : array de caracteres

O Módulo "GRAVAÇÕES" Fig. (5.9)

A função deste módulo é receber um buffer, contendo

um array de caracteres e gravá-los a partir do primeiro registro disponível para garvação do arquivo <DIÁLOGO> /AÇÕES.

Parâmetros de entrada:

buffer : array de caracteres, contendo informações relacionada com as descrições que estão sendo recebidas.

regações: endereço do primeiro registro do arquivo <DIÁLOGO> /AÇÕES, disponível para a gravação do buffer.

Parâmetros de saída:

regações: endereço do último registro utilizado na gravação do buffer, no arquivo <DIÁLOGO> /TEXTOS.

5.3 - A implementação do GERADOR

Esta procedure é a que realiza a tarefa de geração do código fonte ALGOL. O fato de gerar código fonte requer que uma grande quantidade de detalhes de implementação seja levada em conta.

Entretanto, trataremos, enquanto for possível,de afastar o máximo que podermos aqueles detalhes, procurando sempre manter clara a idéia da Geração do código fonte.

5.3.1 - Definição dos módulos principais Fig. (5.13)

1) Como foi visto no Capítulo IV, secção 4.3, o

"G D" basicamente interpreta dois tipos de descrições: o menu e a tela de coleta de parâmetros (TCP).

Isto nos leva a criar, dentro do GERADOR dois grandes módulos, que gerarão código fonte, tanto para menus como para TCP. TAis módulos são os GERARMENU e GERARTCP.

2) Na definição do "G D", capítulo IV secção 4.4.5, foi visto também, que um diálogo para ser gerado precisa, além das procedures dos menus e TCPs, de uma procedure que se encarregue de executar o referido diálogo. Para gerar tal procedure, o GERADOR inclui um outro módulo, ao qual chamaremos de GERADIÁLOGO.

3) As descrições dos menus e das TCPs encontram-se armazenadas nos arquivos, <DIRETORIO> /AÇÕES e <DIRETÓRIO> /TEXTOS.

Para conseguir-se ler aquelas descrições, é preciso conhecer os seus endereços de gravação nesses arquivos, tais endereços encontram-se nas tabelas, ao final de cada arquivo; (secção 4.4. 2, capítulo IV).

Uma maneira de utilizar essas tabelas seria lê-las, no disco, cada vez que fosse necessário, porém, devido à grande diferença entre o tempo de acesso ao disco e um fetch da memória, consideramos conveniente manter na memória as chaves e os endereços das tuplas dessas tabelas. Para a rea

lização desse processo e de outras tarefas prévias à geração de código, o gerador inclui o módulo CARREGA.

4) Conforme os módulos do GERADOR forem gerando código fonte, este código será armazenado como um array de caracteres na memória, no entanto, para que ele possa ser utilizado, o GERADOR utiliza um módulo que escreve aquele código (de maneira ALGOL-LIKE) num arquivo em disco. Este módulo o denominaremos de GRAVAFONTE.

5) Finalmente, para encerrar sua execução, o GERADOR utiliza um outro módulo - FIMARQUIVOS - que salvará definitivamente no disco, os arquivos que tenham sido criados para receber o código fonte, resultado da execução do GERADOR.

5.3.2 - O Funcionamento Básico do GERADOR

1) O GERADOR interage com o usuário para conhecer o nome do diálogo que deverá ser gerado. Este nome permite que os arquivos nome/AÇÕES e nome/TEXTOS, nos quais estão gravadas as descrições deste diálogo, sejam lidos. Nesta apresentação, tais arquivos serão chamados de <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.

2) O GERADOR carrega, na memória, as palavras reservadas, os nomes das procedures que serão utiliza

das pelo diálogo, assim como os endereços de gravação dos menus e TCPs. Isto é feito através da execução do módulo CARREGA.

3) A seguir, o GERADOR traz para a memória as características do diálogo que está sendo gerado. Isto é feito através de uma chamada ao módulo PEGAINFO. (Este módulo será descrito com detalhes na Seção 5.3.5).

4) Uma vez que tais características encontrem-se na memória, o GERADOR executa o módulo GERADIÁLOGO, que produzirá o código fonte, da procedure que será utilizada pelo usuário para executar a interface que está sendo gerada. (Seção 4.4.5). Antes de finalizar o GERADIÁLOGO chamará novamente PEGAINFO, o qual colocará, na memória, a primeira descrição do diálogo que tiver sido gravada no arquivo <DIÁLOGO> /AÇÕES.

5) Dependendo do tipo de descrição que o PEGAINFO houver colocado na memória, o GERADOR poderá executar os módulos GERARMENU ou GERARTCP, que produzirão o código fonte correspondente a uma descrição de tipo menu ou TCP, segundo o caso. Contudo, estes dois módulos, antes de finalizar sua execução, chamarão o módulo PEGAINFO. Tal chamada colocará, na memória, as informações correspondentes à uma nova descrição (menu ou TCP), do arquivo <DIÁLOGO> /AÇÕES.

6) A chamada ao módulo PEGAINFO pode devolver duas opções:

   a) as informações correspondentes a uma nova descrição encontrada no arquivo <DIÁLOGO> /AÇÕES neste caso, o GERADOR realizará o passo 5;

   b) a execução do PEGAINFO devolve uma marca de fim de arquivo, o que indica que não existem outras descrições. Neste caso, o GERADOR realizará o passo 7.

7) Execução do módulo FIMARQUIVOS, que armazenará, no disco, todo o código fonte que houver sido gerado.

5.3.3 - Verificação de Erros

O funcionamento do GERADOR tal como foi descrito,só é válido para uma sequência de descrições (armazenadas nos arquivos <DIÁLOGO> /AÇÕES E <DIÁLOGO> /TEXTOS) que corresponda, às solicitações do "G D" definidas no capítulo IV.

Qualquer descrição, diferente daquela prevista pelo GERADOR, ocasionará nele uma condição de erro.

1) Erros Sintáticos

   As informações contidas nos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS estão, teoricamente, livres de erros sintáticos. No entanto, eles estão preparados para sofrer alterações ou modifi

cações, através do simples uso do editor do B-6700, (capítulo IV, secção 4.5.3).

Dependendo das ações manuais do projetista, tais modificações poderão ou não, acompanhar a sintaxe prevista pelo "G D".

Considerando essa nova circunstância, o GERADOR antes de aceitar qualquer descrição, realiza através do módulo ANALISADOR, toda a análise necessária para garantir uma sequência de descrições livre de erros sintáticos.

2) Erros Semânticos

Para a detecção dos erros semânticos, o GERADOR inclui múltiplos testes. Estas verificações, pela natureza "sensivel ao contexto" do erro semântico, não estão agrupadas num único módulo, mas sim repartidas, ao longo de todo o GERADOR.

Tanto os erros sintáticos como os erros semânticos produzem, no vídeo do projetista, a respectiva mensagem de erro, que ajuda-lo-á no conserto de algum registro daquela descrição.

Depois da apresentação das mensagens de erro, o GERADOR suspende sua execução e retira da memória e do disco, todo o código fonte gerado até a ocorrência de tais erros.

A quantidade de erros que o GERADOR conseguir detectar, antes da suspensão da sua execução, é variável e depende do contexto de onde os erros aconteceram.

5.3.4 - Descrição dos Módulos do GERADOR

O Módulo CARREGA Fig.(5.13)

Finalidade : a função desse módulo é trazer, para a memória, as informações que o GERADOR precisa para começar sua execução. Estas informações incluem:

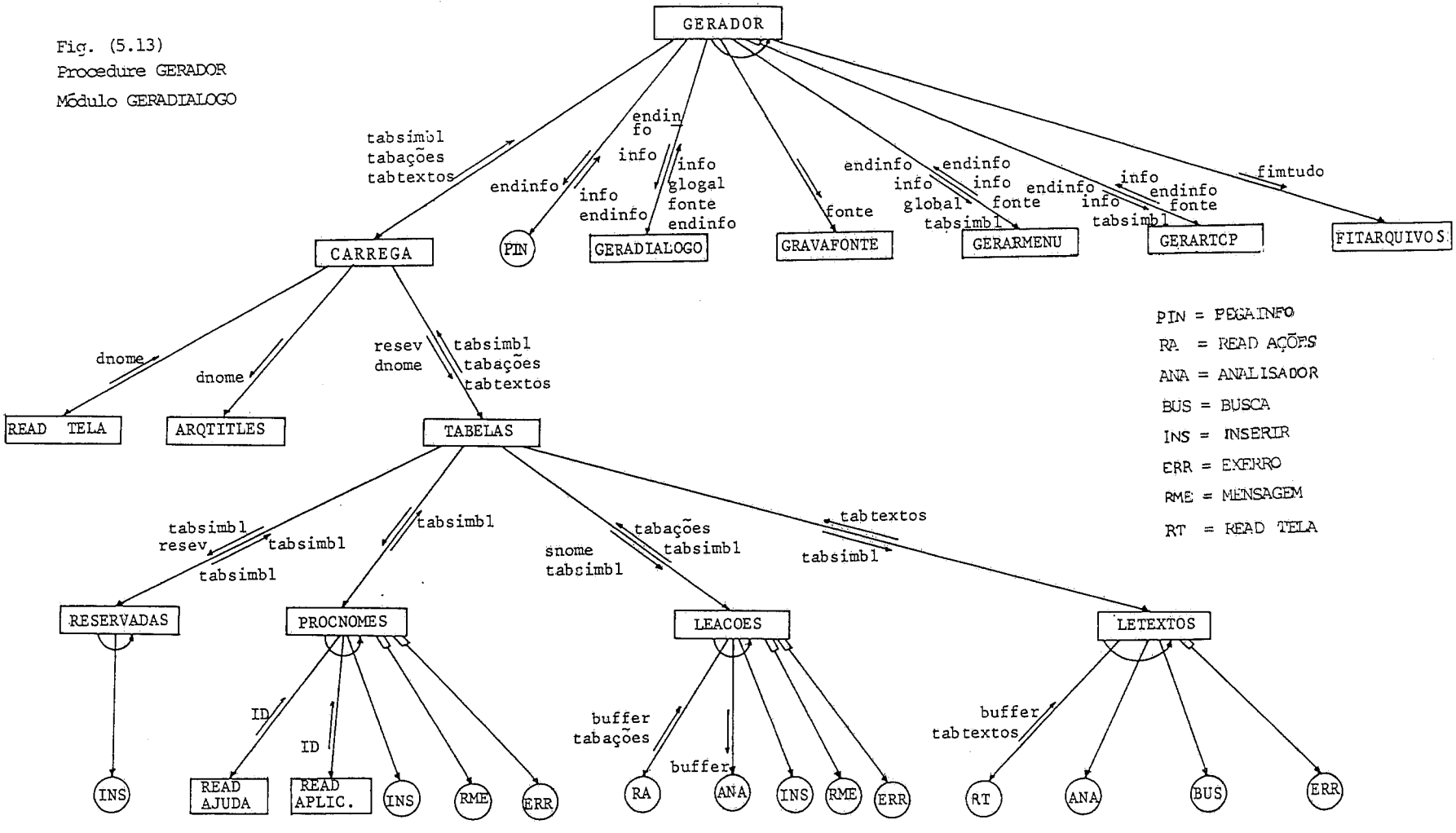
- o nome do diálogo que vai ser interpretado;
- os nomes das palavras reservadas do ALGOL;
- os nomes das procedures de auxílio, crítica de erros, aplicação,telas de tipo menu e das TCP;
- os endereços de gravação dos menus e das TCP nos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.

A execução do CARREGA é feita através dos módulos ARQTITLES e TABELAS e pode ser descrita nos seguintes termos:

1) o CARREGA interage com o projetista, para conseguir o nome do diálogo que deverá ser gerado;

2) chamada ao módulo ARQTITLES - a execução deste módulo produzirá uma associação física dos nomes dos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO>/TEXTOS, com os arquivos nome /AÇÕES e nome /TEXTOS,que deverão encontrar-se no disco;

Fig. (5.13)
Procedure GERADOR
Módulo GERADIALOGO



PIN = PEGAINFO
RA = READ AÇÕES
ANA = ANALISADOR
BUS = BUSCA
INS = INSERIR
ERR = EXERRO
RME = MENSAGEM
RT = READ TELA

3) chamada ao módulo TABELAS - a execução deste módulo colocará, na memória, os nomes das palavras reservadas e das procedures que serão utilizadas pelo diálogo, assim como os endereços de gravação dos menus e TCPs (componentes de diálogo) nos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.
O módulo TABELAS, para realizar suas funções,utiliza os módulos RESERVADAS, PROCNOMES, LEAÇÕES e LETEXTOS.
O módulo RESERVADAS carrega na memória, no caso um array real tabsimbl, as palavras reservadas do ALGOL. Estas palavras reservadas são recebidas em um array de caracteres predefinido no GERADOR (value array resev). Tais palavras reservadas são carregadas no tabsimbl através da execução do módulo INSERIR, que está descrito na Seção 5.3.5, deste capítulo.

O módulo PROCNOMES carrega na memória (tabsimbl) os nomes das procedures de auxílio, crítica de erros e aplicação, que serão utilizadas pelo diálogo.
Para isto, o PROCNOMES lê os arquivos <DIÁLOGO> /AJUDA e <DIÁLOGO> /APLICAÇÃO (estes arquivos foram descritos na Seção 4.4.4, do capítulo IV). Os nomes destas procedures também são carregados no tabsimbl, através da execução do INSERIR. No caso de algum desses nomes não ser válido, o projetista será alertado através da execução dos módulos MENSAGEM e EXERRO, descritos na Seção 5.3.5,

deste capítulo.

O módulo LEAÇÕES carrega na memória (no caso, um array real tabaçÕes), os endereços de gravação dos menus e TCPs dos diálogos, assim como, no tabsimbl, os nomes desses menus e TCPs. O módulo LEAÇÕES obtém tais informações através de repetidas leituras da tabela de endereços, gravada no final do arquivo DIÁLOGO /AÇÕES (esta tabela de endereços foi descrita na Seção 4.4.2, do capítulo IV). É possível, porém,que as informações contidas nessa tabela de endereços não estejam sintaticamente corretas e, por este motivo, o LEAÇÕES realiza, para cada tupla dessa tabela, uma análise sintática através da execução do módulo ANALISADOR (este módulo está descrito na Seção 5.3.5). Após tal análise ter sido realizada, chaves e endereços de cada uma dessas tuplas, são carregados no array tabaçÕes; os nomes dos menus e das TCPs são carregados no tabsimbl através do módulo INSERIR. Em caso de erros, o projetista será alertado através da execução dos módulos MENSAGEM e EXERRO.

O módulo LETEXTOS tem funções similares às do LEAÇÕES, no entanto dizendo respeito ao arquivo <DIÁLOGO> /TEXTOS.

Existe, contudo, uma diferença que deve ser destacada no caso do LETEXTOS, os nomes dos menus e das TCPs encontrados no arquivo <DIÁLOGO>/TEXTOS

já não poderão ser carregados no tabsimbl, pois isto já foi realizado pelo LEAÇÕES, por este motivo, a ação semântica que o LETEXTOS realiza é a procura de cada um desses nomes no array tabsimbl. Esta procura é feita através da execução do módulo BUSCA. Em caso de erros, o projetista será alertado através da execução dos módulos MENSAGEM e EXERRO.

Até aqui, temos definido a forma na qual se procede a execução do CARREGA. Esta descrição não inclui o detalhe dos módulos: INSERIR, MENSAGEM, EXERRO e ANALISADOR.

Tais módulos são utilizados repetidamente ao longo do GERADOR. Por essa razão, as suas respectivas descrições podem ser separadas do contexto da sua utilização. As mesmas estão presentes na secção 5.3.5 deste capítulo.

O Módulo GERADIÁLOGO (Fig. 5.14)

Finalidade: a função deste módulo é a produção do código fonte da procedure que será utilizada pelo usuário para executar o diálogo que está sendo gerado.Esta procedure foi descrita na Seção 4.4.5, do capítulo IV.

O GERADIÁLOGO recebe como parâmetros de entrada:

info : contendo as características do diálogo que está sendo gerado;

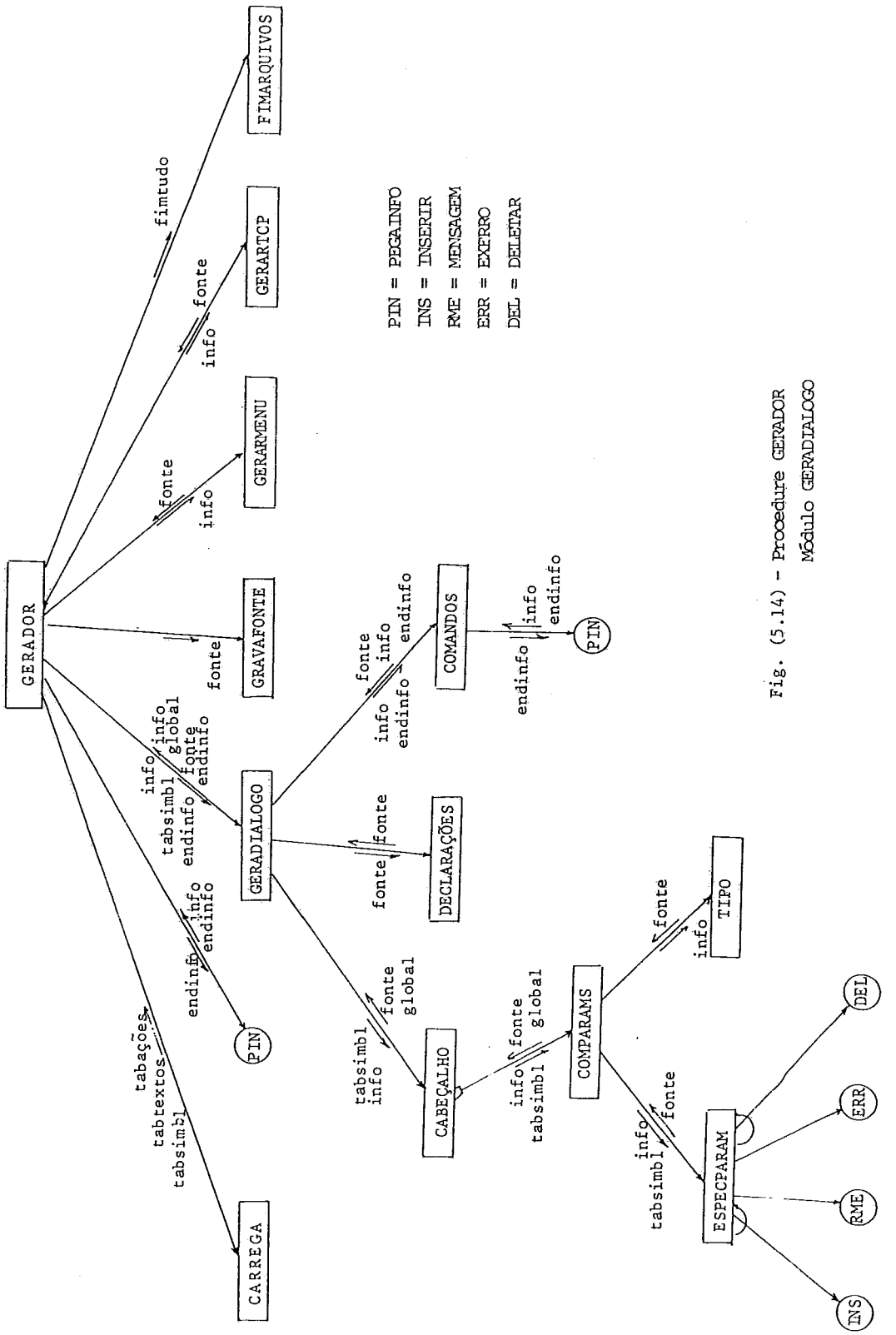


Fig. (5.14) - Procedure GERADOR
Módulo GERADIALOGO

tabsimbl : contendo os identificadores armazenados pelo módulo CARREGA.

O GERADIÁLOGO utiliza os seguintes módulos: CABEÇALHO, DECLARAÇÕES e COMANDOS.

A forma de execução do GERADIÁLOGO pode ser descrita nos seguintes termos:

1) chamada ao módulo CABEÇALHO - esta chamada gerará o código fonte correspondente ao cabeçalho da procedure - Ex.:

```
PROCEDURE DIÁLOGO (ITEM, VALOR, QUANTIDADE);
REAL ITEM;
INTEGER VALOR, QUANTIDADE;
```

2) chamada ao módulo DECLARAÇÕES - esta chamada produzirá o código fonte correspondente às declarações da procedure. Tais declarações constam na descrição realizada na Seção 4.4.5, do capítulo IV;

3) chamada ao COMANDOS - esta chamada gerará o código fonte correspondente aos comandos da procedure; estes comandos também estão descritos na Seção 4.4.5, acima mencionada.

Conforme cada um dos módulos - CABEÇALHO, DECLARAÇÕES e COMANDOS - vão gerando o código fonte, esse código vai sendo carregado em um array de caracteres fonte, que o GERADIÁLOGO enviará de volta para o GERADOR.

A execução do módulo CABEÇALHO apresenta duas opções:

a) o diálogo que está sendo gerado não tem parâmetros;

Ex.: PROCEDURE DIALOG2;

b) O diálogo que está sendo gerado tem parâmetros - neste caso o CABEÇALHO chama o módulo COMPARAMS, para que ele faça a geração do código fonte do cabeçalho da procedure.

O módulo COMPARAMS utiliza dois módulos: o ESPEC PARAM, que realiza as verificações correspondentes à validade dos nomes de cada parâmetro e produz o código fonte da declaração da procedure - Ex.:

PRODEDURE DIÁLOGO3 (ITEM, VALOR, QUANTIDADE);

e o módulo TIPO, que realiza a geração do código fonte, da especificação do tipo de cada parâmetro - Ex.:

INTEGER VALOR, QUANTIDADE;

REAL ITEM;

As verificações feitas pelo ESPECPARAM são realizadas comparando-se os nomes de cada parâmetro com os identificadores armazenados no array tabsimbl - isto é feito através dos módulos INSERIR e DELETAR, que são descritos na Seção 5.3.5. No caso de erros, o projetista receberá as respectivas mensagens através da execução dos módu

los MENSAGEM e EXERRO.

Após sua execução, o módulo COMPARAMS devolverá, em um array de caracteres global, a lista dos parâmetros do diálogo.

Na execução do módulo COMANDOS, faz-se necessário conhecer o nome do primeiro menu do diálogo que está sendo gerado (tal como pode ser visto na apresentação feita na Seção 4.4.5, do capítulo IV). Por tal motivo, o módulo COMANDOS chama o PEGAINFO - esta chamada substituirá as informações contidas no array info pelas que correspondem àquele primeiro menu.

Com o término da execução do módulo COMANDOS, o GERADIALOGO também termina sua execução, devolvendo ao GERADOR, no array fonte,o código gerado, e nos arrays info e endinfo, a informação de uma nova descrição.

O Módulo GRAVAFONTE Fig.(5.15)

Parâmetros de entrada:

Fonte : contendo o código fonte, gerado por algum dos módulos do GERADOR

A função do módulo GRAVAFONTE é gravar num arquivo em disco, a cadeia de caracteres recebida no array fonte.

O GRAVAFONTE realiza esta gravação de maneira que, o resultado dela, é um arquivo contendo o código fonte ALGOL - like ao que estamos acostumados.

No processo de gravação, o número máximo de caracteres gravados em cada registro é 72.

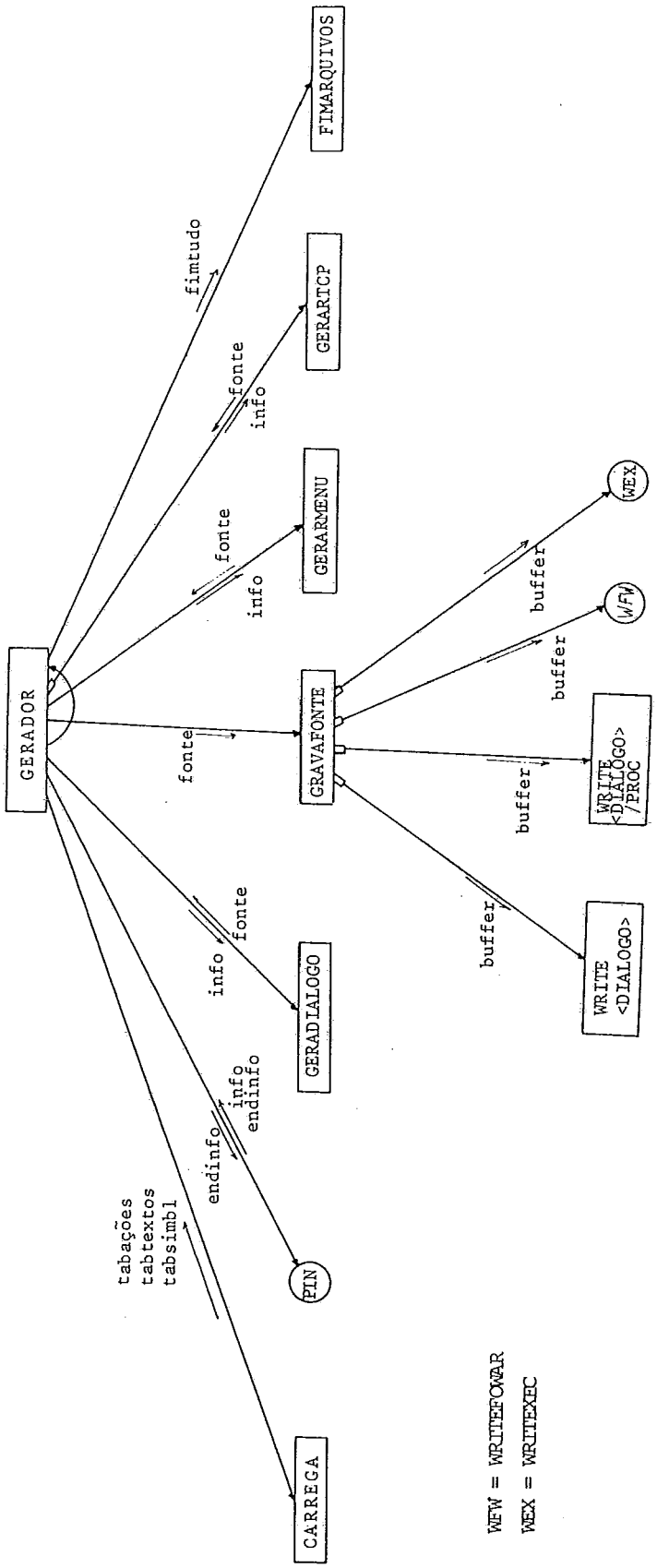


Fig. (5.15) - Procedure GERADOR
Módulo GRAVAFONTE

A execução do módulo GRAVAFONTE apresenta duas opções:

1) O código fonte recebido pertence à procedure que executará a interface que esta sendo gerada:
   Nesse caso, o conteúdo do array info será gravado em um arquivo em disco, de nome <DIÁLOGO>, onde <DIÁLOGO> é o nome do diálogo que está sendo gerado;
   Também é gravado num arquivo em disco, de nome <DIÁLOGO> /EXEC, o cabeçalho da procedure EXEC (descrita no capítulo IV, seção 4.4.5).

2) O código fonte recebido pertence à descrição de um menu ou de uma TCP.
   Neste caso, o conteúdo do array info será gravado no primeiro registro não utilizado, do arquivo <DIÁLOGO> /PROC. .
   Será também gravada, no arquivo <DIÁLOGO>/FOWAR, a declaração FORWARD, correspondente à descrição que esteja sendo gravada. (seção 4.4.5 do capítulo IV).
   Se o código fonte que está sendo gravado pertencer a um menu, o GRAVAFONTE gravará, também, no arquivo <DIÁLOGO>/EXEC a chamada correspondente aquele menu.

O Módulo GERARMENU Fig. (5.16)

Finalidade: a função deste módulo é produzir o códi

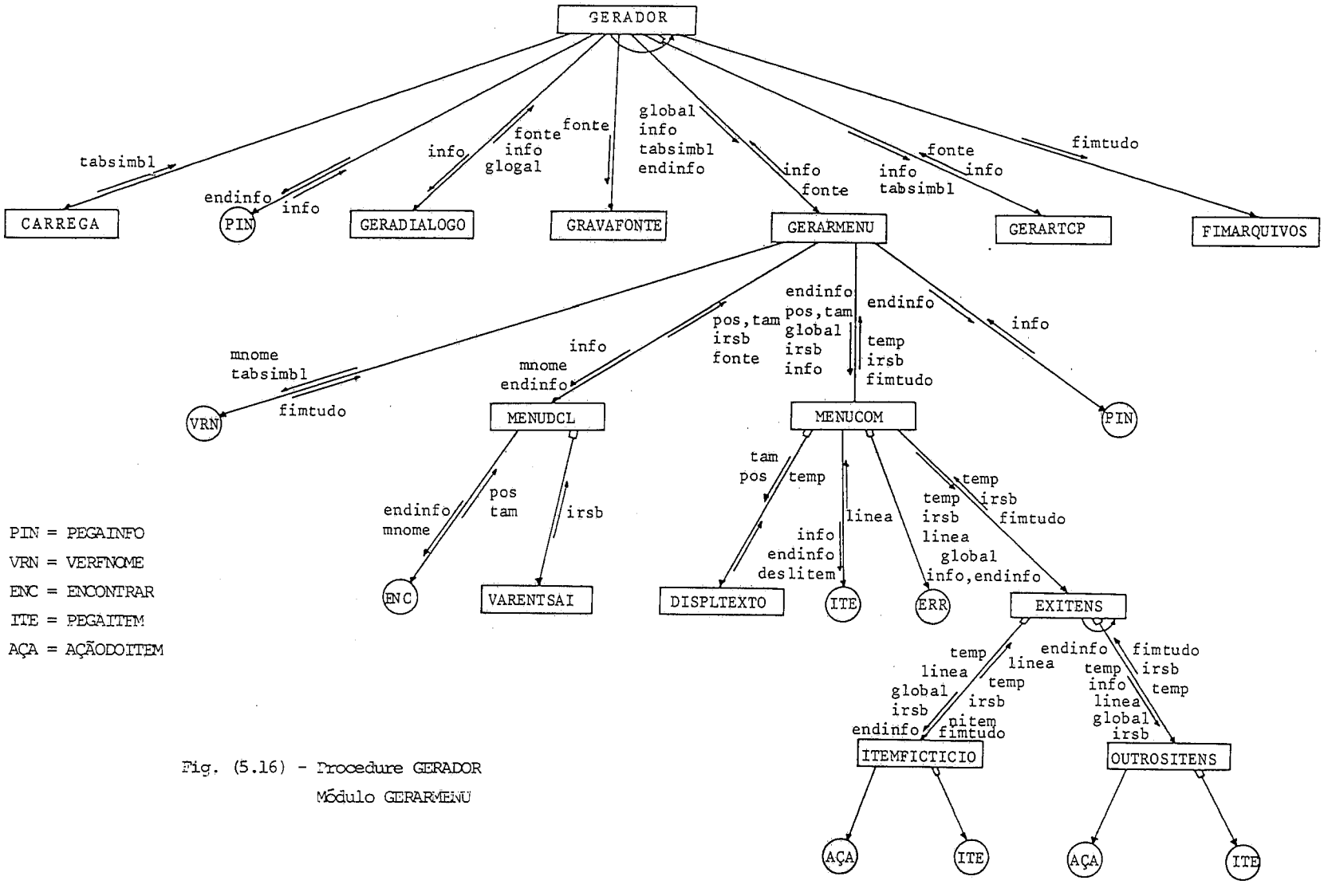


Fig. (5.16) - Procedure GERADOR

Módulo GERARMENU

go fonte correspondente a uma descrição de tipo menu.

Um modelo do código gerado pelo GERARMENU foi apresentado na Seção 4.4.5 do capítulo IV. Outros exemplos podem ser vistos no Apêndice I.

O GERARMENU recebe os seguintes parâmetros:

info : contendo a descrição desse menu, lida do arquivo <DIÁLOGO>/AÇÕES;

global : contendo os parâmetros do diálogo que está sendo gerado (se houver);

tabsimbl : contendo os identificadores que estiverem sendo utilizados nesse diálogo;

endinfo : contendo os endereços de gravação desse menu. Estes endereços são lidos pelo PEGAINFO das tabelas de endereços,no final dos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.

O GERARMENU utiliza os módulos MENUDCL e MENUCOM.

O funcionamento do GERARMENU pode ser descrito nos seguintes termos:

1) verificação da validade do nome do menu. Isto é feito através da execução do módulo VERFNOME,que é descrito na Seção 5.3.5;

2) geração do código fonte das declarações da procedure que corresponde ao menu que está sendo in

terpretado. Este código é obtido através do módulo MENUDCL;

3) geração dos comandos dessa procedure - isto se produz através da execução do módulo MENUCOM;

4) chamada ao módulo PEGAINFO, que colocará, na memória, as informações correspondentes à descrição seguinte, encontrada no arquivo <DIÁLOGO>/AÇÕES.

A execução do módulo MENUDCL apresenta duas opções:

a) o menu que está sendo gerado não tem textos, o que significa que aquele menu tem somente um item fictício - Ex.: um menu cuja única ação seja realizar uma determinada chamada para uma tela de coleta de parâmetros; (o item fictício está descrito na Seção 4.3.1, do capítulo IV);

b) o menu que está sendo gerado tem textos que devem ser apresentados ao usuário (o menu tem, pelo menos, um item que não é fictício).

A presença ou ausência dos textos do menu significa, para o módulo MENUDCL, a geração ou não geração das variáveis que deverão ser utilizadas para a apresentação de tais textos.

A existência desses textos é verificada através da execução do módulo ENCONTRAR, que devolverá em pos o endereço

de gravação e em tam o número de registros do texto daquele menu no arquivo <DIÁLOGO> /TEXTOS.

Se tam for maior que zero, MENUDCL chamará ao módulo VARENTSAI que gerará o código das declarações dessas variáveis. Este código fonte será carregado no array irsb.

Exemplos:

a) Código gerado pelo MENUDCL para um menu que não apresenta textos:

```
no array fonte : procedure MENU1 ;
                          BEGIN

no array irsb  : INTEGER
                 REAL
                 EBCDIC ARRAY
                 BOOLEAN
```

b) Código gerado para um menu que apresenta textos:

```
no array fonte : PROCEDURE MENU1 ;
                          BEGIN
                           POINTER PT;

no array irsb  : REAL ARRAY RESPOSTA [0:12];
                 INTEGER,POS,TAM,POSREL,DIGITO,
                 EBCDIC ARRAY BUFFER [0:1120]
                              CADEIA
                 REAL
                 BOOLEAN
```

O código fonte obtido pela execução de MENUDCL, será devolvido ao GERARMENU nos array fonte irsb.

Tal como foi dito, a finalidade do módulo MENUCOM é produzir o código fonte correspondente aos comandos da procedure que corresponde ao menu. Para isto, o MENUCOM utiliza os módulos DISPLTEXTO e EXITENS e recebe os seguintes parâmetros:

endinfo e info: contendo os endereços e a descrição do menu que está sendo interpretado;

pos e tam : contendo a posição e o número de registros do texto desse menu no arquivo <DIÁLOGO> /TEXTOS;

global : contendo os nomes dos parâmetros do diálogo (se houver);

irsb : contendo o código fonte das declarações das variáveis, gerado pela execução do módulo MENUDCL.

A execução do módulo MENUCOM pode ser descrita como segue:

1) no caso do menu que está sendo gerado tiver textos, o MENUCOM executará o módulo DISPLTEXTO, que produzirá o código fonte correspondente aos comandos que apresentarão tais textos ao usuário em caso contrário, o MENUCOM começará sua execução pelo passo 2.
2) chamada ao módulo PEGAITEM (descrito na Seção 5.3.5), esta chamada colocará, em um array linea,

as informações correspondentes ao primeiro item do menu. Estas informações são obtidas do array info e são necessárias, para facilitar a geração do código fonte dos comandos que realizam as ações de cada item. No caso do primeiro item recebido no linea não ser item fictício e, ao mesmo tempo, o menu não tiver textos, o projetista será alertado da inconsistência, através do módulo EXERRO;

3) chamada ao módulo EXITENS - esta chamada produzirá o código fonte correspondente aos comandos que deverão ser realizados, caso esse primeiro item for o escolhido pelo usuário, ou, no caso dele ser um item fictício. (um item fictício é executado antes do usuário fazer a escolha de qualquer outro item do menu - capítulo IV, Seção 4.3.1).

O código fonte gerado, tanto pelo DISPLTEXTO como pelo EXITENS, é devolvido em um outro array de caracteres temp.

A execução do módulo EXITENS é feita através dos módulos ITEMFICTÍCIO E OUTROSITENS. Esta execução pode ser descrita nos seguintes termos:

1) o EXITENS analisa as informações contidas no linea; no caso delas pertencerem a um item fictício, o EXITENS realizará o passo 2 - caso contrário, será realizado o passo 3;

2) chamada ao módulo ITEMFICTICIO - esta chamada produzirá o código fonte dos comandos que realizem as ações especificadas para esse item - a continuação do EXITENS realizará o passo 4;

3) o EXITENS gera o código fonte para os comandos de leitura da resposta do usuário e executa o módulo OUTROSITENS, que produzirá o código fonte para os comandos especificados como ações do primeiro item do menu - em seguida, será realizado o passo 4;

4) o EXITENS chama repetidamente, o módulo OUTROSITENS, até que sejam esgotados todos os ítens do menu.

Tanto os módulos ITEMFICTÍCIO como o OUTROSITENS, utilizam, para sua execução, os módulos AÇÃODOITEM e PEGAITEM, que encontram-se descritos na Seção 5.3.5.

Os parâmetros de entrada dos módulos ITEMFICTICIO e OUTROITENS são os mesmos que foram descritos para o módulo EXITENS. Os resultados da execução desses módulos são devolvidos nos arrays temp e irsb.

Na execução do módulo AÇÃODOITEM, é possível que sejam encontrados erros - neste caso, uma variável booleana fimtudo será colocada como verdadeira e devolvida ao módulo EXITENS.

Como pode ser observado, o GERARMENU recebe, até o término da execução do MENUCOM, três arrays, cada um deles con

tendo código fonte gerado.

- array fonte : código para o cabeçalho e primeiras declarações da procedure.

- array irsb : código para outras declarações

- array temp : código para o bloco de comandos

Tal separação foi necessária pela dificuldade em se conhecer no momento da geração do código fonte das declarações da procedure o nome e o tipo de cada parâmetro que aparece nas chamadas de procedures de aplicação e TCP's, contidas nas ações de cada item.

No momento me que o MENUCON termina sua execução, o GERARMENU reune, no array fonte, todo o código recebido, nesses três arrays.

No final de sua execução, o GERARMENU devolverá ao GERADOR:

fonte : código fonte gerado para o menu

info : atualizado, com as informações de uma nova descrição

eof : caso não existam novas descrições

fimtudo : no caso de erros, durante a execução de GERARMENU

O Módulo GERARTCP Fig. (5.17)

Finalidade: a função deste módulo é produzir o código fonte correspondente a uma descri

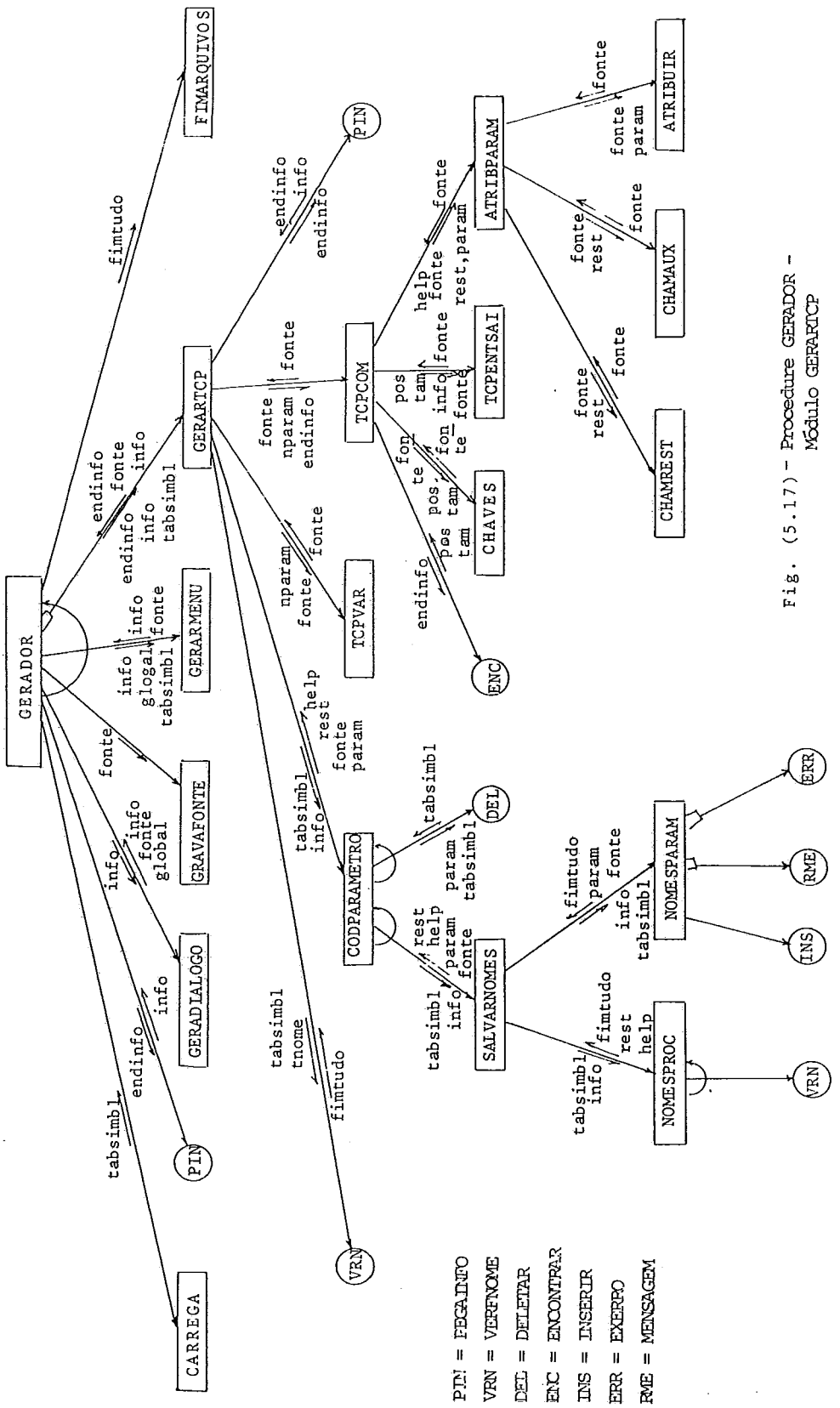


Fig. (5.17) - Procedure GERADOR - Módulo GERARTCP

ção do tipo TCP; (tela para coleta de parâmetro).

Um modelo de código gerado pelo GERARTCP foi apresentado na Seção 4.4.5, do capítulo IV. Outros exemplos podem ser vistos no apêndice I.

O GERARTCP recebe os seguintes parâmetros:

info : contendo a descrição dessa TCP, lida do arquivo <DIÁLOGO> /AÇÕES;

tabsimbl: contendo os identificadores que estejam sendo utilizados no diálogo;

endinfo : contendo os endereços de gravação da TCP; esses endereços são lidos pelo PEGAINFO, das tabelas de endereços, no final dos arquivos <DIÁLOGO> /AÇÕES e <DIÁLOGO> /TEXTOS.

O GERARTCP utiliza os seguintes módulos: CODPARAMETRO, TCPVAR e TCPCOM.

O funcionamento do GERARTCP pode ser descrito nos seguintes termos:

1) verificação da validade do nome da TCP - isto é feito através de uma chamada ao módulo VERFNOME (descrito na Seção 5.3.5);

2) chamada ao módulo CODPARAMETRO - esta chamada produz o código fonte para o cabeçalho e a especificação dos parâmetros da procedure que repre

sente a TCP;

3) chamada ao módulo TCPVAR - que gera o código fonte das declarações das variáveis dessa procedure;

4) chamada ao módulo TCPCOM - que gera o código dos comandos;

5) chamada ao módulo PEGAINFO - que substituirá as informações contidas no array info, pelas que correspondem à descrição seguinte, encontrada no arquivo <DIÁLOGO> /AÇÕES - caso não existam novas descrições, PEGAINFO devolve um parâmetro de fim de arquivo eof

O módulo CODPARAMETRO, para ser executado,recebe os seguintes parâmetros:

info : contendo as informações da TCP;

tabsimbl: contendo os identificadores que estão sendo usados na geração do diálogo.

A execução do CODPARAMETRO é feita através de repetidas chamadas ao módulo SALVARNOMES (uma chamada para cada parâmetro da TCP). Esse módulo, SALVARNOMES, carrega, nos arrays param, help e rest, os nomes de cada parâmetro, assim como das procedures de auxílio ao usuário e crítica de erros. Os nomes das procedures de auxílio e crítica de erros são obtidos através de uma chamada ao módulo NOMESPROC. Os nomes dos parâmetros,

assim como o código fonte das declarações desses parâmetros,são obtidos através da execução do módulo NOMESPARAM.

A validade dos nomes das procedures de auxílio, crítica de erros e aplicação é conferida através da execução do módulo VERFNOME.

A validade dos nomes dos parâmetros da TCP é conferido através do módulo INSERIR. O módulo CODPARAMETRO, antes de terminar sua execução retirará, do tasimbl, os nomes desses parâmetros, através do módulo DELETAR.

Em caso de erros, o projetista será alertado através da execução dos módulos MENSAGEM e EXERRO.

Os resultados da execução do CODPARAMETRO são devolvidos ao GERARTCP nos arrays:

rest : nomes das procedures de crítica de erros

help : nomes das procedures de auxílio

fonte : código fonte gerado

param : nome e tipo dos parâmetros da TCP que está sendo gerada

O módulo TCPCOM, que gera o código fonte correspondente aos comandos da procedure, utiliza os módulos CHAVES, TCPENTSAI, ATRIBPARAM. A execução do TCPCOM pode ser descrita da seguinte maneira:

1) chamada ao módulo ENCONTRAR - esta chamada devolverá a posição pos e o número de registros tam do texto da TCP gravado no arquivo <DIALOGO>/TEXTOS. Estes valores serão utilizados para gerar o código dos comandos, que apresentarão tal texto

ao usuário (o ENCONTRAR é descrito na Seção 5.3.5);

2) chamada ao módulo CHAVES - esta chamada produzirá o código dos comandos, que interpretarão as respostas do usuário, nas diferentes posições do vídeo, que tenham sido formatadas.Tais comandos só tem efeito nos terminais TS-800, para os terminais TD-110 as respostas do usuário serão, sempre, lidas na última linha);

3) chamada ao módulo TCPENTSAI - esta chamada gerará o código fonte para a apresentação dos textos da TCP e leitura das respostas do usuário;

4) chamada ao módulo ATRIBPARAN - que gera o código fonte dos comandos que realizarão a atribuição das respostas fornecidas pelo usuário aos respectivos parâmetros da TCP.

O código fonte gerado por esses três módulos é carregado no array fonte.

O módulo ATRIBPARAM utiliza os módulos: CHAMREST, CHAMAUX e ATRIBUIR.

O módulo CHAMREST realiza a geração do código fonte das chamadas às procedures de crítica de erros de cada parâmetro.

O módulo CHAMAUX gera o código fonte para as chamadas das procedures de auxílio.

O módulo ATRIBUIR realiza a geração do código para

a atribuição dos valores fornecidos pelo usuário, para cada parâmetro da TCP.

O código fonte gerado pelos módulos CHAMREST, CHAMAUX e ATRIBUIR é carregado no ARRAY fonte.

Outros detalhes do código fonte gerado para uma TCP podem ser observados nas ilustrações apresentadas na Seção 4.4.5, do capítulo IV e nos exemplos da Apêndice I.

O GERARTCP, ao final de sua execução, devolverá ao GERADOR:

fonte : código fonte gerado

info : atualizado, com uma nova descrição

eof : caso não existam novas descrições

fimtudo : no caso de terem acontecido erros, durante a execução do GERARTCP.

O Módulo FIMARQUIVOS Fig. (5.18)

Parâmetros de entrada:

fimtudo: variável booleana, que caso seja verdadeira, indicará ao FIMARQUIVOS a existência de erros.

A função do FIMARQUIVOS é encerrar a execução do GERADOR.

Na sua execução, o FIMARQUIVOS tem duas opções:

a) fimtudo = FALSE

Neste caso, a execução do FIMARQUIVOS realiza o

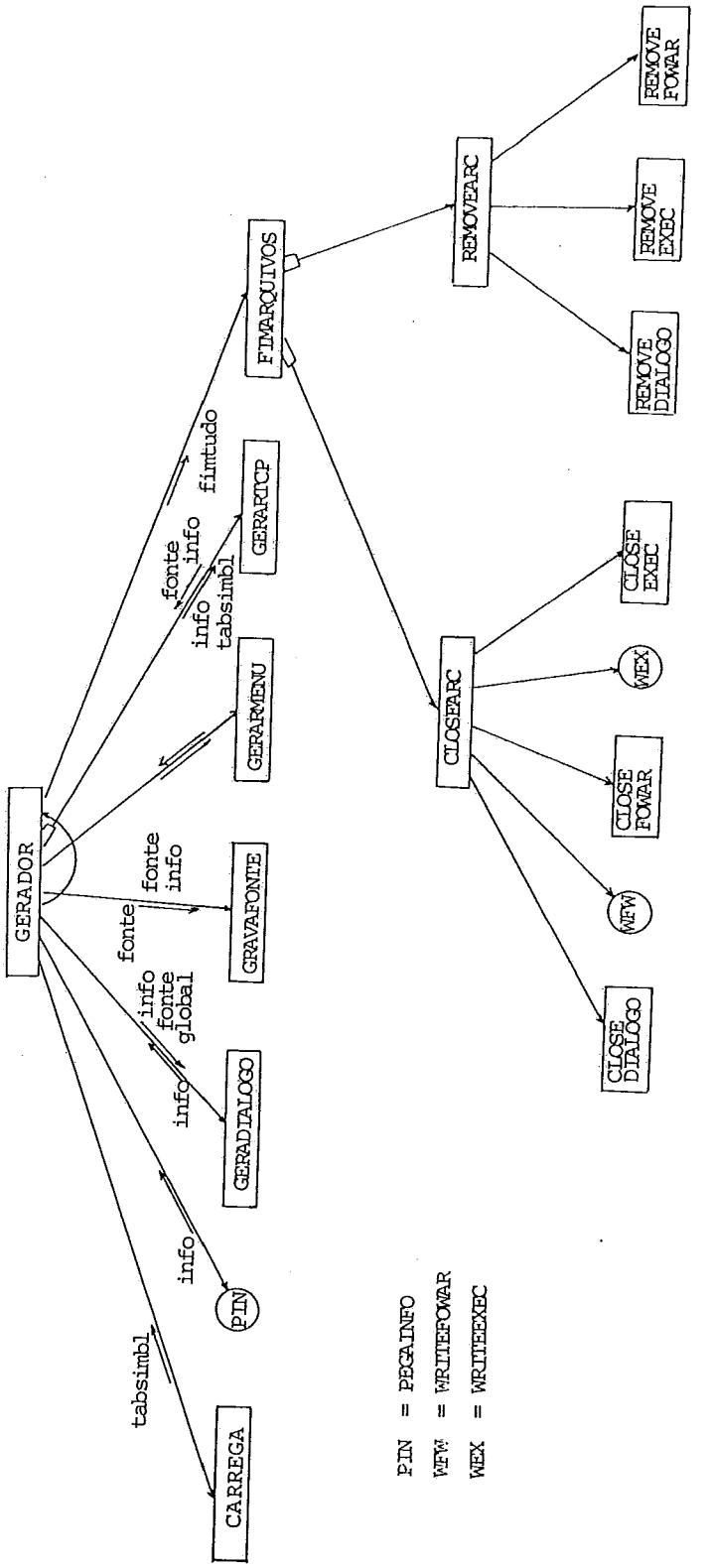


Fig. (5.18)

Procedure GERADOR; Módulo FIMARQUIVOS

fechamento de todos os arquivos que foram abertos durante a execução do GERADOR. Isto é conseguido quando o FIMARQUIVOS executa CLOSEARQ.

O módulo CLOSEARQ fecha os arquivos <DIÁLOGO> e <DIÁLOGO> /PROC, grava as últimas instruções dos arquivos <DIÁLOGO> /FOWAR e <DIÁLOGO> /EXEC; e, finalmente fecha esses dois arquivos.

Quando o CLOSEARQ termina sua tarefa, todo o código fonte, gerado pela execução do GERADOR, ficará armazenado no disco, nos arquivos <DIÁLOGO> <DIÁLOGO> /PROC, <DIÁLOGO> /EXEC e <DIÁLOGO> /FOWAR , encerrando desta maneira a execução do GERADOR.

b) fimtudo = VERDADEIRO

Neste caso, a execução do FIMARQUIVOS retira do disco todo o código que puder ter sido gerado.

Tal fato é conseguido através da execução do REMOVEARQ.

O módulo REMOVEARQ retira do disco os arquivos <DIÁLOGO>, <DIÁLOGO> /PROC, <DIÁLOGO> /EXEC e <DIÁLOGO> /FOWAR, encerrando a execução do GERADOR.

5.3.5 - Módulos Utilizados Repetidamente na Execução do GERADOR Fig. (5.19)

O módulo Inserir

Finalidade: a função desse módulo é inserir no

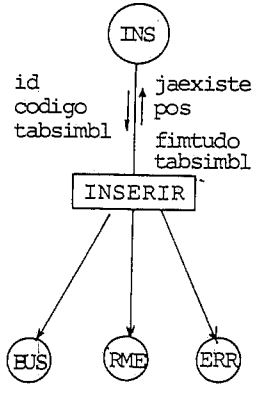


Fig. (5.19)

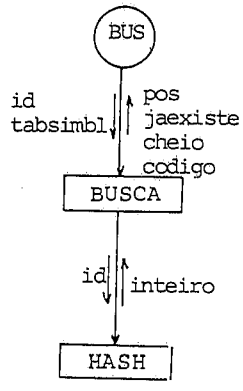


Fig. (5.20)

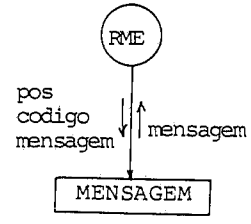


Fig. (5.21)

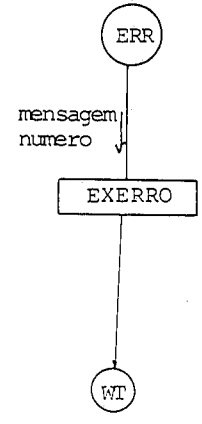


Fig. (5.22)

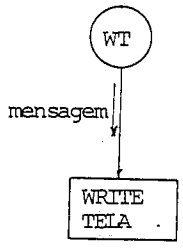


Fig. (5.23)

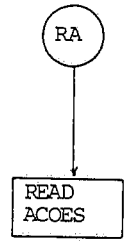


Fig. (5.24)

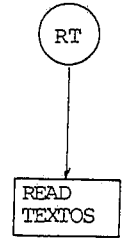


Fig. (5.25)

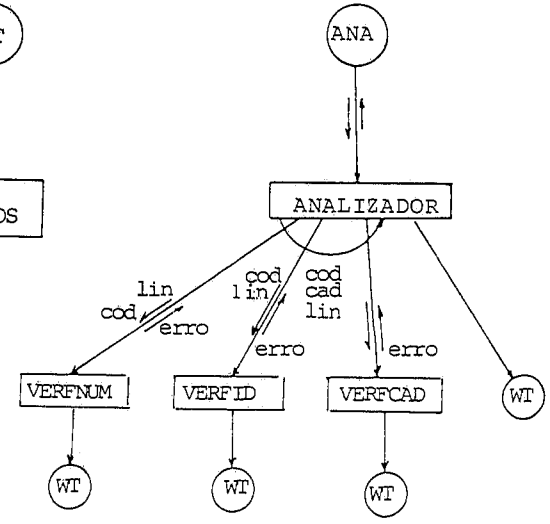


Fig. (5.26)

array de identificadores que esteja sendo utilizado no diálogo, tabsimbl, um novo identificador id.

Para isto o INSERIR recebe os seguintes parâmetros:

id : identificador a ser inserido no tabsimbl;

código : código para diferenciar o id;
Ex.: palavra reservada, nome de procedure, nome de menu, TCP, parâmetro, etc.;

tabsimbl : array com os identificadores utilizados no diálogo

Para realizar a inserção o módulo INSERIR, chamará, primeiramente, o módulo BUSCA. Este módulo devolve ao INSERIR a posição no array tabsimbl, onde o id poderá ser inserido, e a variável booleana jaexiste. Caso jaexiste seja verdadeira, o valor recebido no pos indicará a posição onde aquele id foi encontrado no tabsimbl.

Na presença de erros, o INSERIR executará os módulos MENSAGEM e EXERRO, com os resultados já descritos.

O Módulo BUSCA Fig. (5.20)

A finalidade desse módulo é encontrar a posição de um identificador id no array de identificadores tabsimbl.

Parâmetros de entrada:

id : identificador a ser procurado no tabsimbl

tabsimbl : array com os identificadores utilizados no diálogo

O módulo BUSCA inicia sua execução começando pelo módulo HASH. O HASH recebe de BUSCA o identificador id e gera, a partir deste, um número aleatório pos, que é devolvido ao BUSCA.

Com esse valor pos, o módulo BUSCA procede à procura do id no array tabsimbl. Esta procura, que é realizada com resolução quadrática, só termina no momento em que o conteúdo da posição pos do array tabsimbl corresponda à do identificador procurado, ou no momento em que pos indique uma posição livre do tabsimbl.

O Módulo MENSAGEM Fig. (5.21)

Finalidade: a função deste módulo é carregar num array de caracteres mensagem, um texto que identifique o tipo de algum identificador armazenado no tabsimbl.

Parâmetros de entrada:

código : código desse identificador

Ex.: 0 = palavra reservada

1 = nome de uma procedure de auxílio

2 = nome de menu

mensagem : contendo alguma mensagem de erro

pos : contendo o endereço no array mensagem da primeira posição disponível para a gravação daquele texto.

O funcionamento do MENSAGEM:

Dependendo do valor recebido em código, o MENSAGEM carregará na primeira posição livre do array mensagem (apuntada por pos), o "texto" que identifique o tipo de um determinado identificador.

Ex.: código = 0 (palavra reservada)

array mensagem = ... não poderá ser o nome de menu pois ele é uma palavra reservada
pos "texto"

O Módulo EXERRO Fig. (5.22)

Finalidade: a função do EXERRO é apresentar, ao projetista, uma mensagem de erro e um código que identifique o tipo de erro.

Parâmetros de entrada:

- mensagem: contendo o texto a ser apresentado;

- código : contendo o código do erro

O Módulo ANALISADOR Fig. (5.26)

Finalidade: a função deste módulo é realizar a análise sintática de uma determinada cadeia de caracteres.

Parâmetros de entrada:

buffer : array contendo a cadeia de caracteres

caso : código que identifica o conjunto de regras sintáticas, que deverão ser utilizadas;

reg : endereço da cadeia de caracteres que está sendo analisada nos arquivos ‹DIÁLOGO› / AÇÕES ou ‹DIÁLOGO› /TEXTOS.

O ANALISADOR utiliza os seguintes módulos: VERFNUM, VERFID e VERFCAD.

O funcionamento do ANALISADOR pode ser descrito nos seguintes termos:

1) chamada para um dos módulos que realiza a análise sintática; a escolha do módulo é feita em função do código caso:

   a) VERFID - analisa o conteúdo do buffer, com as regras previstas para um identificador;

   b) VERFNUM - similar ao VERFID, porém para um valor inteiro;

   c) VERFCAD - compara o conteúdo do buffer com uma determinada cadeia de caracteres cad.

2) em caso de erros, os módulos VERFID, VERFNUM ou VERFCAD apresentam, ao projetista, as respectivas mensagens de erro e devolvem, em erro, um código para o tipo de erro detetado.

Caso tenham ocorrido erros, o analisador devolverá fimtudo colodada com um valor verdadeiro.

O Módulo PEGAINFO Fig. (5.27)

Finalidade: a função deste módulo é carregar, no array info, as informações de uma nova descrição, lida no arquivo <DIÁLOGO> /AÇÕES.

Parâmetros de entrada:

info : array de caracteres contendo as informações de alguma descrição do arquivo <DIÁLOGO> /AÇÕES;

endinfo : contendo os endereços dessa descrição.

O funcionamento do PEGAINFO pode ser descrito nos seguintes termos:

1) atualização dos endereços recebidos no endinfo, com os endereços da descrição seguinte, que puder ser encontrada no <DIÁLOGO> /AÇÕES;

2) substituição das informações no info por aquelas que correspondem a essa nova descrição - esta

Fig. (5.27)

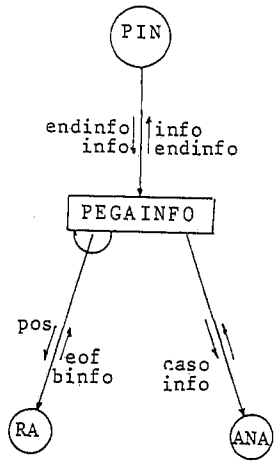


Fig. (5.28)

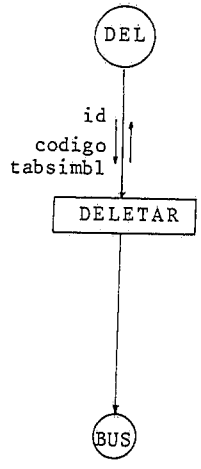


Fig. (5.29)

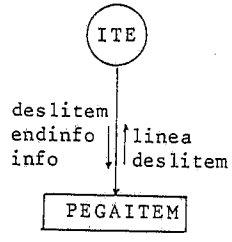


Fig. (5.30)

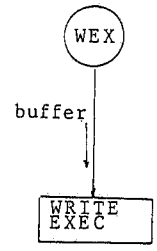


Fig. (5.31)

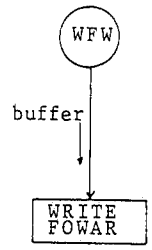
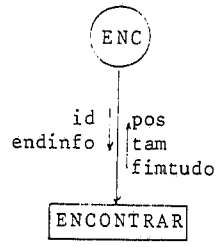


Fig. (5.32)



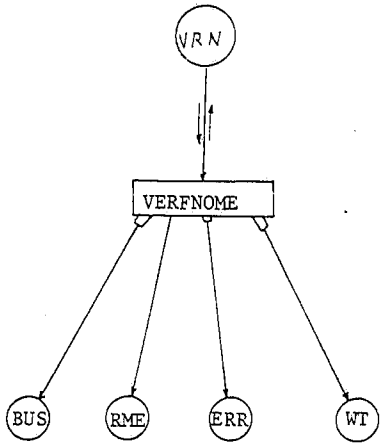
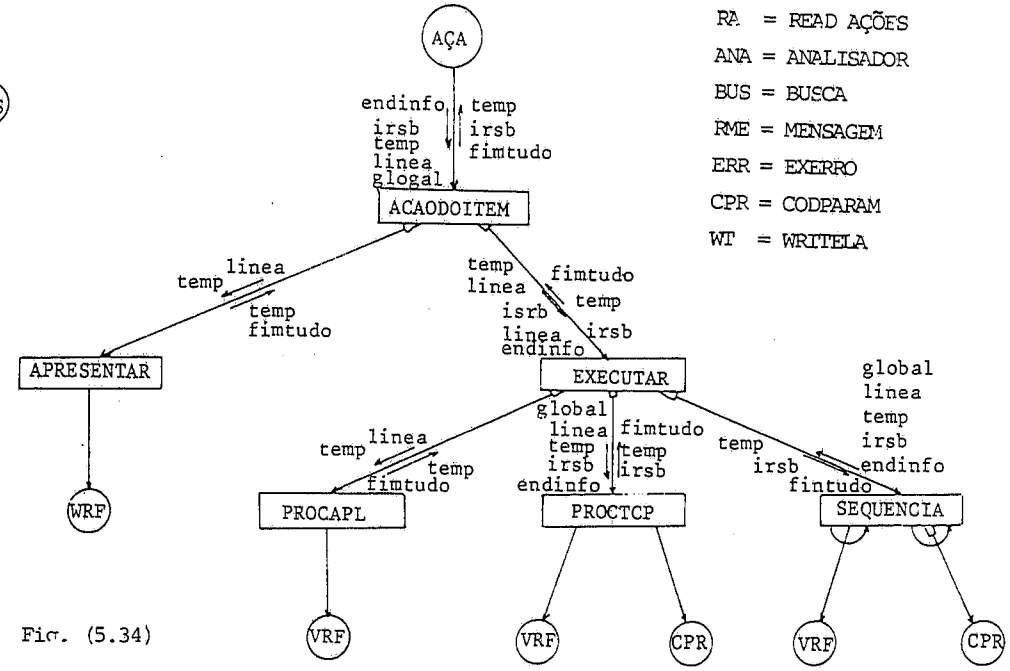


Fig. (5.33)



RA = READ AÇÕES
ANA = ANALISADOR
BUS = BUSCA
RME = MENSAGEM
ERR = EXERRO
CPR = CODPARAM
WT = WRITELA

Fig. (5.34)

substituição é feita através de repetidas leituras do arquivo <DIÁLOGO> /AÇÕES;

3) quando o conteúdo do info tiver sido totalmente atualizado, o PEGAINFO chama o módulo ANALISADOR para realizar a análise sintática dessas novas informações.

No caso do PEGAINFO não encontrar novas descrições no arquivo <DIÁLOGO>/AÇÕES, ele devolverá uma marca de fim de arquivo eof.

O Módulo DELETAR Fig. (5.28)

Finalidade: a função do DELETAR é retirar do tabsimbl um determinado identificador id.

Parâmetros de entrada:

id : identificador a ser retirado do tabsimbl;

tabsimbl: array de caracteres contendo os identificadores que estejam sendo utilizados no diálogo;

código : tipo de identificador - Ex.: palavra reservada, parâmetro,nome do menu etc.

Para retirar o id, o DELETAR precisa conhecer a posição pos desse identificador no tabsimbl. Para isto, o DELETAR executa o módulo BUSCA.

A remoção do id é feita colocando-se uma marca de

deleção no lugar que ocupa o código daquele id no tabsimbl.

A tentativa de remover identificadores não existentes em tabsimbl será considerada como erro, fato que será detetado pelo BUSCA.

O Módulo PEGAITEM Fig. (5.29)

Finalidade: a função deste módulo é colocar em um array de caracteres as informações correspondentes a um item de um determinado menu.

O PEGAITEM recebe os seguintes parâmetros de entrada:

info : contendo todas as informações de uma descrição de tipo menu;

endinfo : contendo os endereços de gravação desse menu;

deslitem : contendo o deslocamento no array info do começo das informações correspondentes àquele item.

Após ter sido executado, o PEGAITEM devolve, em linea, as informações do item e, em deslitem, o deslocamento do item seguinte nesse menu.

O Módulo ENCONTRAR Fig. (5.32)

Finalidade: a função do ENCONTRAR é achar, no arquivo <DIÁLOGO> /TEXTOS, a posição pos e o

número de registros tam de uma descri
ção de nome id.

O Módulo VERFNOME Fig. (5.33)

Parâmetros de entrada:

id : identificador

caso : especifica o tipo esperado daquele iden
tificador - ex.: nome de menu,de uma TCP
etc.;

reg : número do registro do arquivo <DIÁLOGO>
/AÇÕES, ao qual pertence o id;

tabsimbl : array com os identificadores utilizados
no diálogo

Parâmetro de saída:

fimtudo: que será verdadeiro, caso aconteçam erros.

A função do VERFNOME é comprovar se o identificador recebido no id pertence ao tabsimbl e se o tipo esperado, daque le id (recebido no caso) é igual ao tipo do identificador en contrado no tabsimbl.

A pesquisa do id é feita através de uma chamada do módulo BUSCA.

Caso a procura do tabsimbl não seja bem sucedida, o VERFNOME executará as seguintes ações:

1) mensagens de "warning" no caso do tipo esperado ser o de procedures de auxílio, crítica ou apli cação;

2) se o tipo esperado for uma descrição (menu ou TCP), o VERFNOME executará o MENSAGEM e EXERRO, com os resultados já previstos, e devolverá fim-tudo com valor verdadeiro

O Módulo AÇÃODOITEM Fig. (5.34)

Finalidade: a função deste módulo é gerar o código fonte dos comandos que executam as ações de um item de um menu.

Parâmetro de entrada:

irsb : código fonte gerado para as declarações das variáveis, até o momento utilizadas;

temp : código fonte gerado para os comandos da procedure;

linea : informações com a descrição do item;

global : contendo os nomes dos parâmetros do diálogo (se houver);

endinfo: endereços de gravação do menu, no arquivo <DIÁLOGO> /AÇÕES.

O funcionamento do AÇÃODOITEM pode ser descrito da seguinte forma:

- caso o tipo de ação do item for Apresentar, o AÇÃODOITEM chama o módulo APRESENTAR;
- caso o tipo de ação do item for Executar ou Coletar, o AÇÃODOITEM chama o módulo EXECUTAR.

O módulo APRESENTAR confere, mediante uma chamada do VERFNOME, se o nome do menu recebido em linea é válido.

No caso daquele nome ser válido, o APRESENTAR carrega no array temp o código da chamada daquele menu.

Caso o nome recebido no LINEA não seja o de um menu, APRESENTAR devolverá fimtudo com valor verdadeiro.

A execução do módulo EXECUTAR apresenta duas opções: chamar uma procedure de aplicação ou uma tela para coleta de parâmetros (TCP).

a) Chamada de uma procedure de aplicação:

- neste caso, o EXECUTAR chama o módulo PROCAPL, para que ele gere o código fonte para a chamada dessa procedure; antes da geração desse código, o PROCAPL verifica, através do VERFNOME, a validade do nome dessa procedure.

b) Chamada de uma tela para coleta de parâmetros (TCP)

- neste caso, o EXECUTAR chama o módulo PROCTCP, o PROCTCP procede, primeiramente, à verificação da validade do nome da TCP e, a seguir, deve gerar o código fonte para o comando que realiza a chamada dessa TCP, assim como para as declarações dos parâmetros, no cabeçalho da procedure - isto é feito através da execução do módulo CODPARAM.

Em seguida, PROCTCP verificará se existe uma procedure de aplicação com os parâmetros da TCP, que deva ser execu

tada após a chamada dessa TCP. Se tal procedure existir,o PROC TCP acrescentará, no temp, a chamada dessa procedure.

Os resultados da execução do PROCTCP são devolvidos ao EXECUTAR nos arrays temp e irsb. Na presença de erros o PROC TCP devolverá, ao EXECUTAR, a variável fimtudo como verdadeira.

Como foi visto no capítulo IV, Seção 4.3.1, "G D" permite que novas procedures, com ou sem parâmetros, possam ser executadas, após a execução de uma procedure de aplicação ou de uma TCP.

Por essa razão, o módulo EXECUTAR, depois do término da execução dos módulos PROCAPL ou PROCTCP (segundo o caso), procura no array linea pela presença desta sequência de procedures.

Se tal sequência existir, o EXECUTAR chamará o módulo SEQUÊNCIA, através dos parâmetros temp, irsb, linea e global. (que já foram descritos).

O módulo SEQUÊNCIA conferirá, através de chamadas do VERFNOME, o nome de cada procedure de aplicação ou TCP recebidos. Quando a execução do VERFNOME for realizada, o SEQUÊNCIA procederá a geração de código fonte para executar a chamada dessa procedure de aplicação ou TCP.

A geração desse código fonte apresenta duas opções:

1) o nome analisado pelo VERFNOME é uma procedure de aplicação - neste caso a geração de código para a chamada da procedure será feita pelo SEQUÊNCIA;

2) o nome analisado pelo VERFNOME é uma TCP - nesta opção, a especificação dos parâmetros da TCP se

rá feita através de uma chamada ao CODPARAM; em seguida, o SEQUÊNCIA verificará se, após a chamada da TCP, deve ser executada alguma procedure de aplicação (com os mesmos parâmetros dessa TCP). Caso tal procedure existir, o SEQUÊNCIA acrescentará no TEMP o respectivo código fonte. O código fonte obtido pela execução do SEQUÊNCIA é acrescentado e devolvido ao EXECUTAR nos arrays temp e irsb. Havendo erros, o SEQUÊNCIA devolverá fimtudo como verdadeira.

O Módulo CODPARAM Fig. (5.35)

A função do CODPARAM é gerar o código fonte da especificação dos parâmetros da chamada de uma TCP.

Parâmetros de entrada:

endinfo : array de endereços do menu;

irsb : tipos das variáveis até o momento definidas na geração do código fonte da procedure

temp : código fonte gerado até o momento para o bloco de comandos da procedure

global : nomes dos parâmetros através dos quais o diálogo que está sendo gerado pode ser chamado (se tiver);

idtcp : nome da TCP

O CODPARAM inicia sua execução procurando informa

ções sobre aquela determinada TCP. TAl procura é realizada com uma chamada do PEGATCP, através dos parâmetros idtcp e endinfo.

A execução do PEGATCP devolve ao CODPARAM, no array inftcp , todas as informações da descrição daquela TCP. Ocorrendo erros, o PEGATCP chama o EXERRO e devolve fimtudo com valor verdadeiro.

Com a informação recebida no inftcp, o CODPARAM carrega no temp o código fonte correspondente à especificação dos parâmetros dessa TCP.

Junto com esse código fonte, o CODPARAM também carrega, no irsb, o código correspondente à declaração de cada parâmetro da TCP, que ainda não tenha sido declarado na geração de código da procedure e que não faça parte do array global (nesse caso, ele teria sido declarado como parâmetro da procedure <DIÁLOGO>, gerada pelo GERADIALOGO).

Havendo erros, CODPARAM devolverá fimtudo com valor verdadeiro.

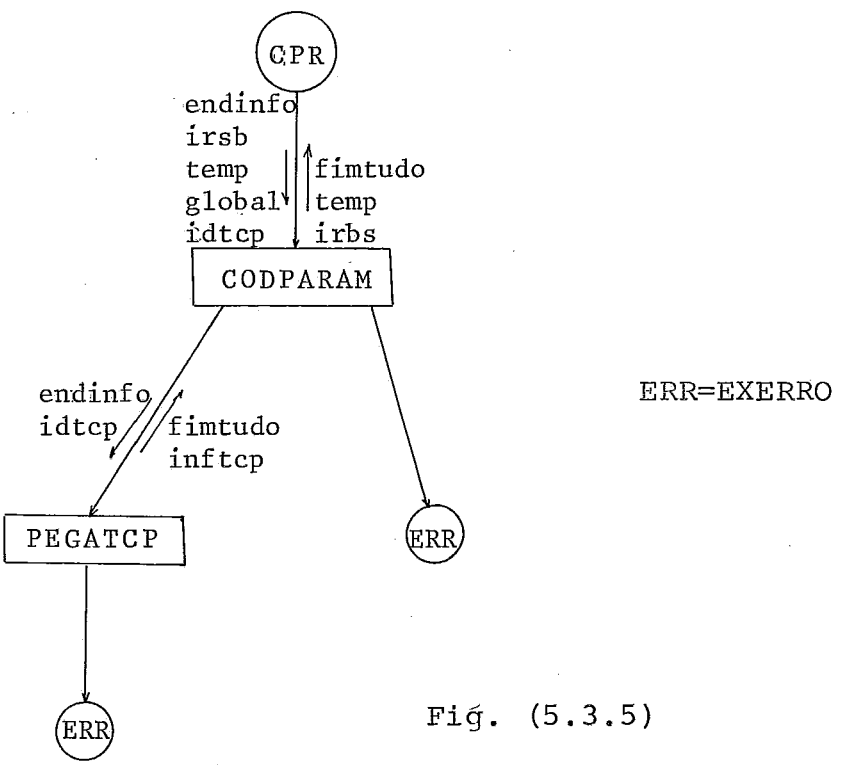


Fig. (5.3.5)

VI - CONCLUSÕES

6.1 - Resumo e Discussão

A produção de interfaces que sejam exequíveis para um grande número de usuários requer consideráveis esforços na produção de software.

A utilização de programas auxiliares (ferramentas de software), que automatizem uma parte ou todo o processo da criação da interface, contribui definitivamente na diminuição da complexidade, assim como dos custos de produção.

As ferramentas de software para produção de interfaces que foram pesquisadas, apresentam em geral, uma estrutura que pode ser descrita nos seguintes termos: um programa editor que faz a recepção das descrições da interface e um programa que traduza e/ou execute as descrições recebidas.

Essas interfaces podem ser constituidas de:

- instruções ao usuário (perguntas-respostas);
- comandos que executem tarefas;
- menus (para a escolha em uma lista de possibilidades);

Uma característica comum a todas essas ferramentas de software é a utilização dos programas de aplicação estruturados em módulos que realizam determinadas funções. A interface que é obtida após o uso desses programas auxiliares é, de alguma maneira, o módulo que controla a execução dos módulos da aplicação.

O Gerador de Diálogos proposto neste trabalho apresenta-se como uma alternativa na forma de um gerador de código fonte da interface cujas características essenciais são as seguintes:

- criação de interfaces baseadas em uma estrutura de apresentação de menus e de telas para coleta de parâmetros;

- interpretação das descrições dos diálogos dividida em:

  a) uma etapa de a entrada e/ou modificações das descrições;

  b) uma fase para a geração do código fonte ALGOL;

  c) uma etapa para a compilação e utilização do código fonte criado.

Esta divisão permite uma clara separação das atividades relacionadas ao projeto da interface daquelas que correspondem ao software da aplicação.

- a utilização de um arquivo que armazene as descrições recebidas permite que o prejetista da interface possa, através do simples uso do programa editor de textos do computador, realizar facilmente modificações ou atualizações sobre descrições previamente recebidas;

- a possibilidade do código fonte criado pelo "G D" poder ser utilizado tanto como o módulo de controle para a execução de programas de

aplicação, como no contexto das instruções de um determinado programa (macros - pré-processadas para interagir com o usuário), permite que o "G D" possa ser utilizado em um maior número de aplicações;

- a utilização de telas de tipo menu, assim como de procedures de crítica e recuperação de erros, permite que o usuário possa receber auxílio, em cada entrada de informações possível;

- os valores que o usuário deve fornecer como parâmetros de entrada para execução de determinadas procedures de aplicação somente são atribuídos se todos eles satisfizerem as necessidades colocadas no projeto da interface; este teste prévio à atribuição dos parâmetros garante a coerência na execução do software da aplicação;

Experiência no uso do "G D"

A realização das aplicações apresentadas no Apêndice I permite a apresentação dos seguintes resultados:

- a interface com o usuário, produzida pelo "G D" nessas aplicações, foi, em sua totalidade, considerada fácil de ser utilizada pelos usuários desses sistemas;

- a utilização do "G D" dispensou em grande parte o trabalho de elaboração de manuais do usuário dessas aplicações;

- o relacionamento interativo do "G D" com os projetistas facilitou muito o fornecimento das descrições dos diálogos;

- a estrutura do "G D" permitiu que a produção das interfaces fosse feita de maneira evolutiva, partindo de menu TCPs constituídas por umas poucas linhas e que logo foram depuradas através do simples uso do editor de textos, assim como, das procedures de aplicação, auxílio ou crítica de erros, que, inicialmente, eram pequenos blocos de código fonte de tipo "dummy" e que logo foram substituidos por seus verdadeiros segmentos de código fonte;

- este processo evolutivo foi altamente benéfico, pois permitiu, por exemplo, que durante a depuração das telas que deviam ser apresentadas ao usuário pudessem ser percebidas muitas das sutilezas relacionadas à atenção dos fatores humanos nessas interfaces.

Esse processo evolutivo facilitou, também, a otimização das procedures de crítica e recuperação de erros, assim como a total separação do projeto da interface, daquele que cor

responde aos programas de aplicação.

O "overhead" produzido pelo uso do "G D"

Os "overhead" produzidos pelo uso do "G D" foram pequenos, comparados às vantagens que têm sido descritas. Em seguida, apresentaremos algumas medidas que descrevem um custo médio do uso do "G D":

1 - Tempo de processador gasto na compilação de um ou mais menus ou TCPs: (Fig. 6.1)

Tcomp = C + C1 (# menus) + C2 (# TCP)

C=4.0 s gasto na compilação das procedures de auxílio que acompanham qualquer código fonte gerado pelo "G D" (capítulo IV, Seção 4.4.5)

c1=1.5 s para menus com uma média de três ítens

C2=1.5 s para TCP com uma média de 4 parâmetros

Tcomp = tempo de compilação

Fig. (6.1) - Tempos de Processador gastos na compilação de diálogos interpretados pelo "G D".

2 - Tempo de processador gasto na execução de um ou mais menus ou TCP's:

- menus com uma média de três ítens = 0.3 s
- TCP's com uma média de 4 parâmetros = 0.5 s

3 - A quantidade de Código Fonte gerado:

O número de linhas de código fonte ALGOL, gasto nas procedures obrigatórias que devem acompanhar qualquer código gerado pelo "G D" (Capítulo IV, Seção 4.4.5) é de 129 + 2 para cada descrição.

Talves seja esse o maior "overhead" no uso do "G D" porém sua utilização é totalmente justificada, pois permite que o usuário possa responder a uma tela de menu com o nome de outro menu, assim como faz com que os valores que o usuário deve fornecer como parâmetros possam ser entregues com formato livre.

O número de linhas do código fonte de uma procedure gerada pelo "G D", para a interpretação de um menu, é de 32 + 1 para cada item desse menu.

No caso das TCP, esse número é de 35 + 4 para cada parâmetro.

O espaço em disco que deve ser alocado para salvar todo esse código fonte gerado, será sensivelmente diminuido após a compilação de cada diálogo, quando somente será necessário salvar o código objeto desses diálogos.

O número de linhas contidas no arquivo <DIÁLOGO> /AÇÕES depende do número de menus ou TCPs contidas no diálogo, porém o uso desse arquivo é dispensável após a geração do código fonte.

O número de linhas do arquivo <DIÁLOGO> /TEXTOS é, também, ligado ao número de menus e TCPs contidas no diálogo, sendo que o tamanho das telas que devem ser apresentadas ao usuário influirá, definitivamente, no comprimento total desse arquivo. No entanto, o tamanho do <DIÁLOGO> /TEXTOS será sensivelmente reduzido através do uso do utilitário "COMPACTADOR" do computador.

A implementação do "G D"

A implementação do "G D" no computador B-6700 da da UFRJ foi feita visando a otimização dos recursos disponíveis.

Nesse sentido, a linguagem ALGOL, na qual foi implementado o "G D", tem sido de grande ajuda, especialmente na manipulação de cadeias de caracteres.

O programa BINDER do B-6700 permitiu que o código fonte gerado possa ser utilizado por programas de aplicação que estejam escritos em FORTRAN, COBOL ou ALGOL, o que amplia o número de aplicações que poderão ser feitas sobre o "G D".

A velocidade de transmissão da rede de terminais TS-800 ligados ao B-6700 (2400Bauds), tem garantido uma rápida apresentação das telas de menus ou TCPs.

O aproveitamento da formatação da tela dos terminais TS-800 faz com que a interface gerada pelo "G D" seja mui

to atraente para o usuário. Contudo, para o caso dos terminais TD-110 que não apresentam a facilidade de formatação da tela, o código fonte gerado pelo "G D" é perfeitamente aceitável,tal como foi descrito na Seção (4.3.2), do Capítulo IV.

6.2 - Recomendações

Em prosseguimento ao trabalho ora apresentado, recomenda-se que após um período de observação dos resultados obtidos com a utilização do "G D" na criação de interfaces para diversos tipos de aplicações, que se procedam aos seguintes estudos:

- avaliação dos benefícios reais, obtidos com o uso do "G D" comparando-se os resultados conseguidos com a utilização e a não utilização do mesmo, na construção de interfaces para aplicações similares. Podem ser comparados, estatisticamente, os tempos de programação e outras medidas da performance das duas formas de solução (número de erros, tempos de execução, aceitação por parte do usuário etc.).

- o "G D", atualmente implementado, gera código fonte ALGOL, que é utilizado em aplicações no B-6700; entretanto, a mesma estrutura da solução do "G D" pode ser aproveitada para gerar código fonte em outras linguagens de programação - PASCAL, por exemplo - esta nova versão poderia ser utilizada perfeitamente, para que as interfaces produzidas

pudessem ser utilizadas em aplicações sobre micro computadores;

- na solução proposta neste trabalho, foram utilizados três tipos de técnicas para interagir com o usuário: perguntas e respostas, menus e telas para coleta de parâmetros; futuras pesquisas poderiam ser levadas a efeito, a fim de aumentar esse número de técnicas - a interação com gráficos poderia ser uma delas;

- as modificações sobre descrições previamente recebidas pelo "G D" são, atualmente, realizadas através do programa editor do B-6700 - uma alternativa seria a criação de um Editor de Descrições próprio para o "G D"; contudo, a implementação desta nova facilidade deveria ser o resultado da análise da performance do projetista no uso do "G D".

- a utilização de procedures de crítica e avaliação de erros é uma solução bastante eficiente para fornecer auxílio ao usuário na entrada de parâmetros; no entanto, a implementação dessas procedures não é feita através do "GD" - esta forma de solução (que também foi encontrada nas diversas abordagens que foram pesquisadas - Capítulo III), talves possa ser otimizada, se uma parte do software dessas procedures conseguisse ser produzida automaticamente;

- a forma de solução utilizada no "GD", que associa uma procedure ALGOL para cada menu ou TCP da interface que deve ser gerada, funciona perfeitamente para os casos nos quais o número de menus ou TCPs não seja excessivamente elevado, entretanto, nos casos em que for necessário a apresentação de uma grande quantidade de telas, Ex.: Instrução assistida pelo computador, o modelo de solução proposto pelo "G D" talvez não seja muito eficiente - esta possibilidade poderá ser avaliada nas pesquisas sobre a performance do uso do "G D";

- na execução de interfaces produzidas através do uso do "G D", o usuário pode responder à apresentação de um menu com o nome de outro menu, o que permite que essa interface seja muito flexível; ela não tem, todavia, instrumentos implementados que permitam conhecer a história das ações até o momento realizadas - a implementação desses instrumentos traria, às interfaces produzidas pelo "G D", meios ainda mais atraentes para o relacionamento usuário-computador.

A P E N D I C E I

ILUSTRAÇÃO DO USO DO "G D"

EXEMPLO I

Neste primeiro exemplo, aproveitamos o desenvolvimento de um programa de aplicação, escrito na linguagem de programação FORTRAN do B-6700.

1.1 - Os Objetivos deste Programa de Aplicação

Este software de aplicação faz parte do trabalho de pesquisa da tese de Mestrado "Planejamento Industrial em Face da Programação Linear, com Objetivos Múltiplos", que atualmente está sendo realizada no Programa de Engenharia de Sistemas - Otimização da COPPE.

A função do programa FORTRAN, em particular, é resolver problemas de programação linear, com objetivos múltiplos.

1.2 - Os Parâmetros de Entrada do Programa de Aplicação

A execução deste programa FORTRAN requer, do usuário, o fornecimento dos seguintes parâmetros:

a) Parâmetros que definem o modelo:

NOBJ: variável inteira, que representa o número de objetivos.

Restrição ..... $(0 < NOBJ \leq 40)$

NPRI: variável inteira que representa o número de níveis de prioridade.

Restrição ..... $(0 < NPRI \leq 10)$

NVAR: variável inteira que representa o número de variáveis.

Restrição ..... ($0 < NVAR \leq 60$)

NTAF: variável inteira que representa o número de termos da função execução.

Restrição ..... ($0 < NTAF \leq 40$)

b) Uma matriz bidimensional de máximo 40 X 40 elementos, contendo os coeficientes que valorizam os objetivos.

c) Os parâmetros que definem a função de execução:

IPRI: variável inteira, que recebe o nível de prioridade para um determinado termo.

Restrição ..... ($0 < IPRI \leq 10$)

ISUB: variável inteira, que recebe um código associado a uma variável de desvio.

Restrição ..... (não tem)

WHTF: variável real, que recebe um fator ponderável, correspondente à respectiva variável de desvio.

Restrição ..... ($WHTF > 0$)

1.3 - O Desenvolvimento da Interface Usuário-Computador

1.3.1 - A primeira tentativa do desenvolvimento da interface com o usuário deste programa pode ser vista no esquema apresentado na Fig. (A-1.1)

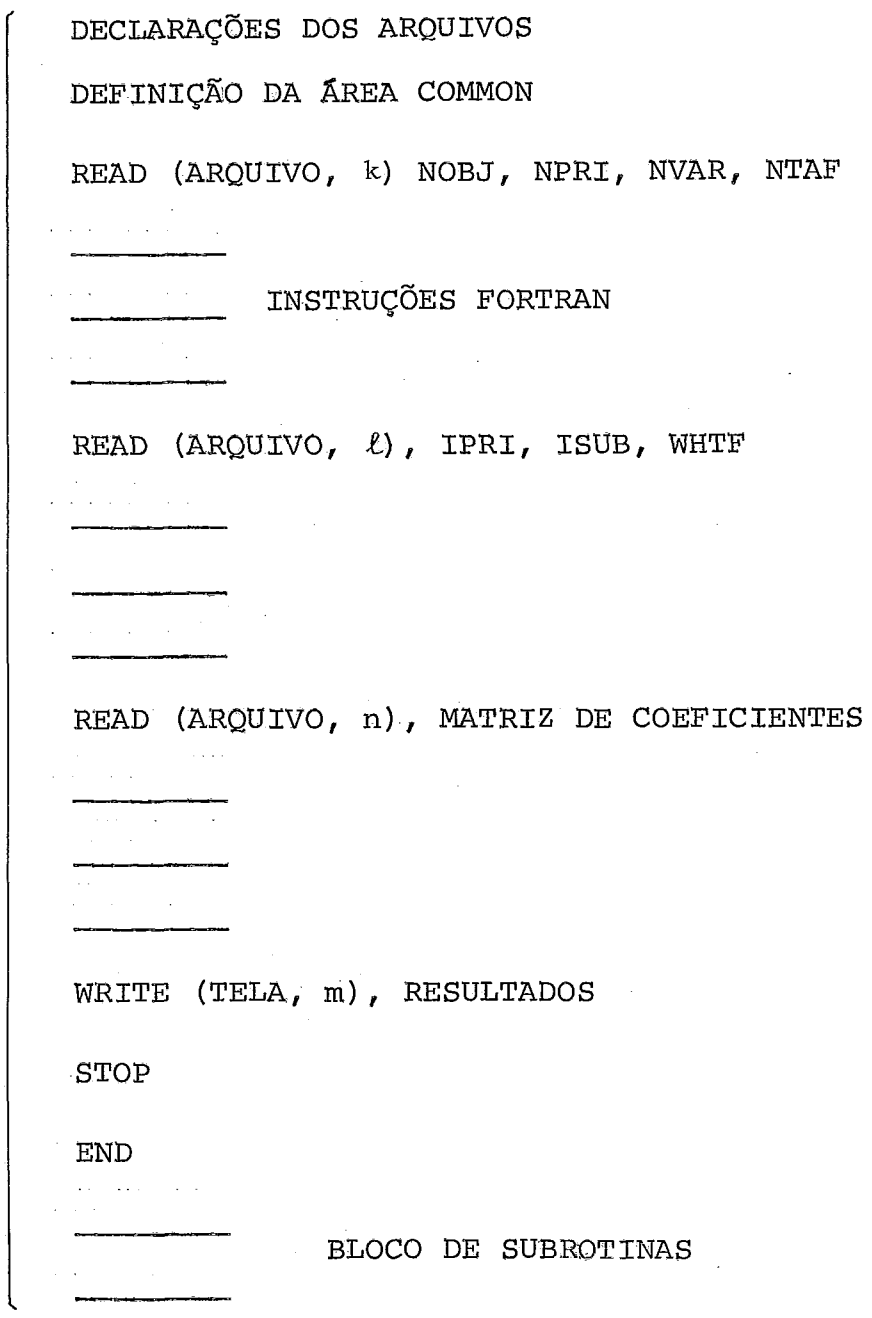


Fig. (1.1) - Desenvolvimento da Interface Usuário-Computador (primeira tentativa).

Como pode ser deduzido da análise do esquema da figura (A-1.1), a participação do usuário na execução deste programa estaria limitada aos seguintes aspectos:

a) gravar em um arquivo em disco, os parâmetros de entrada;

b) executar um comando que rode o programa FORTRAN;

c) esperar pelos resultados.

Na presença de erros nos parâmetros de entrada, as mensagens de erro que o usuário deveria esperar, seriam as que usualmente, fornece o sistema operacional do B-6700, no momento da execução de algum programa. No entanto, essas mensagens precisam que o usuário esteja familiarizado com o uso do programa em questão e com a linguagem FORTRAN; caso contrário, elas apareceram confusas e não poderão servir como um auxílio efetivo para a recuperação de tais erros.

1.3.2 - Uma segunda versão da interface usuário-computador foi projetada e desenvolvida através do uso do "G D". Um esquema de programa FORTRAN, já incluindo esta nova interface, é apresentado na Fig. (A-1.2)

```
$ SET AUTOBIND                                          ┐
$ BIND DIAL1      FROM OBJECT/=                         │ INSTRUÇÕES DO
$ BIND DIAL2      FROM OBJECT/=                         ┘ BINDER DO B-6700

DECLARAÇÕES DOS ARQUIVOS
DEFINIÇÕES DA ÁREA COMMON
CALL DIAL1        (NOBJ, NPRI, NVAR, NTAF)  ←┐
                                             │
      INSTRUÇÕES FORTRAN                     │ MÓDULOS GERADOS ATRAVÉS DO "G D "
                                             │
CALL DIAL2     (IPRI, ISUB, WHTF)        ←┘
      INSTRUÇÕES FORTRAN

READ (ARQUIVO, m) MATRIZ DE COEFICIENTES
      INSTRUÇÕES FORTRAN

WRITE (TELA, n) RESULTADOS
STOP
END
      BLOCO DE SUBROTINAS
```

Fig. (A-1.2) - Programa FORTRAN incluindo chamadas para IUC produzida através do "G D"

Como pode ser visto no esquema da Fig. (A-1.2),as únicas instruções que foram acrescentadas ao programa FORTRAN foram as que acionam o programa BINDER do B-6700.

As subrotinas DIAL1 e DIAL2 são, na verdade, procedures ALGOL, que foram produzidas pelo "G D" e previamente compiladas, antes da sua utilização no programa FORTRAN.

A maneira como estas procedures foram geradas pelo "G D" acompanha as normas do uso do "G D", descritas no capítulo IV.

1.3.3 - Os resultados da execução da procedure DIAL1, no decorrer da execução do programa FORTRAN, podem ser vistos nos esquemas das figuras (A-1.3) até (A-1.6).

As ações resultantes da escolha dos itens são apresentada na fig. (A-1.8).

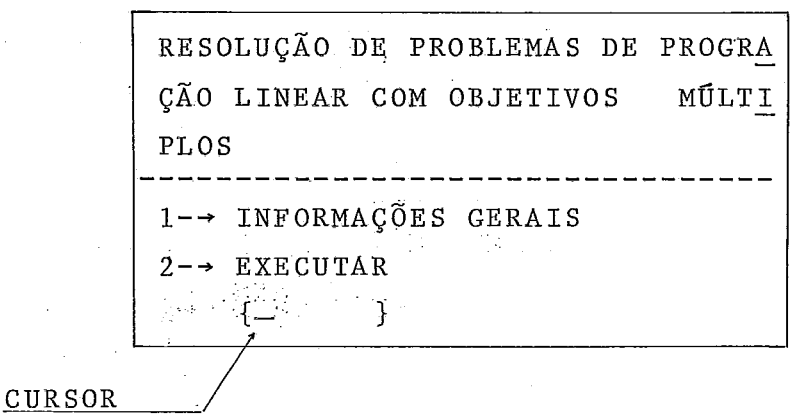


Fig. (A-1.3) - TELA 1

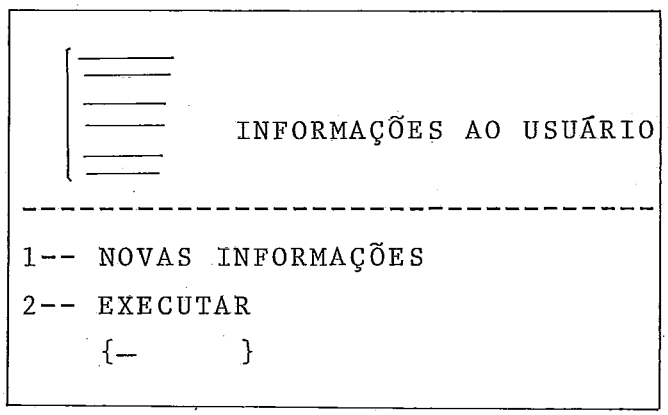


Fig. (A-1.4) TELA 2

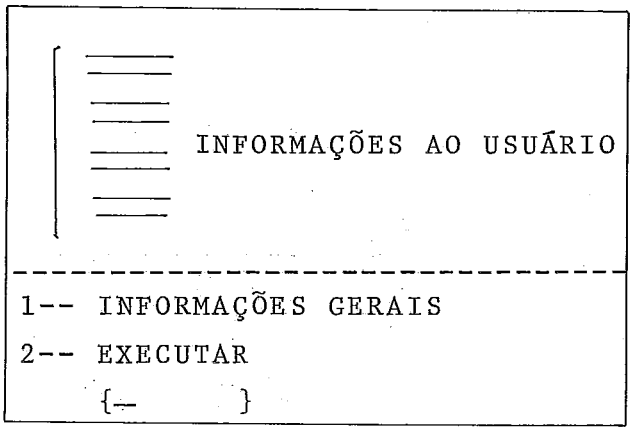


Fig. (A-1.5) TELA 3

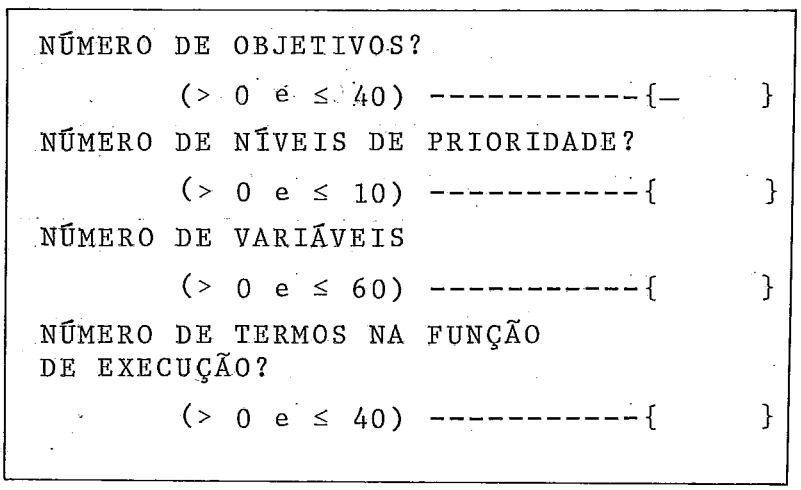


Fig. (A-1.6) TELA 4

1.3.4 - Os resultados da execução da procedure DIAL2, no decorrer da execução do programa FORTRAN, podem ser vistos no esquema da Fig. (A-1.7)

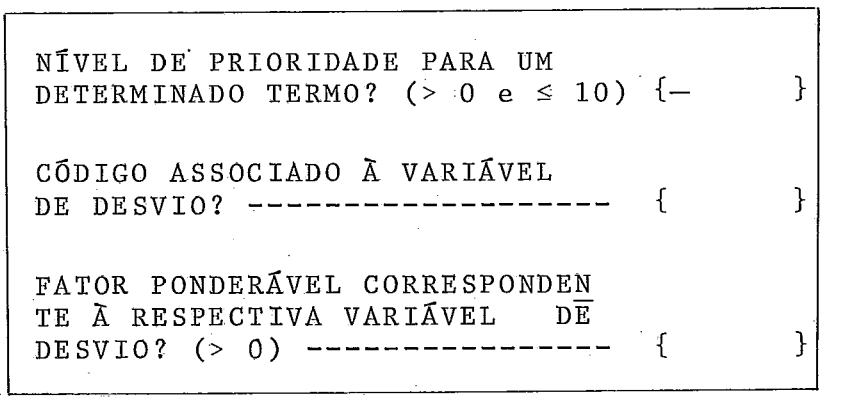


Fig. (A-1.7) TELA 5

1.3.5 - O Relacionamento com o Usuário

Nesta versão do programa FORTRAN,o primeiro relacionamento do usuário com o computador produz-se no momento em que a execução do programa atinge a chamada da procedure DIAL1.

Esta procedure começa sua execução apresentando, no vídeo o usuário a TELA 1.

- Na frente da TELA 1 e do restante das telas, que foram preparadas para serem apresentadas durante a execução da procedure DIAL1, as respostas do usuário e as consequentes ações do computador podem ser esquematizadas no resumo que é apresentado na Fig. (A-1.8).

TELA	RESPOSTA DO USUÁRIO	AÇÃO DO COMPUTADOR
1	Valor inteiro = 1 Valor inteiro = 2 Outra coisa	Apresentar TELA 2 " TELA 4 " TELA 1
2	Valor inteiro = 1 Valor inteiro = 2 Outra coisa	Apresentar TELA 3 " TELA 4 " TELA 2
3	Valor inteiro = 1 Valor inteiro = 2 Outra coisa	Apresentar TELA 1 " TELA 4 " TELA 3
4	Preenchimento certo de cada parâmetro	Fim da execução da procedure DIAL1 e continuação da execução do programa.
	Preenchimento errado de um ou mais parâmetros	Apresentação de mensagens de auxílio que interagem com o usuário na procura de novas informações

Fig. (A-1.8) - Ações do Computador diante das respostas do usuário.

No momento em que a execução do programa FORTRAN atinge a chamada da procedure DIAL2, produz-se o segundo relacionamento do usuário com o computador. Desta vez a execução da procedure DIAL2 apresenta, no vídeo do usuário a TELA 5.Fig. (A-1.7).

Na presença da TELA 5, o usuário poderá responder da seguinte maneira:

- preenchendo corretamente todos os parâmetros. Neste caso, a ação do computador será a continuação da execução do programa FORTRAN.

- preenchendo de maneira errada um ou mais parâmetros. Neste caso o computador apresentará as respectivas mensagens de auxílio, que interagem com o usuário na procura de novas informações.

1.3.6 - As Mensagens de Auxílio

Exemplo:

VÍDEO - Número de objetivos?
(> 0 < 40)

USUÁRIO - 4A

VÍDEO - A resposta recebida -- 4A
não é um número
Deseja fornecer um novo valor? S/N

USUÁRIO - S

VÍDEO - Número de objetivos?
(> 0 E < 40)

USUÁRIO - 20

VÍDEO - (Continua a execução)

Estas mensagens de auxílio, tal como foi visto no capítulo IV, são o resultado da execução das procedures de crítica e recuperação de erros que são compiladas junto a cada procedure gerada pelo "G D".

1.4 - A Sequência da Utilização do "G D" pelo Projetista

1.4.1 - A Geração da procedure DIAL1

a) Entrada das descrições foi realizada, através do uso de uma das procedures do "G D", a procedure CRIAR.

Os resultados da execução desta procedure podem ser vistos nas figuras (A-1.9) e (A-1.10).

```
tabel 000021
siste nome  DIAL1       param S     param 01    NOBJ        tipo  I
param 02    NPRI        tipo  I     param 03    NVAR        tipo  I
param 04    NTAF
descricao   MENU1        tipo  G
item  1     acao  C     tela  colec COLEC1      exec              displ
####
item  2     acao  A     displ MENAJUDA1
descricao   MENAJUDA1   tipo  G
item  1     acao  C     tela  colec COLEC1      exec              displ
####
item  2     acao  A     displ MENAJUDA1
descricao   COLEC1      tipo  D     numer param 04
param 01    nome  NOBJ        tipo  I     rest  nome  VERFNOBJ    help
nome  AJUDANOBJ
param 02    nome  NPRI        tipo  I     rest  nome  VERFNPRI    help
nome  AJUDANOBJ
param 03    nome  NVAR        tipo  I     rest  nome  VERFNVAR    help
nome  AJUDANOBJ
param 04    nome  NTAF        tipo  I     rest  nome  VERFNTAF    help
nome  AJUDANOBJ
MENU1          4.    4.G         2.    1.
MENAJUDA1      8.    4.G         2.    2.
COLEC1        12.    9.D         4.   -1.
```

Fig. (A-1.9)

Arquivo DIAL1/AÇÕES

```
TABEL 000040
              RESOLUCAO DE PROBLEMAS DE PROGRAMACAO LINEAR
                       COM  OBJETIVOS  MULTIPLOS
-------------------------------------------------------------------------
1--> E X E C U C A O
2--> INSTRUCOES

[             ]
          O OBJETIVO DESTE PACOTE COMPUTACIONAL E' RESOLVER
        PROBLEMAS DE PROGRAMACAO LINEAR COM OBJETIVOS MULTIPLOS
-------------------------------------------------------------------------

  1) INICIALMENTE DEVEM-SE DEFINIR OS SEGUINTES PARAMETROS, EN FACE
     AO MODELO MATEMATICO CORRESPONDENTE:
     A)  NUMERO DE OBJETIVOS;  B)  NUMERO DE NIVEIS DE PRIORIDADE
     C)  NUMERO DE VARIAVEIS DE DECISAO;
     D)  NUMERO DE TERMOS NA FUNCAO DE EXECUCAO.
  2) SEGUIDAMENTE O USUARIO TERA' QUE CRIAR UM ARQUIVO DE DADOS
     DE NOME "DAT01" CONTENDO OS VALORES DOS COEFICIENTES CON-
     FORME EXPLICA-SE NO MANUAL CORRESPONDENTE.
  3) APOS O FORNECIMENTO DESTES VALORES, O USUARIO DEVERA REPETI-
     DAMENTE DIGITAR, OS VALORES DAS VARIAVEIS DE DESVIO NA FUN -
     CAO DE EXECUCAO.
  UM EXEMPLO COMPLETO DA PREPARACAO DE TODOS ESTES DADOS, E' APRE-
  SENTADAO NO MANUAL CORRESPONDENTE.
-------------------------------------------------------------------------
1--> E X E C U C A O
2--> INSTRUCOES
[             ]
NUMERO DE OBJETIVOS ?
 ( MAIOR QUE 0  E  MENOR QUE  40  ) ........[   ]

NUMERO DE NIVEIS DE PRIORIDADES ?
 ( MAIOR QUE 0  E  MENOR OU IGUAL A 10).....[   ]

NUMERO DE VARIAVEIS ?
 ( MAIOR QUE 0  E  MENOR OU IGUAL A 60 ).....[   ]

NUMERO DE TERMINOS NA FUNCAO DE EXECUCAO ?
 ( MAOIR QUE 0  E  MENOR OU IGUAL A 40 ).....[   ]
MENU1           1.    7.G          1.
MENAJUDA1       8.   21.G          2.
COLEC1         29.   11.D         -1.
```

Fig. (A-1.10)

Arquivo DIAL1/TEXTOS

b) Na geração de código fonte ALGOL

A geração das procedures ALGOL, foi feita mediante a utilização de outra procedure do "G D", a procedure GERADOR.

A execução desta procedure teve como entradas:

O arquivo DIAL1/ACOES, o arquivo DIAL1/TEXTOS e o nome do diálogo, neste caso "DIAL1".

Os resultados da execução da procedure gerador podem ser vistos nas fig. (A-1.11), (A-1.12), (A-1.13), (A-1.14).

```
$SET SEPARATE
PROCEDURE DIAL1 (NOBJ,NPRI,NVAR,NTAF);
INTEGER NOBJ,NPRI,NVAR,NTAF;
BEGIN
  BOOLEAN PULAR,VALIDA;
  INTEGER PREL,TOTREG;REAL ARRAY TABDADC0:99,0:2];
  FILE ACOES(KIND=DISK);
  FILE TEXTOS(KIND=DISK);
  FILE TELA(KIND=DC,MYUSE=IO,FILETYPE=7,
            MAXRECSIZE=30,BLOCKSIZE=30);
  FORMAT CLEAR (4"0C");
  FORMAT FORMS (4"12",4"3C",4"03");
$ INCLUDE "DIAL1/FOWAR."
$ INCLUDE "APLICACAO."
$ INCLUDE "AJUDA."
$ INCLUDE "DIAL1/EXEC."
$ INCLUDE "DIAL1/PROC."


  BEGIN
    REPLACE TEXTOS.TITLE BY "DIAL1/TEXTOS.";
    CARREGA;
    MENU1;
    WHILE PULAR DO  EXEC(PREL);
    CLOSE(TELA);
  END;
END;
```

Fig. (A-1.11)- Arquivo DIAL1

```
PROCEDURE MENU1 ;
FORWARD;
PROCEDURE MENAJUDA1 ;
FORWARD;
PROCEDURE COLEC1 (NOBJ,NPRI,NVAR,NTAF);
INTEGER NOBJ,NPRI,NVAR,NTAF;
FORWARD;
PROCEDURE EXEC(PREL);
INTEGER PREL; FORWARD;
```

Fig. (A-1.12) - Arquivo DIAL1/FOWAR

```
PROCEDURE EXEC(PREL);
INTEGER PREL;
BEGIN
  CASE PREL OF
  BEGIN
    01 : MENU1 ;
    02 : MENAJUDA1 ;
  END;
END;
```

Fig. (A-1.13)

Arquivo DIAL1/EXEC

```
PROCEDURE MENU1 ;
BEGIN
  POINTER PT;
  REAL ARRAY RESPOSTA[0:12];
  INTEGER CONT,POS,TAM,POSREL,DIGITO ;
  EBCDIC ARRAY BUFFER[0:2000],CADEIA[0:72] ;
  BEGIN
    PULAR:=FALSE;
    VALIDA:=TRUE;
    POS:=000001;  TAM:=000007;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL
                  TAM -1 DO PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72),FOR CONT:=0 STEP 1 UNTIL TAM -1 DO
                    PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    READ (TELA,(A72),CADEIA);
    ANALISE(CADEIA,RESPOSTA[*],DIGITO,POSREL);
    IF (DIGITO GEQ 0)
       THEN CASE DIGITO OF
       BEGIN
         1   :      COLEC1 (NOBJ,NPRI,NVAR,NTAF);
         2   :      MENAJUDA1 ;
          ELSE:MENU1 ;
       END
       ELSE BEGIN
              IF POSREL GEQ 0
                 THEN BEGIN
                        PULAR:=TRUE;
                        PREL:=POSREL;
                      END
                 ELSE MENU1 ;
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-1.14) - Arquivo DIAL1/PROC

```
PROCEDURE MENAJUDA1 ;
BEGIN
  POINTER PT;
  REAL ARRAY RESPOSTA[0:12];
  INTEGER CONT,POS,TAM,POSREL,DIGITO ;
  EBCDIC ARRAY BUFFER[0:2000],CADEIA[0:72] ;
  BEGIN
    PULAR:=FALSE;
    VALIDA:=TRUE;
    POS:=000008;  TAM:=000021;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL
                    TAM -1 DO PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72),FOR CONT:=0 STEP 1 UNTIL TAM -1 DO
                      PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    READ (TELA,(A72),CADEIA);
    ANALISE(CADEIA,RESPOSTA[*],DIGITO,POSREL);
    IF (DIGITO GEQ 0)
       THEN CASE DIGITO OF
       BEGIN
         1    :      COLEC1 (NOBJ,NPRI,NVAR,NTAF);
         2    :      MENAJUDA1 ;
          ELSE:MENAJUDA1 ;
       END
       ELSE BEGIN
              IF POSREL GEQ 0
                 THEN BEGIN
                        PULAR:=TRUE;
                        PREL:=POSREL;
                      END
                 ELSE MENAJUDA1 ;
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-1. 14)

Arquivo DIAL1/PROC (continuação)

```
PROCEDURE COLEC1 (NOBJ,NPRI,NVAR,NTAF);
INTEGER NOBJ,NPRI,NVAR,NTAF;
BEGIN
  INTEGER POS,TAM,POSREL,ID,IL,CONT;
  INTEGER ARRAY ERRO,NUMCAR[0:12];
  REAL ARRAY RESPOSTA[1:04,0:12];
  EBCDIC ARRAY BUFFER[0:2000],CADEIA[0:72];
  POINTER PT,PBU;
  BEGIN
    NUMCAR[01]:=03 ;
    NUMCAR[02]:=03 ;
    NUMCAR[03]:=03 ;
    NUMCAR[04]:=03 ;
    POS:=000029;
    TAM:=000011;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL TAM-1 DO

        PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72), FOR CONT:=0 STEP 1  UNTIL TAM-1 DO
        PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    ID:=04 ;
    READ(TELA,(A180),BUFFER);
    PBU:=BUFFER; REPLACE CADEIA BY " " FOR 72;
    FOR IL:=1 STEP 1 UNTIL ID DO
    BEGIN
      REPLACE CADEIA BY PBU:PBU FOR NUMCAR[IL];
      ANALISE(CADEIA,RESPOSTA[IL,*],CONT,POSREL);
    END;
    VERFNOBJ(RESPOSTA[01,*],ERRO[01]);
    VERFNPRI(RESPOSTA[02,*],ERRO[02]);
    VERFNVAR(RESPOSTA[03,*],ERRO[03]);
    VERFNVAR(RESPOSTA[04,*],ERRO[04]);
    FOR IL:=1 STEP 1 UNTIL ID DO
    IF VALIDA AND  (ERRO[IL] > 0)
       THEN CASE IL OF
            BEGIN
             01 : AJUDANOBJ(RESPOSTA[01,*],VALIDA,ERRO[IL],IL
                            );
             02 : AJUDANOBJ(RESPOSTA[02,*],VALIDA,ERRO[IL],IL
                            );
             03 : AJUDANOBJ(RESPOSTA[03,*],VALIDA,ERRO[IL],IL
                            );
             04 : AJUDANOBJ(RESPOSTA[04,*],VALIDA,ERRO[IL],IL
                            );
            END;
    IF VALIDA
       THEN BEGIN
              NOBJ:=RESPOSTA[01,0];
              NPRI:=RESPOSTA[02,0];
              NVAR:=RESPOSTA[03,0];
              NTAF:=RESPOSTA[04,0];
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-1.14)

Arquivo DIAL1/PROC (continuação)

c) A seguir apresentamos as procedures de critica e recuperação de erros não produzidas mediante o uso de "G D", porém necessárias para a verificação das respostas do usuário e para a emissão das respectivas mensagens de auxílio. Fig. (A-1.15).

```
PROCEDURE VERFID(RESP,ERR,LIMI,LIMS);
 REAL ARRAY RESP[*]; INTEGER ERR; REAL LIMI, LIMS;
BEGIN
  IF RESP[2] NEQ "R      " THEN ERR:=1;
  IF ((ERR=0 ) AND ((RESP[0] LEQ LIMI) OR (RESP[0] GTR LIMS)))
     THEN ERR:=2;
END;

PROCEDURE VERFNOBJ(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*]; INTEGER ERRO;
BEGIN
  ERRO:=0; VERFID(RESPOSTA,ERRO,0.,40);
END;

  PROCEDURE VERFNPRI(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*]; INTEGER ERRO;
BEGIN
   ERRO:=0; VERFID(RESPOSTA,ERRO,0.,10);
END;

   PROCEDURE VERFNVAR(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*]; INTEGER ERRO;
BEGIN
    ERRO:=0; VERFID(RESPOSTA,ERRO,0.,60);
END;

   PROCEDURE VERFNTAF(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*]; INTEGER ERRO;
BEGIN
     ERRO:=0; VERFID(RESPOSTA,ERRO,0.,40);
END;
```

Fig. (A-1.15) - Arquivo DIAL1/AJUDA

(Procedures de Critica e recuperação de erros)

```
PROCEDURE NOVAMENTE(RESP,LIMI,LIMS,PARM,SIM,ERR);
REAL ARRAY RESP[*]; INTEGER ERR; REAL LIMI,LIMS,PARM,SIM;
BEGIN
  EBCDIC ARRAY CAD[0:72]; INTEGER POSREL,DIGITO;
  WRITE(TELA,CLEAR);
  IF ERR=1 THEN WRITE(TELA,("RESPOSTA(",I2,"). CADEIA RECEBIDA:)",
                A6,/,"NAO E' UM NUMERO"),PARM,RESP[0])
           ELSE WRITE(TELA,("RESPOSTA(",I2,"). CADEIA RECEBIDA:)",
                F12.6,/,"ESTA FORA DOS LIMITES ",A6,"---",A6),
                PARM,RESP[0],LIMI,LIMS);
  WRITE(TELA,("DESEJA FORNECER UM NOVO VALOR ?  S/N  "));
  READ(TELA,<A6>,SIM);
  IF SIM="S     "
     THEN BEGIN
            WRITE(TELA,(" ")); READ(TELA,<A72>,CAD);
            ANALISE(CAD,RESP,DIGITO,POSREL);
            END;
END; % DA NOVAMENTE

PROCEDURE AJUDANOBJ(RESPOSTA,CERTA,ERRO,PARM);
  REAL ARRAY RESPOSTA[*]; BOOLEAN CERTA; INTEGER ERRO,PARM;
BEGIN
  REAL SN,LIMI,LIMS;
  CASE PARM OF
  BEGIN
    1: LIMS:="40    ";
    2: LIMS:="10    ";
    3: LIMS:="60    ";
    4: LIMS:="40    ";
  END;                          LIMI:="0     ";
   NOVAMENTE(RESPOSTA,LIMI,LIMS,PARM,SN,ERRO);
IF SN="S     "
   THEN BEGIN
          CASE PARM OF
          BEGIN
            1: VERFNOBJ(RESPOSTA,ERRO);
            2: VERFNPRI(RESPOSTA,ERRO);
            3: VERFNVAR(RESPOSTA,ERRO);
            4: VERFNTAF(RESPOSTA,ERRO);
          END;
          IF ERRO >0 THEN AJUDANOBJ(RESPOSTA,CERTA,ERRO,PARM)
                     ELSE CERTA:=TRUE;
        END
    ELSE CERTA:=FALSE;
END; % DA AJUDANOBJ
```

Fig. (A-1.15)

Arquivo DIAL1/AJUDA (continuação)

1.4.2 - Na Geração da Procedure DIAL2

A sequência da utilização do "G D" neste caso, é similar à que foi apresentada para o caso da geração da procedure DIAL1, por esta razão, nos limitaremos à apresentação dos resultados.

a) No ingresso das "descrições dos diálogos" contidos na execução da procedure DIAL2. Figs. (A-1.16) e (A-1.17).

```
tabel 000013
siste nome  DIAL2       param S     param 01    IPRI        tipo  I
param 02    ISUB        tipo  I     param 03    WHTF        tipo  R
descricao   MENU1       tipo  G
item  ####  acao  C     tela  colec COLEC2      exec              displ
####
descricao   COLEC2      tipo  D     numer param 03
param 01    nome  IPRI        tipo  I     rest  nome  VERFIPRI    help
nome  AJUDAIPRI
param 02    nome  ISUB        tipo  I     rest  nome  VERFISUB    help
nome  AJUDAIPRI
param 03    nome  WHTF        tipo  R     rest  nome  VERFWHTF    help
nome  AJUDAIPRI
MENU1          3.     3.G          1.    1.
COLEC2         6.     7.D          3.   -1.
```

Fig. (A-1.16)

Arquivo DIAL2/AÇÕES

```
TABEL 000009
NIVEL DE PRIORIDADE PARA ESTE TERMO  DA FUNCAO DE EXECUCAO ?
(MAIOR QUE 0  E  MENOR OU IGUAL A 10 ).........[      ]

CODIGO ASSOCIADO A' VARIAVEL DE DESVIO  ?
( - PARA M     E   + PARA P ) ................[      ]

FATOR  CORRESPONDENTE A' RESPECTIVA VARIAVEL DE DESVIO  ?
( MAIOR QUE CERO )     ........................[          ]
MENU1            1.    0.G          1.
COLEC2           1.    8.D         -1.
```

Fig. (A-1.17) - Arquivo DIAL2/TEXTOS

b) Na geração do código fonte ALGOL;

Figs. (A-1.18), (A-1.19), (A-1.20) e (A-1.21)

```
$SET SEPARATE
PROCEDURE DIAL2 (IPRI,ISUB,WHTF);
INTEGER IPRI,ISUB;
REAL    WHTF;
BEGIN
  BOOLEAN PULAR,VALIDA;
  INTEGER PREL,TOTREG;REAL ARRAY TABDAD[0:99,0:2];
  FILE ACOES(KIND=DISK);
  FILE TEXTOS(KIND=DISK);
  FILE TELA(KIND=DC,MYUSE=IO,FILETYPE=7,
            UNITS=CHARACTERS,MAXRECSIZE=80,BLOCKSIZE=1600);
  FORMAT CLEAR (4"0C", 79 ( 4"00" ) );
  FORMAT FORMS (4"12",4"3C",4"03");
$ INCLUDE "DIAL2/FOWAR."
$ INCLUDE "APLICACAO."
$ INCLUDE "AJUDA."
$ INCLUDE "DIAL2/EXEC."
$ INCLUDE "DIAL2/PROC."


  BEGIN
    REPLACE TEXTOS.TITLE BY "DIAL2/TEXTOS.";
    CARREGA;
    MENU1;
    WHILE PULAR DO  EXEC(PREL);
    CLOSE(TELA);
  END;
 END;
```

Fig. (A-1.18) - Arquivo DIAL2

```
PROCEDURE MENU1 ;
FORWARD;
PROCEDURE COLEC2 (IPRI,ISUB,WHTF);
INTEGER IPRI,ISUB;
REAL    WHTF;
FORWARD;
PROCEDURE EXEC(PREL);
INTEGER PREL; FORWARD;
```

Fig. (A-1.19)

Arquivo DIAL2/FOWAR

```
PROCEDURE EXEC(PREL);
INTEGER PREL;
BEGIN
  CASE PREL OF
  BEGIN
    01 : MENU1 ;
  END;
END;
```

Fig. (A-1.20)

Arquivo DIAL2/EXEC

```
PROCEDURE MENU1 ;
BEGIN
  BEGIN
    PULAR:=FALSE;
    VALIDA:=TRUE;
    COLEC2 (IPRI,ISUB,WHTF);
  END;
  CLOSE(TELA);
END;
```

Fig. (A-1.21) - Arquivo DIAL2/PROC

```
PROCEDURE COLEC2 (IPRI,ISUB,WHTF);
INTEGER IPRI,ISUB;
REAL    WHTF;
BEGIN
  INTEGER POS,TAM,POSREL,ID,IL,CONT;
  INTEGER ARRAY ERRO,NUMCAR[0:12];
  REAL ARRAY RESPOSTA[1:03,0:12];
  EBCDIC ARRAY BUFFER[0:2000],CADEIA[0:72];
  POINTER PT,PBU;
  BEGIN
    NUMCAR[01]:=06 ;
    NUMCAR[02]:=06 ;
    NUMCAR[03]:=11 ;
    POS:=000001;
    TAM:=000008;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL TAM-1 DO

        PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72), FOR CONT:=0 STEP 1  UNTIL TAM-1 DO
        PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    ID:=03 ;
    READ(TELA,(A80),BUFFER);
    PBU:=BUFFER; REPLACE CADEIA BY " " FOR 72;
    FOR IL:=1 STEP 1 UNTIL ID DO
    BEGIN
      REPLACE CADEIA BY PBU:PBU FOR NUMCAR[IL];
      ANALISE(CADEIA,RESPOSTA[IL,*],CONT,POSREL);
    END;
    VERFIPRI(RESPOSTA[01,*],ERRO[01]);
    VERFISUB(RESPOSTA[02,*],ERRO[02]);
    VERFWHTF(RESPOSTA[03,*],ERRO[03]);
    FOR IL:=1 STEP 1 UNTIL ID DO
    IF VALIDA AND  (ERRO[IL] > 0)
       THEN CASE IL OF
            BEGIN
             01 : AJUDAIPRI(RESPOSTA[01,*],VALIDA,ERRO[IL],IL
                                  );
             02 : AJUDAIPRI(RESPOSTA[02,*],VALIDA,ERRO[IL],IL
                                  );
             03 : AJUDAIPRI(RESPOSTA[03,*],VALIDA,ERRO[IL],IL
                                  );
            END;
    IF VALIDA
       THEN BEGIN
              IPRI:=RESPOSTA[01,0];
              ISUB:=RESPOSTA[02,0];
              WHTF:=RESPOSTA[03,0];
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-1.21)

Arquivo DIAL2/PROC (continuação)

c) As procedures de critica e recuperação de erros não produzidas pelo "G D", porém necessarias para a verificação das respostas do usuário e para a emissão das respectivas mensagens de auxílio. Fig. (A-1.22).

```
 PROCEDURE VERFIPRI(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*]; INTEGER ERRO;
BEGIN
   ERRO:=0; IF RESPOSTA[2] NEQ "R      " THEN ERRO:=1;
   IF ((ERRO=0) AND ((RESPOSTA[0] LEQ 0) OR (RESPOSTA[0] GTR 10)))
      THEN ERRO:=2;
END;

PROCEDURE VERFISUB(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*]; INTEGER ERRO;
BEGIN
  ERRO:=0; IF RESPOSTA[2] NEQ "R      " THEN ERRO:=1;
END;

PROCEDURE VERFWHTF(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*]; INTEGER ERRO;
BEGIN
  ERRO:=0; IF RESPOSTA[2] NEQ "R      " THEN ERRO:=1;
  IF ((ERRO=0) AND (RESPOSTA[0] LEQ 0)) THEN ERRO:=2;
END;

PROCEDURE NOVAMENTE(RESP,LIMI,LIMS,PARM,SIM,ERR);
REAL ARRAY RESP[*]; INTEGER ERR; REAL LIMI,LIMS,PARM,SIM;
BEGIN
  EBCDIC ARRAY CAD[0:72]; INTEGER POSREL,DIGITO;
  WRITE(TELA,CLEAR);
  IF ERR=1 THEN WRITE(TELA,("RESPOSTA(",I2,"). CADEIA RECEBIDA:)",
                A6,/,"NAO E' UM NUMERO"),PARM,RESP[0])
           ELSE WRITE(TELA,("RESPOSTA(",I2,"). CADEIA RECEBIDA:)",
                F12.6,/,"ESTA FORA DOS LIMITES ",A6,"---",A6),
                PARM,RESP[0],LIMI,LIMS);
  WRITE(TELA,("DESEJA FORNECER UM NOVO VALOR ?  S/N  "));
  READ(TELA,(A6),SIM);
  IF SIM="S     "
     THEN BEGIN
            WRITE(TELA,(" ")); READ(TELA,(A72),CAD);
            ANALISE(CAD,RESP,DIGITO,POSREL);
            END;
END; % DA NOVAMENTE
```

Fig. (A-1.22)

Arquivo DIAL2/AJUDA - Procedures de Crítica e Recuperação de Erros).

```
PROCEDURE AJUDAIPRI(RESPOSTA,CERTA,ERRO,PARM);
 REAL ARRAY RESPOSTA[*]; BOOLEAN CERTA; INTEGER ERRO,PARM;
BEGIN
  REAL SN,LIMI,LIMS;
  CASE PARM OF
  BEGIN
     1: LIMS:="10     "; LIMI:="0.0    ";
     2: LIMS:="       "; LIMI:="       ";
     3: LIMS:="       "; LIMI:="0.0    ";
  END;                      LIMI:="0      ";
   NOVAMENTE(RESPOSTA,LIMI,LIMS,PARM,SN,ERRO);
IF SN="S      "
   THEN BEGIN
          CASE PARM OF
          BEGIN
            1: VERFIPRI(RESPOSTA,ERRO);
            2: VERFISUB(RESPOSTA,ERRO);
            3: VERFWHTF(RESPOSTA,ERRO);
          END;
          IF ERRO >0 THEN AJUDAIPRI(RESPOSTA,CERTA,ERRO,PARM)
                     ELSE CERTA:=TRUE;
        END
    ELSE CERTA:=FALSE;
END; % DA AJUDANOBJ
```

Fig. (A-1.22) - Arquivo DIAL2/AJUDA - Procedures de Crítica e Recuperação de Erros). (continuação)

A P Ê N D I C E 1

EXEMPLO 2

Neste segundo exemplo, a ilustração do uso do "G D" será feita utilizando-se programas escritos na linguagem ALGOL do B-6700.

2.1 - Os Objetivos destes Programas de Aplicação

O software de aplicação, que será descrito a seguir, faz parte do trabalho de pesquisa da Tese de Doutorado "Gerador de Analisadores Sintáticos R*S, que atualmente está sendo desenvolvida na área de sistemas-computação da COPPE-UFRJ.

- Procedure GRAMATICA

  Esta procedure realiza a análise de uma gramática R*S, que previamente deverá estar gravada em um arquivo de disco.

  Ela recebe, como parâmetro de entrada, o nome desse arquivo e seus resultados são armazenados em outros arquivos no disco.

- Procedure GERADOR/G

  Esta procedure gera as tabelas de empilhamento,salto e redução que compõem o algoritmo de análise R*S, proposto por J.L. Rangel e S. M. Schneider (1983).

- Procedure RESULTADOS

  Ela recebe, como parâmetros de entrada, valores Sim

ou Não fornecidos pelo usuário do sistema R*S e que condicionarão a emissão dos resultados do sistema R*S.

2.2 - A Interface com o Usuário

Como pode ser visto na descrição das procedures de aplicação, elas podem ser tomadas perfeitamente como módulos, com suas funções totalmente definidas.

Por esta razão, o projeto da interface com o usuário, deste software de aplicação, foi voltado para outra das possibilidades do uso do "G D", em particular, aquela que permite a geração da interface com o usuário, necessária para a utilização de programas de aplicação já existentes.

Esquemas dessa interface, que começam com a apresentação da "TELA1", podem ser vistos nas Figs. (A-2.1), (A-2.2), (A-2.3), (A-2.4) e (A-2.5). As ações resultantes da escolha dos ítens, são apresentadas na fig. (A-2.10).

```
MENU1
     GERADOR DE ANALISADORES
              R*S
        PARA GRAMÁTICAS BNF
-----------------------------------
1-- INFORMAÇÕES GERAIS
2-- EXECUTAR
3-- ENCERRAR
    {       }
```

CURSOR

Fig. (A-2.1) TELA1

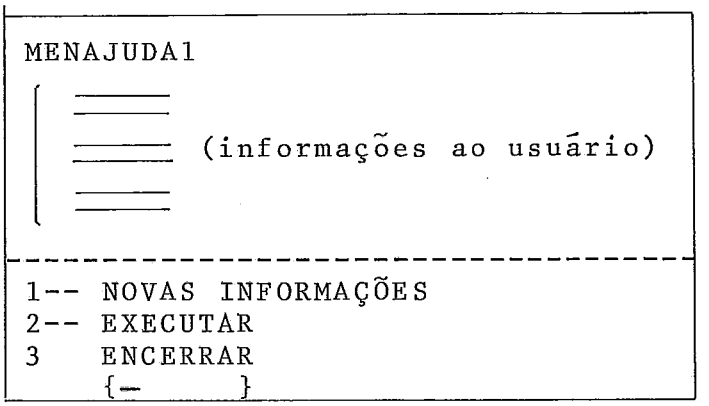


Fig. (A-2.2) TELA 2

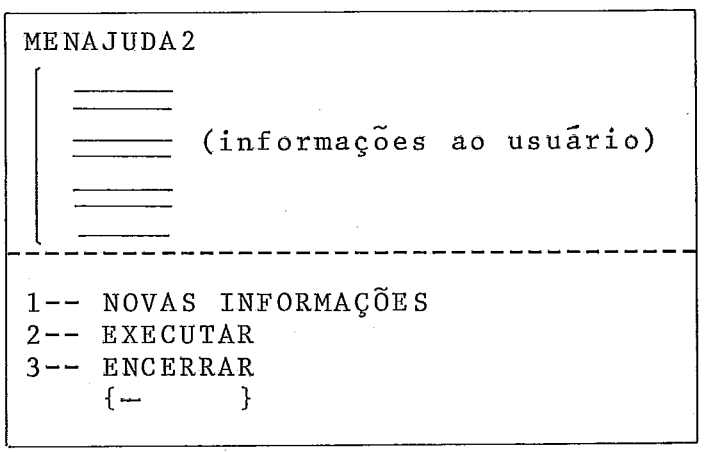


Fig. (A-2.3) TELA 3

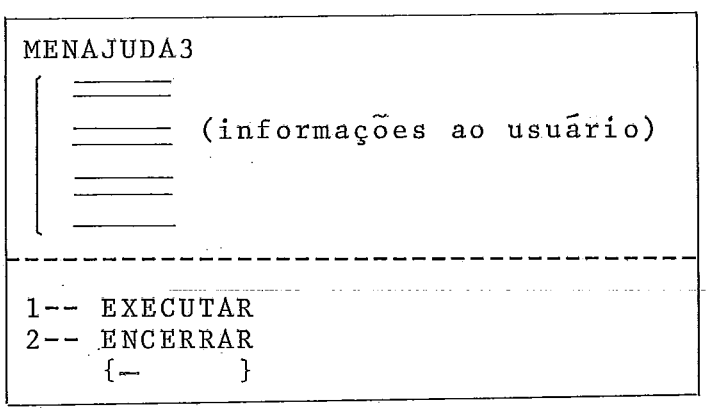


Fig. (A.2.4) TELA 4

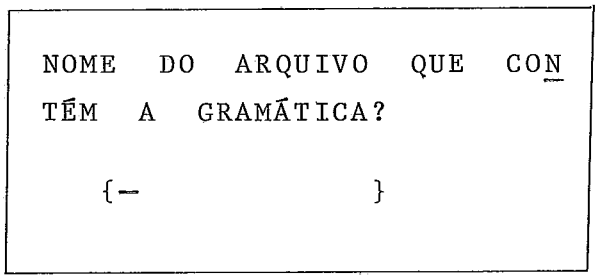


Fig. (A-2.5) TELA 5

A execução da procedure GRAMATICA apresenta duas opções:

a) análise de um arquivo, contendo uma gramática livre de erros;

b) análise de um arquivo, contendo uma gramática na qual foram detetados erros em tempo de execução.

Devido a esta dualidade, a procedure GRAMÁTICA inclui, nas suas instruções ALGOL, uma chamada a uma procedure gerada pelo "G D", concretamente uma chamada ao MENU2. Tal chamada permite a continuidade da execução, caso não tenham acontecido erros.

Na presença de erros na análise das gramáticas, eles são tratados separadamente pela procedure GRAMÁTICA. Este fato é apresentado no esquema da Fig. (A-2.6).

```
PROCEDURE GRAMÁTICA (NOMEARQUIVO)
EBCDIC ARRAY NOMEARQUIVO  *
BEGIN
   ========
   ========   INSTRUÇÕES ALGOL
   ========
   IF "ACONTECERAM ERROS"
     THEN      BEGIN
                  ======
                  ======
               END
     ELSE MENU2;
END;
```

A execução da procedure MENU2 apresenta, no vídeo do usuário, a "TELA6" Fig. (A-2.7).

```
MENU2
      FIM DA ANÁLISE DA GRAMATICA
                RECEBIDA
-----------------------------------

1-- GERAÇÃO
2-- ENCERRAR
    {-      }
```

Fig. (A-2.7) TELA 6

A execução da procedure GERADOR/G é seguida pela execução de procedure RESULTADOS.

Como esta procedure RESULTADOS é uma procedure com parâmetros, antes da sua execução, o usuário recebe a TCP apresentada na TELA 7, Fig. (A-2.9).

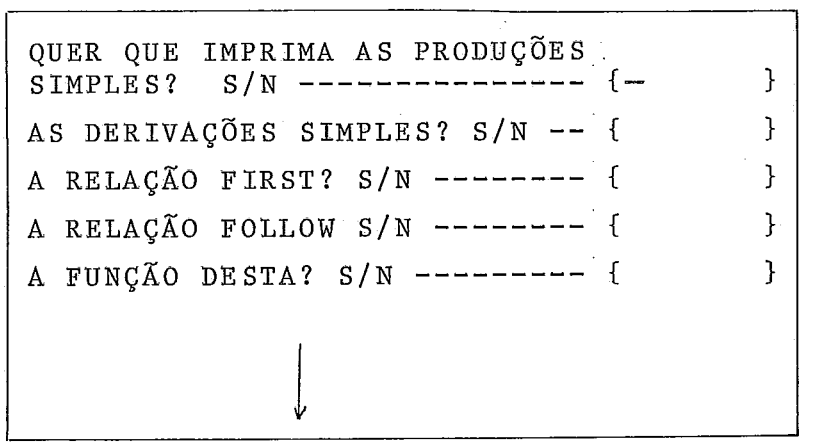


Fig. (A-2.9) TELA 7

2.3 - As Respostas do Usuário

As respostas do usuário, a cada uma das telas descritas, junto com as correspondentes ações do computador, podem ser vistas no resumo da Fig. (A-2.10).

TELA	RESPOSTA AO USUÁRIO	AÇÃO DO COMPUTADOR
1	Valor inteiro 1 Valor inteiro 2 Valor inteiro 3	Apresentar TELA 2 Apresentar TELA 5 Fim da Execução
2	Valor inteiro 1 Valor inteiro 2 Valor inteiro 3	Apresentar TELA 3 Apresentar TELA 5 Fim da Execução
3	Valor inteiro 1 Valor inteiro 2 Valor inteiro 3	Apresentar TELA 4 Apresentar TELA 5 Fim da Execução
4	Valor inteiro 1 Valor inteiro 2	Apresentar TELA 5 Fim da Execução
5	Identificador Válido	Execução da procedure gramática e possível apresentação da TELA6
	Identificador Invalido	Mensagem de Auxílio
6	Valor inteiro 1	Execução da procedure gerador, seguida pela apresentação da TELA7
	Valor inteiro 2	Fim da execução
	Preenchimento correto dos Parâmetros	Execução da procedure resultados, seguida pelo fim da execução
	Preenchimento errado de um ou mais parâmetros	Mensagem de Auxílio

Tal como foi visto no capítulo IV, Secção 4.3.1, as respostas do usuário, na presença de menus, podem ser os nomes de outros menus. Com estas alternativas, completa-se o resumo de respostas do usuário e ações do computador - Fig. (A-2.11).

EM QUALQUER UMA DAS SEGUINTES TELAS	O USUÁRIO PODERÁ RESPONDER COM ALGUM DOS SEGUINTES NOMES	EM CUJO CASO O COMPUTADOR APRESENTARÁ, A
1, 2, 3, 4, 5, 6	MENU1	TELA 1
	MENU2	TELA 6
	MENAJUDA1	TELA 3
	MENAJUDA2	TELA 3
	MENAJUDA3	TELA 4
	Outra cadeia de caracteres não prevista	A mesma tela que estava sendo apresentada

Fig. (A-2.11)

2.4 - Mensagens de Auxílio

Acompanham a descrição feita no exemplo 1 deste apêndice, porém de acordo com as destrições e procedimentos estabelecidos para este exemplo 2.

2.5 - A Sequência da Utilização do "G D" pelo Projetista

a) Na definição da interface usuário-computador:

Nesta etapa, foram definidos todos os diálogos necessários para a utilização dos programas: GRAMATICA, GERADOR/G e RESULTADOS.

Nas figuras (A-2.12) e (A-2.13) apresentamos resulta

dos da entrada das descrições desses diálogos.

```
tabel 000045
siste nome  DIALOGO      param N
descricao   MENU1        tipo  G
item  1     acao  A      displ MENAJUDA1
item  2     acao  C      tela  colec COLEC1        exec  GRAMATICA   displ
####
item  3     acao  E      exec  FIMSERGIO   displ ####
descricao   COLEC1       tipo  D     numer param 01
param 01    nome  NARQ         tipo  S     rest  nome  VERFNARQ    help
nome  AJUDANARQ
descricao   MENAJUDA1    tipo  G
item  1     acao  A      displ MENAJUDA2
item  2     acao  C      tela  colec COLEC1        exec  GRAMATICA   displ
####
item  3     acao  E      exec  FIMSERGIO   displ ####
descricao   MENAJUDA2    tipo  G
item  1     acao  A      displ MENAJUDA3
item  2     acao  C      tela  colec COLEC1        exec  GRAMATICA   displ
####
item  3     acao  E      exec  FIMSERGIO   displ ####
descricao   MENAJUDA3    tipo  G
item  1     acao  C      tela  colec COLEC1        exec  GRAMATICA   displ
####
item  2     acao  E      exec  FIMSERGIO   displ ####
descricao   MENU2        tipo  G
item  1     acao  E      exec  GERADOR     exec  RESULTADOS  tela colec
COLEC2      displ ####
item  2     acao  E      exec  FIMSERGIO   displ ####
descricao   COLEC2       tipo  D     numer param 08
param 01    nome  P1           tipo  R     rest  nome  VERF        help
nome  HELP
param 02    nome  P2           tipo  R     rest  nome  VERF        help
nome  HELP
param 03    nome  P3           tipo  R     rest  nome  VERF        help
nome  HELP
param 04    nome  P4           tipo  R     rest  nome  VERF        help
nome  HELP
param 05    nome  P5           tipo  R     rest  nome  VERF        help
nome  HELP
param 06    nome  P6           tipo  R     rest  nome  VERF        help
nome  HELP
param 07    nome  P7           tipo  R     rest  nome  VERF        help
nome  HELP
param 08    nome  P8           tipo  R     rest  nome  VERF        help
nome  HELP
MENU1          2.    2.G          3.    1.
COLEC1         7.    3.D          1.   -1.
MENAJUDA1     10.    5.G          3.    2.
MENAJUDA2     15.    5.G          3.    3.
MENAJUDA3     20.    4.G          2.    4.
MENU2         24.    4.G          2.    5.
COLEC2        28.   17.D          8.   -1.
```

Fig. (A-2.12)

Arquivo <DIALOGO>/ AÇÕES

```
TABEL 000089
        G E R A D O R    D E    A N A L I S A D O R E S
                              R * S
          P A R A   G R A M A T I C A S   " B N F "
-------------------------------------------------------------------------
 1--> INFORMACOES  GERAIS  SOBRE  O  USO  DESTE  GERADOR
 2--> E X E C U T A R
 3--> E N C E R R A R
 [      ]
 NOME DO ARQUIVO QUE CONTEM A GRAMATICA ? [                          ]
```

INFORMACOES GERAIS

ESTE GERADOR FORNECE AS TABELAS DE EMPILHAMENTO, SALTO E REDUCAO QUE SAO PRODUZIDAS PELO GERADOR R*S. PARA ISSO , A GRAMATICA SOBRE A QUAL SE DESEJA CONSTRUIR UM ANALISADOR R*S DEVE SER GRAVADA ANTES EM UM ARQUIVO EM DISCO NO B-6700.

AS SEGUINTES REGRAS DEVEM SER OBSERVADAS NA MONTAGEM DAS GRAMATICAS:

1- TERMINAIS DEVEM APARECER ENTRE APOSTROFES.

2- O SINAL DE IGUAL SEPARA OS LADOS DIREITO E ESQUERDO DE CADA PRODUCAO.

```
1--> AS INFORMACOES RESTANTES
2--> E X E C U T A R
3--> E N C E R R A R
[      ]
```

3- QUANDO VARIAS PRODUCOES POSSUEM O MESMO LADO ESQUERDO, NAO E' NECESSARIO REPETIR O LADO ESQUERDO DA PRODUCAO. BASTA COLOCAR UM PONTO DE EXCLAMACAO SEPARANDO DUAS PRODUCOES.
AO FINAL DE UMA PRODUCAO DEVERA APARECER SEMPRE UM PONTO E VIRGULA.

```
                EX:   E = E '+' T
                        ! T ;
```

4- PRODUCOES NULAS NAO TEM REPRESENTACAO ESPECIAL.

```
                EX.   COMANDO-VAZIO =  ;
```

5- BRANCOS DEVEM SER USADOS PARA SEPARAR SIMBOLOS.

```
1--> AS INFORMACOES  RESTANTES
2--> E X E C U T A R
3--> E N C E R R A R
  [           ]
```

Fig. (A-2.13)

Arquivo <DIALOGO>/TEXTOS

```
     6-  SIMBOLOS NAO DEVEM CONTER BRANCOS.
               EX:   ERRADO:  LISTA DE EXPRESSOES
                     CORRETO: LISTA-DE-EXPRESSOES

     7-  A PRODUCAO EQUIVALENTE A      S' -> S $
         NAO DEVE SER DADA.  ELA E' COLOCADA PELO GERADOR R*S.

     8-  ( I M P O R T A N T E )
         O NAO-TERMINAL DA PRIMEIRA PRODUCAO E' TOMADO COMO
         "SI'MBOLO INICIAL DA GRAMA'TICA".

         E'  SO'.  BOA  SORTE !!!  RECLAMACOES: SERGIO DE M. SCHNEIDER.

 1--> E X E C U T A R
 2--> E N C E R R A R
   [           ]
   MENU2          F I M   D A   A N A L I S E   D A
                        G R A M A T I C A
-------------------------------------------------------------------------------
1--> G E R A C A O
2--> E N C E R R A R
  [           ]
          ****       R E S U L T A D O S   *****
-------------------------------------------------------------------------------

 RESPONDA DIZENDO QUAIS RESULTADOS QUER QUE IMPRIMA:

 AS PRODUCOES SIMPLES ? .....................................[S/N].[    ]1
 AS DERIV. SIMPLES E MULTIPLAS E AS PRODUCOES SIMPLES?  [S/N].[    ]2
 A RELACAO FIRST ?...........................................[S/N].[    ]3
 A RELACAO FOLLOW ?..........................................[S/N].[    ]4
 A FUNCAO DELTA ?............................................[S/N].[    ]5
 AS TERNAS DE REDUCAO E AS REDUCOES ASSOCIADAS ?.........[S/N].[    ]6
 OS ITENS QUE COMPOEM CADA ESTADO ANALISADO ?............[S/N].[    ]7
 A GRAMATICA DADA ?..........................................[S/N].[    ]8

-------------------------------------------------------------------------------
       SE O CURSOR DO SEU TERMINAL ESTIVER NA PARTE INFERIOR DA TELA,
 COLOQUE AS RESPOSTAS NOS ESPACOS ENTRE AS EXCLAMACOES.

! 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7 ! 8 !
MENU1            1.    8.G          1.
COLEC1           9.    1.D         -1.
MENAJUDA1       10.   19.G          2.
MENAJUDA2       29.   19.G          3.
MENAJUDA3       48.   16.G          4.
MENU2           64.    6.G          5.
COLEC2          70.   19.D         -1.
```

Fig. (A-2.13) (continuação)

b) Na geração de código fonte ALGOL :

A geração das procedures ALGOL, foi feita mediante a utilização da procedure GERADOR.

A execução desta procedure teve como entradas: O nome do diálogo que foi gerado DIALOGO, e os arquivos DIALOGO/AÇÕES e DIALOGO/TEXTOS. Os resultados da execução desta procedure podem ser vistos nas figuras: (A-2.14), (A-2.15), (A-2.16) e (A-2.17).

```
PROCEDURE DIALOGO;
BEGIN
  BOOLEAN PULAR,VALIDA;
  INTEGER PREL,TOTREG;REAL ARRAY TABDADI0:99,0:2];
  FILE TEXTOS(KIND=DISK);
  FILE TELA(KIND=DC,MYUSE=IO,FILETYPE=7,UNITS=CHARACTERS
          ,MAXRECSIZE=80,BLOCKSIZE=1600);
  FORMAT CLEAR (4"0C",79(4"00"));
  FORMAT FORMS (4"12",4"3C",4"03");
$ INCLUDE "DIALOGO/FOWAR."
$ INCLUDE "APLICACAO."
$ INCLUDE "AJUDA."
$ INCLUDE "DIALOGO/EXEC."
$ INCLUDE "DIALOGO/PROC."


  BEGIN
    REPLACE TEXTOS.TITLE BY "SERGIO2/TEXTOS.";
    CARREGA;
    MENU1;
    WHILE PULAR DO  EXEC(PREL);
    CLOSE(TELA);
  END;
END;
```

Fig. (A-2.14)

Arquivo DIALOGO

```
PROCEDURE MENU1 ;
FORWARD;
PROCEDURE COLEC1 (NARQ);
EBCDIC ARRAY NARQ[*];
FORWARD;
PROCEDURE MENAJUDA1 ;
FORWARD;
PROCEDURE MENAJUDA2 ;
FORWARD;
PROCEDURE MENAJUDA3 ;
FORWARD;
PROCEDURE MENU2 ;
FORWARD;
PROCEDURE COLEC2 (P1,P2,P3,P4,P5,P6,P7,P8);
REAL    P1,P2,P3,P4,P5,P6,P7,P8;
FORWARD;
```

Fig. (A-2.15)
Arquivo <DIALOGO> /FOWAR

```
PROCEDURE EXEC(PREL);
INTEGER PREL; FORWARD;
PROCEDURE EXEC(PREL);
INTEGER PREL;
BEGIN
  CASE PREL OF
  BEGIN
    01 : MENU1 ;
    02 : MENAJUDA1 ;
    03 : MENAJUDA2 ;
    04 : MENAJUDA3 ;
    05 : MENU2 ;
  END;
END;
```

Fig. (A-2.16)
Arquivo <DIALOGO> /EXEC

```
PROCEDURE MENU1 ;
BEGIN
  POINTER PT;
  REAL ARRAY RESPOSTA[0:12];
  INTEGER CONT,POS,TAM,POSREL,DIGITO ;
  EBCDIC ARRAY BUFFER[0:2000],CADEIA,NARQ[0:72] ;
  BEGIN
    PULAR:=FALSE;
    VALIDA:=TRUE;
    POS:=000001;  TAM:=000008;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL
                  TAM -1 DO PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72),FOR CONT:=0 STEP 1 UNTIL TAM -1 DO
                     PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    READ (TELA,(A72),CADEIA);
    ANALISE(CADEIA,RESPOSTA[*],DIGITO,POSREL);
    IF (DIGITO GEQ 0)
       THEN CASE DIGITO OF
       BEGIN
         1   :      MENAJUDA1 ;
         2   :      COLEC1 (NARQ);
               IF VALIDA THEN GRAMATICA(NARQ);
         3   : FIMSERGIO ;
          ELSE:MENU1 ;
       END
       ELSE BEGIN
              IF POSREL GEQ 0
                 THEN BEGIN
                        PULAR:=TRUE;
                        PREL:=POSREL;
                      END
                 ELSE MENU1 ;
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-2.17)

Arquivo <DIALOGO> /PROC

```
PROCEDURE COLEC1 (NARQ);
EBCDIC ARRAY NARQ[*];
BEGIN
  INTEGER POS,TAM,POSREL,ID,IL,CONT;
  INTEGER ARRAY ERRO,NUMCAR[0:12];
  REAL ARRAY RESPOSTA[1:01,0:12];
  EBCDIC ARRAY BUFFER[0:2000],CADEIA[0:72];
  POINTER PT,PBU;
  BEGIN
    NUMCAR[01]:=20 ;
    POS:=000009;
    TAM:=000001;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL TAM-1 DO

        PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72), FOR CONT:=0 STEP 1  UNTIL TAM-1 DO
        PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    ID:=01 ;
    READ(TELA,(A80),BUFFER);
    PBU:=BUFFER; REPLACE CADEIA BY " " FOR 72;
    FOR IL:=1 STEP 1 UNTIL ID DO
    BEGIN
      REPLACE CADEIA BY PBU:PBU FOR NUMCAR[IL];
      ANALISE(CADEIA,RESPOSTA[IL,*],CONT,POSREL);
    END;
    VERFNARQ(RESPOSTA[01,*],ERRO[01]);
    FOR IL:=1 STEP 1 UNTIL ID DO
    IF VALIDA AND  (ERRO[IL] > 0)
       THEN CASE IL OF
            BEGIN
             01 : AJUDANARQ(RESPOSTA[01,*],VALIDA,ERRO[IL],IL
                                  );
            END;
    IF VALIDA
       THEN BEGIN
               PT:=POINTER(RESPOSTA[01,*]);
             REPLACE NARQ BY PT FOR 72;
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-2.17) (continuação)

```
PROCEDURE MENAJUDA1 ;
BEGIN
  POINTER PT;
  REAL ARRAY RESPOSTA[0:12];
  INTEGER CONT,POS,TAM,POSREL,DIGITO ;
  EBCDIC ARRAY BUFFER[0:2000],CADEIA,NARQ[0:72] ;
  BEGIN
    PULAR:=FALSE;
    VALIDA:=TRUE;
    POS:=000010;  TAM:=000019;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL
                    TAM -1 DO PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72),FOR CONT:=0 STEP 1 UNTIL TAM -1 DO
                      PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    READ (TELA,(A72),CADEIA);
    ANALISE(CADEIA,RESPOSTA[*],DIGITO,POSREL);
    IF (DIGITO GEQ 0)
       THEN CASE DIGITO OF
       BEGIN
         1    :      MENAJUDA2 ;
         2    :      COLEC1 (NARQ);
                IF VALIDA THEN GRAMATICA(NARQ);
         3    : FIMSERGIO ;
          ELSE:MENAJUDA1 ;
       END
       ELSE BEGIN
              IF POSREL GEQ 0
                 THEN BEGIN
                        PULAR:=TRUE;
                        PREL:=POSREL;
                      END
                 ELSE MENAJUDA1 ;
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-2.17) (continuação)

```
PROCEDURE MENAJUDA2 ;
BEGIN
  POINTER PT;
  REAL ARRAY RESPOSTA[0:12];
  INTEGER CONT,POS,TAM,POSREL,DIGITO ;
  EBCDIC ARRAY BUFFER[0:2000],CADEIA,NARQ[0:72] ;
  BEGIN
    PULAR:=FALSE;
    VALIDA:=TRUE;
    POS:=000029;  TAM:=000019;
    READ (TEXTOS[POS],<A72>,FOR CONT:=0 STEP 1 UNTIL
                  TAM -1 DO PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,<A72>,FOR CONT:=0 STEP 1 UNTIL TAM -1 DO
                     PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    READ (TELA,<A72>,CADEIA);
    ANALISE(CADEIA,RESPOSTA[*],DIGITO,POSREL);
    IF (DIGITO GEQ 0)
       THEN CASE DIGITO OF
       BEGIN
         1    :      MENAJUDA3 ;
         2    :      COLEC1 (NARQ);
                IF VALIDA THEN GRAMATICA(NARQ);
         3    : FIMSERGIO ;
          ELSE:MENAJUDA2 ;
       END
       ELSE BEGIN
              IF POSREL GEQ 0
                 THEN BEGIN
                        PULAR:=TRUE;
                        PREL:=POSREL;
                      END
                 ELSE MENAJUDA2 ;
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-2.17) (continuação)

```
PROCEDURE MENAJUDA3 ;
BEGIN
  POINTER PT;
  REAL ARRAY RESPOSTA[0:12];
  INTEGER CONT,POS,TAM,POSREL,DIGITO ;
  EBCDIC ARRAY BUFFER[0:2000],CADEIA,NARQ[0:72] ;
  BEGIN
    PULAR:=FALSE;
    VALIDA:=TRUE;
    POS:=000048;  TAM:=000016;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL
                    TAM -1 DO PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72),FOR CONT:=0 STEP 1 UNTIL TAM -1 DO
                      PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    READ (TELA,(A72),CADEIA);
    ANALISE(CADEIA,RESPOSTA[*],DIGITO,POSREL);
    IF (DIGITO GEQ 0)
       THEN CASE DIGITO OF
       BEGIN
         1    :      COLEC1 (NARQ);
                IF VALIDA THEN GRAMATICA(NARQ);
         2    : FIMSERGIO ;
          ELSE:MENAJUDA3 ;
       END
       ELSE BEGIN
              IF POSREL GEQ 0
                 THEN BEGIN
                        PULAR:=TRUE;
                        PREL:=POSREL;
                      END
                 ELSE MENAJUDA3 ;
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-2.17) (continuação)

```
PROCEDURE MENU2 ;
BEGIN
  POINTER PT;
  REAL ARRAY RESPOSTA[0:12];
  INTEGER CONT,POS,TAM,POSREL,DIGITO ;
  REAL    P1,P2,P3,P4,P5,P6,P7,P8 ;
  EBCDIC ARRAY BUFFER[0:2000],CADEIA[0:72] ;
  BEGIN
    PULAR:=FALSE;
    VALIDA:=TRUE;
    POS:=000064;  TAM:=000006;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL
                  TAM -1 DO PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72),FOR CONT:=0 STEP 1 UNTIL TAM -1 DO
                     PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    READ (TELA,(A72),CADEIA);
    ANALISE(CADEIA,RESPOSTA[*],DIGITO,POSREL);
    IF (DIGITO GEQ 0)
       THEN CASE DIGITO OF
       BEGIN
         1    : GERADOR ;
                COLEC2(P1,P2,P3,P4,P5,P6,P7,P8);
                IF VALIDA THEN RESULTADOS(P1,P2,P3,P4,P5,P6,P7
                                          ,P8);
         2    : FIMSERGIO ;
          ELSE:MENU2 ;
       END
       ELSE BEGIN
              IF POSREL GEQ 0
                 THEN BEGIN
                        PULAR:=TRUE;
                        PREL:=POSREL;
                      END
                 ELSE MENU2 ;
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-2.17) (continuação)

```
PROCEDURE COLEC2 (P1,P2,P3,P4,P5,P6,P7,P8);
REAL     P1,P2,P3,P4,P5,P6,P7,P8;
BEGIN
  INTEGER POS,TAM,POSREL,ID,IL,CONT;
  INTEGER ARRAY ERRO,NUMCAR[0:12];
  REAL ARRAY RESPOSTA[1:08,0:12];
  EBCDIC ARRAY BUFFER[0:2000],CADEIA[0:72];
  POINTER PT,PBU;
  BEGIN
    NUMCAR[01]:=04 ;
    NUMCAR[02]:=04 ;
    NUMCAR[03]:=04 ;
    NUMCAR[04]:=04 ;
    NUMCAR[05]:=04 ;
    NUMCAR[06]:=04 ;
    NUMCAR[07]:=04 ;
    NUMCAR[08]:=04 ;
    POS:=000070;
    TAM:=000019;
    READ (TEXTOS[POS],(A72),FOR CONT:=0 STEP 1 UNTIL TAM-1 DO

        PT:=BUFFER[CONT*72]);
    WRITE(TELA,CLEAR);
    WRITE(TELA,(A72), FOR CONT:=0 STEP 1  UNTIL TAM-1 DO
        PT:=BUFFER[CONT*72]);
    WRITE(TELA,FORMS);
    ID:=08 ;
    READ(TELA,(A80),BUFFER);
    PBU:=BUFFER; REPLACE CADEIA BY " " FOR 72;
    FOR IL:=1 STEP 1 UNTIL ID DO
    BEGIN
      REPLACE CADEIA BY PBU:PBU FOR NUMCAR[IL];
      ANALISE(CADEIA,RESPOSTA[IL,*],CONT,POSREL);
    END;
    VERF(RESPOSTA[01,*],ERRO[01]);
    VERF(RESPOSTA[02,*],ERRO[02]);
    VERF(RESPOSTA[03,*],ERRO[03]);
    VERF(RESPOSTA[04,*],ERRO[04]);
    VERF(RESPOSTA[05,*],ERRO[05]);
    VERF(RESPOSTA[06,*],ERRO[06]);
    VERF(RESPOSTA[07,*],ERRO[07]);
    VERF(RESPOSTA[08,*],ERRO[08]);
    FOR IL:=1 STEP 1 UNTIL ID DO
    IF VALIDA AND  (ERRO[IL] > 0)
       THEN CASE IL OF
            BEGIN
             01 : HELP(RESPOSTA[01,*],VALIDA,ERRO[IL],IL);
             02 : HELP(RESPOSTA[02,*],VALIDA,ERRO[IL],IL);
             03 : HELP(RESPOSTA[03,*],VALIDA,ERRO[IL],IL);
             04 : HELP(RESPOSTA[04,*],VALIDA,ERRO[IL],IL);
             05 : HELP(RESPOSTA[05,*],VALIDA,ERRO[IL],IL);
             06 : HELP(RESPOSTA[06,*],VALIDA,ERRO[IL],IL);
```

Fig. (A-2.17) (continuação)

```
              07 : HELP(RESPOSTA[07,*],VALIDA,ERRO[IL],IL);
              08 : HELP(RESPOSTA[08,*],VALIDA,ERRO[IL],IL);
            END;
    IF VALIDA
       THEN BEGIN
              P1:=RESPOSTA[01,0];
              P2:=RESPOSTA[02,0];
              P3:=RESPOSTA[03,0];
              P4:=RESPOSTA[04,0];
              P5:=RESPOSTA[05,0];
              P6:=RESPOSTA[06,0];
              P7:=RESPOSTA[07,0];
              P8:=RESPOSTA[08,0];
            END;
  END;
  CLOSE(TELA);
END;
```

Fig. (A-2.17)
(continuação)

c) As procedures de crítica e recuperação de erros:

A seguir fig. (A-2.10) apresentamos as procedures de crítica e recuperação de erros embora não produzidas pelo "G D", porém necessárias para a apresentação de mensagens de auxílio ao usuário, assim como para a avaliação das suas respostas.

```
 PROCEDURE VERFNARQ(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*]; INTEGER ERRO;
BEGIN
  POINTER PC;
  TRUTHSET NUMEROS("0123456789");
  TRUTHSET LETRAS (ALPHA AND NOT NUMEROS);
  ERRO:=0;  PC:=RESPOSTA;
  IF RESPOSTA[2]="R      " THEN ERRO:=1  % NUMERO
     ELSE BEGIN
            WHILE (( PC NEQ " ") AND (ERRO=0))
            DO BEGIN
                 IF PC IN LETRAS
                    THEN SCAN PC:PC WHILE IN ALPHA
                    ELSE ERRO:=1 ;
                 IF ((PC=" ") OR (PC="/")) THEN PC:=PC+1
                                           ELSE ERRO:=1;
               END; % DO WHILE
          END;
  END;     % DA VERFNARQ

PROCEDURE VERF(RESPOSTA,ERRO);
REAL ARRAY RESPOSTA[*];  INTEGER ERRO;
BEGIN
  IF ((RESPOSTA[0] EQL "S      ") OR (RESPOSTA[0] EQL "N      "))
     THEN ERRO:=0  ELSE ERRO:=1;
END;
```

Fig. (A-2.18) - Arquivo <DIALOGO> /AJUDA

```
PROCEDURE AJUDANARQ(RESPOSTA,CERTA,ERRO ,PARM);
REAL ARRAY RESPOSTA[*]; BOOLEAN CERTA; INTEGER ERRO,PARM;
BEGIN
  REAL TEMP; EBCDIC ARRAY CAD[0:72]; POINTER PC;
  WRITE(TELA,<" A RESP. RECEBIDA NAO E' UM IDENTIFICADOR VALIDO">);
  WRITE(TELA,<"DESEJA PRENCHER NOVAMENTE O NOME DO ARQUIVO ? S/N">);
  READ (TELA,<A6>,TEMP);
   IF TEMP = "S      "
       THEN BEGIN
              WRITE(TELA,CLEAR);
              WRITE(TELA,<"NOME DO ARQUIVO ? [                    ]">);
              WRITE(TELA,FORMS);  READ(TELA,<A72>,CAD);
              PC:=RESPOSTA[0]; REPLACE PC BY CAD FOR 72;
              VERFNARQ(RESPOSTA,ERRO);
              IF ERRO=0 THEN CERTA:=TRUE
                        ELSE AJUDANARQ(RESPOSTA,CERTA,ERRO,PARM);
            END
       ELSE CERTA:=FALSE;
END;      % DO HELPNARQ

PROCEDURE HELP (RESPOSTA,CERTA,ERRO,PARM);
  REAL ARRAY RESPOSTA[*]; BOOLEAN CERTA; INTEGER ERRO,PARM;
BEGIN
  REAL TEMP;
  WRITE(TELA,<"A RESPOSTA(",I1,")  CADEIA RECEBIDA )",A6,
              /, "NAO E' UM  S/N">,PARM,RESPOSTA[0]);
  WRITE(TELA,<" DESEJA FORNACER UM NOVO VALOR ? S/N">);
   READ(TELA,<A6>,TEMP);
   IF TEMP = "S      "
     THEN BEGIN
            WRITE(TELA,CLEAR);
            WRITE(TELA,<" S/N ?? [                ]">);
            WRITE(TELA,FORMS);
            READ(TELA,<A6>,RESPOSTA[0]);
            VERF(RESPOSTA,ERRO);
            IF ERRO > 0 THEN HELP(RESPOSTA,CERTA,ERRO,PARM)
                        ELSE CERTA:=TRUE;
          END
    ELSE CERTA:=FALSE;
END;      % DA HELP
```

Fig. (A-2.18)

Arquivo <DIALOGO> /AJUDA (continuação)

REFERÊNCIAS BIBLIOGRÁFICAS

1 - WILDER, M.A. & MAXEMCHUK, N.F. Virtual editing: II. the user interface. ACM SIGOA Newsletter, New York, 3 (1/2): 41-46, 1982. Proceedings SIGOA CONFERENCE ON OFFICE INFORMATION SYSTEMS. Philadelphia, Pennsylvania, June 21-23, 1982.

2 - HAMMER, et alii. The Implementation of etude, an integrated and interactive document production system. ACM SIGOA Newsletter, New York, 2 (1/2): 137-46, spr./sum. 1981 Proceedings of the ACM SIGPLAN SIGOA SYMPOSIUM ON TEXT MANIPULATION- Portland, Oregon, June 8-10, 1981.

3 - GOOD, M. Etude and the folklore of user interface design. ACM SIGOA Newsletter, New York, 2 (1/2): 34-43, Spr./Sum. 1981. Proceedings of the ACM SIGPLAN SIGOA SYMPOSIUM ON TEXT MANIPULATION - Portland, Oregon, June 8-10, 1981

4 - WASTON, R.W. User interface design issues for a large interactive system. In: AFIPS CONFERENCE PROCEEDINGS. New York, NY, June 7-10, 1976. National Computer Conference; Montvale, N. J. AFIPS Press, 1976. v.45 - p. 357-64

5 - HARDY JR., T.M. The Syntax of interactive command languages: a framework for design. Software Practice & Experience, New York, John Wiley & Sons, 12 (1): 67-75, Jan. 1982

6 - COELHO, M. Man-machine communications in portuguese: a friendly library service system. Information Systens, Oxford, Eng., Pergamon Press, 7 (2): 163-81, 1982.

7 - HAYES, P. et alii. Braking the man. machine communication barrier. JEEE Computer, Piscataway, N.J., 14 (3): 19-30,

Mar. 1981.

8 - Bo, K. Human computer interaction. IEEE Computer, Piscataway, N.J. 15 (11): 9-11, Nov. 1982.

9 - PFAFF, G. et alii. Constructing user interface based on logical input devices. IEEE Computer, Piscataway, N. J. 15 (11): 62-8, Nov. 1982.

10 - FRANK, G.R. & THEAKER, C.J. Muss - the user interface. Software Practice & Experience, New York, John Wiley & Sons, 9 (8): 621-31, Aug. 1979.

11 - PALME, J. A Human computer interface for non computer specialists. Software Practice & Experience, New York, John Wiley & Sons, 9 (9): 741-47, Sept. 1979

12 - SCHOFIELD, D. et alii. MM1, a man-machine interface. Software Practice & Experience, New York, John Wiley & Sons, 10 (9): 751-63, Sept. 1980

13 - ALTY, J.L. & COOMBS, M.J. University computer advisory services; the Study of the man-computer interface. Software Practice & Experience, New York, John Wiley & Sons, 10 (11): 919-34, Nov. 1980.

14 - DWYER, B. A User friendly algorithm. ACM Communications, New York, 4 (9):556-61, Sept. 1981

15 - BASS, L.J. & BUNKER, R.E. A Generalized user interface for applications programs. ACM Communications, New York, 24 (12): 796-800, Dec. 1981.

16 - KAISER, P. & STETINA, I. A Dialogue generator Software Practice & Experience, N.Y., John & Sons, 12 (8): 693-707.

17 - CASEY, B. & DASARATHY, B. Modelling and validating the man machine interface. Software Practice and Experience, New York, John Wiley & Sons, 12 (6): 557-69, Jun. 1982.

18 - NEGUS, B. et alii. Dialog: a sheme for the quick and effective production of interactive applications software. Software Practice & Experience. New York, John Wiley & Sons, 11 (3): 205-24, Mar. 1981.

19 - HEFFLER, M. J. Description of a menu creation and interpretation system. Software Practice & Experience, Ney York, John Wiley & Sons, 12 (3): 269-81, Mar. 1982.

20 - ROBERSTON, K. et alii. Experimental evaluation of the zog frame editor. Pittsburg, PA, Carnegie-Mellon University, 1981. 14p. (CMV- CS - 81-112).

21 - BARRON, J. Dialogue and process design for interactive information systems using taxis. ACM SIGOA News letter, New York, 3 (1/2): 12-20, 1982. Proceedings SIGOA CONFERENCE ON OFFICE INFORMATION SYSTEMS. Philadelphia, Pennsylvania, June 21-23, 1982.

22 - SABANI, C. O Uso de diálogos na interação homem. máquina. IN: CONGRESSO NACIONAL DE INFORMÁTICA, 14. São Paulo, 16-23 out. 1981. Anais do XIV Congresso Nacional da Informática. São Paulo, SUCESU, 1981. p. 521-7.

23 - PEARSALL, R. J. Technique for assessing external design of software. IBM Systems Journal, Armonk, N.J., 21 (2): 211-20, 1982.

24 - DAVIS, A. M. The Design of a family of application oriented requeriments languages. IEEE Computer. Piscataway, N.J., 15 (5): 21-8, Mar 1982.

25 - HUMMEL, H. Lektor - a list oriented, machine independent programming system for conventional applications. Software Practice & Experience, New York, John Wiley & Sons, 6 (4): 447-62, Oct/Dec. 1976.

26 - KREBS, C. E. et alii. Terminal tranparent display language (TTDL). IN: AFIPS CONFERENCE PROCEEDINGS. New York, N.Y. June 7-10, 1976. National Computer Conference. Montvale, N.J., AFIPS Press, 1976. v.45. p. 365-71.

27 - HAYES, P. & REDDY, R. An Anatony of graceful interaction in Spoken and written machine communication. Pittsburg, PA, Carnegie- Mellon University, 1979. 70 p. CMU-CS-79-144).

28 - ROWE, N.C. On Some arguable claims in B. Shneiderman's evaluation of natural language interaction with data base systems. s.n.t.

29 - DZIDA, W. et alii. User-perceived quality of interactive systems. IEEE Transactions on Software Engineering. Piscataway, N.J., 4 (4): 270-6, July 1978.

30 - REISNER, P. et alii. Human factors evaluation of two data base query languages - Square and Sequel. IN: AFIPS CONFERENCE PROCEEDINGS, Anaheim, CA, May 19-22, 1975. National computer conference, Montvale, N. J. AFIPS Press, 1975. v.44. p. 447-51

31 - WATERS, R. C. The Programmer's apprentice: knowledge based program editing. IEEE Transactions on Software Engineering, Piscataway, N.J., 8 (1): 1-12, Jan. 1982.

32 - REISNER, P. Formal grammar and human factors design of an interactive graphics system. IEEE Transactions on software Engineering, Piscataway, N.J., 7 (2): 229-40, Mar. 1981.

33 - CODD, E. F. How about recently? IN: SHNEIDERMAN, B. ed. Databases: improving usability and responsiveness. London, Academic Press, 1978. p. 3-28.

34 - SHNEIDERMAN, B. Human factors experiments in designing interactive systems. IEEE Computer, Piscataway, N.J., 12 (12): 9-19, Dec. 1979.

35 - TRACZ, W. J. Computer programming and the human thought process. Software Practice and Experience, New York, John Wiley & Sons, 9 (2): 127-37, Feb. 1979.

36 - ZLOOF, M. M. Design aspects of the query-by- example data base management language. In: SHNEIDERMAN, B. ed Data-bases: improving usability and responsiveness. London, Academia Press, 1978. p. 29-53.

37 - SHNEIDERMAN. Improving the human factors aspect of data-base interactions. ACM Transactions on Database systems, New York, 3 (4): 417-39, Dec. 1978.

38 - BRANSCOMB, L.B. Bringing computing to people the broadening challenge. IEEE Computer,Piscataway, N.J., 15 (7): 68-75, July 1982.

39 - MARTIN, J. Design of man-computer dialogues. Englewood Cliffs, N.J. Prentice Hall, 1973. 559 p. (Prentice-Hall series in Automatic Computation).

40 - STEVENS, W. Using structured design. New York, John Wiley & Sons, 1981. 213 p.

41 - DAVIS, G. Management information systems.Tokyo, McGraw - Hill Kogakusha, 1974.

42 - MILLER, L. M. A Study in man-machine interaction.IN: AFIPS CONFERENCE PROCEEDINGS. Dallas, Tx, June 13-16, 1977 National Computer Conference. Montvalle, N.J. AFIPS Press, 1977. v.46, p. 409-21.

43 - JACOB, R. Using formal specifications in the design of a human-computer interface. ACM Communications, New York,

26 (4): 259-64, Apr. 1983.

44 - QUERY languages - IN: SHNEIDERMAN, B. Database management Systems. Montvale, N.J., AFIPS Press, 1976 p. 59-61. (The Information Technology Series, 1).

45 - EHRICH, R.W. Research on human- computer interfaces at Virginia tech. ACM SIGCHI Bulletin, New York, 14 (3): 16-9, Jan. 1973

46 - NORMAN, D. Design rules based on analyses of human error ACM Communications, New York, 26 (4): 254.8, Apr. 1983.

47 - WEIZENBAUM, J. ELIZA - a computer program for the stuty of natural language communication between man and machine. ACM Communications, New York, 9 (1): 36-45, Jan. 1966.

48 - BLACK, J.L. A General purpose dialogue processor. IN: AFIPS CONFERENCE PROCEEDINGS. Dallas, TX, June 13-16, 1977. National Computer Conference, Montvale, N.J., 1977. v.46, p. 197-408.

49 - TESLER, L. Enlisting user help in software design. ACM SIGCHI Bulletin, New York, 14 (3): 5-9, Jan. 1983.

50 - MATHEWSON,S.C. User acceptance: design considerations for a program generator. Software Practice and Experience, New York, Hohn Wiley & Sons, 13 (2): 101-17, Feb. 1983.

51 - CODD, E. F. A Relational model of data for large shared data banks. ACM Communications, New York, 13 (6): 377-87, June 1970.

52 - DATE, C. J. An introduction to database Systems. 2. ed. Reading, Mass., Addison. Wesley, 1977,536p.(The Systems Programming Series).

53 - SIMPSON, M. A Human-factors style guide for program

design. Byte Peterborough, NH, Byte Publications, 7 (4): 108.32, Apr. 1982.

54 - BARDEN JR., W. Color computer from A to D; make your color computer see and feel better. Byte, Peterborough NH, Byte Publications, 6 (12): 134-58, Dec. 1981.

55 - BURROUGHS. ALGOL language reference manual. Detroit, Michig., 1974. Relative to mark 11.5 release. 5000649.

56 - SOUSA. A.C.G. Interface para usuário casual - IUC linguagem de comunicação para usuário casual de banco de dados, baseada em linguagem natural. Rio de Janeiro, COPPE/UFRJ, 1983. 168p. Tese (Mestrado defendida em 03/83 na COPPE/UFRJ).