CRIPTA : UMA FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS DE DADOS ESTATÍSTICOS

Dulce Maria Rocha Barbosa

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

Aprovado por:

Prof. Ana Regina Cavalcanti da Rocha, D.Sc.
(Presidente)

Prof. Jano Moreira de Souza, Ph.D.

Prof. Antonio Claudio C. M. Silva, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 1989

BARBOSA, DULCE MARIA ROCHA

Cripta: uma ferramenta para desenvolvimento de sistemas de dados estatísticos [Rio de Janeiro] 1989

xvi, 287 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1989)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Ferramenta para desenvolver sistemas estatísticos de apuração de dados I. COPPE/UFRJ

II. Título (série).

Agradecimentos

No transcorrer deste trabalho enfrentei vários momentos difíceis, mas, graças a Deus e ao estímulo de grandes amigos, consegui vencê-los.

Agradeço a paciência e dedicação de minha orientadora Ana Regina Rocha.

Agradeço o grande incentivo e carinho de minha família e amigos, em especial a Sérgio Baia, Sérgio Cortes, Valéria Roitman e Reina Marta Hanono.

E agradeço ainda à Instituição IBGE pela oportunidade da pesquisa e implementação deste trabalho. Agradeço especialmente a Aluizio Guedes e Marco Antonio Duarte por estarem sempre dispostos a comunicar sua experiência em pesquisas de apuração de dados.

Gostaria de relacionar aqui o nome de todos que me ajudaram em sinal de meu agradecimento, porém, estes estarão sempre em minha memória.

Dulce Maria Rocha Barbosa

COPPE/UFRJ em Março de 1989

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências (M.Sc.).

CRIPTA : UMA FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS DE DADOS ESTATÍSTICOS

Dulce Maria Rocha Barbosa

Março de 1989

Orientadora : Ana Regina Cavalcanti da Rocha

Programa : Engenharia de Sistemas e Computação

Analisando-se o desenvolvimento de sistemas de computação, nas últimas décadas, verificamos uma busca à produção de software de boa qualidade, baixo custo e bom nível de produtividade em seu desenvolvimento, o que tem levado os especialistas a pesquisar novas metodologias e ferramentas para desenvolvimento de sistemas.

Observando-se o desenvolvimento de sistemas de apuração de dados para grande volume de dados, verificamos que estes possuem características muito particulares a este tipo de sistema.

Este trabalho apresenta um conjunto de ferramentas para especificação, documentação e geração de aplicações de sistemas de apuração de dados estatísticos.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.).

CRIPTA : A TOOL FOR THE DEVELOPMENT OF STATISTICAL DATA SYSTEMS

Dulce Maria Rocha Barbosa

March, 1989

Chairwoman : Ana Regina Cavalcanti da Rocha

Department : Computing and Systems Engineering

Analysing the development of computing systems throughout the last decades, we can notice the specialists' concern with the production of software of good quality, of low cost, and of high level of efectiveness. This has led them to research new methodologies and tools for the development of these systems.

When we observe the development of survey data systems, we realize that there are inherent features in this kind of system. This Thesis presents a set of tools for the specification, documentation and code generation of statistical data systems.

INDICE

LISTA DE FIGURAS

CAPÍTULO I

INTRODUÇÃO

A necessidade de se expor objetivos e soluções de problemas a serem elaborados com o uso de computador de modo eficiente, têm levado, nos últimos tempos, a estudos e pesquisas que têm como meta o estabelecimento de métodos para que o desenvolvimento de sistemas de software se torne uma tarefa organizada, com resultados eficientes e menos onerosa.

Vários métodos têm sido apresentados e todos eles, embora não resolvam totalmente os problemas ligados ao desenvolvimento de sistemas, têm apresentado consideráveis benefícios em cada uma de suas respectivas áreas de aplicação.

Juntamente com os métodos, têm sido desenvolvidos softwares que visam facilitar o uso dos mesmos, automatizando algumas de suas ferramentas. No contexto brasileiro os exemplos mais usuais destas ferramentas automatizadas são os editores de dicionários de dados e os editores de diagramas de fluxo de dados.

Porém os problemas de desenvolvimento de software continuam: projetos fora de prazos, excedendo orçamentos e

produtos que não satisfazem as necessidades de seus usuários.

Alguns dos fatores para estes problemas, como citado por CAMPOS [1], podem ser relacionados: falhas nas técnicas de gerência de projetos, dificuldades culturais no processo de implantação de métodos de desenvolvimento de software e, talvez uma das mais importantes causas, a integração não muito perfeita entre as diferentes fases de desenvolvimento - definição, análise, programação, testes e manutenção, usando cada uma delas ferramentas diferentes e não integradas.

Algumas possibilidades para redução destes problemas são sugeridas por GUIMARÃES [2] como:

- aumento da produtividade dos profissionais de informática;

- transferência para o usuário final de uma parte do desenvolvimento.

No que concerne ao aumento da produtividade dos profissionais da informática existem duas possibilidades:

1) Continuação do uso de uma metodologia de desenvolvimento já existente, ou baseada no Ciclo de Vida tradicional, mas com um aumento da produtividade graças a ferramentas de apoio ao desenvolvimento.

Estas podem ser classificadas em diferentes grupos:

- Ferramentas de Base, onde estão os softwares de apoio ao processo de desenvolvimento como Editores de Dicionários de Dados e Diagramas de Fluxo de Dados, softwares para concepção de telas e geradores de relatórios;

- Geradores de Esqueleto, que tentam fazer uma integração de diferentes funções em uma só ferramenta, gerando um programa fonte em linguagem de alto nível (COBOL ou PLI) que deve ser completado pelo programador;

- Geradores de Programas, que se constituem numa segunda etapa de integração do conjunto de funções necessárias aos profissionais da informática, integrando principalmente as funções de dicionário de dados, geração de documentação e de programas e, uso de novas linguagens de programação.

2)- Uso de Novos Métodos para desenvolvimento, dentre os quais podemos citar o uso de prototipação.

Concentrando-nos na área de grandes centros de processamento de dados estatísticos, onde por grandes centros de processamento se entende o processamento de grandes volumes de dados, podemos verificar que alguns tipos de tratamento de dados são comuns à maioria dos sistemas desenvolvidos nestas áreas.

Os tipos de tratamento de dados mais característicos por possuirem esta afinidade de processamento entre os sistemas são as fases de crítica, correção (atualização), imputação e tabulação dos dados.

Analisando-se estes tipos de processamento, no que se refere ao Ciclo de Vida, podemos identificar algumas fases que se caracterizam como pontos críticos para as causas do surgimento dos problemas de desenvolvimento de software citados anteriormente, quais sejam:

- "Especificação pelo Usuário" dos procedimentos necessários para realização da tarefa, normalmente caracterizados por um Plano de Crítica, Imputação ou Tabulação.

- "Análise das Especificações" fornecidas pelos usuários, efetuada pela equipe de Análise, constatando sua viabilidade ou não, e especificação destes para implementação.

- "Implementação da Especificação", que é caracterizada pela codificação e testes dos programas correspondentes aos procedimentos especificados.

O objetivo deste trabalho é estudar os problemas existentes nestas fases, as ferramentas existentes que podem ser usadas, suas vantagens e desvantagens, e o desenvolvimento de uma ferramenta que visa atender a uma

melhor forma de especificação de procedimentos de crítica, imputação e tabulação. Esta ferramenta deve possibilitar uma melhor comunicação entre as fases envolvidas e, sempre que possível, permitir a geração automática dos programas de Crítica, criando-se uma documentação e formalização entre as fases e atributos que tornem estes programas representantes fiéis de sua especificação. Cabe ressaltar, ainda, que todo este estudo e análise estão voltados para ambientes de grandes processamentos de dados, principalmente para processamento de sistemas de pesquisas estatísticas através de levantamento de dados.

Para o estudo desta nova ferramenta foram consideradas, além das ferramentas existentes com suas vantagens e desvantagens, as sugestões para a solução dos problemas de Engenharia de Software apresentadas anteriormente.

Este trabalho de tese encontra-se dividido nos seguintes capítulos:

-Capítulo II, "SISTEMA ATUAL", onde será descrito o modo como são realizadas as pesquisas de coleta de dados, e a forma como são desenvolvidos os sistemas de apuração destas pesquisas. São analisados também os problemas de software encontrados em cada uma das fases de desenvolvimento destes sistemas.

-Capítulo III, "ESTUDO DAS FERRAMENTAS UTILIZADAS", que descreverá as principais ferramentas encontradas no mercado para uso em sistemas de apuração de pesquisas de coleta de

dados, e que se propõem à redução dos problemas encontrados no desenvolvimento destes sistemas. Para cada uma delas é feita uma análise citando suas vantagens e desvantagens.

-Capítulo IV, "SISTEMA PROPOSTO", onde será apresentada a proposta de um novo sistema que visa atender os problemas citados na descrição dos objetivos e especificação lógica de cada uma das ferramentas apresentadas.

-Capítulo V, "PROJETO FÍSICO", onde será apresentada a arquiterura do sistema proposto, e a especificação dos principais módulos que compõem o sistema.

-Capítulo VI, "CONCLUSÃO", onde serão feitas a avaliação do sistema proposto e sugestões para novas pesquisas.

CAPÍTULO II

DESCRIÇÃO DO SISTEMA ATUAL

O desenvolvimento desta tese está voltado para grandes ambientes de tratamento de dados de pesquisas estatísticas. Assim, toda a análise e especificação dos requisitos estão baseados no ambiente e procedimentos de trabalho usados neste tipo de empresas.

Normalmente as pesquisas são efetuadas através de coleta de dados, ou seja, distribuição de questionários, que serão devolvidos contendo as informações particulares, a cada pesquisa, e a cada informante.

Estes questionários, por sua vez, são codificados e digitados. Aqui tem início um ciclo de "Crítica" para verificação dos erros de transcrição e consistência interna dos dados, envolvendo correções tanto dos erros de digitação, quanto das informações iniciais dos informantes. Isto gera um volume de programação que consome semanas até o teste final.

As definições de quais críticas devem ser efetuadas sobre os questionários, para verificação da qualidade dos dados, normalmente são de responsabilidade do usuário, que as define sob uma forma denominada de Plano de Crítica.

Quando da execução destes Planos de Crítica, sobre os dados coletados para uma pesquisa, tem de ser estipulado também o tratamento a ser adotado com relação aos registros que não passarem pelo Plano de Crítica como corretos.

Um tratamento que pode ser adotado, é contactar o responsável pelo preenchimento do questionário, para verificação e correção das informações prestadas. Mas isto nem sempre é possível, e em alguns casos é considerado inviável.

Rejeitar todos os registros que apresentem erro de Crítica, é um tratamento perigoso, pois exige que se tenha uma certeza de que estas eliminações não irão interferir nos resultados da pesquisa.

A técnica de tratamento mais usada é a de Imputação Automática, que consiste na correção dos registros por máquina, através da estipulação de um conjunto de regras pelas quais se identificam conjuntos de campos do questionário, cujos valores podem ser alterados de acordo com os valores dos demais campos do conjunto, de tal modo que o registro resultante satisfaça as condições do Plano de Crítica. Estas regras de Imputação, normalmente, também são definidas pelo usuário.

Outro método usado para tratamento dos dados incorretos é o de Imputação Automática por "HOT-DECK" que consiste do armazenamento numa matriz dos valores das variáveis de controle associadas à variável a ser corrigida, mais o valor para a correção da variável, encontrado no último registro processado sem erros de

crítica. Este método consiste basicamente em se atribuir a uma variável, que esteja errada, o valor observado, para esta variável no último registro processado corretamente.

Após este ciclo de crítica e correção dos dados, tem início o ciclo de tratamento dos dados a nível estatístico, incluindo as tabulações necessárias para obtenção dos dados finais da pesquisa. A seguir são apresentados o Diagrama de Fluxo de Dados do sistema atual, e a descrição de cada processo.

II.1 - DIAGRAMA DE FLUXO DE DADOS

A figura II.1 apresenta o diagrama de fluxo de dados do sistema atual representando cada uma das atividades realizadas para o desenvolvimento da fase de crítica de um sistema de apuração de pesquisas.

II.2 - DESCRIÇÃO DOS PROCESSOS

Serão descritos neste item os processos apresentados no diagrama de fluxo de dados do sistema atual.

(1) - Especificar tarefa

Neste processo, o usuário, de posse de uma tarefa a realizar, estabelece os procedimentos necessários para sua

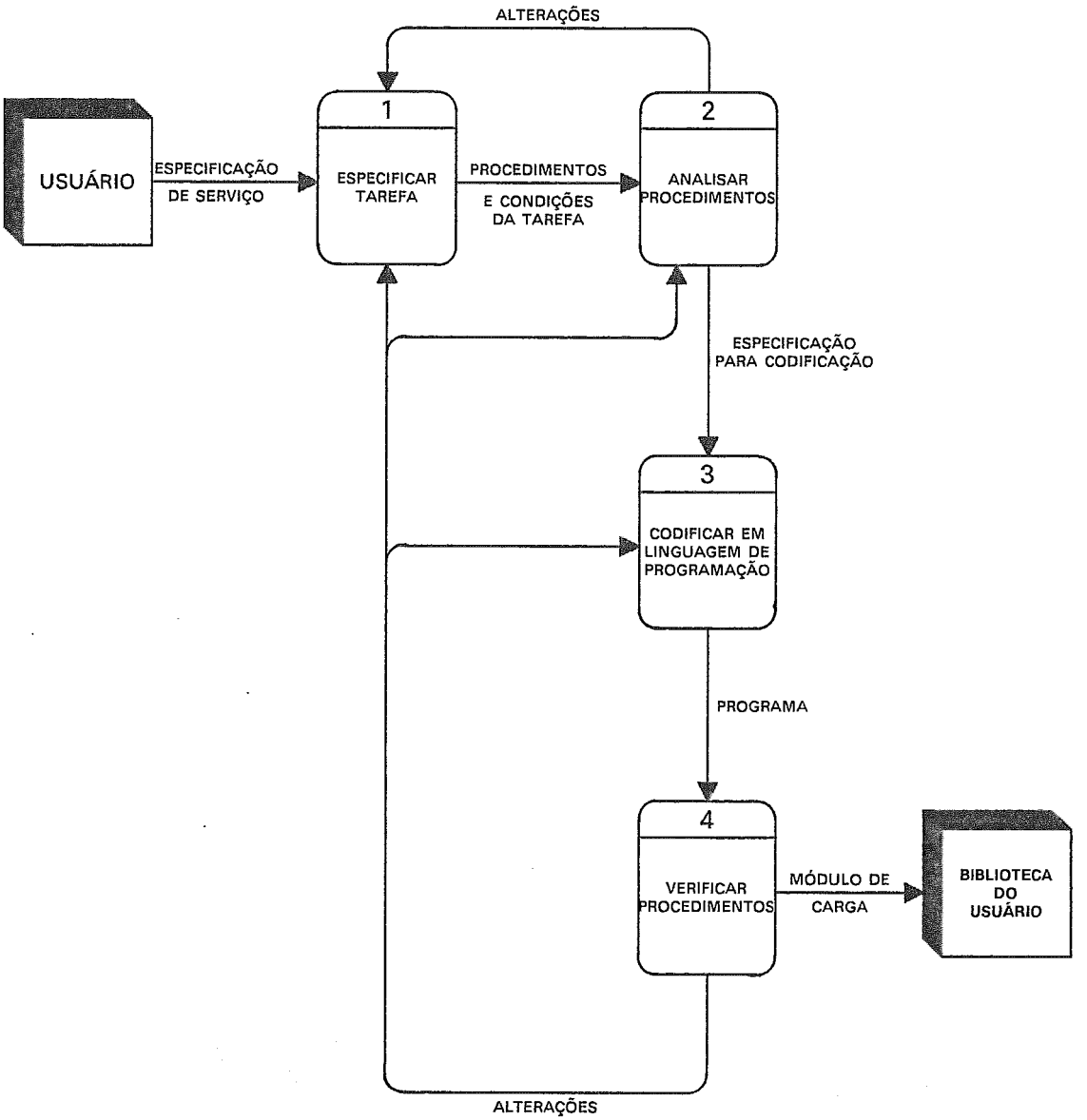


FIGURA II.1 - Diagrama do fluxo de dados do sistema atual

execução. Estes podem ser, um plano de crítica, um processo de imputação automática, uma tabulação ou uma consulta a seus arquivos.

A realização deste processo é feita, atualmente, sem nenhum padrão de apresentação comum por parte da comunidade dos usuários, resultando em esforços diversificados para a realização desta tarefa, que tem se tornado uma tarefa árdua e onerosa pela falta de ferramentas automatizadas para auxiliar, dar maior segurança ao desenvolvimento destas tarefas e promover uma formalização. Entretanto, pode-se constatar uma crescente forma técnica de apresentação destes planos, tanto por parte dos usuários (usuário final) como, também, por parte dos analistas de sistemas. No entanto, esta tecnologia torna-se desperdiçada, ou não tão proveitosa, tendo em vista encerrar-se na especificação dos planos.

(2) - Analisar procedimentos

Após estabelecidos, pelo usuário, os procedimentos necessários à realização de sua tarefa, cabe aos analistas responsáveis pelos serviços, realizarem a análise destes procedimentos, a fim de se constatar sua viabilidade para implementação em máquina.

Nesta fase constata-se um grande consumo de tempo para absorção dos procedimentos realmente desejados pelos usuários podendo ocorrer a necessidade de se voltar ao processo anterior, para reavaliação, junto ao usuário, dos

procedimentos por ele estabelecidos.

Após avaliados e acertados os procedimentos, o analista tem agora de especificar os procedimentos a nível lógico e físico para passagem das definições de programas. Aqui verifica-se também a inexistência de uma ferramenta automatizada que o auxilie nesta tarefa, que poderia chegar até à implementação de código, podendo inclusive aproveitar alguma parte das especificações feitas pelos usuários, e que amenizasse os problemas de comunicação entre usuários e analistas.

(3) - Codificar em linguagem de programação

Neste processo o programador é responsável por transformar a especificação física em código reconhecido por máquina, ou seja, escrever um programa em linguagem convencional.

Observa-se, nesta fase, que os processos dos programas normalmente são funções já generalizadas dentro do âmbito de processamento de dados, como críticas e consistências que envolvem pesquisas a tabelas, armazenamento de registros em memória e impressão de relatórios de erros. Isto, nas linguagens convencionais, consome um tamanho razoável de código a ser escrito, compilado e depurado. O uso de linguagens de programação de quarta geração, também, não é muito aconselhável devido à quantidade de tempo e memória necessária para seu uso com grande volume de dados.

Verifica-se, também, que o programa gerado não contém

nenhum atributo que o qualifique como representante fiel dos procedimentos pré-estabelecidos, nem tão pouco, os processos anteriores a este contribuem para que, nesta fase, se tenha uma parametrização que venha a ajudar na certificação desta codificação.

(4) - Verificar procedimentos

Este processo é caracterizado pela execução em máquina dos procedimentos estabelecidos pelo usuário e transformados num programa.

Este processo pode causar um retorno aos três processos anteriores, por motivo de erros na própria codificação ou lógica do programa, por uma má definição dos procedimentos pelos analistas, ou por um mal entendimento entre os objetivos reais do usuário e os interpretados pelo analista.

Outra característica a ser observada neste ciclo de atividades, é a dinâmica das alterações efetuadas sobre um plano de crítica e regras de imputação.

Isto se deve em parte à falta de um método para especificação de um plano de crítica e seu correspondente plano de imputação, de um modo fechado, pelos profissionais de estatística. Quando da especificação de um plano de crítica com o plano de imputação correspondente a ser usado, pode ocorrer o fato de uma regra de imputação, especificada para correção de um dado de modo a atender uma

crítica, ao ser efetuada provoque um novo erro, devido ao não atendimento de outra crítica, pertencente ao mesmo plano de crítica.

Normalmente os testes que se realizam não conseguem detectar todos estes tipos de erros, antes que a pesquisa entre em produção.

Outro problema que se oberva também na especificação dos planos de crítica, relaciona-se à crítica de limite de valores, principalmente para pesquisas econômicas, na qual os limites são estabelecidos previamente, com base em pesquisas anteriores e projetados para o ano corrente. No entanto, só quando se começam a processar os dados coletados é que se pode ter uma idéia dos valores de limites aceitáveis para a pesquisa, devido a mudanças nos quadros econômicos de uma pesquisa para outra ou, às vezes, no decorrer da própria pesquisa.

Estas alterações efetuadas após o sistema entrar em fase de produção, causam os mesmos tipos de problemas que ocorrem em qualquer tipo de sistema e que são tão abordados pela Engenharia de Software: Estouro de prazos e, o aumento do custo do sistema, uma vez que o custo para manutenção corretiva num sistema aumenta numa escala logaritmica de acordo com a fase do Ciclo de Vida em que será efetuada a manutenção, conforme evidenciado por BOEHM[3], e deficiência na documentação do sistema, que por motivos de prazos, geralmente é sempre relegada, fazendo com que muitas vezes, em sistemas de apuração de dados, o documento final do plano de crítica e ou imputação seja

encontrado apenas no código do programa.

O uso das técnicas sugeridas pelos especialistas em Engenharia de Software, como por exemplo Gane[4], DE MARCO[5] e BOEHM[3], embora atenuem bastante estes problemas, não os resolvem totalmente, particularmente para este tipo de sistemas de apuração de dados, onde as alterações para um sistema que já entrou na fase de produção, não podem ser previstas e analisadas nas fases anteriores do sistema, devido a serem provocadas por falta de uma metodologia de consistência das regras de crítica e, para os casos de sistemas de apuração de pesquisas econômicas, a mudanças nos quadros econômicos do país. Assim, o capítulo seguinte apresenta um estudo de algumas ferramentas disponíveis no mercado que visam atender estes problemas.

CAPÍTULO III

ESTADO DA ARTE EM FERRAMENTAS

Neste capítulo são apresentadas algumas ferramentas automatizadas, existentes no mercado, para aplicação em sistemas de apuração de dados, buscando atenuar os problemas de software encontrados no desenvolvimento destes sistemas. Para cada uma destas ferramentas é feita uma análise citando suas vantagens e desvantagens para uso genérico em qualquer tipo de pesquisa.

III.1 - CONCOR :

"CONCOR DATA EDITING PACKAGE" [6], é um sistema desenvolvido pelos membros do CELADE, no ano de 1974 para CONsistência e CORreção de dados.

É formado por um conjunto integrado de programas que podem identificar e corrigir dados inválidos ou inconsistentes que estejam sendo preparados para tabulação e análise.

Foi desenvolvido, especialmente, para tratamento de dados censitários e é um dos primeiros softwares

desenvolvidos para estes objetivos.

Requisitos para o arquivo de entrada:

- o arquivo tem de ser sequencial;
- todos os registros têm de ter tamanho fixo;
- tem de existir uma única identificação de questionário, formada por dois campos de identificação, podendo ser no caso dos censos demográficos, o código da região geográfica e o número do domicílio;
- cada registro tem de ter um código de tipo de registro.

Através da linguagem CONCOR, o usuário descreve o arquivo a ser processado, as edições (críticas e correções) e os tipos de estatísticas a serem efetuadas.

Esta linguagem consiste de três divisões:

"DICTIONARY DIVISION" - Contém os comandos que definem e descrevem os arquivos de entrada/saída, os itens de dados e as variáveis a serem usadas nas operações de edição do arquivo.

"EXECUTION DIVISION" - Contém os comandos para edição que se desejam sobre os dados e os tipos de estatísticas que deverão ser efetuadas.

"REPORT DIVISION" - Contém comandos para descrição de relatórios a serem impressos a partir das estatísticas realizadas sobre os dados do arquivo.

Nestas três divisões deve ser utilizado um formato padrão para a entrada das especificações.

A linguagem CONCOR é procedural e tem dois tipos de comandos:

1 - Comandos não executáveis (instruções para o compilador);

2 - Comandos executáveis, que podem ser divididos nas seguintes categorias:

- comandos para recodificação e conversão de dados;
- comandos aritméticos;
- comandos de verificação;
- comandos de controle de fluxo de processamento;
- comandos de alocação dinâmica de matrizes;
- comandos de geração de registros de saída.

As informações destas três divisões são a entrada para o sistema CONCOR que as analisa e, caso corretas, gera um programa COBOL correspondente. Este, quando em execução, lê um registro do arquivo de entrada, processa-o de acordo com as especificações fornecidas pelo usuário no programa

CONCOR e produz um arquivo "Limpo".

O CONCOR possui as seguintes funções de edição de dados:

- Verificação de estrutura - verifica se todos os registros que formam um questionário estão presentes, e se não existem registros extras para um questionário;

- Verificação de limite de valores - determina se o valor de uma variável está compatível com os limites de valores estipulados para ela;

- Verificação de consistências - verifica se dois ou mais valores de variáveis, do mesmo registro ou não, estão consistentes de acordo com as condições permitidas especificadas pelo usuário;

- Correção automática - permite que além da identificação de um erro numa variável, esta possa ser corrigida de acordo com as especificações do usuário. O método de correção através de matriz "HOT DECK" também é permitido.

- Arquivo de saída - permite a criação de um arquivo contendo os dados corrigidos, sendo seu formato idêntico ao do arquivo de entrada;

- Arquivos derivados - o CONCOR permite a criação de um arquivo que poderá conter qualquer combinação dos dados do arquivo de entrada, ou variáveis derivadas, permitindo assim a criação de um arquivo contendo dados recodificados ou reformatados;

- Relatórios de erros - o CONCOR gera estatísticas sobre os testes de condições executados e o número de imputações feitas. Os relatórios destas estatísticas são gerados de acordo com as descrições fornecidas pelo usuário.

- Relatórios de Crítica - o CONCOR pode produzir quatro tipos de relatórios:

  . número e percentagem de erros encontrados e correções efetuadas pelo CONCOR;
  . número e percentagem de erros encontrados e correções efetuadas para cada item de dados editado;
  . listagem de todos ou alguns itens de dados de um registro ou questionário;
  . número e percentagem dos valores que foram usados como correções para itens de dados particulares.

As informações para cada um destes relatórios são de completo controle do usuário, sendo a formatação dos

relatórios feita pelo CONCOR, podendo ser feita quebra ou seleção por área geográfica.

Para esta ferramenta, em relação aos objetivos propostos, podemos citar como vantagens e desvantagens os seguintes itens:

VANTAGENS:

- atribuição automática de valores a variáveis quando da ocorrência de incoerências com suas especificações;
- acesso a registros diferentes através de um índice de ocorrência para comparação de variáveis inter-registros;
- correção automática através de comandos de atribuição, podendo-se utilizar ou não uma matriz de imputação;
- geração automática de relatórios de estatísticas de ocorrências de erros e correções.

DESVANTAGENS:

- só permite arquivos sequenciais;
- todos os registros têm de ter o mesmo tamanho, sendo o tamanho máximo permitido de 999 bytes;
- o teste de validade para variáveis categorizadas tem de ser programado pelo usuário;

- só permite processamento por quebra de até dois níveis.

III.2 - UNEDIT :

O UNEDIT [7], é um software para crítica e correção de Dados Censitários a serem preparados para análise e tabulação. Este sistema embora possua algumas opções de correção automática, tem como seu maior objetivo a pesquisa de erro que precede a correção manual. Foi desenvolvido tendo como objetivo principal a aplicação em Censos Demográficos.

O sistema UNEDIT consiste de dois módulos. O primeiro examina as especificações do programa escrito pelo programador em linguagem UNEDIT, verificando os erros sintáticos e lógicos. O segundo módulo lê o arquivo de entrada e identifica os registros contendo os erros, de acordo com as especificações fornecidas pelo usuário.

As estatísticas dos erros, por tipo de erro e nome do campo errado, são impressas para cada área geográfica separada.

O usuário especifica seu programa utilizando cinco tipos de formulários. Três deles para o dicionário, um para especificação dos cálculos (aritméticos ou lógicos) a serem efetuados e um para especificação de críticas e imputações:

FORMULÁRIO 1 - DESCRICÃO DE ARQUIVOS: Contém comandos para definir e descrever os arquivos de entrada e saída, a forma de processamento (registro a registro ou um grupo de registros a cada vez) e o tamanho e composição da chave do registro.

FORMULÁRIO 2 - DESCRICÃO DE REGISTROS: Contém comandos para definir os registros do arquivo de entrada e suas variáveis. Admite como número máximo a definição de nove tipos de registros, e somente permite definir variáveis zonadas (numéricas ou não). Para o caso em que exista mais de uma ocorrência de um determinado tipo de registro num mesmo questionário, permite a especificação de uma variável para identificar cada ocorrência, variável esta que será utilizada para se fazer verificação de estrutura e/ou de consistência de valores de variáveis inter-registros.

FORMULÁRIO 3 - ESPECIFICAÇÃO DE CÓDIGOS: Especifica, por item de dado, os códigos válidos e valores "default" para os casos de erro (código inválido e branco), a serem utilizados nas operações. Este formulário é opcional.

FORMULÁRIO 4 - ESPECIFICAÇÃO DE CÁLCULOS PRÉVIOS À EDIÇÃO: É utilizado para atribuição de resultados de cálculos e comparações a novas variáveis, ou variáveis já existentes, definidas no formulário 2, e que serão

utilizadas no formulário 5. Permite também criar uma nova variável, a partir de uma variável já definida no formulário 1, para uma ocorrência identificada de um tipo de registro, o que facilitará a verificação inter-registros (por exemplo: para o caso de um questionário que contém mais de um tipo de registro de pessoa, podemos criar a variável IDADE-CHEFE a partir da variável IDADE do registro de pessoa correspondente ao chefe). Este formulário é opcional.

FORMULÁRIO 5 - ESPECIFICAÇÃO DE INCONSISTÊNCIAS: Especifica combinações de valores de variáveis que são considerados inconsistentes (erro). Também é utilizado para imputação de valores a variáveis erradas.

Para esta ferramenta podem-se destacar os seguintes itens como vantagens e desvantagens:

VANTAGENS:

- verificação automática da estrutura de um questionário, acusando ausência de tipo de registro no questionário;
- atribuição automática de valores a variáveis quando da ocorrência de inconsistência com suas especificações;
- verificação automática da validade de uma variável categorizada;

- identificação de um registro num grupo de registros repetidos do mesmo tipo;
- geração automática de relatórios de estatísticas de erros, e relatório de crítica.

DESVANTAGENS:

- sistema de correção automática muito limitado, consistindo apenas de comandos de atribuição;
- só permite a definição de nove tipos de registros;
- o uso de vários tipos de formulários e a especificação da linguagem em campos formatados torna difícil o aprendizado e a compreensão de um programa nesta linguagem.

III.3 - ATLAS :

ATLAS [8], é um sistema para geração de programas de Crítica, Imputação e Tabulação de Dados na linguagem PL/I.

O processamento deste sistema é constituido de duas etapas básicas: Descrição dos Dados e Descrição dos Resultados.

Na primeira, o usuário descreve o arquivo de entrada, seu tipo, os registros, as variáveis e as categorias existentes para cada uma das variáveis categorizadas. Além disto, descreve as variáveis auxiliares a serem utilizadas no programa, bem como os demais arquivos a serem usados no

processamento.

Na segunda parte, o usuário define o seu programa utilizando as estruturas permitidas pela linguagem que são formadas pelo uso de processos automatizados, pré-programados, cada um tendo uma função específica, e um conjunto de instruções inerentes a cada processo.

O sistema ATLAS suporta as seguintes funções de edição de dados:

. Correção Automática (através de atribuição de valores e/ou através do processo IMPUTACAO podendo usar o modelo de matriz "HOT-DECK").

. Crítica - Geração automática do relatório de crítica, opcionalmente, com as estatísticas dos erros emitidos ou gravação do arquivo de crítica;

. Geração automática de procedimentos da atualização bem como do relatório de ocorrências da atualização;

. Geração do arquivo de Saída, arquivo de entrada corrigido ou arquivos formatados pelo usuário;

. Geração automática de rotinas de pesquisa de códigos;

. Geração de relatórios de frequências, tabulações, e relatórios formatados pelo usuário;

A linguagem de especificação consiste de dois módulos:

DICIONÁRIO: contém comandos para definir e descrever os arquivos de entrada e saída, os registros, as variáveis do registro de entrada (que poderão ser matrizes) assim como as variáveis de trabalho a serem usadas no programa e a descrição dos arquivos auxiliares;

PROGRAMA: contém os processos e procedimentos para a Crítica, Imputação e Tabulação dos Dados, assim como para a emissão de relatórios;

Nos dois módulos deverá ser utilizado um formato padrão para a entrada das especificações, um formulário para a descrição dos dados e outro para a descrição dos resultados.

A informação contida nestas duas divisões constitui a entrada para o sistema ATLAS. O sistema faz uma análise sintática dos comandos e, se não houver erros, gera um programa fonte em PL/I. Este é compilado, linkeditado e poderá ser testado antes de entrar em produção.

O sistema oferece a vantagem de se poder tratar o módulo dicionário separadamente, fazendo uma análise deste, permitindo assim que vários programas utilizem o mesmo dicionário.

O ATLAS possui os seguintes processos automatizados:

. processos de início e término do programa;

. processos de leitura dos arquivos de entrada;

. processo para controle de execução;

. processos para criação de dados;

. processos para acompanhamento;

. processos para crítica;

. processos para atualização;

. processos para emissão de relatórios;

. processos para gravação de arguivos;

. processos para tabulação de dados;

No que se refere à capacidade de processamento o sistema ATLAS permite processar qualquer tipo de arquivo, com registros de tamanho até 32767 bytes, e admite para as variáveis qualquer tipo de formato, e redefinição. Possibilita o processamento dos registros do arquivo de entrada de forma sequencial, assim como um grupo de registros armazenados em memória a cada vez, em função das variáveis especificadas para a quebra. Permite, também, o tipo de processamento "balanced-line" entre o arquivo de entrada e os arquivos paralelos declarados como arquivos de referência.

A linguagem deste sistema permite as facilidades de chamadas de rotinas externas, geração de rotinas de pesquisas de códigos e passagem de parâmetros externos para o programa.

O sistema permite, ainda, que a listagem impressa pelo processo CRITICA sirva para que o usuário realize as

correções manuais necessárias na própria listagem, e esta sirva como o formulário para a entrada de dados. Por sua vez, o arquivo gravado na entrada de dados poderá entrar diretamente no processo de ATUALIZACAO, definido nesta linguagem, que fará automaticamente as atualizações no arquivo principal.

Como vantagens e desvantagens deste sistema podemos citar os seguintes itens:

VANTAGENS:

- atribuição automática de valores a variáveis, dependendo dos valores de outras variáveis;
- atribuição automática de valores a variáveis, dependendo de intervalos de valores de outras variáveis;
- permite processar qualquer tipo de arquivo: sequencial, fixo, variável, indexado,etc;
- permite processamento por quebra de até 10 variáveis especificadas para cada processo;
- permite a geração automática de rotinas de pesquisa de códigos declarados ou armazenados em arquivo;
- permite o processamento de arquivos paralelos, podendo ser lidos em forma de "balanced-line" com o arquivo principal;
- emissão do relatório de crítica de forma a poder ser usado na entrada de dados para posterior atualização;

- crítica automática de variáveis categorizadas.

DESVANTAGENS:

- especificação da linguagem em formulários formatados, obrigando que cada processo e comandos sejam especificados em colunas pré-determinadas dificultando, assim, seu aprendizado e o entendimento dos programas escritos;
- não permite o aninhamento dos processos da linguagem;
- não permite identificação automática de um registro dentro de um grupo de registros do mesmo tipo;
- não permite verificação automática da estrutura dos questionários;
- o relatório de crítica, gerado, só permite a listagem de variáveis pertencentes ao mesmo registro criticado, não permitindo a listagem de variáveis de registros diferentes envolvidas numa mesma crítica.

III.4 - SISQUE :

O SISQUE - Sistema para processamento de questionários, [9], consiste de uma pseudo-linguagem dirigida para verificação de erros em questionários, e geração de variáveis e/ou tabelas a partir de variáveis pertencentes a um questionário armazenados num arquivo.

Foi desenvolvido pela Central de Processamento de Dados da Universidade Federal de Viçosa, para o trabalho de preparação dos dados de questionários de pesquisas sócio-econômicas.

A sintaxe da linguagem SISQUE é análoga à da linguagem de programação SPSS, e o sistema funciona em quatro modalidades no tratamento dos dados: Fechamento, Consistência interna, Agregação e Análise.

Os cartões-comandos que compõem a linguagem utilizam as colunas 1 a 10 para o nome do comando. Da coluna 11 até a coluna 72 devem ser colocados os parâmetros do comando, e da coluna 73 a 80, a identificação da sequência dos cartões.

São permitidos os seguintes comandos:

- OBJETIVO: este comando pode ser especificado mais de uma vez no início do programa e especifica a modalidade de processamento a ser executada;

- ARQUIVO: neste comando são descritas as características do arquivo que contém os dados a serem trabalhados;

- CAMPOS: este comando descreve a distribuição lógica das variáveis dentro do arquivo, seu formato, tamanho e registro a que pertencem;

- COMPUTE: este comando tem por finalidade a geração de novas variáveis por meio de uma expressão aritmética

qualquer envolvendo outras variáveis e/ou constantes, funções matemáticas, etc;

- IF: este comando permite a tomada de decisão dependendo da expressão lógica-relacional fornecida;

- CONSISTE: este comando faz a verificação lógica da expressão contida entre parênteses. Se a expressão for falsa, o sistema imprimirá uma mensagem de erro acompanhada dos nomes e valores das variáveis envolvidas no comando e a identificação do questionário;

- IMPRIME: causa a impressão das variáveis numéricas, tanto originais quanto criadas no processamento, descritas no campo de especificações de parâmetros do comando (colunas 11-72);

- GRAVA: com o uso deste comando é possível gerar um novo arquivo de dados com as variáveis selecionadas (tanto originais como criadas no processamento);

- TABELA: serve para a geração de tabelas de variáveis com até três quebras. Estas tabelas contêm somatório, média, desvio, variância e número de ocorrências para cada célula da matriz de quebras;

- SELECIONE: sua função é a de selecionar parte dos

registros do arquivo de entrada que satisfaçam as condições explicitadas na expressão lógica-aritmética fornecida.

O primeiro comando da linguagem SISQUE a ser especificado é o comando OBJETIVO que definirá a modalidade da tarefa a ser realizada. Estas podem ser FECHAR, CONSISTIR, AGREGAR, ou ANALISAR:

FECHAR: O programa executará os fechamentos horizontal e vertical para cada questionário, ignorando os demais comandos especificados no programa.

O fechamento horizontal serve para verificar a exata transcrição dos dados em cada registro do questionário, para o qual são somados todos os campos definidos para o registro, sendo verificado se a soma confere com o campo de soma especificado no comando CAMPOS, definido com o nome padronizado FECHA. Para os registros em que não tiver sido especificado este campo, não será efetuado o fechamento deste registro.

O fechamento vertical tem a função de verificar a exata presença dos registros que compõem o questionário, isto é, os registros podem estar ausentes por não conterem informações, ou devem estar presentes se as informações existem. Se este tipo de teste for para ser efetuado, então deve existir uma declaração de registro no comando CAMPOS, com nome do campo da variável FECHAVERT. O sistema consistirá o

somatório dos números de identificação dos registros com este campo, para cada questionário.

CONSISTIR: O programa executará todos os comandos especificados no programa, exceto os comandos GRAVA, IMPRIME, e TABELA. O processamento é feito questionário a questionário e os comandos CONSISTE que tiverem resultado lógico falso acusarão saída em disco para posterior classificação e impressão. A saída consta de número do questionário, nomes e valores das variáveis envolvidas no teste.

AGREGAR: Esta opção está relacionada com a criação de um novo arquivo de dados, contendo os valores de variáveis originais ou geradas a partir dos comandos COMPUTE e IF. As variáveis para este arquivo deverão ser descritas no comando GRAVA ou IMPRIME.

ANALISAR: Esta opção prepara o sistema para a geração das tabelas que deverão ser impressas ao final do programa. Estas tabelas são definidas através dos comandos TABELA.

Para este sistema podem ser considerados como vantagens e desvantagens os seguintes itens:

VANTAGENS:

- crítica automática dos fechamentos horizontal e

vertical;

- opção de especificação da modalidade de processamento desejada, sem que seja necessário, a especificação de programas separados para cada modalidade.

DESVANTAGENS:

- campo de identificação do questionário consistindo apenas de dois campos: número do questionário e tipo de registro;
- só permite processamento de arquivos sequenciais;
- não permite crítica automática de consistência das variáveis com suas especificações;
- especificação da linguagem de modo formatado obrigando a especificação dos comandos em colunas pré-formatadas;
- não permite especificação nem crítica automática de variáveis categorizadas.

III.5 - AERO :

O sistema AERO - Generalized Data Editing System foi especificado por K. Szász, A. Szabó e I. Varasdy [10] do Departamento de Desenvolvimento de Aplicações do Centro de Computação do CSO (Oficina Central de Estatística) da Hungria.

Sua primeira versão foi implementada em 1976 com o

objetivo de sua utilização no censo demográfico de 1980, e foi usado satisfatoriamente na realização dos censos demográficos da Espanha, Hungria e Iugoslávia.

O AERO foi incluido como um dos pontos básicos de trabalho no UNDP Statistical Computing Project (SPC) e como resultado desta cooperação foi criada sua segunda versão incrementada.

O sistema se caracteriza pela geração de programas para depuração de dados qualitativos, em linguagem PLI, que permitem as seguintes funções de processamento:

- validação de variáveis individuais com base numa lista de códigos armazenados num Dicionário de Dados;

- verificação de relacionamentos lógicos ou aritméticos entre as variáveis de um mesmo registro, usando regras de conflito especificadas (relações lógicas inaceitáveis entre os valores de duas ou mais variáveis);

- imputação automática dos registros errados através de um procedimento probabilístico baseado nas próprias regras de conflito ou através de um procedimento determinístico baseado em regras de imputação determinística especificadas;

- verificação da composição de grupos de registros

(por exemplo: o conjunto de pessoas que habitam o mesmo domicílio) e correção automática destes erros;

- produção de uma listagem com diagnóstico detalhado dos registros errados e das correções realizadas;

- produção de estatísticas sobre o resultado da validação de variáveis e verificação das regras de conflito;

Todas estas funções são executadas por programas gerados a partir de especificações fornecidas pelo usuário para cada situação específica.

Para cada aplicação o usuário deve especificar:

. a descrição do registro do arquivo de entrada;
. a lista de valores aceitáveis para cada variável pertencente ao registro;
. os parâmetros que descrevem as funções de depuração necessárias que podem ser: validação das variáveis, verificação do cumprimento das regras de conflito, listagem dos erros, imputação automática, etc.

A lista de valores aceitáveis para cada variável é especificada no módulo Dicionário de Dados do sistema, que pode ser usado por vários programas e contém comandos para inserção, deleção e modificação de variáveis no Dicionário.

Para cada variável é necessário especificar: nome,

número máximo de caracteres que compõem a variável, tipo da variável (N-numérica, C-alfanumérica) e lista de códigos válidos.

A descrição do registro de entrada é feita fazendo-se referência às variáveis especificadas no Dicionário, indicando-se para cada variável do registro de entrada o nome, posição inicial e final dentro do registro, tipo de variável e tipo de codificação (se a variável deve ser considerada na depuração ou não).

As regras de conflito, ou regras Y, que especificam as condições inaceitáveis para as variáveis e, portanto, a serem detectadas como erradas têm o seguinte formato:

Y(número) : (Expressão Lógica).

As regras de imputação determinística, ou regras X, que determinam através da especificação de uma situação inaceitável a imputação a ser realizada para estes casos, têm o seguinte formato:

X(número):(Expressão Lógica) DO (Atribuição).

Os parâmetros que especificam as funções a serem executadas pelo programa gerado podem ser:

CVAL - verifica se as variáveis têm valores válidos com os especificados no Dicionário de Dados;

RELA - verificação do cumprimento das regras de conflito;

ACOR - imputação automática;

ESTAT - elaboração das estatísticas de erros;

ELIST - listagem dos registros errados com informações dos erros ocorridos em cada um;

SEQN - listagem dos registros errados com informações dos erros ocorridos em cada um indicando o número de ordem do registro;

O sistema AERO possui ainda a opção de uso de um Analisador de Regras para verificação da consistência lógica do conjunto de regras de conflito, regras Y, especificado pelo usuário num programa AERO.

Este analisador, além de eliminar redundâncias no conjunto de regras especificado, detecta também inconsistências entre elas.

Como vantagens e desvantagens deste sistema podem ser citados os seguintes itens:

VANTAGENS:

- uso de um Dicionário de Dados contendo a descrição das variáveis com seus códigos válidos, armazenados em máquina, permitindo que as variáveis possam ser usadas em arquivos diferentes sem necessidade de se redefini-las novamente;
- uso de imputação automática em dois modelos: o determinístico e o probabilístico;
- conjunto de comandos simples a serem usados na

especificação do usuário;

- analisador de regras para consistência das regras de conflito especificadas pelo usuário;

DESVANTAGENS:

- restrições do arquivo de entrada que só permite o processamento de arquivos sequenciais, com variáveis do tipo qualitativo com formato alfanumérico ou decimal zonado e todos os registros de um mesmo tipo com tamanho máximo de 9999 bytes;
- na descrição do registro do arquivo de entrada, têm de ser especificadas todas as variáveis que compõem o registro, mesmo que estas não sejam usadas no processamento;
- não permite o uso nem geração de arquivos paralelos ao de entrada;

III.6 - CAN_EDIT:

O sistema CAN_EDIT [11] foi desenvolvido pelo centro de estatísticas do Canadá, para aplicação no censo demográfico de 1976 com o objetivo de detecção de erros e correção automática dos dados coletados.

Este sistema é voltado totalmente para a análise e correção de dados qualitativos.

Para a operação deste sistema é necessário que os

dados sejam armazenados no RAPID e o dicionário de dados seja definido pelo MDM (Meta Data Memory) do RAPID.

O sistema CAN_EDIT é operado em duas fases chamadas de Preparação e Produção.

A fase de Preparação é composta de duas atividades:

- Definição dos Dados: onde é feita a criação do dicionário de dados em termos de variáveis e conjuntos de valores permitidos para cada uma das variáveis, os arquivos e campos que descrevem os arquivos que contêm os dados coletados, e as relações, atributos e valores que definirão a base de dados do RAPID.

- Especificação e Análise das Regras de Crítica: nesta atividade são especificadas as regras de crítica a serem aplicadas aos dados, que são expressas sob forma de expressões lógicas que especificam combinações de valores de variáveis não permitidas.

  Uma vez especificadas estas regras, o sistema as analisa fazendo diagnósticos de erros de consistência encontrados entre elas e gerando o conjunto de regras derivadas a partir das regras especificadas.

A fase de Produção é, também, composta de duas atividades:

- Carregamento da Base de Dados: esta etapa consiste da

conversão e armazenamento dos arquivos que contêm os dados coletados na Base de Dados do RAPID.

- Execução da Crítica e Correção dos Dados: O CAN_EDIT, a partir do conjunto de regras especificado pelo usuário e das regras derivadas por ele, gera um programa de crítica e correção das variáveis detectadas como erradas.

O processo de imputação automática usado pelo CAN_EDIT usa a filosofia de matrizes HOT-DECK e todas as correções efetuadas são determinadas pelo sistema a partir das regras de crítica especificadas sem que haja especificação de comandos do usuário. Assim a fase de pré-crítica e crítica são executadas simultaneamente.

Como vantangens e desvantagens deste sistema podemos citar os seguintes itens:

VANTAGENS:

- analisador das regras de crítica especificadas;
- geração das regras de imputação para correção do arquivo, a partir das regras de crítica especificadas;

DESVANTAGENS:

- necessidade do RAPID para uso do sistema;
- só trabalha com dados qualitativos;

III.7 - ANÁLISE DAS FERRAMENTAS APRESENTADAS

Analisando-se cada uma das ferramentas apresentadas constata-se sua eficiência no atendimento dos objetivos propostos por cada uma delas.

Embora cada um destes sistemas apresentados se proponha atender a uma situação específica de processamento de dados e se promulgue para uso genérico em outros tipos de situações, isto nem sempre é garantido devido às restrições apresentadas por cada uma destas ferramentas, tais como: limitações para o arquivo de entrada, chaves que compõem o registro, tipos de variáveis, estrutura dos questionários, etc.

Outra característica que se observa nestas ferramentas é que cada uma oferece uma ou mais funções altamente vantajosas, mas que se esbarram com outras funções do sistema a que pertencem fazendo que este não possa ser usado de modo genérico.

Como exemplo disto pode ser citado o fato do UNEDIT que permite uma verificação automática e correção para os casos de inconsistência dos valores das variáveis com suas especificações e verificação automática da estrutura de um questionário, mas no entanto, possuindo a função de correção automática muito limitado, consistindo apenas de comandos de atribuição não fazendo uso de correção automática por matrizes de imputação (HOT DECK), como é o caso dos sistemas CONCOR, ATLAS e AERO.

Um outro aspecto que se pode observar nestas

ferramentas é que embora construidas para serem usadas pelos usuários, e suas linguagens sejam simples, em todas elas é exigido um certo grau de conhecimento, por parte dos usuários, de lógica de processamento dos programas, o que nem sempre existe, nem deveria ser necessário. O usuário técnico estatístico responsável pela pesquisa, normalmente faz seus planos de crítica dos formulários baseado nas variáveis que compõem estes questionários sem se preocupar se as variáveis pertencem ao mesmo registro ou não, o que normalmente é definido numa etapa posterior de arquitetura do sistema pela equipe de analistas que desenvolvem o projeto físico e, neste caso, para o uso destas ferramentas não podem aproveitar diretamente as especificações das críticas especificadas pelos usuários.

Uma situação ideal seria a possibilidade de termos uma ferramenta em que pudessemos reunir todas as vantagens, oferecidas por cada uma das ferramentas descritas, incorporadas num só sistema, acrescida de novas características visando atender os objetivos propostos.

Analisando-se as ferramentas apresentadas, verificamos que o sistema de maior abrangência de aplicação é o ATLAS, devido a incorporar tanto processos de crítica como imputação e tabulação, e a sua grande variedade de funções permitidas que são:

- possibilidade de processamento de qualquer tipo de arquivo com qualquer tipo de variáveis;
- funções para grupamento de variáveis;

- uso de arquivos paralelos tanto de leitura como gravação;
- correção automática por atribuição ou matrizes de imputação;
- emissão de relatórios de crítica para correção manual e posterior atualização pelo próprio sistema;
- impressão de relatórios formatados pelo usuário;
- pesquisa automática de códigos;
- formatação das chaves de controle de quebra para os processos;
- funções de tabulação,

No entanto, o sistema ATLAS não incorpora características, altamente vantajosas, encontradas nos outros sistemas analisados, que o tornariam uma ferramenta de maior potencialidade, tais como:

- verificação automática dos valores dos conteúdos das variáveis com suas especificações;
- checagem automática da estrutura de um questionário;
- crítica direta de variáveis inter-registros.

Observando-se os problemas apresentados no capítulo II, referente à descrição do sistema atual, verifica-se que nenhuma destas ferramentas atende ao requisito de uma passagem automática entre as especificações do usuário e o programa correspondente, devido a todas estas ferramentas serem construídas para uso na fase de implementação do

Ciclo de desenvolvimento de sistemas de apuração de dados. Uma situação que resolveria este problema seria a possibilidade de termos uma linguagem de programação para a especificação da lógica de processamento e ao invés de recodificarmos os procedimentos estabelecidos no Plano de Crítica, pelo usuário, pudessemos inclui-los diretamente no programa.

Assim, se fosse criada uma ferramenta, tendo como base as características do sistema ATLAS, escolhido, como já citado, por sua maior abrangência e potencialidade das funções oferecidas, com a incorporação das características mais vantajosas, encontradas nos outros sistemas, e incluindo-se novas características que:

- permitam que as próprias especificações dos Plano de Crítica do usuário possam ser inseridas para geração automática;
- dêem uma nova forma de especificação da linguagem tornando-a mais livre sem necessidade de formulários;
- permitam o aninhamento de suas funções;
- permitam a especificação de novos comandos como macros, para especificação de expressões lógicas, aritméticas e acesso a variáveis de registros diferentes;

teríamos uma ferramenta poderosíssima que, se não resolve totalmente, atenua num alto grau os problemas de software

apresentados nos capítulos anteriores deste trabalho.

Seguindo esta linha de raciocínio, baseado nas ferramentas analisadas e nos problemas citados, é apresentado no capítulo seguinte a proposta de uma nova ferramenta para o desenvolvimento de sistemas de software de apuração de pesquisas.

CAPÍTULO IV

SISTEMA PROPOSTO

Neste capítulo é apresentada a proposta de um novo sistema para atendimento dos problemas citados nos capítulos anteriores, apresentando a especificação lógica de cada uma das ferramentas propostas.

IV.1 - OBJETIVOS:

Analisando-se o sistema atual descrito anteriormente e o conjunto de ferramentas apresentadas, é constatada a necessidade de uma nova ferramenta que automatize e auxilie aos usuários, analistas e programadores, a elaborarem seus procedimentos de forma mais rápida, segura e independente de excessivo conhecimento técnico específico.

Os objetivos a serem atendidos pelo sistema proposto podem ser especificados como:

- Elaboração de um conjunto de ferramentas que:

. automatize e auxilie os usuários na elaboração da

especificação dos procedimentos de crítica (Planos de Crítica);

. automatize a geração automática dos programas de crítica, através da elaboração de uma linguagem de programação de altíssimo nível, onde as especificações dos usuários possam ser aproveitadas;
. promova uma interação entre a especificação de uma tarefa e o desenvolvimento de sua implementação.

Os benefícios que serão alcançados com estas novas ferramentas são:

. a especificação dos procedimentos de crítica de forma mais rápida, segura e independente de conhecimentos específicos;
. uma maior integração entre as fases de análise, implementação e certificação;
. redução no tempo de especificação e implementação dos programas de crítica;
. redução dos problemas de comunicação entre usuários, analistas e programadores.

IV.2 - INTERFACES EXTERNAS E FLUXO DE DADOS

A figura IV.1 apresenta o Diagrama de Fluxo de Dados do sistema proposto representando o conjunto de atividades do desenvolvimento da fase de Crítica de um sistema de

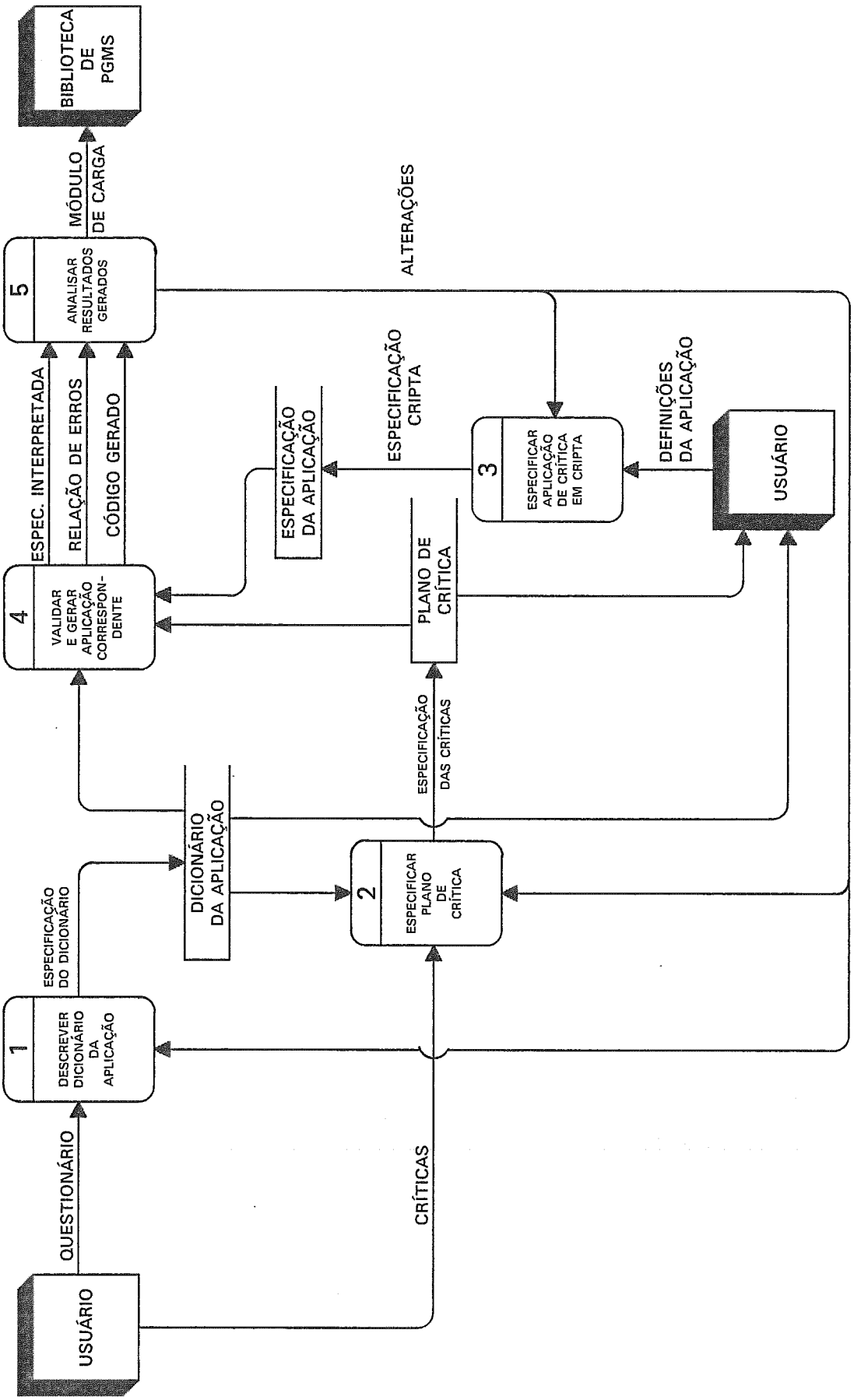


FIGURA IV.1 - Diagrama de fluxo de dados do sistema proposto

apuração de uma pesquisa de coleta de dados, utilizando as novas ferramentas propostas.

Neste diagrama temos as seguintes entidades externas:

"USUÁRIO": representa o conjunto de pessoas que usarão o sistema, incluindo tanto o usuário técnico da pesquisa quanto o usuário da área de informática, analista ou programador.

"BIBLIOTECA DO USUÁRIO": esta entidade representa o destino do produto gerado pelo sistema, ou seja, os programas de crítica gerados. Poderia-se representar esta entidade como um armazenamento, mas representando-a como entidade externa fica explícito que esta biblioteca é de uso particular do usuário.

A atividade 1, "Descrever Dicionários", consta de se fazer a descrição de cada variável, registro e características do arquivo que conterá os questionários a serem criticados e dos arquivos a serem usados paralelamente a este no processamento e a descrição das variáveis auxiliares necessárias na aplicação.

Para cada arquivo são especificados os tipos e formatos de registros permitidos, se obrigatórios ou não no questionário e o número máximo de ocorrências de cada um.

Para cada registro são especificadas as variáveis que o formam e, para cada variável, são especificadas a posição, tamanho, formato e, se categorizadas, as

categorias permitidas.

A ferramenta utilizada para esta atividade, DICAPLIC, Dicionário da Aplicação, será descrita adiante.

De posse do Dicionário da Aplicação, já definido e criticado, o usuário(técnico) passa então à atividade 2, "Especificar Plano de Crítica", onde através da ferramenta PCAUT, a ser descrita adiante, ele definirá o conjunto de críticas a serem aplicadas ao questionário ou questionários descritos no dicionário.

A partir das atividades 1 e 2 realizadas, Especificação do Dicionário da Aplicação e Plano de Crítica, o analista/programador pode especificar a aplicação de Crítica correspondente, atividade 3, usando a ferramenta Cripta que é uma linguagem de programação de altíssimo nível a ser usada com interação do Dicionário da Aplicação, DICAPLIC, e Planos de Crítica associados, PCAUT.

A atividade 4, "Validar Especificação e Gerar Aplicação", consta de a partir das especificações dos dicionários fornecidas, da especificação do Plano de Crítica, opcional, e da especificação da aplicação desejada, verificar a consistência e validade destas entradas e gerar a aplicação correspondente de forma a poder ser executada em máquina.

IV.3 - REQUISITOS PARA O SISTEMA PROPOSTO

A seguir são descritos os requisitos a serem considerados para o desenvolvimento de cada uma das ferramentas citadas no item anterior de descrição das atividades do sistema proposto.

IV.3.1 - DICIONÁRIO DA APLICAÇÃO - DICAPLIC

Através desta ferramenta serão descritas as características dos arquivos a serem processados, as variáveis que o compõem, bem como as variáveis auxiliares a serem usadas na aplicação.

Para especificação desta ferramenta foram considerados os seguintes requisitos operacionais:

- permitir a especificação das características do arquivo de modo flexível, atendendo às opções permitidas pela linguagem Cripta (arquivo de tamanho de registros fixo, variável por tipo de registro ou variável por ocorrência);
- permitir definição de arquivos com registros de tamanho variável por número de ocorrências;
- permitir a definição de vários tipos de registros;
- permitir a definição da estrutura de um questionário através da especificação dos tipos de registros que o compõem, informando para cada um deles se são

obrigatórios ou não e o número mínimo e máximo de cada tipo de registro permitido por questionário;

- permitir a identificação das variáveis que compõem cada um dos tipos de registros, as variáveis que compõem a chave do arquivo e a variável que identifica o tipo de registro;
- permitir definição de variáveis quantitativas e qualitativas;
- permitir a definição de variáveis com formato alfanumérico, numérico (zonado, compactado, binário e ponto flutuante) e categorizadas;
- permitir a especificação das categorias que compõem uma variável categorizada;
- permitir a especificação de limites mínimos e máximos de valores para uma variável quantitativa;
- permitir a redefinição de variáveis;
- permitir a especificação de vetores;
- permitir a especificação de variáveis auxiliares.

IV.3.2 - PLANO DE CRÍTICA AUTOMATIZADO - PCAUT

Esta ferramenta servirá para os usuários técnicos das pesquisas especificarem seus planos de crítica dos questionários que compõem a pesquisa.

Para especificação desta ferramenta foram considerados os seguintes requisitos operacionais a serem atendidos:

. permitir a especificação da maioria das críticas a serem efetuadas neste tipo de sistemas, que normalmente constam da verificação dos conteúdos dos campos do questionário com seus formatos e valores permitidos, das críticas de totais para verificação dos totais informados com a soma das parcelas fornecidas e a crítica de consistência, para verificação dos valores de uma variável com um conjunto de outras variáveis;

. criação de comandos para especificação das críticas de modo simples que permita uma fácil compreensão e aprendizado pelos usuários, oferecendo:

  . funções para especificação dos cálculos de totais, frequências e expressões condicionais largamente utilizadas na especificação dos planos de crítica;

  . facilidades para o tratamento de variáveis inter-registros;

  . possibilidade para especificação dos casos em que o usuário não consiga expressar suas críticas através dos comandos permitidos como, por exemplo, a verificação de um código pertencer ou não a um cadastro, onde para estes casos, o usuário poderá especificar suas críticas especiais em linguagem natural e o programador desenvolverá a rotina correspondente;

. permitir o aproveitamento das especificações contidas no plano de crítica para inclusão nos programas de crítica correspondentes a serem gerados;

Os benefícios a serem alcançados com esta ferramenta são:

. criação de um formalismo para a especificação dos planos de crítica pelos usuários e consequentemente uma documentação;
. melhoria na comunicação entre analistas e usuários para entendimento dos planos de crítica especificados;
. redução no tempo de implementação dos programas de crítica , devido à possibilidade de inclusão da especificação dos usuários, para geração automática de código. Para as críticas em que não seja possível a geração automática das críticas, por estas estarem em linguagem natural, estas servirão como parametrização no programa de crítica correspondente, onde a rotina associada, especificada pelo programador, poderá ser verificada com os reais objetivos expressos pelo usuário.

IV.3.3 - A LINGUAGEM CRIPTA

A linguagem Cripta será usada para a especificação e codificação dos programas necessários à execução das fases de crítica, imputação e tabulação dos sistemas de apuração de dados.

Nestes tipos de sistemas pode-se constatar uma semelhança na arquitetura lógica dos programas e quando implementados em linguagens padrões como Cobol e PLI, isto faz com que haja um desperdício de esforços nas construções de algorítmos, que muitas vezes já foram desenvolvidos e testados, induzindo a erros de lógica, codificação e digitação que são responsáveis por grande parte dos erros que ocorrem e que, na maioria das vezes, são erros evitáveis.

Estes problemas se relacionam ao assunto que atualmente é muito discutido sobre a produtividade dos programadores [12] que pode ser definida como:

- a produtividade é diretamente proporcional ao número de funções corretas emitidas por unidade de tempo;

- a produtividade é inversamente proporcional ao número de defeitos introduzidos durante as fases de projeto, codificação, depuração e testes;

- a produtividade é diretamente proporcional à percentagem do código fonte não escrito a partir do

nada.

Ao analisarmos o código de um programa COBOL ou PLI verificamos que ele recai nas seguintes categorias:

. conjunto de dados, estruturas e procedimentos definidos pelo programador;
. acionadores lógicos: lógica para iniciar, processar e encerrar o programa. Esta parte do código pode ser 90% padronizada se forem utilizados modelos de programas padrões;
. lógica para entrada e saída de dados: leitura e gravação de um conjunto limitado e pré-definido de estruturas e arquivos. Mais de 90% deste código pode ser padronizado com o uso de rotinas de entrada/saida padronizadas;
. código para transformação de dados: consistindo na manipulação de dados, computações aritméticas e instruções para tomadas de decisão. Uma boa parte deste código não precisa ser escrita, repetidamente, a partir do zero.

Usando esta abordagem das quatro categorias, um programador só precisa escrever novo código para apenas 20 a 40% do código fonte necessário.

Com relação aos sistemas de apuração de dados, objeto deste trabalho, podemos acrescentar algumas categorias referentes à fase de processamento de crítica:

. emissão de relatórios de crítica, consistindo da indicação de erros de forma a este relatório servir para a codificação das alterações a serem efetuadas e entrada de dados das correções indicadas;

. atualização do arquivo, a partir das correções transcritas pela entrada de dados.

Os procedimentos de imputação e tabulação estão incluidos na categoria já citada de código para transformação de dados.

Assim, a linguagem Cripta se propõe a ser uma linguagem de altíssimo nível, que agilize e automatize a especificação dos procedimentos definidos pelos usuários, atendendo de modo mais específico, os tipos mais característicos dos procedimentos de crítica, imputação e tabulação.

Os tipos de procedimentos mais característicos na crítica, imputação e tabulação podem ser classificados como:

- Manipulação de dados:

. especificação de cálculos e críticas envolvendo variáveis de registros diferentes como:

. consistência de uma variável de um registro do questionário com outra variável de outro registro do mesmo questionário;

. cálculo do total de uma variável dentro do questionário (fechamento vertical);

. cálculo da frequência de registros, dentro de um questionário, que atendam a determinada condição;

. verificação da condição de uma variável pertencer ou não a uma tabela;

. consistência do valor de uma variável categorizada com seus códigos permitidos;

- Atribuição de novos valores a variáveis:

. designação de valores a uma variável dependendo do valor de outra variável pertencer a intervalos especificados;

. designação de valores a uma variável dependendo dos valores especificados de outras variáveis;

. designação de valores a uma variável dependendo da combinação dos valores de outras variáveis armazenadas e atualizadas numa matriz de imputação;

- Gravação de arquivos de saída:

. cópia de alguns registros especificados do arquivo de entrada;

. cópia de de uma lista de variáveis de determinados registros do arquivo de entrada, adicionados

opcionalmente de algumas variáveis auxiliares;

- Emissão de relatórios:

  . impressão de um relatório de crítica padronizado contendo a identificação dos registros, o código do erro e a lista de variáveis envolvidas no erro;
  . impressão de um relatório contendo uma lista de variáveis;

- Tabulação de dados:

  . cálculo de uma matriz contendo a frequência de ocorrências da combinação de duas variáveis;
  . cálculo de uma matriz contendo o somatório de uma variável de acordo com a combinação de duas variáveis;
  . cálculo de uma matriz contendo o somatório de uma variável com várias categorias por uma lista de variáveis quantitativas;

Para especificação desta linguagem foram considerados como requisitos os seguintes itens:

  . uso de comandos simples;
  . tipos de usuários a serem atendidos: analistas, programadores e usuário final;
  . permitir a especificação de programas concisos,

inteligíveis e modulares;

. permitir o uso de funções pré-definidas (procedimentos ou macros) para funções padrões nos processos de crítica, imputação e tabulação;

. permitir a inclusão de procedimentos de crítica, especificados pelo usuário final no Plano de Crítica, para geração automática de código;

. fazer a especificação e implementação desta linguagem de modo a permitir alterações futuras, novas versões, que poderão enriquecer ainda mais a linguagem.

CAPÍTULO V

PROJETO FÍSICO

Neste capítulo será especificado o projeto físico do sistema proposto, especificado no capítulo anterior. Assim sendo, este capítulo contém a especificação das entradas e saídas do sistema, sua arquitetura e projeto detalhado.

Para elaboração do projeto físico serão seguidas as etapas sugeridas por ROCHA [13] referentes ao "Roteiro para elaboração da especificação de projeto". No entanto, por este ser um sistema altamente atípico dos normalmente desenvolvidos, algumas adequações serão efetuadas sendo estas devidamente documentadas.

O projeto físico será desenvolvido tendo como objetivo sua implementação em máquina de grande porte IBM, com o uso da linguagem de programação PLI. Estas restrições de implementação, são feitas por razões do ambiente onde o sistema será operacional.

V.1 - ARQUITETURA DO SISTEMA

Observando-se o diagrama do fluxo de dados do sistema

proposto, apresentado no capítulo IV, podemos definir como limite de automatização a área tracejada no diagrama da figura V.1. As identificações de entrada e saída do sistema também estão indicadas neste diagrama.

O novo diagrama apresentado na figura V.2 apresenta o diagrama de fluxo de dados onde já foi eliminada a região de processamento manual e foram acrescentados os dispositivos de entrada e saída, representando o fluxo de dados físico do sistema.

Como podemos verificar neste último diagrama, o sistema consiste de apenas um programa. Isto se deve ao fato deste sistema ser constituido de apenas uma função automatizada com um comportamento semelhante a um "compilador", ou seja, a partir das especificações de entrada, fazer sua interpretação, validação e geração de código correspondente.

Um "compilador" é definido, pela literatura convencional como um programa que aceita como entrada um programa em linguagem de alto nível (programa fonte) e produz como saída um programa em linguagem de máquina (programa objeto).

Uma variação deste tipo de programa pode ser definido como um programa que aceite como entrada um programa em linguagem de alto nível e produza como saída, não um programa em linguagem de máquina, mas sim um programa em outra linguagem. Estes tipos de programas são denominados de "tradutores".

Podemos definir a estrutura de um "compilador" segundo

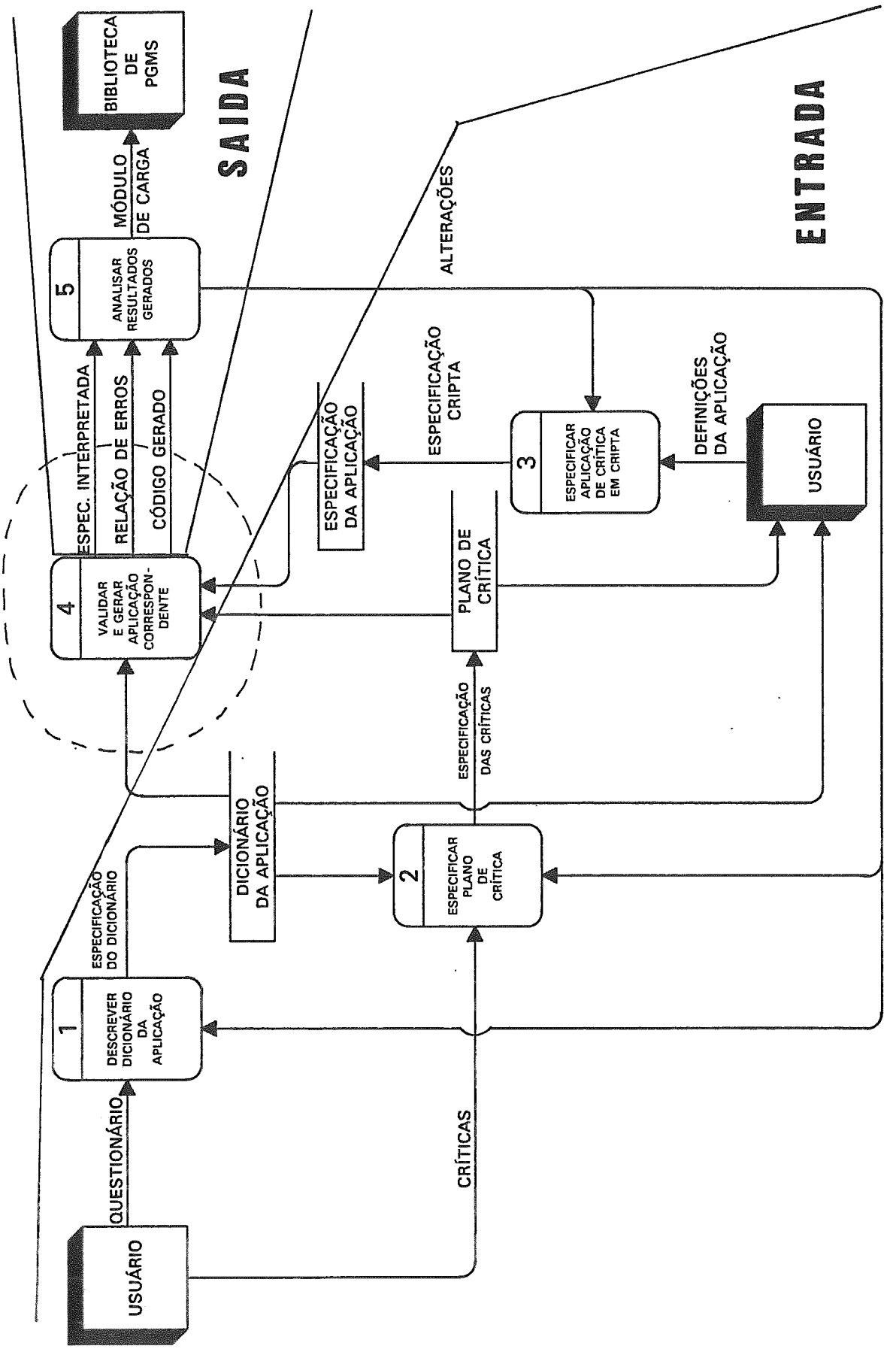


FIGURA V.1 - Diagrama de fluxo de dados do sistema proposto com limitação da parte automatizada e identificação das entradas e saídas do sistema

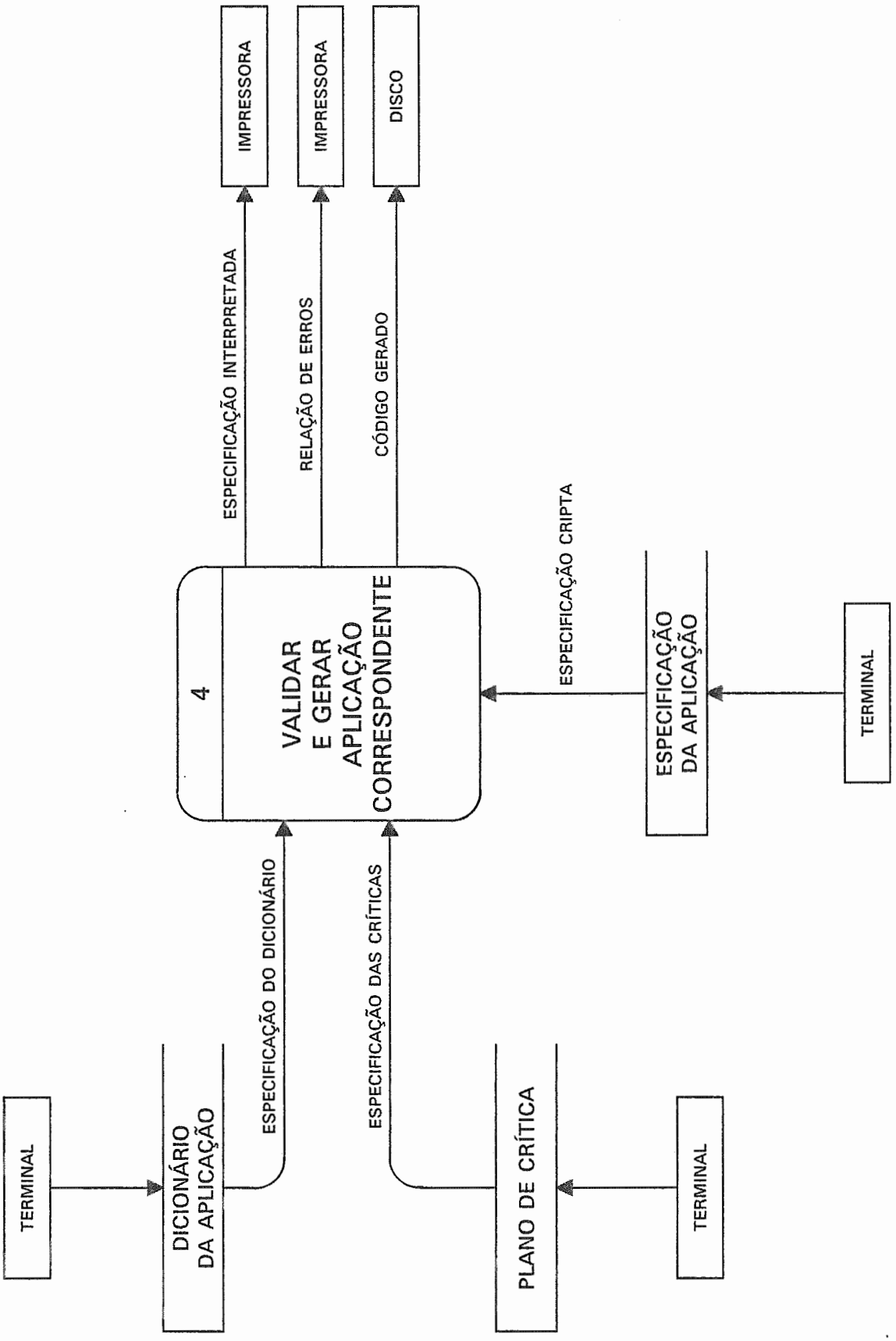


FIGURA V.2 - Diagrama do fluxo de dados físico do sistema

AHO e ULLMAN [14], como sendo formada pelos módulos representados na figura V.3, cujas funções são descritas a seguir.

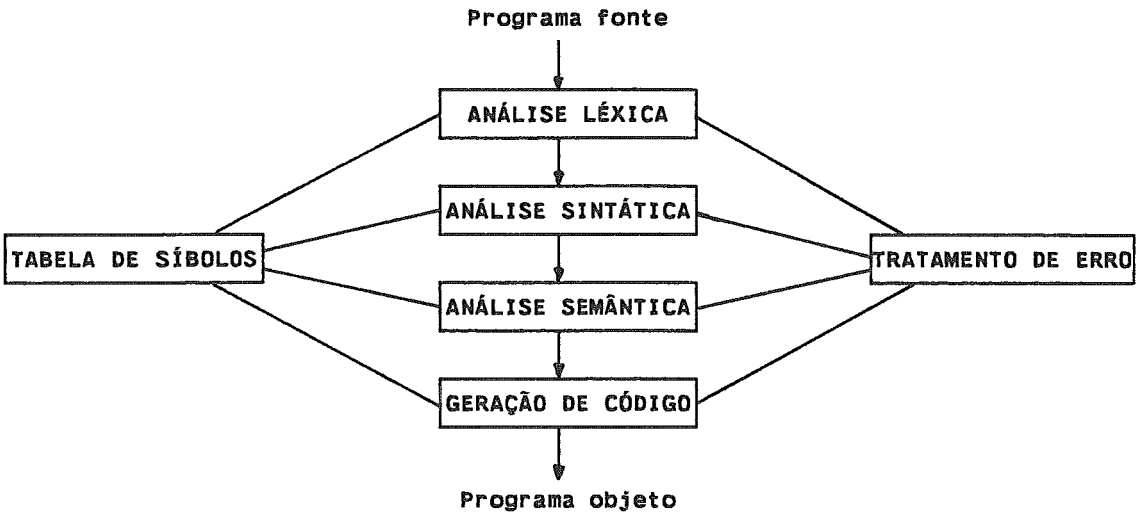


FIGURA V.3 - Fases de um compilador. Extraído de AHO e HULLMAN[11]

ANÁLISE LÉXICA :

É o módulo responsável pela interface entre a especificação de entrada e o processo de compilação. Este módulo lê a entrada caracter a caracter estruturando-a numa sequência de unidades atômicas chamadas "tokens", que representam uma sequência de caracteres que podem ser tratados como uma entidade lógica separada.

Como exemplo de "tokens" podem ser citados os identificadores, palavras reservadas, constantes, símbolos de pontuação, etc.

ANÁLISE SINTÁTICA :

Este módulo tem a função de verificar se a sequência de "tokens" da especificação de entrada, fornecida pelo analisador léxico, obedece aos moldes, ou regras de sintaxe, estabelecidos pela gramática da linguagem que está sendo compilada, identificando estruturas para as sequências de "tokens" como expressões, comandos de atribuição, declaração de rotinas, etc.

Assim, este módulo tem também como função identificar e diagnosticar os erros de sintaxe ocorridos, como por exemplo, ao analisar a sequência A + B + * C, para uma gramática com especificação de expressões análoga às expressões aritméticas convencionais, deve ser indicado o erro relativo à presença de dois operadores consecutivos.

ANÁLISE SEMÂNTICA :

Este módulo tem a função de verificar a obediência às regras semânticas especificadas para a linguagem, que determinam a consistência dos atributos de cada comando da linguagem.

Como exemplo de regra semântica, pode ser citada a consistência dos atributos de cada operando especificado numa expressão aritmética, ou comando de atribuição.

Este módulo tem também a função de diagnosticar a ocorrência de erros de semântica. Ex: não permitir a atribuição de um valor alfanumérico a uma variável

numérica.

GERAÇÃO DE CÓDIGO :

De posse das estruturas identificadas pelo analisador léxico e verificadas sem erros pelos analisadores sintático e semântico, este módulo tem a função de produzir um código executável em máquina correspondente às funções especificadas na entrada.

TRATAMENTO DE ERRO :

Este módulo será chamado sempre que seja detectado um erro por uma das fases anteriores. Sua função é fazer um diagnóstico do erro ocorrido, informando-o ao programador, e ajustar o programa de modo a que este possa continuar seus procedimentos de modo mais completo possível.

É desejável que o programa compilador consiga continuar compilando programas com erros, pelo menos a nível de análise sintática, de modo a permitir que sejam detectados o maior número possível de erros numa única compilação.

Na implementação de um "compilador", estes módulos podem ser executados sequencialmente para o programa fonte completo, gerando uma representação do programa interpretado em código intermediário, que poderá ser lido pelo módulo ou fase subsequente. Este tipo de processamento é chamado de multi-passo.

Outra forma de implementação, é o grupamento destas várias fases num único passo, onde o controle do programa compilador ficará sendo intercalado entre as várias fases. Este tipo de processamento é denominado de compilador de passo único ou de um passo.

A definição da construção de um compilador de um único ou vários passos não é determinada por um critério matemático, mas sim por uma variedade de considerações sobre as linguagens e máquinas a serem usadas.

A estrutura da linguagem do programa fonte a ser interpretado influi, fortemente, na determinação do número de passos de um compilador. Por exemplo, se tivermos uma linguagem fonte que permita declaração de variáveis após seu uso dentro dos comandos da linguagem, será necessário um compilador de no mínimo dois passos, pois não será possível a geração de código imediata de comandos que contenham variáveis, uma vez que as declarações das variáveis podem ser feitas posteriormente a este comando no programa.

O ambiente em que o compilador será usado também pode influir na determinação do número de passos em que será composto. Para o caso de disponibilidade de pouca área de memória, é mais aconselhável um compilador de vários passos, uma vez que a área acupada por um módulo do programa compilador correspondente a uma fase poderá ser reusada para o módulo da fase seguinte. No entanto, um compilador de multi-passos é mais lento do que um compilador de passo único, devido a cada passo ler e gravar

um arquivo contendo a representação do programa fonte em código intermediário. Assim as características da máquina a ser usada podem influir diretamente na decisão da estrutura do compilador a ser construido.

Assim, antes de se definir a estrutura do interpretador Cripta a ser usado, é necessário analisar-se a forma e estrutura do programa fonte a ser interpretado, que será descrita a seguir.

V.1.1 - DEFINIÇÃO DAS ENTRADAS

Este sistema possui três entradas, especificadas separadamente pelo usuário, que constituirão a especificação da aplicação a ser compilada pelo sistema.

V.1.1.1 - DICIONÁRIO DA APLICAÇÃO

A especificação das variáveis a serem usadas pelo sistema Cripta, se divide em duas partes. A primeira se refere à especificação das variáveis pertencentes aos arquivos de entrada, isto é, arquivos que serão processados pelo programa gerado. A outra se refere às variáveis auxiliares, ou de trabalho, usadas para especificação dos procedimentos.

Assim, temos dois tipos de dicionários a serem processados: Dicionário das Variáveis do Arquivo e

Dicionário das Variáveis Auxiliares. A seguir são descritos o conteúdo e a forma de especificação de cada um deles.

A nomenclatura e funções de alguns campos destes formulários são idênticos aos do sistema ATLAS, devido ao sistema proposto incorporar as características mais importantes deste sistema, como já citado anteriormente na proposta do novo sistema.

V.1.1.1.1 - DICIONÁRIO DAS VARIÁVEIS DO ARQUIVO

A especificação das características do arquivo de entrada e a descrição das variáveis que o compõem, é feita através dos formulários A0, R0, V1, V2 e V3 descritos a seguir.

A forma de entrada das especificações do dicionário para o sistema Cripta será feita através do armazenamento das informações de cada formulário em arquivo, onde cada formulário corresponderá a um registro de 80 posições.

FORMULÁRIO A0 - DESCRIÇÃO DO ARQUIVO :

Este formulário é usado para se descrever as características do arquivo de entrada, com o "lay-out" apresentado na figura V.4.

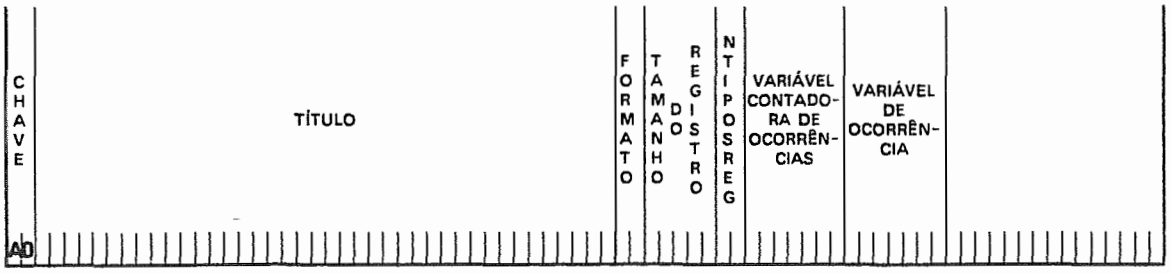


FIGURA V.4 - "Lay-out" do registro de descrição do arquivo

É formado pelos campos:

CHAVE(=A0)(2 posições) - especifica o número do formulário;

TÍTULO (40 posições) - descreve o nome do arquivo;

FORMATO (2 posições) - descreve o formato do arquivo e pode ter os seguintes códigos:

FB - fixo blocado

VB - variável blocado

Esta informação é opcional e se não informada assume como valor "default" o código "FB";

TAMANHO (5 posições) - especifica o número de posições que o registro de entrada possui. O tamanho máximo permitido é de 32767 posições. Esta restrição se deve à capacidade máxima da máquina para tamanho de registros. Para o caso de arquivos com registros de

tamanho variável deve ser especificado o tamanho do maior registro;

Um arquivo de entrada é considerado como uma sequência de um ou mais tipos de registros em formato fixo ou variável. Assim, no dicionário pode ser definido um arquivo com um dos seguintes tipos:

(A) - todos os registros são do mesmo tipo e têm tamanho fixo;
(B) - todos os registros têm tamanho fixo mas tipos diferentes;
(C) - os registros têm tamanho variável de acordo com o tipo de registro;
(D) - todos os registros são do mesmo tipo, mas têm tamanho variável dependendo do número de ocorrências de um conjunto de campos.

A opção de se ter tipos de registros diferentes com cada um de tamanho variável dependendo do número de ocorrências de um conjunto de campos foi desprezado, devido a seu pouco ou raríssimo uso nos tipos de sistemas a que este trabalho está dirigido.

Para o caso (A) basta especificar os campos CHAVE, TÍTULO, FORMATO e TAMANHO do formulário A0. Os demais campos devem ficar em branco.

Para os casos (B) e (C) além dos campos anteriores deve ser especificado o campo N. DE TIPOS DE REGISTROS.

N. DE TIPOS DE REGISTROS (2 posições) - especifica o número de tipos de registros que formam o arquivo. O número máximo de 99 tipos de registros foi considerado suficiente para atender à maioria dos diferentes tipos de questionários das pesquisas.

Para o caso (D) devem ser preenchidos os campos CHAVE, TÍTULO, FORMATO e TAMANHO. O campo N. DE TIPOS DE REGISTROS deve ficar em branco, e deverão ser preenchidos os seguintes campos:

VARIÁVEL CONTADORA DE OCORRÊNCIAS (7 posições) - especifica o número da variável que contém o número de ocorrências do campo repetido para o registro (7 posições). Esta variável deverá estar descrita no formulário V2.

VARIÁVEL DE OCORRÊNCIA (7 posições) - especifica o número da variável que descreve o campo de ocorrência do registro (7 posições). Esta variável também deverá ser especificada no formulário V2.

FORMULÁRIO R0 - DESCRIÇÃO DOS TIPOS DE REGISTROS:

Este formulário é usado para se descrever cada um dos tipos de registros que compõem o arquivo e só deve ser usado quando no formulário A0 (Descrição do Arquivo) o campo N. DE TIPOS DE REGISTROS tiver sido preenchido. Deve

haver tantos formulários deste tipo, quanto o conteúdo deste campo. Este formulário possui o "lay-out" apresentado na figura V.5 formado pelos campos:

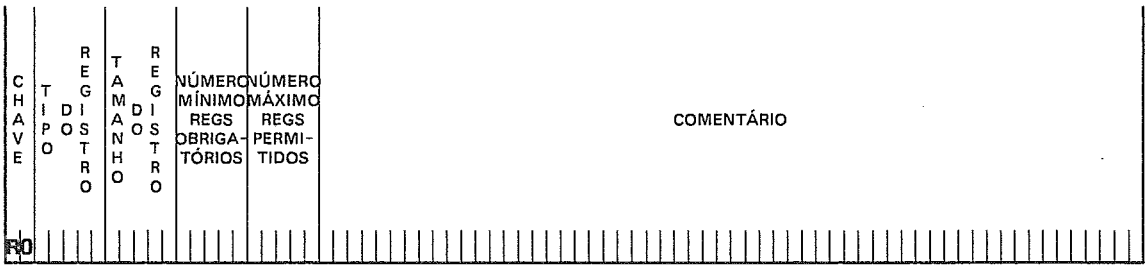


FIGURA V.5 - "Lay-out" do registro de descrição dos tipos de registros do arquivo

CHAVE(=R0) (2 posições)- especifica o número do formulário;

TIPO DE REGISTRO (5 posições) - Este campo especifica o código do tipo de registro. Embora o número de tipos de registros seja no máximo 99, as cinco posições para o valor do código do tipo de registro se deve à formação usada para classificação dos registros nas pesquisas que normalmente são formados por n. do formulário, n. do bloco e linha dentro do bloco.

TAMANHO DO REGISTRO (5 posições) - Embora o tamanho dos registros do arquivo tenha sido especificado no formulário A0, este campo deverá ser preenchido se o formato do arquivo tiver sido especificado como variável, VB, onde para este caso os tipos de

registros têm tamanhos diferentes. Este campo servirá para se fazer uma consistência quando da especificação das variáveis que compõem cada registro.

N. MÍNIMO DE REGISTROS OBRIGATÓRIOS (5 posições) - Este campo informa a quantidade mínima de registros com o código de tipo de registro especificado neste formulário que são obrigatórios dentro de cada questionário. Se o tipo do registro que está sendo especificado é opcional dentro de um questionário, este campo deverá ser preenchido com zeros.

N. MÁXIMO DE REGISTROS PERMITIDOS (5 posições) - Este campo informa a quantidade máxima de registros com o código de tipo especificado neste formulário que são permitidos dentro de cada questionário da pesquisa. As informações contidas neste campo e no anterior servirão para se fazer a consistência de estrutura do questionário.

DESCRIÇÃO DAS VARIÁVEIS :

As informações a respeito de cada uma das variáveis, que compõem o arquivo de entrada, se dividem em dois grupos. O primeiro relativo às características físicas das variáveis e o outro relativo às características de processamento a serem aplicadas a cada variável.

FORMULÁRIO V1 - DESCRIÇÃO DAS CARACTERÍSTICA FÍSICAS DAS VARIÁVEIS:

Este formulário ø usado para se descrever cada uma das variáveis pertencentes ao arquivo em relação às suas características físicas, como posicionamento, formato, nome, etc, seguindo o "lay-out" apresentado na figura V.6.

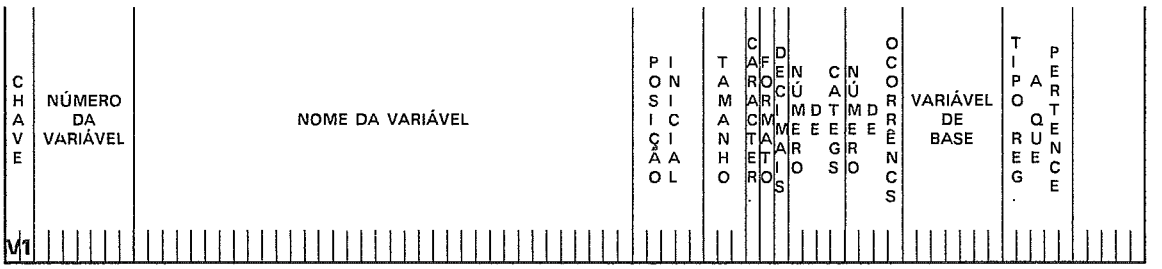


FIGURA V.6 - "Lay-out" do registro de descrição das características físicas da variável

Está composto pelos campos:

CHAVE(=V1) (2 posições) - especifica o número do formulário correspondente às características físicas das variáveis;

NÚMERO DA VARIÁVEL (7 posições) - Número que identifica a variável no arquivo associado da aplicação. Este número deve estar compreendido entre 0000000 e 9999999.

A razão da identificação das variáveis ser feita através de um número, se deve ao fato de num questionário grande, composto por vários formulários, ser muito mais fácil encontrarmos uma variável pela sua identificação numérica, composta por número do formulário, quadro e quesito, do que por seu nome por extenso. A limitação em 7 caracteres é considerada como suficiente para identifição de variáveis, atendendo aos questionários mais complexos, podendo ser formada em até 4 níveis como: número do formulário com 1 posição, número do quadro, número da linha e quesito com 2 posições cada um.

NOME DA VARIÁVEL (35 posições) - Nome da variável. Este campo serve para termos o nome por extenso da variável para documentação, a ser impresso nos relatórios, gerados automaticamente, pela linguagem CRIPTA.

POSIÇÃO INICIAL (5 posições) - Este campo indica a posição inicial da variável dentro do registro do arquivo associado. É permitido que uma variável do registro de entrada redefina as posições do registro de entrada já definidas por outras variáveis.

TAMANHO (3 posições) - Especifica o número de algarismos reservados para a variável.

CARACTERÍSTICA (1 posição) - Este campo é usado para se

indicar que uma variável tem uma característica especial e pode ser representada por um dos códigos listados abaixo. Caso a variável não possua nenhuma destas características especiais citadas, este campo deverá estar em branco.

"T" - Indica que esta variável contém o código do tipo de registro. Só pode ser definida uma variável com esta característica no arquivo;

"P" - Indica que a variável deve ser protegida, isto é, não pode receber nenhuma atribuição nos programas que usem este dicionário. Esta característica serve para se evitar erros de digitação ou lógica nos programas. Um exemplo de variável com esta característica pode ser uma variável da chave do arquivo;

"V" - Indica que a variável define um item de grupo de um conjunto de uma ou mais variáveis que formam uma ocorrência (elemento) de um vetor, cujo número de ocorrências (dimensão) deve ser especificado no campo N. DE OCORRÊNCIAS deste formulário. O campo POSIÇÃO INICIAL deve indicar o início das ocorrências no registro. O formato permitido para variáveis com esta característica é "L". Para o caso da variável que esteja sendo definida, com esta caracterís-

tica, ser a mesma especificada no formulário de Descrição do Arquivo (A0), como variável de ocorrência, o campo NÚMERO DE OCORRÊNCIAS, deste formulário, deve ser preenchido com o número máximo de ocorrências. Isto porque o número correto de ocorrências será especificado pela variável contadora de ocorrências especificada.

"B" - Esta característica indica que a variável está redefinindo um item de grupo definido como ocorrência cujo número da variável é especificado no campo VARIÁVEL BASE deste formulário e que deve ter sido especificada com característica "V". O campo POSIÇÃO INICIAL deve ser preenchido com a posição inicial relativa à variável de base associada.

FORMATO (1 posição) - Este campo indica o formato a ser declarado para a variável e pode conter um dos seguintes códigos:

"N" - variável zonada que ocupa o número de bytes especificado no campo TAMANHO;

"C" - variável compactada: a variável ocupa o número de bytes igual a (TAMANHO/2)+1;

"B" - variável binária: a variável ocupa 2 bytes se TAMANHO for < 5 e 4 bytes se TAMANHO > 4;

"F" - variável de ponto flutuante: a variável ocupa 4

bytes se TAMANHO < 7 e 8 bytes se TAMANHO > 6;

"L" - variável alfanumérica que ocupa o número de bytes especificado no campo TAMANHO.

Se este campo estiver em branco é assumido como default o código "N".

DECIMAIS (1 posição) - Este campo só deve ser preenchido quando a variável for especificada com formato numérico de tamanho maior que 1, e indica o número de casas decimais para a variável.

NÚMERO DE CATEGORIAS (4 posições) - Este campo só deve ser preenchido se a variável que está sendo definida é categorizada e especifica o número de categorias que esta variável possui. Quando este campo estiver preenchido, após este formulário devem ser preenchidos tantos formulários do tipo V3, DESCRIÇÃO DE CATEGORIAS, quanto o conteúdo deste campo.

NÚMERO DE OCORRÊNCIAS (4 posições) - Este campo só pode ser preenchido quando o campo CARACTERÍSTICA tiver sido preenchido com "V" (definição de uma ocorrência de um vetor), e especifica a dimensão deste vetor.

VARIÁVEL DE BASE (7 posições) - Este campo só pode ser preenchido quando o campo CARACTERÍSTICA tiver sido especificado com "B", item da ocorrência de um vetor,

e especifica o número da variável que está sendo redefinida que deve ter sido preenchida com CARACTERISTICA "V".

TIPO DO REGISTRO A QUE PERTENCE (5 posições) - O valor deste campo especifica o código do tipo de registro a que a variável pertence. Quando a variável pertence a todos os tipos de registros este campo deverá ser preenchido com a palavra "CHAVE". Quando o arquivo à qual pertence a variável só tenha um tipo de registro, este campo deverá ser deixado em branco ou opcionalmente ser preenchido com a palavra "CHAVE". Para o caso do arquivo ter mais de um tipo de registro, este campo deverá ser preenchido com um dos valores identificadores de tipo de registro especificados.

FORMULÁRIO V2 - DESCRIÇÃO DAS CARACTERÍSTICA DE PROCESSAMENTO DAS VARIÁVEIS:

Este formulário é usado para se descrever as características de processamento a serem adotadas para cada uma das variáveis do arquivo de entrada, como crítica de intervalo de valores permitidos, valor da representação a ser assumida como valor ignorado, etc, seguindo o "lay-out" apresentado na figura V.7.

FIGURA V.7 - "Lay-out" do registro de descrição das características de processamento da variável

É formado pelos campos:

CHAVE(=V2) (2 posições) - especifica o número do formulário correspondente às características de processamento das variáveis;

LIMITES INFERIOR E SUPERIOR (15 posições cada) - Estes dois campos só devem ser preenchidos para variáveis com formato numérico e especificam os valores mínimos e máximos permitidos para preenchimento da variável. Estas informações serão usadas para se fazer a consistência automática das variáveis.

POSSUI IGNORADO (1 posição) - Este campo informa se a variável possui um código de representação especial para representação de valor ignorado ou não. Deve ser preenchida com "S" ou "N", sendo assumido o valor "N" como "default" caso este campo esteja em branco.

IGNORADO (15 posições) - Este campo informa o valor usado para a representação de informação ignorada para a variável.

No tratamento dos dados coletados, tem de ser feito um tratamento para as variáveis que não foram informadas. Por exemplo, para cálculo do total de uma coluna de um questionário, não se pode simplesmente fazer o somatório das parcelas diretamente, devido à possibilidade de algumas parcelas não terem sido preenchidas. Estes problemas ocorrem não só nas totalizações dos questionários, como também na comparação de variáveis, devido a problemas de representação interna dos dados, em seus respectivos formatos, e à impossibilidade de operações de dados numéricos com alfanuméricos.

Normalmente estes problemas são resolvidos, em tempo de programação, pelo programador que redefine as variáveis necessárias e executa as operações condicionalmente.

Cabe lembrar, que a conversão de um dado de uma variável numérica, que esteja com valor não preenchido, para zero, não resolve o problema devido a, em termos estatísticos, um valor zero e branco terem significados diferentes. Por exemplo, um campo de valor da produção, de um questionário qualquer, quando preenchido com zeros, significa que o informante não produziu nada, ao passo que, este mesmo campo, quando em branco, significa que o valor não foi

informado, ou seja, ele é ignorado.

Assim, para estes tipos de variáveis, é escolhido um valor para representação de informação ignorada, que não necessariamente precisa ser o valor branco. A inclusão deste campo no dicionário viabiliza sua crítica automática, além da geração automática dos testes condicionais para operações que o incluam. O valor especificado como ignorado para a variável passa a ser considerado como elemento neutro na adição e subtração e como elemento absorvente na multiplicação e divisão.

O preenchimento deste campo só é permitido caso o campo POSSUI IGNORADO tenha sido preenchido com 'N'.

VALOR A SER ASSUMIDO (15 posições) - Este campo só deve ser preenchido para variáveis com formato numérico ou categorizada, e especifica um valor a ser atribuido quando da inconsistência do valor desta variável com a declaração especificada. Quando a variável tiver sido declarada como categorizada, o valor deste campo deverá ser uma das categorias especificadas para esta variável.

FORMULÁRIO V3 - DESCRIÇÃO DAS CATEGORIAS:

Este formulário é utilizado para se descrever os códigos das categorias que compõem uma variável categorizada. O "lay-out" deste formulário é representado

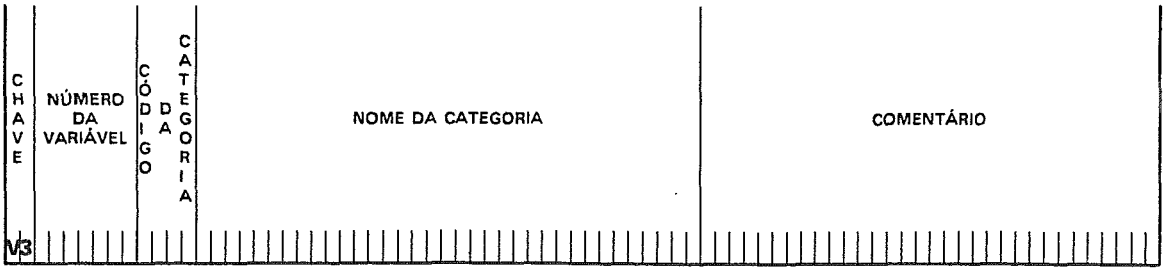


FIGURA V.8 - "Lay-out" do registro de descrição das categorias da variável

na figura V.8, sendo formado pelos campos:

CHAVE(=V3)(2 posições) - especifica o número do formulário;

NÚMERO DA VARIÁVEL (7 posições) - Número da variável associada à qual pertence a categoria.

CÓDIGO DA CATEGORIA (4 posições)- Neste campo deve ser especificado o valor do código da categoria.

NOME DA CATEGORIA (35 posições) - Este campo especifica o nome da categoria que será impresso nos relatórios gerados automaticamente pela linguagem CRIPTA.

V.1.1.1.2 - DICIONÁRIO DAS VARIÁVEIS AUXILIARES

A especificação das variáveis auxiliares, de trabalho,

a serem usadas num programa CRIPTA, é feita de modo análogo à especificação das variáveis do arquivo de entrada.

Este dicionário usa apenas dois formulários, um para descrição das variáveis e outro para a descrição das categorias das variáveis.

O formulário para descrição das variáveis auxiliares não pode ser o mesmo usado para a descrição das variáveis do arquivo de entrada devido às variáveis auxiliares possuirem características e formas de tratamento diferentes. No entanto, para o caso de uma variável auxiliar ser categorizada, pode ser usado o mesmo formulário de descrição das categorias das variáveis do arquivo de entrada, uma vez que o tratamento será o mesmo.

FORMULÁRIO W0 - DESCRIÇÃO DAS VARIÁVEIS AUXILIARES:

Este formulário é usado para se descrever as características das variáveis auxiliares possuindo o "lay-out" apresentado na figura V.9.

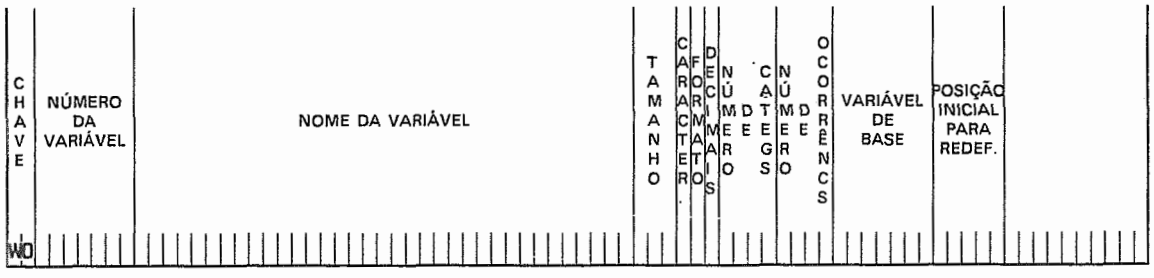


FIGURA V.9 - "Lay-out" do registro de descrição das variáveis auxiliares

É formado pelos campos:

CHAVE(=W0)(2 posições) - especifica o número do formulário;

NÚMERO DA VARIÁVEL (7 posições) - Número que identifica a variável auxiliar a ser usada na aplicação e que deve estar compreendido entre 0000000 e 9999999. A razão para a identificação das variáveis auxiliares ser feita através de um número se deve a facilitar uma melhor associação com as variáveis do arquivo de entrada. Por exemplo, podemos ter a variável, do arquivo de entrada, V0125, e a W0125, uma variável auxiliar que guarda o valor da variável, do arquivo de entrada V0125.

NOME DA VARIÁVEL (35 posições) - Nome da variável a ser impresso nos relatórios gerados automaticamente pela linguagem CRIPTA.

CARACTERÍSTICA (1 posição) - Este campo é usado para indicar que a variável possui uma característica especial representada por um dos códigos abaixo:

"V" - Como para as variáveis do arquivo de entrada esta característica indica que a variável define um item de grupo de um conjunto de uma ou mais variáveis que formam uma ocorrência(elemento) de um vetor, cujo número de ocorrências(dimensão)

deve ser especificado no campo NÚMERO DE OCORRÊNCIAS deste formulário;

"B" - Esta característica indica que a variável está redefinindo um item de grupo definido como ocorrência cujo número da variável é especificado no campo VARIÁVEL BASE deste formulário que deve ter sido especificada com característica "V". O campo POSIÇÃO INICIAL PARA REDEFINIÇÃO deve ser preenchido com a posição inicial relativa da variável de base associada.

"P" - Indica que a variável deverá ser definida no parâmetro passado para o programa gerado quando for executado. O campo POSIÇÃO INICIAL PARA REDEFINIÇÃO indica a posição inicial desta variável no parâmetro. O formato correspondente só poderá ser "N" ou "L".

"X" - Indica que a variável tem escopo externo ao programa.

"R" - Esta característica indica que a variável está redefinindo uma outra variável auxiliar já especificada, cuja identificação deve ser preenchida no campo VARIÁVEL DE BASE, e a posição inicial da variável de base a partir de onde será feita a redefinição no campo POSIÇÃO

INICIAL PARA REDEFINIÇÃO.

FORMATO (1 posição) - Este campo indica o formato a ser declarado para a variável e pode conter um dos seguintes códigos:

"N" - variável zonada que ocupa o número de bytes especificado no campo TAMANHO;

"C" - variável compactada: a variável ocupa o número de bytes igual a (TAMANHO/2)+1;

"B" - variável binária: a variável ocupa 2 bytes se TAMANHO for < 5 e 4 bytes se TAMANHO > 4;

"F" - variável de ponto flutuante: a variável ocupa 4 bytes se TAMANHO < 7 e 8 bytes se TAMANHO > 6;

"L" - variável alfanumérica que ocupa o número de bytes especificado no campo TAMANHO.

Se este campo estiver em branco é assumido como "default" o código "N".

NÚMERO DE CATEGORIAS (4 posições) - Este campo só deve ser preenchido se a variável que está sendo definida é categorizada e especifica o número de categorias que esta variável possui. Quando este campo estiver preenchido, após este formulário devem ser preenchidos tantos formulários V3, DESCRIÇÃO DE CATEGORIAS, quanto o conteúdo deste campo.

NÚMERO DE OCORRÊNCIAS (4 posições) - Este campo só pode ser preenchido quando o campo CARACTERÍSTICA tiver sido preenchido com "V" (definição de uma ocorrência de um vetor), e especifica a dimensão deste vetor.

VARIÁVEL DE BASE (7 posições) - Este campo deve ser preenchido quando a característica da variável tiver sido especificada com "B" indicando a variável de item de grupo que representa a ocorrência do vetor, ou "R", indicando a variável auxiliar a ser redefinida.

POSIÇÃO INICIAL PARA REDEFINIÇÃO (5 posições) - Este campo deve ser preenchido quando a característica da variável for "B", indicando a posição inicial relativa dentro da ocorrência, "P", indicando a posição inicial dentro do Parm, ou "R", indicando, a posição inicial da redefinição da variável.

V.1.1.1.3 - COMPOSIÇÃO DO DICIONÁRIO DA APLICAÇÃO

Um dos requisitos, especificados para o sistema, é a utilização, por uma aplicação, do processamento de mais de um arquivo de entrada.

Assim, o usuário pode definir para uso numa aplicação, a especificação de até 10 arquivos, um tipo mestre e os outros para processamento paralelo. Neste caso o Dicionário da Aplicação será formado por 10 especificações de arquivos de entrada e, opcionalmente, a especificação de

um dicionário auxiliar.

A especificação dos dicionários dos arquivos de entrada é feita de modo análogo para todos eles, sendo feita a distinção de qual o arquivo mestre e os de processamento paralelo dentro da especificação da aplicação CRIPTA a ser descrita adiante.

Esta facilidade, permite que a mesma especificação de um dicionário de um arquivo de entrada possa ser aproveitada para diferentes aplicações, onde ele seja o arquivo principal de processamento, arquivo mestre, ou de processamento paralelo.

Os "lay-outs" dos formulários para especificação dos dicionários dos arquivos de entrada e auxiliar, encontram-se no Anexo A.

V.1.1.2 - PLANO DE CRÍTICA AUTOMATIZADO

Analisando-se as especificações dos Planos de Crítica convencionais, verifica-se que para cada crítica especificada, esta é dividida em três campos distintos, conforme representado na figura V.10.

O primeiro deles consiste da identificação da crítica, correspondendo na maioria das vezes ao código do erro a ser indicado. Opcionalmente, dependendo da crítica e do detalhamento da especificação do usuário, este campo possui associadamente a mensagem de erro correspondente à crítica e a lista de variáveis associadas à ocorrência do erro.

CRÍTICA	CAUSA	EFEITO

FIGURA V.10 - Exemplo de plano de crítica convencional

A segunda parte contém as especificações para a detecção do erro da crítica, podendo incluir procedimentos a serem efetuados antes do teste de condição da crítica.

A terceira e última parte contém a especificação dos procedimentos a serem efetuados em resposta ao teste de condição da crítica especificada no campo anterior.

A figura V.11 apresenta o "lay-out" do formulário de um Plano de Crítica, a ser usado no sistema Cripta, cuja descrição de cada campo é apresentada a seguir.

REFERÊNCIA: Consiste de uma identificação composta de uma cadeia de até 6 caracteres, que identifica um conjunto de procedimentos e/ou uma crítica. Este código

REFERÊNCIA	DESCR	CAUSA	DESCR	EFEITO

FIGURA V.11 - Formulário do plano de crítica para o sistema Cripta

corresponderá ao código de erro a ser impresso pelo programa de crítica correspondente, como execução de um procedimento especificado no campo EFEITO correspondente à condição especificada em CAUSA.

Este campo servirá também como um índice do plano de crítica a ser usado para inclusão de trechos deste no programa de crítica correspondente.

CAUSA: Este campo do formulário é usado para se especificar os procedimentos e condições a serem executadas para detecção de um erro. Os procedimentos e condições são especificados por um subconjunto de instruções da linguagem "CRIPTA", a ser apresentada

posteriormente, formado pelos comandos de atribuição e especificação de condições.

No entanto, podem ocorrer situações em que não seja possível a especificação por estes comandos permitidos. Para estes casos o subcampo DESCR (descrição) do campo CAUSA, deve ser marcado com um '*' e os procedimentos e condições desejadas devem ser especificados em linguagem natural, terminando com ';'

EFEITO: Este campo é usado para se especificar os procedimentos a serem executados em resposta a uma condição especificada no campo CAUSA. Como resultado de uma expressão condicional, temos duas alternativas: Verdadeiro ou falso. Assim, para cada condição especificada em CAUSA podemos ter dois efeitos correspondentes.

Se para uma condição especificada em CAUSA não for especificado nenhum procedimento no campo EFEITO, é assumido como default para atendimento da condição, o procedimento de indicação de erro no relatório de críticas pelo programa de crítica correspondente.

As instruções a serem usadas no campo EFEITO são os comandos de atribuição da linguagem Cripta, o comando ERRO que especifica que deverá ser indicado no relatório de críticas, gerado pelo programa de crítica correspondente, o erro cujo código corresponde ao campo REFERÊNCIA associado, e o comando DESPREZE que especifica que o registro que está sendo processado

deve ser desprezado, isto é, as restantes críticas para o registro não devem ser processadas e o controle do processamento deve passar para o próximo registro.

De modo idêntico ao campo CAUSA, podem existir situações em que não seja possível se especificar os procedimentos desejados através dos comandos permitidos. Para estes casos, como no campo CAUSA, deve ser preenchido o campo DESCR, descrição, do campo EFEITO, com um "*" e os procedimentos desejados devem ser especificados em linguagem natural terminando com ";".

A forma de entrada da especificação do Plano de Crítica para o sistema Cripta, deverá ser feito através do armazenamento das informações num arquivo com registros de 80 posições cada, seguindo o "lay-out" da figura V.12.

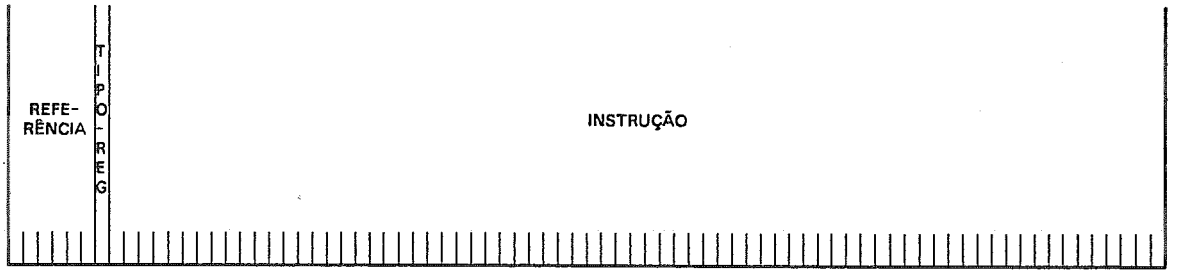


FIGURA V.12 - "Lay-out" do registro de especificação do plano de crítica

Cada crítica é identificada pelo código de referência que deve ser repetido para os demais registros que compõem a crítica.

Para cada crítica só pode existir um registro de tipo 'M' opcional; mas, se presente, deverá ser o primeiro, que conterá a mensagem de erro a ser indicada.

O registro tipo 'L' conterá a especificação da lista de variáveis associadas a serem listadas, quando da indicação do erro. Caso seja necessário, podem ser usados mais de um registro deste tipo, que devem ser consecutivos e obrigatoriamente após um registro tipo 'M'.

O registro tipo 'C' conterá as especificações referentes ao campo CAUSA/PROCEDIMENTO, podendo ser usados tantos registros deste tipo, quantos necessários.

O subcampo DESCRIÇÃO do campo CAUSA/PROCEDIMENTO é especificado na la. coluna do campo CONTEÚDO deste registro.

Os registros tipo 'V' conterão as especificações referentes ao campo EFEITO em resposta ao atendimento à condição especificada no campo CAUSA do Plano de Crítica.

O subcampo DESCRIÇÃO do campo EFEITO é especificado na la. coluna do campo CONTEÚDO deste registro.

Este tipo de registro é opcional numa crítica; mas, se existente, deverá ser consecutivo aos registros de tipo 'C'.

Os registros tipo 'F' conterão as especificações referentes ao campo EFEITO ao não atendimento à condição especificada no campo do Plano de Crítica.

Este tipo de registro é opcional numa crítica; mas, se existente, deverá ser consecutivo aos registros de tipo 'C' e 'V', se este último existir.

Os registros com tipo '*' conterão um comentário. Este tipo de registro é opcional e poderão existir tantos quantos necessários numa crítica, e em qualquer lugar.

V.1.1.3 - ESPECIFICAÇÃO DA APLICAÇÃO

A especificação da aplicação desejada pelo usuário será feita através do uso da linguagem Cripta, criada para este sistema.

Esta linguagem foi especificada com base nos requisitos especificados para esta fase, no capítulo IV, referente à especificação do sistema proposto.

Para especificação da sintaxe desta nova linguagem é usada a notação de especificação de gramática livre de contexto, apresentada em AHO e ULLMAN [14] também chamada de BNF, Backus-Naur-Form, que usa os conceitos de regras de reescrita.

A vantagem da utilização desta notação é que além de ser uma ferramenta para a especificação precisa e de fácil entendimento da sintaxe da linguagem, oferece a facilidade das estruturas do módulo analisador léxico e sintático poderem ser construídos diretamente a partir desta especificação.

As palavras em maiúsculo delimitadas pelos símbolos <

e > representam os símbolos não terminais da gramática.

As palavras escritas em maiúsculas e os símbolos sem estarem entre os delimitadores citados representam os símbolos terminais da gramática a serem identificados pelo módulo analisador léxico.

Informações referentes a cada um dos comandos da linguagem podem ser obtidos no Manual da Linguagem apresentado no Anexo B.

. GRAMÁTICA DA LINGUAGEM CRIPTA:

```
<CRIPTA> --> <PROGRAMA>

<PROGRAMA> --> PROGRAMA <STRING> ;
               <DICIONARIO>
               <ESTPROGR1>
               FIM PROGRAMA ;

<DICIONARIO> --> DICIONARIO = <LISTADICIONARIOS> ;

<LISTADICIONARIOS> --> <LISTADICIONARIOS> <NOMEDICIONARIO>
                   --> <NOMEDICIONARIO>

<NOMEDICIONARIO> --> BASICO
                 --> AUXILIAR
                 --> REF1
                 --> REF2
                 --> REF3
```

--> REF4

--> REF5

--> REF6

--> REF7

--> REF8

--> REF9

<ESTPROG1> --> <PROCERRO> <ESTPROG2>

--> <ESTPROG2>

<ESTPROG2> --> <PROCINICIALIZACAO> <ESTPROG3>

--> <ESTPROGS3>

<ESTPROG3> --> <PROCENTRADA> <ESTPROG4>

--> <ESTPROG4>

<ESTPROG4> --> <PROCQUEBRA> <ESTPROG5>

--> <ESTPROG5>

<ESTPROG5> --> <PROCCRITICA> <ESTPROG6>

--> <ESTPROG6>

<ESTPROG6> --> <LISTAROTINAS> <ESTPROG7>

--> <ESTPROG7>

<ESTPROG7> --> <LISTAPROCEDIMENTOS> <PROCEDIMENTOSFINAIS>

--> <LISTAPROCEDIMENTOS>

```
<PROCERRO> --> PROC ERRO ;
                <CORPOERRO>
                FIM ERRO ;

<CORPOERRO> --> LISTA = <LISTAVARIAVEIS> ;
            --> <PROCPLI>

<PROCINICIALIZACAO> --> PROC INICIALIZACAO ;
                        <LISTACOMDSINICIALIZACAO>
                        FIM INICIALIZACAO ;

<LISTACOMDSINICIALIZACAO> --> <LISTACOMDSINICIALIZACAO>
                                <COMDINICILIZACAO>
                          --> <COMDINICIALIZACAO>

<COMDINICIALIZACAO> --> <COMDATRIB>
                    --> <PROCPLI>

<PROCENTRADA> --> PROC ENTRADA ;
                   <CORPOENTRADA>
                  FIM ENTRADA ;
              --> PROC ENTRADA ( PLI ) ;
                   <CORPOENTRADAPLI>
                  FIM ENTRADA ;

<CORPOENTRADA> --> <CORPOENTRADA> <COMDCORPOENTRADA>
               --> <COMDCORPOENTRADA>
```

<CORPOENTRADAPLI> --> <CORPOPLI>

<COMDCORPOENTRADA> --> <COMDATRIB>
--> <COMDCONDICIONAL>
--> <COMDREJEITE>
--> <COMDEXECUTE>
--> <COMDLEIA>
--> <PROCPLI>
--> <PROCREFERENCIA>

<PROCQUEBRA> --> PROC QUEBRA ;
<CORPOQUEBRA>
FIM QUEBRA ;

<CORPOQUEBRA> --> <INICIOQUEBRA> <LISTACOMDSQUEBRA>
--> <INICIOQUEBRA>

<INICIOQUEBRA> --> CHAVE = <LISTAVARIAVEIS> ;
TAMANHO = <CONSTNUMERICA> ;
--> CHAVE = <LISTAVARIAVEIS> ;

<LISTACOMDSQUEBRA> --> <LISTACOMDSQUEBRA> <COMANDOQUEBRA>
--> <COMANDOQUEBRA>

<COMANDOQUEBRA> --> <COMANDOTOTALIZE>
--> <COMANDOFREQUENCIA>
--> <COMANDOMAIORMENORVALOR>

<COMANDOTOTALIZE> --> <INICCOMDTOTAL> <COMPCOMDTOTAL> ;

<INICCOMDTOTAL> --> CALCULE TOTAL (<LISTATOTAIS>)

<COMPCOMDTOTAL> --> <CONDICAOPARA> <COMPREF>
--> <COMPFRE>
--> <CONDICAOPARA>

<LISTATOTAIS> --> <LISTATOTAIS> <TOTAL>
--> <LISTATOTAIS> , <TOTAL>
--> <TOTAL>

<TOTAL> --> "VARIAVEL" EM "VARIAVEL"

<CONDICAOPARA> --> PARA <CONDICAO>

<COMDFREQ> --> COM FREQUENCIA EM "VARIAVEL"

<COMANDOFREQUENCIA> --> CALCULE FREQUENCIA EM "VARIAVEL"
<CONDICAOPARA> ;

<COMANDOMAIORMENORVALOR> --> CALCULE <MAIORMENORVALOR>
(<LISTATOTAIS>)
<CONDICAOPARA> ;
--> CALCULE <MAIORMENORVALOR>
(<LISTATOTAIS>) ;

<MAIORMENORVALOR> --> MAIORVALOR

--> MENORVALOR

<LISTAVARIAVEIS> --> <LISTAVARIAVEIS> "VARIAVEL"

--> <LISTAVARIAVEIS> , "VARIAVEL"

--> "VARIAVEL"

<PROCCRITICA> --> PROC CRITICA ;

<CORPOCRITICA>

FIM CRITICA ;

<CORPOCRITICA> --> <INICIOCRITICA>

<LISTACOMDSOPCCRITICA>

<LISTACOMDSERRO>

--> <INICIOCRITICA>

<LISTACOMDSERRO>

<INICIOCRITICA> --> IDENTIFICACAO = <LISTAVARIAVEIS> ;

<LISTACOMDSOPCCRITICA> --> <LISTACOMDSOPCCRITICA>

<COMANDOOPCIONALCRITICA>

--> <COMANDOOPCIONALCRITICA>

<COMANDOOPCIONALCRITICA> --> TITULO = <TITULO> ;

--> FREQUENCIA = <SIM/NAO> ;

--> DDNAME = <STRING> ;

--> SAIDA = "IMPRIMIR/GRAVAR" ;

<TITULO> --> <TITULO> + <TERMOTITULO>

--> <TERMOTITULO>

<TERMOTITULO> --> <STRING>

--> <VARIAVEL>

<LISTACOMDSERRO> --> <LISTACOMDSERRO> <COMANDOERRO>

--> <COMANDOERRO>

<COMANDOERRO> --> ERRO = <STRING> : <MENSAGEM>

(<LISTAVARERRO>);

<LISTAVARERRO> --> <LISTAVARERRO> , <VARERRO>

--> <LISTAVARERRO> <VARERRO>

--> <VARERRO>

<VARERRO> --> <VARIAVEL>

<MENSAGEM> --> <MENSAGEM> + <TERMOMENSAGEM>

--> <TERMOMENSAGEM>

<TERMOMENSAGEM> --> <STRING>

--> <VARIAVEL>

<LISTAROTINAS> --> <LISTAROTINAS> <ROTINA>

--> <ROTINA>

<ROTINA> --> <ROTINAEXTERNA>

--> <ROTINAPESQUISA>

```
          --> <ROTINAINTERNA>

          --> <ROTINAPLANCRIT>

          --> <ROTINAIMPUTACAO>


<LISTAPROCEDIMENTOS> --> <LISTAPROCEDIMENTOS>
                              <PROCEDIMENTO>
                         --> <PROCEDIMENTO>


<PROCEDIMENTO> --> <PROCGRUPAMENTO>
                --> <PROCTRANSFORMACAO>
                --> <PROCFREQUENCIA>
                --> <PROCIMPRESSAO>
                --> <PROCGRAVACAO>
                --> <PROCTRASLADO>
                --> <PROCCRUZAMENTO>
                --> <PROCQUANTIFICADO>
                --> <PROCTABELA>
                --> <PROCREFERENCIA>
                --> <PROCPESQUISA>
                --> <PROCMARCA>
                --> <PROCPLI>
                --> <PROC%NOME>


<ROTINAEXTERNA> --> ROTEXT <NOME> ;
                 --> ROTEXT <NOME> ( <LISTAPARMROTEXT> ) ;


<LISTAPARMROTEXT> --> <LISTAPARMROTEXT> <PARMROTEXT>
                   --> <PARMROTEXT>
```

<PARMROTEXT> --> <VARIAVEL>

--> <CONSTNUMERICA>

--> <STRING>

<ROTINAPESQUISA> --> ROTPESQ <NOME> (<PARMROTPESQ>) ;

<PARMROTPESQ> --> "VARIAVEL" , "VARIAVEL"

--> "VARIAVEL" "VARIAVEL"

<PROCGRUPAMENTO> --> PROC GRUPAMENTO

<CORPOGRUPAMENTO>

FIM GRUPAMENTO ;

<CORPOGRUPAMENTO> --> (<LISTAVARIAVEIS>) ;

<LISTACONSTNUMERICAS>

<PROCTRANSFORMACAO> --> PROC TRANSFORMACAO

<CORPOTRANSFORMACAO>

FIMTRANSFORMACAO ;

<CORPOTRANSFORMACAO> --> (<LISTAVARIAVEIS>) ;

<LISTACONSTNUMERICAS> ;

--> (<LISTAVARIAVEIS>) ;

<LISTACONSTNUMERICAS> ;

<COMANDOINVALIDO>

<COMANDOINVALIDO> --> INVALIDO = <CONSTNUMERICA> ;

```
<PROCFREQUENCIA> --> PROC FREQUENCIA ;
                     <CORPOFREQUENCIA>
                     FIM FREQUENCIA ;

<CORPOFREQUENCIA> --> <LISTAINSTRUCOESFREQUENCIA>
                       <INSTRFREQUENCIA>
                    --> <INSTRFREQUENCIA>

<INSTRFREQUENCIA> --> SAIDA = <IMPRIMIR/GRAVAR> ;
                  --> RECORRIDA = <LISTAVARIAVEIS> ;
                  --> PESO = "VARIAVEL" ;
                  --> MEDIA;
                  --> VARIAVEIS = <LISTAVARFREQ> ;
                  --> DDNAME = <STRING> ;

<LISTAVARFREQ> --> <LISTAVARFREQ> <VARFREQ>
               --> <LISTAVARFREQ> , <VARFREQ>
               --> <VARFREQ>

<VARFREQ> --> "VARIAVEL" A "VARIAVEL"
          --> "VARIAVEL"

<PROCIMPRESSAO> --> PROC IMPRESSAO
                    <CORPOIMPRESSAO>
                    FIM IMPRESSAO ;

<CORPOIMPRESSAO> --> <LISTAINSTRUCOESIMPRESSAO>
```

```
<LISTAINSTRUCOESIMPRESSAO> --> <LISTAINSTRUCOESIMPRESSAO>
                                   <INSTRUCAOIMPRESSAO>
                              --> <INSTRUCAOIMPRESSAO>

<INSTRUCAOIMPRESSAO> --> TITULO = <STRING> ;
                     --> RECORRIDA = <LISTAVARIAVEIS> ;
                     --> NOME = <SIM/NAO> ;
                     --> ERRO = "VARIAVEL" ;
                     --> DDNAME = <STRING> ;
                     --> VARIAVEIS = <LISTAVARIMP> ;

<LISTAVARIMP> --> <LISTAVARIMP> <VARIMP>
              --> <LISTAVARIMP> , <VARIMP>
              --> <VARIMP>

<VARIMP> --> "VARIAVEL" TOTAL
         --> "VARIAVEL" , TOTAL

<PROCGRAVACAO> --> PROC GRAVACAO ;
                   <CORPOGRAVACAO>
                   FIM GRAVACAO ;

<CORPOGRAVACAO> --> <INSTRUCAOGRAVACAO>

<INSTRUCAOGRAVACAO> --> DDNAME = <STRING> ;

<PROCTRASLADO> --> PROC TRASLADO ;
                   <CORPOTRASLADO>
```

FIM TRASLADO ;

<CORPOTRASLADO> --> <LISTAINSTRUCOESTRASLADO>

<LISTAINSTRUCOESTRASLADO> --> <LISTAINSTRUCOESTRASLADO>
<INSTRUCAOTRASLADO>
--> <INSTRUCAOTRASLADO>

<INSTRUCAOTRASLADO> --> FORMATO = <FB/VB> ;
--> TAMANHO = <CONSTNUMERICA> ;
--> DDNAME = <STRING> ;
--> VARIAVEIS = <LISTAVARIAVEIS> ;

<PROCCRUZAMENTO> --> PROC CRUZAMENTO úCONSNUMERICA>;
<CORPOCRUZAMENTO>
FIM CRUZAMENTO ;

<CORPOCRUZAMENTO> --> <LISTAINSTRUCOESCRUZAMENTO>

<LISTAINSTRUCOESCRUZAMENTO> --> <LISTAINSTRUCOESCRUZAMENTO>
<INSTRUCAOCRUZAMENTO>
--> <INSTRUCAOCRUZAMENTO>

<INSTRUCAOCRUZAMENTO> --> LINHA = "VARIAVEL" ;
--> COLUNA = "VARIAVEL" ;
--> POR = "VARIAVEL" ;
--> TITULO = <STRING> ;
--> RECORRIDA = <LISTAVARIAVEIS> ;

```
                        --> PESO = <LISTAVARIAVEIS> ;
                        --> INVALIDO = <SIM/NAO> ;
                        --> DDNAME = <STRING> ;
                        --> SAIDA = <IMPRIMIR/GRAVAR> ;
<PROCQUANTIFICADO> --> PROC QUANTIFICADO ;
                        <CORPOQUANTIFICADO>
                        FIM QUANTIFICADO ;

<CORPOQUANTIFICADO> --> <LISTAINSTRUCOESQUANTIFICADO>

<LISTAINSTRUCOESQUANTIFICADO> --> <INSTRUCAOQUANTIFICADO>

<INSTRUCAOQUANTIFICADO> --> LINHA = "VARIAVEL" ;
                        --> COLUNA = "VARIAVEL" ;
                        --> SOMA = "VARIAVEL" ;
                        --> TITULO = <STRING> ;
                        --> POR = "VARIAVEL" ;
                        --> RECORRIDA = <LISTAVARIAVEIS> ;
                        --> INVALIDO = <SIM/NAO> ;
                        --> DDNAME = <STRING> ;
                        --> SAIDA = <IMPRIMIR/GRAVAR> ;

<PROCTABELA> --> PROC TABELA ;
                    <CORPOTABELA>
                    FIM TABELA ;

<CORPOTABELA> --> <LISTAINSTRUCOESTABELA>
```

```
<LISTAINSTRUCOESTABELA> --> <LISTAINSTRUCOESTABELA>
                                <INSTRUCAOTABELA>
                            --> <INSTRUCAOTABELA>

<INSTRUCAOTABELA> --> LINHA = "VARIAVEL" ;
                  --> COLUNAS = <LISTAVARIAVEIS> ;
                  --> TITULO = <STRING> ;
                  --> FREQUENCIA = <SIM/NAO> ;
                  --> POR = "VARIAVEL" ;
                  --> RECORRIDA = <LISTAVARIAVEIS> ;
                  --> INVALIDO = <SIM/NAO> ;
                  --> DDNAME = <STRING> ;
                  --> SAIDA = <IMPRIMIR/GRAVAR> ;

<PROCREFERENCIA> --> PROC REFERENCIA <CONSTNUMERICA> ;
                     <CORPOREFERENCIA>
                     FIM REFERENCIA ;

<CORPOREFERENCIA> --> <LISTAINSTRUCOESREF>

<LISTAINSTRUCOESREF> --> <LISTAINSTRUCOESREF>
                             <INSTRUCAOREF>
                         --> <INSTRUCAOREF>

<INSTRUCAOREF> --> "VARIAVEL" = "VARIAVEL" ;

<COMDATRIB> --> "VARIAVEL" = <EXPRESSAO>
```

```
<EXPRESSAO> --> ( <EXPRESSAO> )
            --> <EXPRESSAO> <OPARITMETICO> <EXPRESSAO>
            --> <VARIAVEL>
            --> <CONSTNUMERICA>
            --> <STRING>
            --> DIGITO10 ( <LISTAVARIAVEIS> )
            --> DIGITO11 ( <LISTAVARIAVEIS> )
            --> SOMAVAR ( <LISTAOPSOMA> )
            --> SOMAVAR ( <VARIAVELSOMA> )
            --> TOTAL ( "VARIAVEL" PARA <CONDICAO> )
            --> TOTAL ( "VARIAVEL" )
            --> MAIORVALOR ( "VARIAVEL" PARA <CONDICAO> )
            --> MAIORVALOR ( "VARIAVEL" )
            --> MENORVALOR ( "VARIAVEL" PARA <CONDICAO> )
            --> MENORVALOR ( "VARIAVEL" )
            --> FREQUENCIA ( <CONDICAO> )

<LISTAOPSOMA> --> <LISTAOPSOMA> <OPSOMA>
              --> <LISTAOPSOMA> , <OPSOMA>
              --> <OPSOMA>

<OPSOMA> --> "VARIAVEL" A "VARIAVEL"
         --> "VARIAVEL"

<COMCONDICIONAL> --> SE <CONDICAO>
                     ENTAO
                     <LISTAINSTRUCOESSE>
                     FIM SE ;
```

```
                --> SE <CONDICAO>
                    ENTAO
                    <LISTAINSTRUCOESSE>
                    SENAOE ;
                    <LISTAINSTRUCOESSE>
                    FIM SE ;

<CONDICAO> --> ( <CONDICAO> )
           --> <OPNEGACAO> <CONDICAO>
           --> <EXPRESSAO> <OPRELACIONAL> <EXPRESSAO>
           --> <EXPRESSAO> <OPRELACIONAL> <FUNCAOLOGICA>
           --> <FUNCAOLOGICA> <OPRELACIONAL> <EXPRESSAO>
           --> <FUNCAOBOOLEANA>
           --> <EXPRESSAO> <OPRELACIONAL> <FUNCAOLOGICA>

<LISTAINSTRUCOESSE> --> <LISTAINSTRUCOESSE> <INSTRUCAOSE>
                    --> <INSTRUCAOSE>

<INSTRUCAOSE> --> <COMDATRIB>
              --> <COMDCONDICIONAL>
              --> <COMDREJEITE>
              --> <COMSEXECUTE>
              --> <COMDCOPIAPLANO>
              --> <COMDLEIA>
              --> <COMDSALVE>
              --> <COMDIMPUTACAO>
              --> <COMDERRO>
              --> <PROCEDIMENTO>
```

```
<FUNCAOBOOLEANA> --> PRIMEIRO
                 --> ULTIMO
                 --> CORRENTE
                 --> ANTERIOR
                 --> POSTERIOR
                 --> <NOME>
                 --> <STRING>
                 --> INTERVALO
                     ("VARIAVEL",<TERMOINT>,<TERMOINT>)
                 --> NUMERICO ( "VARIAVEL" )
                 --> INVALIDO ( "VARIAVEL" )

<FUNCAOLOGICA> --> TODAS ( <LISTATERMOSFUNCAOLOGICA> )
               --> QUALQUERUMA (<LISTARTERMOSFUNCAOLOGICA>)
               --> APENASUMA ( <LISTATERMOSFUNCAOLOGICA> )

<LISTATERMOSFUNCAOLOGICA> --> <LISTATERMOSFUNCAOLOGICA>
                                  <TERMOFUNCAOLOGICA>
                          --> <LISTATERMOSFUNCAOLOGICA>
                              <TERMOFUNCAOLOGICA>
                          --> <TERMOFUNCAOLOGICA>

<TERMOFUNCAOLOGICA> --> <CONSTNUMERICA>
                    --> <VARIAVEL>

<TERMOINTERVALO> --> <TERMO>
                 --> = <TERMO>
```

```
<TERMO> --> <CONSTNUMERICA>
        --> <VARIAVEL>
        --> <EXPRESSAO>

<ROTINAIMPUTACAO> --> IMPUTACAO <CONSTNUMERICA>
                      ( <LISTAVARIAVEIS> ) ;
                      <CORPOIMPUTACAO> ;
                      FIM IMPUTACAO ;

<CORPOIMPUTACAO> --> <LISTAVALORESIMPUTACAO>
                 --> <LISTAVALORESIMPUTACAO> INICIALIZE;

<LISTAVALORESIMPUTACAO> --> <LISTAVALORESIMPUTACAO>
                                <VALORIMPUTACAO>
                            --> <LISTAVALORESIMPUTACAO> ,
                                <VALORIMPUTACAO>
                            --> <VALORIMPUTACAO>

<VALORIMPUTACAO> --> <CONSTNUMERICA>
                 --> INV

<PROCPESQUISA> --> PROC PESQUISA ;
                   <CORPOPESQUISA> ;
                   FIM PESQUISA ;
               --> PROC PESQUISA ( <CONDICAO> ) ;
                   <CORPOPESQUISA>
                   FIM PESQUISA ;
```

```
                --> PROC PESQUISA ( ANTERIOR ) ;
                    <CORPOPESQUISA>
                    FIM PESQUISA ;
                --> PROC PESQUISA ( POSTERIOR ) ;
                    <CORPOPESQUISA>
                    FIM PESQUISA ;

<CORPOPESQUISA> --> <CORPOPESQUISA> <COMDSPESQUISA>
                --> <COMDSPESQUISA>

<COMDSPESQUISA> --> <COMDATRIB>
                --> <COMDCONDICIONAL>
                --> <COMDSALVE>
                --> <COMDERRO>
                --> <COMDSEXECUTE>
                --> <COMDCOPIAPLANO>
                --> <COMDIMPUTACAO>
                --> <COMDLEIA>
                --> <PROCEDIMENTO>
                --> RESTAURE ;

<COMDERRO> --> ERRO = 'coderro' ;

<COMDSEXECUTE> --> EXECUTE 'nomeROTEXT' ;
               --> EXECUTE 'nomeROTEXT'
                   ( <LISTAPARAMETROS> );
               --> EXECUTE 'nomeROTINT' ;
```

```
<ROTINAPLANCRIT> --> ROTPLAN <NOME> ;
                         <CORPOROTPLAN>
                         FIM ROTPLAN ;
                    --> ROTPLAN <NOME> (SIM) ;
                         <CORPOROTPLAN>
                         FIM ROTPLAN ;
                    --> ROTPLAN <NOME> (NAO) ;
                         <CORPOROTPLAN>
                         FIM ROTPLAN ;

<CORPOROTPLAN> --> <CORPOROTPLAN> <COMDROTPLAN>
                --> <COMDROTPLAN>

<COMDROTPLAN> --> <COMDATRIB>
               --> <COMDCONDICIONAL>
               --> <COMDERRO>
               --> <COMDEXECUTE>
               --> <COMDLEIA>
               --> <COMDIMPUTACAO>
               --> <COMDSALVE>
               --> <PROCEDIMENTO>
               --> EXCUTE EFEITO = SIM ;
               --> EXCUTE EFEITO = NAO ;

<ROTINAINTERNA> --> ROTINT <NOME>;
                        <CORPOROTINT>
                       FIM ROTINT;
```

<CORPOROTINT> --> <CORPOROTIN> <COMDSROTINT>

--> <COMDSROTINT>

<COMSDROTINT> --> <COMDATRIB>

--> <COMDCONDICIONAL>

--> <COMDERRO>

--> <COMDEXECUTE>

--> <COMDLEIA>

--> <COMDCOPIAPLANO>

--> <COMDIMPUTACAO>

--> <COMDSALVE>

--> <PROCEDIMENTO>

<PROCMARCA> --> PROC MARCA <NOME> ;

<CORPOPROCMARCA>

FIM MARCA ;

--> PROC MARCA <NOME> (MARQUE) ;

<CORPOPROCMARCA>

FIM MARCA ;

<CORPOPROCMARCA> --> <CORPOPROCMARCA> <COMDPROCMARCA>

--> <COMDPROCMARCA>

<COMDPROCMARCA> --> <COMDATRIB>

--> <COMDCONDICIONAL>

--> <COMDERRO>

--> <COMDEXECUTE>

--> <COMDMARQUE>

```
                --> <COMDLEIA>
                --> <COMDCOPIAPLANO>
                --> <COMDIMPUTACAO>
                --> <PROCEDIMENTO>

<PROC%NOME> --> PROC <NOME> ;
                <CORPOPROC%NOME>
                FIM <NOME> ;

<CORPOPROC%NOME> --> <CORPOPROC%NOME>
                     <COMDSPROC%NOME>
                 --> <COMDSPROC%NOME>

<COMDSPROC%NOME> --> <COMDATRIB>
                 --> <COMDSCONDICIONAL>
                 --> <COMDEXECUTE>
                 --> <COMDERRO>
                 --> <PROCEDIMENTO>
                 --> <COMDLEIA>
                 --> <COMDIMPUTACAO>
                 --> <COMDCOPIAPLANO>
                 --> <COMDSALVE>

<LISTAVARIAVEIS> --> <LISTAVARIAVEIS> "VARIAVEL"
                 --> <LISTAVARIAVEIS> , "VARIAVEL"
                 --> "VARIAVEL"
```

<PROCPLI> --> PROC PLI ; <CORPOPLI> FIM PLI ;

<CORPOPLI> --> 'qualquercomandopli'

<LISTACONSTNUMERICAS> --> <LISTACONSTNUMERICAS>
<CONSTNUMERICA>
--> <LISTACONSTNUMERICAS> ,
<CONSTNUMERICA>
--> <CONSTNUMERICA>

<COMDREJEITE> --> REJEITE ;

<LISTAPARAMETROS> --> <LISTAPARAMETROS> <PARAMETRO>
--> <LISTAPARAMETROS> , <PARAMETRO>
--> <PARAMETRO>

<PARAMETRO> --> "VARIAVEL"
--> <CONSTNUMERICA>
--> <STRING>

<COMDCOPIAPLANO> --> COPIAPLANO <STRING> (<LISTAREFPLANO>);

<LISTAREFPLANO> --> <LISTAREFPLANO> <REFPLANO>
--> <LISTAREFPLANO> , <REFPLANO>
--> <REFPLANO>

```
<REFPLANO> --> <STRING>
           --> <STRING> A <STRING>

<COMDLEIA> --> LEIA <NUMARQREF> ;

<NUMARQREF> --> REF1 ;
            --> REF2 ;
            --> REF3 ;
            --> REF4 ;
            --> REF5 ;
            --> REF6 ;
            --> REF7 ;
            --> REF8 ;
            --> REF9 ;

<COMDIMPUTACAO> --> ATUALIZE <CONSTNUMERICA> ;
                --> IMPUTE <CONSTNUMERICA> ;

<PROCEDIMENTOSFINAIS> --> <PROCSAIDA> <PROCFINALIZACAO>
                      --> <PROCSAIDA>
                      --> <PROCFINALIZACAO>

<PROCSAIDA> --> PROC SAIDA ;
                <CORPOSAIDA>
                FIM SAIDA ;
            --> PROCSSAIDA;(PLI) ;
                <CORPOPLI>;
                FIM SAIDA ;
```

<CORPOSAIDA> --> <CORPOSAIDA> <COMDSAIDA>

--> <COMDSAIDA>

<COMDSAIDA> --> <COMDATRIB>

--> <COMDCONDICIONAL>

--> <COMDSEXECUTE>

--> <COMDREJEITE>

--> <COMDLEIA>

--> <PROCREFERENCIA>

--> <PROCPLI>

<PROCFINALIZACAO> --> PROC FINALIZACAO ;

<CORPOFINALIZACAO>

FIM FINALIZACAO ;

<CORPOFINALIZACAO> --> <CORPOFINALIZACAO> <COMDFINALIZACAO>

--> <COMDFINALIZACAO>

<COMDFINALIZACAO> --> <COMDATRIB>

--> <COMDCONDICIONAL>

--> <COMDEXECUTE>

--> <PROCPLI>

<OPARITMETICO> --> +

--> -

--> /

--> *

--> **

<OPRELACIONAL> --> =

--> IGUAL

--> IGUAL A

--> >

--> MAIOR

--> MAIOR QUE

--> <

--> MENOR

--> MENOR QUE

--> <>

--> ^=

--> NAO IGUAL

--> NAO IGUAL A

--> DIFERENTE

--> DIFERENTE DE

--> ^>

--> NAO MAIOR

--> NAO MAIOR QUE

--> ^<

--> NAO MENOR

--> NAO MENOR QUE

--> >=

--> MAIOR OU IGUAL

--> MAIOR OU IGUAL A

--> <=

--> MENOR OU IGUAL

--> MENOR OU IGUAL A

```
<OPLOGICO> -->   &
           -->   E
           -->   |
           -->   OU

<OPNEGACAO> -->   ^
            -->   NAO

<SIM/NAO> --> SIM ;
          --> NAO ;

<SIMPLES/DUPLA> --> SIMPLES ;
                --> DUPLA ;

<UMA/DUAS> --> UMA ;
           --> DUAS ;

<FB/VB> --> FB ;
        --> VB ;

<IMPRIMIR/GRAVAR> --> IMPRIMIR ;
                  --> GRAVAR ;
```

Os símbolos terminais da gramática a serem reconhecidos pelo analisador léxico são as palavras reservadas, correspondentes às palavras escritas em maiúsculo, e símbolos sem estarem delimitados pelos símbolos < e >. Os demais símbolos terminais a serem

reconhecidos são:

"VARIÁVEL" --> Símbolo composto por um prefixo V, W, S, A, B, C, D, E, F, G, H, I, que indica o arquivo ao qual pertence a variável, seguido de uma cadeia de até 7 dígitos, que fornecem a identificação da variável.

<CONSTNUMERICA> --> As constantes numéricas são formadas por uma sequência de até 15 caracteres numéricos, podendo ser imediatamente precedidas de sinal e incluir ponto decimal.

<STRING> --> Representa as constantes alfanuméricas que são compostas por uma cadeia de até 256 caracteres delimitados por plics (') ou aspas (").

É o único símbolo da linguagem que permite continuação na linha seguinte. Esta continuação é considerada imediatamente a partir da primeira posição do campo de procedimentos do registro seguinte.

<NOME> --> Representa os identificadores criados pelo programador. São compostos pelo prefixo % seguidos de uma cadeia de até 30 caracteres formada por letras, dígitos e o caracter especial underscore (_).

A forma de entrada da especificação da Aplicação para o sistema Cripta, deverá ser feita através da digitação das especificações, obedecendo à gramática apresentada, num arquivo sequencial com o "lay-out" apresentado na figura V.13.

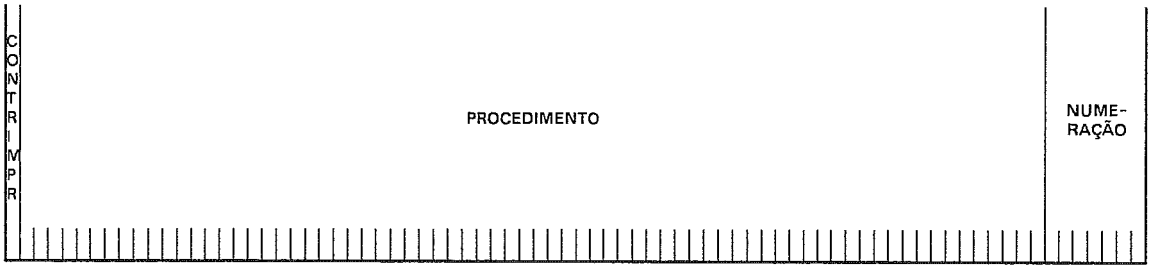


FIGURA V.13 - "Lay-out" do registro de especificação da aplicação Cripta

A forma de especificação usando a linguagem Cripta, é feita em formato livre, nos campos de cada registro relativos aos procedimentos, podendo um comando da linguagem ocupar tantas linhas quanto as desejadas. A única restrição imposta é a não permissão da continuação de um símbolo terminal, token da linguagem, na linha seguinte, com excessão para constantes alfanuméricas que possuirão um tratamento em separado.

A primeira coluna de cada registro é usada para controle da impressão do relatório de compilação e as oito últimas posições podem ser usadas para numeração ou comentário de livre escolha do programador.

O Anexo C apresenta um exemplo de especificação de uma aplicação mostrando o uso destas três ferramentas.

V.1.2 - DEFINIÇÃO DAS SAÍDAS

Analisando-se a figura V.2, Diagrama do fluxo de dados físico, observa-se que o sistema possui três saídas:

. especificação interpretada;

. relação de erros;

. código gerado;

A primeira e segunda saídas, especificação interpretada e relação de erros, podem ser analisadas como uma única saída, correspondente à saída da compilação das entradas fornecidas ao sistema, contendo as especificações das entradas, juntamente com os diagnósticos dos erros detectados pelo compilador.

A última saída, código gerado, corresponde ao programa em código executável em máquina, gerado pelo sistema de acordo com as especificações de entrada fornecidas.

A seguir são apresentadas cada uma destas saídas.

V.1.2.1 - RELATÓRIO DA COMPILAÇÃO

A especificação da aplicação, é formada por três partes distintas, conforme relacionadas no item V.1.1, Especificação das entradas, que serão "compiladas" pelo

sistema.

Devido a cada uma destas especificações possuir uma forma distinta de especificação, o relatório da compilação será formado por três lay_outs diferentes, referentes a cada uma das entradas com os respectivos diagnósticos de erros.

V.1.2.1.1 - RELATÓRIO DA COMPILAÇÃO DO DICIONÁRIO DA APLICAÇÃO

Como já citado anteriormente, uma aplicação pode usar mais de um arquivo de entrada, especificados em dicionários isolados, e, opcionalmente, um dicionário descrevendo as variáveis auxiliares.

Para a saída da compilação do dicionário serão usados dois tipos de relatórios. Um para quando for detectado algum erro de compilação, onde será impresso o arquivo que contém as especificações do dicionário tal como foram digitadas, com as respectivas indicações dos erros detectados. Este relatório possui o lay_out, apresentado na figura V.14, para qualquer um dos dicionários que esteja sendo analisado tendo sua identificação no título do cabeçalho.

O outro tipo de relatório de compilação do dicionário será usado para os casos em que o dicionário sendo analisado não contenha nenhum erro de compilação. Para este caso, as informações podem ser agrupadas como

```
-------------------------------------------------------------------------------------------------------------------------------------
SISTEMA CRIPTA                                                                                                       DATA - 99/99/99
                                            RELATORIO DE COMPILACAO DO DICIONARIO : BASICO                           HORA - 99:99:99
                                                                                                                     PAG. -   ZZZZZ9
-------------------------------------------------------------------------------------------------------------------------------------
NUMERO  |                                        DESCRICAO                               |                ERROS
  DA    |0        1         2         3         4         5         5         7         8|
LINHA   |12345678901234567890123456789012345678901234567890123456789012345678901234567890|  CODIGO    MENSAGEM
-------------------------------------------------------------------------------------------------------------------------------------

ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                            *****                                        ->E999DIC-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                     *******                                                             ->E999DIC-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                                                   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
           *********                                                                     ->E999DIC-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                           ********                                                      ->E999DIC-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                                                   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                  **                                     ->E999DIC-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ZZZZ9    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

FIGURA V.14 - Relatório de compilação de um dicionário com erros

apresentadas nos lay_outs das figuras V.15 e V.16, onde a primeira se refere à primeira página do relatório contendo as características do arquivo e a outra às demais páginas do relatório contendo as descrições das variáveis. Este relatório é usado para qualquer um dos dicionários de arquivo de entrada, sendo sua identificação feita no título do cabeçalho.

Como o dicionário das variáveis auxiliares tem características diferentes de um dicionário para um arquivo de entrada, o relatório de compilação deste, quando sem erros, tem um lay_out diferente, que é apresentado na figura V.17.

V.1.2.1.2 - RELATÓRIO DA COMPILAÇÃO DA APLICAÇÃO CRIPTA E PLANO DE CRÍTICA

A especificação da aplicação é formada pelas entradas Especificação da Aplicação Cripta e, opcionalmente, pelo Plano de Crítica, PCAUT. Estas duas entradas possuem formas de especificação diferentes, mas como o Plano de Crítica está proposto para que trechos deste possam ser inseridos em um ou mais pontos do programa, a saída da compilação destas duas entradas é feita conjuntamente compondo-se o relatório conforme as especificações fornecidas. O lay_out deste relatório é apresentado na figura V.18.

A forma de indicação de erro escolhida, se deve a obter-se uma melhor legibilidade do relatório da

FIGURA V.15 - Relatório de compilação de um dicionário concreto (características do arquivo)

SISTEMA CRIPTA

DESCRICAO DO DICICONARIO : BASICO - CARACTERISTICAS DO ARQUIVO

DATA - 99/99/99
HORA - 99:99:99
PAG. - ZZZZZ9

NOME : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

FORMATO : FIXO SEM TIPOS DE REGISTROS
FIXO COM TIPOS DE REGISTROS
VARIAVEL POR TIPO DE REGISTROS
VARIAVEL POR OCORRENCIA

NUMERO DE TIPOS DE REGISTROS : 99 VARIAVEL DO TIPO DE REGISTRO : 9999999

DESCRICAO DOS TIPOS DE REGISTROS:

INDICADOR	TAMANHO	N.MIN REGS OBRIGATORIOS	N.MAX REGS PERMITIDOS
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999
XXXXX	99999	99999	99999

VARIAVEL CONTADORA DE OCORRENCIAS : 9999999

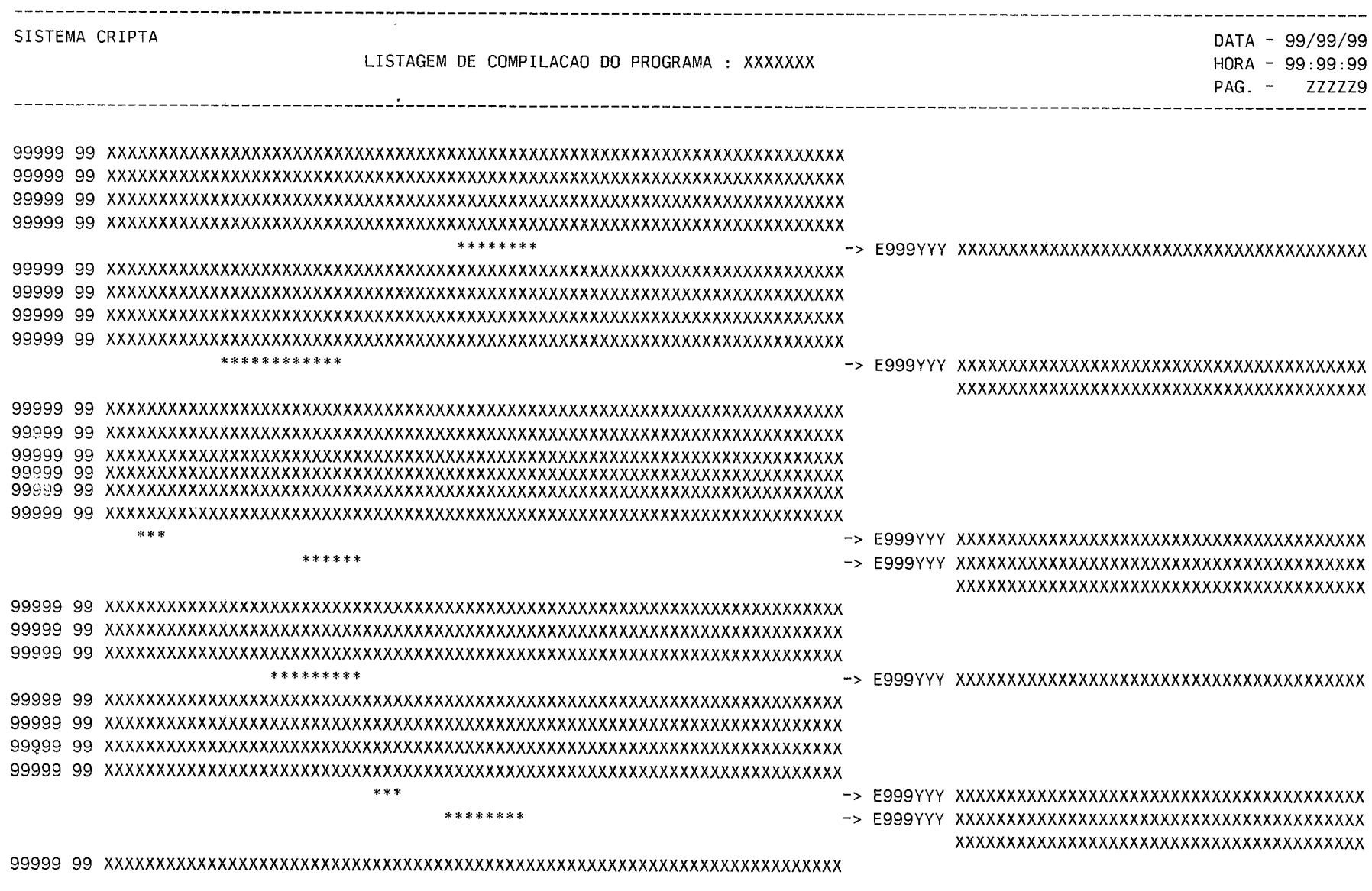
VARIAVEL DE OCORRENCIA : 9999999

```
------------------------------------------------------------------------------------------------------------------------------------
SISTEMA CRIPTA                                                                                                       DATA - 99/99/99
                                  DESCRICAO DO DICICONARIO : BASICO - DESCRICAO DAS VARIAVEIS                        HORA - 99:99:99
                                                                                                                     PAG. -   ZZZZZ9
------------------------------------------------------------------------------------------------------------------------------------
VARIAVEL NOME
         POSINIC   TAM CARACT FORMATO DEC NCAT NOCOR VARBASE LIMITE-INFERIOR LIMITE-SUPERIOR REPRES-IGNORADO VALOR-ASSUMIDO TIPO_REG
         CATEGORIAS: CODIGO  NOME
------------------------------------------------------------------------------------------------------------------------------------

9999999  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          99999    999    X      X     Z  9999 9999  9999999 ZZZZZZZZZZZZZZ9 ZZZZZZZZZZZZZZ9 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX  XXXXX

9999999  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          99999    999    X      X     Z  9999 9999  9999999 ZZZZZZZZZZZZZZ9 ZZZZZZZZZZZZZZ9 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX  XXXXX

9999999  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          99999    999    X      X     Z  9999 9999  9999999 ZZZZZZZZZZZZZZ9 ZZZZZZZZZZZZZZ9 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX  XXXXX

9999999  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          99999    999    X      X     Z  9999 9999  9999999 ZZZZZZZZZZZZZZ9 ZZZZZZZZZZZZZZ9 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX  XXXXX

9999999  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          99999    999    X      X     Z  9999 9999  9999999 ZZZZZZZZZZZZZZ9 ZZZZZZZZZZZZZZ9 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX  XXXXX
         CATEGORIAS:
                      9999 - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                      9999 - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                      9999 - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                      9999 - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                      9999 - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                      9999 - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                      9999 - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

9999999  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          99999    999    X      X     Z  9999 9999  9999999 ZZZZZZZZZZZZZZ9 ZZZZZZZZZZZZZZ9 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX  XXXXX

9999999  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          99999    999    X      X     Z  9999 9999  9999999 ZZZZZZZZZZZZZZ9 ZZZZZZZZZZZZZZ9 XXXXXXXXXXXXXXX XXXXXXXXXXXXXXX  XXXXX
```

FIGURA V.16 - Relatório de compilação de um dicionário correto (descrição das variáveis)

```
------------------------------------------------------------------------------------------------------------------------------------
SISTEMA CRIPTA                                                                                                       DATA - 99/99/99
                                  DESCRICAO DO DICICONARIO DAS VARIAVEIS AUXILIARES                                  HORA - 99:99:99
                                                                                                                     PAG. -   ZZZZZ9
------------------------------------------------------------------------------------------------------------------------------------
VARIAVEL    NOME
            TAMANHO   CARACTERISTICA   FORMATO   DECIMAIS   NUM-CATEGIRIAS   NUM-OCORRENCIAS   VARIAVEL-BASE   POSINIC-REDEFINICAO
           CATEGORIAS: CODIGO  NOME
------------------------------------------------------------------------------------------------------------------------------------

9999999     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              999            X             X         9          9999              9999            9999999             9999999

9999999     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              999            X             X         9          9999              9999            9999999             9999999

9999999     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              999            X             X         9          9999              9999            9999999             9999999



9999999     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              999            X             X         9          9999              9999            9999999             9999999

9999999     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              999            X             X         9          9999              9999            9999999             9999999

9999999     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
              999            X             X         9          9999              9999            9999999             9999999
            CATEGORIAS:
                          9999   - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                          9999   - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                          9999   - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                          9999   - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                          9999   - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                          9999   - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                          9999   - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

FIGURA V.17 - Relatório de compilação de um dicionário correto das variáveis auxiliares



```
----------------------------------------------------------------------------------------------------------------------------------
SISTEMA CRIPTA                                                                                                     DATA - 99/99/99
                                  LISTAGEM DE COMPILACAO DO PROGRAMA : XXXXXXX                                     HORA - 99:99:99
                                                                                                                   PAG. -   ZZZZZ9
----------------------------------------------------------------------------------------------------------------------------------

99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                           ********                            -> E999YYY XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                    ************                                               -> E999YYY XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
            ***                                                                -> E999YYY XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                            ******                                             -> E999YYY XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                         *********                                             -> E999YYY XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                   ***                                         -> E999YYY XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                         ********                              -> E999YYY XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                                                                                          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
99999 99 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

FIGURA V.18 - Relatório de compilação do programa de crítica

compilação. Para cada mensagem de erro, são sublinhados os símbolos da linha da especificação envolvidos no erro, permitindo uma identificação, no código fonte, do erro indicado. A razão do código do erro de cada mensagem ser precedida pelo símbolo "->", é permitir uma busca mais rápida das ocorrências de erro, principalmente, quando a saída da compilação estiver sendo analisada em terminal.

V.1.2.2 - CÓDIGO GERADO

Esta saída corresponde ao programa, em código executável, gerado pelo sistema.

A linguagem de programação na qual o programa será gerado é a linguagem PLI, escolhida por ser uma linguagem que atende tanto a aplicações comerciais como científicas e, sobretudo, por ser a que mais se adaptava a este tipo de sistema, no ambiente IBM usado para desenvolvivento deste trabalho. Outra linguagem que se adapta perfeitamente para este tipo de sistema, é a linguagem C. Entretanto, esta linguagem ainda não se encontra instalada nas máquinas a serem usadas na implementação.

Este tipo de saída do sistema, por suas características, não possui um "lay-out". O que se pode definir para este tipo de saída, é o esqueleto do programa a ser gerado.

Conforme os requisitos especificados para o sistema, existem dois tipos de processamento para os programas de

crítica: processamento do arquivo de entrada sequencialmente ou por grupos de registros, para os casos de crítica de questionários formados por mais de um tipo de registro, com crítica de variáveis inter-registros.

As figuras V.19 e V.20 apresentam os esqueletos de processamento dos programas gerados para cada um dos tipos de processamento citados. A forma de notação usada é o pseudo-código.

V.1.3 - ESTRUTURA MODULAR DO SISTEMA

Por este ser um sistema altamente atípico dos normalmente analisados pela literatura de Engenharia de Software, torna-se difícil o uso de uma das notações recomendadas pelas metodologias propostas. No entanto, as ferramentas propostas para esta fase do sistema, serão usadas sempre que possível de forma a tornar a documentação desta parte do sistema o mais claro possível.

Observando-se o fluxo de dados físico para o processamento de uma especificação da aplicação fornecida ao sistema, representado na figura V.2, podemos verificar que o sistema consta de um único módulo com a função de "compilar" a especificação fornecida. No entanto, devido à "Especificação da aplicação" fornecida ao sistema ser formada por três entradas distintas, este módulo deve ser dividido em três partes, cada uma tendo a função de "compilar" a sua entrada específica.

```
Declarações
Inicializações
Leitura do arquivo de entrada
Enquanto não ocorrer fim do arquivo de entrada
        Executa procedimentos especificados para o registro
        Grava registro no arquivo de saída
        Lê próximo registro do arquivo de entrada
Fim-enquanto
Finalizações
```

FIGURA V.19 - Esqueleto de um programa para processamento sequencial

```
Declarações
Inicializações
Leitura do arquivo de entrada
Enquanto chave do reg do arquivo de entrada igual à anterior
        Armazena registro na tabela
        Lê próximo registro do arquivo de entrada
Fim-enquanto
Inicializa índice da tabela
Enquanto houver registro na tabela
        Executa procedimentos especificados para o registro
        Incrementa índice da tabela
Fim-enquanto
Inicializa índice da tabela
Enquanto houver registro na tabela
        Grava registro no arquivo de saída
        Incrementa índice da tabela
Fim-enquanto
Finalizações
```

FIGURA V.20 - Esqueleto de um programa para processamento por grupos de registros

Analisando-se a especificação das entradas, contida no item V.1.1 deste trabalho, verificamos que a especificação principal é a especificação Cripta, que referenciará as demais entradas a serem usadas, isto é, que dicionários serão analisados e que trechos do Plano de Crítica deverão ser "compilados". Assim, podemos definir como módulo gerente o módulo "compilador" da Especificação da Aplicação, que chamará os módulos "compiladores" de análise dos comandos da linguagem Cripta e das entradas "Dicionário da Aplicação" e "Plano de Crítica".

No início deste capítulo foram descritas as fases de um "compilador" e suas formas de implementação. A forma de implementação escolhida é a de um compilador de passo único, devido às características da linguagem especificada e das instalações a serem usadas.

Analisando-se as características da linguagem Cripta, verificamos que sua compilação em um passo, se torna simples de implementar devido à sua forma de especificação que não permite a especificação de comandos para os quais sua análise e geração de código dependa de comandos que possam ser especificados posteriormente.

Outra característica da linguagem Cripta, que a torna possível de compilação em um só passo, é a hierarquia da especificação de seus comandos, de tal modo que a geração de código pode ser feita de forma linear à análise do programa sendo compilado.

O ambiente em que o compilador será usado não causa problemas à implementação, devido à instalação a ser

utilizada ser de grande porte, não impondo grandes restrições à quantidade de memória necessária, permitindo o uso de estruturas de "over-lay", conforme a hierarquia dos módulos a serem especificados.

Um programa Cripta a ser "compilado" é formado pelos seguintes construtores da gramática:

PROGRAMA - Representa a unidade básica a ser executada, que é formada por blocos de procedimentos que podem ser subrotinas ou procedimentos automatizados.

BLOCOS DE PROCEDIMENTOS - São unidades compostas por comandos da linguagem, que formam outro construtor, particulares a cada bloco específico.

COMANDOS - São formados pelas instruções que compõem a linguagem podendo incluir os construtores de Expressões dependendo da gramática de cada comando.

EXPRESSÕES - São formadas pelos contrutores de FUNÇÕES, OPERADORES e IDENTIFICADORES.

Observando-se a especificação da gramática da linguagem Cripta, nota-se que grande parte dos blocos de procedimentos são bem distintos entre eles, cada um possuindo um conjunto de comandos particulares. Isto torna possível a especificação de módulos analisadores da "compilação" a nível macro para cada um destes processos de

modo isolado, facilitando a hierarquia do sistema e estrutura de "over-lay", otimizando a performance do "compilador" em máquina.

Quando é citado a especificação em nível macro do módulo analisador da "compilação" dos blocos de procedimentos, significa que este módulo se encarregará de analisar a estrutura de mais alto nível, resultante do grupamento da análise dos demais componentes que o formam, que será feita por módulos comuns a todos estes grandes módulos.

A figura V.21 apresenta o esquema da especificação de uma aplicação em linguagem Cripta, que será uma das entradas para o sistema, usando todas as estruturas permitidas pela linguagem. Os símbolos "(" e ")" significam que seu conteúdo é opcional num programa, e os símbolos "[" e "]" significam que pelo menos uma das estruturas especificadas em seu conteúdo é obrigatório no programa. O símbolo de elevado a "*" significa repetição de estruturas. As estruturas estão representadas somente pelos seus símbolos iniciais.

Cada uma destas estruturas representadas corresponde a um módulo do sistema que tem como funções todas as fases de um "compilador" como análise léxica, sintática e semântica, incluindo os módulos de geração de código e tratamento de erros.

Pela gramática da linguagem, a maioria destas estruturas é formada por estruturas internas de comandos particulares a cada uma. No entanto, existem casos em que

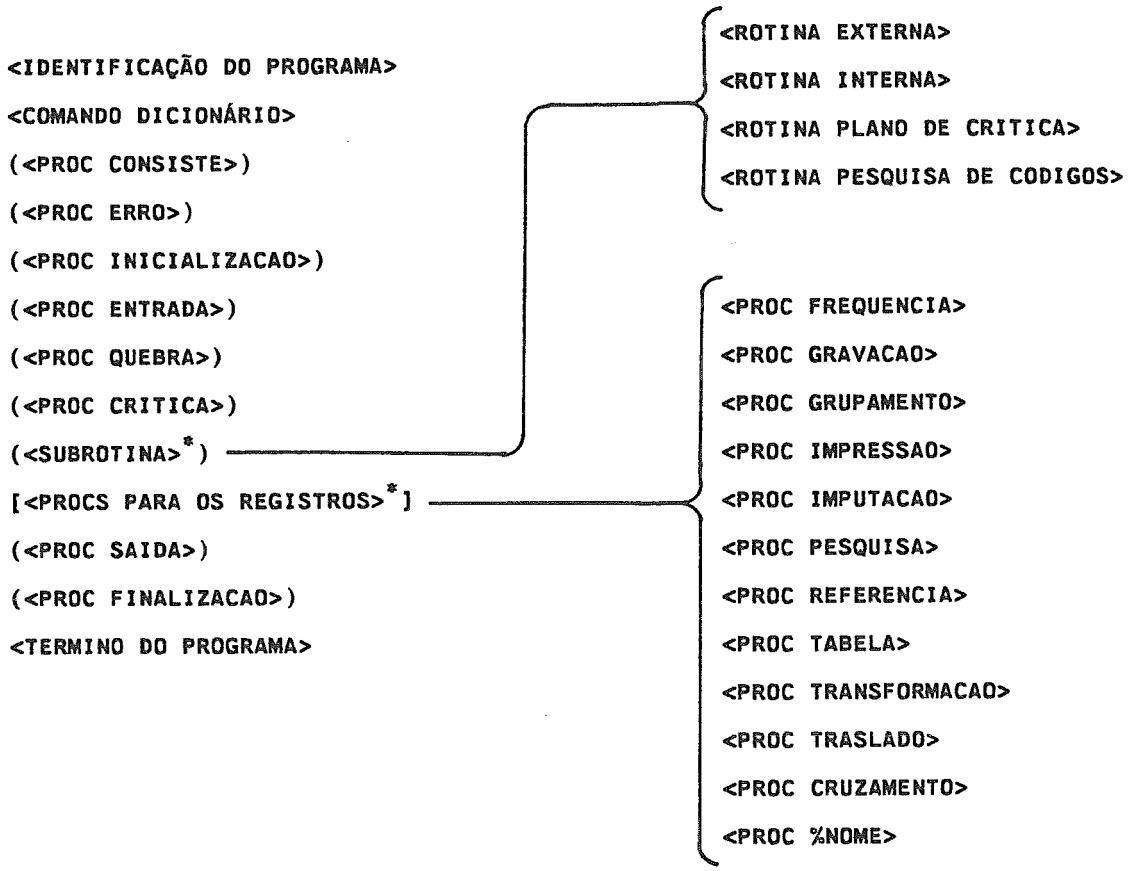


FIGURA V.21 - Esquema da entrada "Especificação da Aplicação Cripta" com as estruturas permitidas pela linguagem

estas estruturas internas podem pertencer também a outras estruturas. Por exemplo, o módulo de análise das estruturas de subrotinas, pela característica da linguagem, permite que sejam usadas estruturas comandos que também serão permitidos no módulo de análise de estruturas de blocos de procedimentos para o registro.

Assim, existirá um submódulo de análise destas estruturas comandos a serem chamados por ambos os módulos. Também, todos estes módulos, com excessão dos módulos "compilação do dicionário" e "compilação do plano de

crítica" que possuirão seus submódulos particulares usarão um único submódulo de análise léxica que será único para todo o sistema, por sua função de reconhecedor dos símbolos da linguagem, que o torna necessário em cada módulo que fará o reconhecimento das estruturas da linguagem, o mesmo acontecendo com o módulo de indicação de erros, uma vez que esta função será padrão para toda a análise da linguagem.

As funções de análise e geração de código em um único passo, de forma isolada para cada estrutura, são facilitadas, como já citado, pela hierarquia da linguagem. Se analisarmos as figuras V.21 e V.19 verificamos que a forma em que um programa Cripta é especificado, é muito semelhante à do programa a ser gerado.

Para representação deste tipo de arquitetura com este aninhamento de módulos, o diagrama hierárquico do sistema se encontra dividido nas figuras V.22 a V.70. A descrição funcional dos principais módulos do sistema é apresentada a seguir.

V.1.4 - DESCRIÇÃO DOS PRINCIPAIS MÓDULOS DO SISTEMA

- ANALISADOR LÉXICO (Fig. D.1)

Este módulo é usado por quase todos os demais módulos do sistema sendo o responsável pela identificação de cada símbolo da linguagem.

A cada execução, procura o próximo símbolo no arquivo do programa Cripta fornecido ao sistema e analisa este símbolo classificando-o ou emitindo mensagem de erro, para o caso de um símbolo não reconhecido.

A comunicação com os módulos que o chamam é feita por uma área do programa declarada com escopo externo, formada por:

- Código
  - classe
  - identificação
- número do registro
- posição inicial no registro
- tamanho do símbolo
- símbolo reconhecido
- indicação de erro
- índice da variável no dicionário

- INDICAÇÃO DE ERRO (Fig. D.2)

Responsável pela indicação dos erros de compilação. A partir do código do erro, da mensagem indicada e das informações fornecidas pelo analisador léxico, este módulo grava um registro com estas informações num arquivo auxiliar a ser usado posteriormente pelo impressor do relatório de compilação.

- PROCESSAR SISTEMA CRIPTA (Fig. D.1)

É o módulo gerente do sistema sendo o responsável pela verificação da estrutura do programa Cripta sendo compilado.

Verifica a sequência das estruturas de comandos permitidas pela linguagem, usadas no programa sendo compilado, fazendo o diagnóstico de erro ou chamando os módulos associados a cada uma das estruturas.

- GERAR IDENTIFICAÇÃO DO PROGRAMA (Fig. D.2)

Analisa o comando de declaração do programa Cripta, fazendo a indicação de erro, caso necessário e gerando o trecho do programa PLI correspondente à declaração do programa e das variáveis de trabalho a serem usadas em qualquer programa gerado como data, hora, contadores de registros lidos, etc.

- MONTAR DICIONÁRIOS DAS VARIÁVEIS (Fig. D.3)

Analisa o comando DICIONARIO especificado no programa Cripta e chama o analisador do dicionário, informando o tipo de dicionário a ser processado.

- ANALISAR DICIONÁRIO (Fig. D.7)

A partir da informação recebida, este módulo faz a análise do dicionário correspondente de acordo com a especificação do Dicionário da Aplicação fornecida no item V.1.1.1.

As informações do dicionário analisado são armazenadas em tabelas e ao final da análise do dicionário é impresso o relatório de compilação associado conforme especificado no item V.1.2.1.1.

- GERAR PROGRAMA DE CONSISTÊNCIA DOS QUESTIONÁRIOS (Fig. D.4)

Este módulo é responsável pela geração de um programa para consistência do arquivo a ser processado.

De acordo com as informações analisadas no bloco de comandos "PROC CONSISTE" e a descrição do arquivo a ser processado, fornecida pelo dicionário especificado, o programa gerado pode fazer as seguintes consistências:

. Consistência do preenchimento dos valores de todas as variáveis que compõem o arquivo, emitindo relatório de ocorrências e/ou estatísticas.

. Consistência do preenchimento dos valores de uma lista de variáveis especificadas, pertencentes ao arquivo descrito no dicionário, emitindo relatório de ocorrências e/ou estatísticas.

. Verificação da estrutura dos registros que formam um arquivo, de acordo com os tipos de registros especificados no dicionário, sendo agrupados pela chave composta pelas variáveis fornecidas como identificação.

- GERAR CONDIÇÃO "ON ERROR" DO PROGRAMA (Fig. D.4)

Este módulo gera a cláusula "ON ERROR" no programa PLI gerado, que especifica que ações devem ser tomadas no caso de ocorrência de erro de execução no programa gerado.

Caso o usuário tenha especificado uma estrutura "PROC ERRO" no programa Cripta, esta é analisada e as ações, para a cláusula "ON ERROR", são geradas de acordo com o especificado, senão, será gerado o procedimento "default" de impressão do último registro processado no programa gerado.

- GERAR PROCEDIMENTOS DE INICIALIZAÇÃO DO PROGRAMA (Fig. D.10)

Se o usuário especificou uma estrutura "PROC INICIALIZAÇÃO", este módulo faz sua análise e gera os procedimentos correspondentes em PLI.

- GERAR ROTINA DE LEITURA DO ARQUIVO DE ENTRADA (Fig. D.1)

Gera a rotina de leitura do arquivo a ser processado pelo programa gerado. Se não foi especificada uma estrutura "PROC ENTRADA" no programa Cripta, é gerada uma rotina de leitura sequencial de um registro do arquivo.

Caso tenha sido especificada a estrutura "PROC ENTRADA", é gerada a rotina de leitura do arquivo com a leitura sequencial do registro e os procedimentos correspondentes, em PLI, aos especificados nesta estrutura.

Se foi especificada uma estrutura "PROC ENTRADA(PLI)", a rotina de leitura do arquivo de entrada é gerada com o código PLI especificado nesta estrutura.

- GERAR LOOP DE PROCESSAMENTO DO ARQUIVO DE ENTRADA (Fig. D.12)

Gera o loop de processamento principal do arquivo de entrada. Caso tenha sido especificada uma estrutura "PROC QUEBRA", esta é analisada e é gerado o loop e procedimentos

para armazenamento dos registros numa tabela em memória. A dimensão da tabela, a chave de controle de quebra e os totais são definidos de acordo com as especificações fornecidas nesta estrutura.

- MONTAR TABELA DE ERROS DE CRITICA (Fig. D.13)

Caso tenha sido especificada uma "PROC CRITICA" no programa Cripta sendo compilado, esta é analisada, e as informações especificadas são armazenadas numa tabela com escopo externo.

Estas informações são usadas para geração da declaração do relatório de crítica a ser emitido pelo programa gerado e, posteriormente, serão usadas pelo módulo analisador da instrução "ERRO" do Cripta, para geração da indicação de um erro neste relatório.

- GERAR SUBROTINAS (Fig. D.14)

De acordo com as estruturas "ROTEXT", "ROTINT", "ROTPLAN", "IMPUTACAO" e "ROTPESQ", estas são analisadas e é gerado o código PLI correspondente, de acordo com a função de cada uma destas estruturas conforme apresentado no Anexo B, "Manual da Linguagem Cripta".

- ANALISAR PROCEDIMENTOS PARA O REGISTRO (Fig. D.15)

De acordo com as estruturas de procedimentos

especificadas, cada uma destas é analisada e é gerado o código PLI correspondente de acordo com a função de cada estrutura especificada conforme apresentado no Anexo B, "Manual da Linguagem Cripta".

- GERAR GRAVAÇÃO DO ARQUIVO DE SAÍDA (Fig. D.16)

Gera a rotina de gravação do arquivo a ser processado pela aplicação. Se não foi especificada uma estrutura "PROC SAIDA", no programa Cripta, é gerada a gravação de modo sequencial do registro processado.

Caso tenha sido especificada uma estrutura "PROC SAIDA", no programa Cripta, esta é analisada e são gerados os procedimentos especificados nesta estrutura, além da gravação do registro processado.

Se foi especificada uma estrutura "PROC SAIDA(PLI)", a gravação do registro processado é gerada diretamente a partir do código PLI especificado nesta estrutura.

- GERAR TÉRMINO DO PROGRAMA (Fig. D.6)

Analisa o comando de término do programa Cripta e gera os procedimentos de fechamento do programa gerado, conforme as funções geradas e imprime o relatório de compilação da aplicação.

- EMITIR RELATÓRIO DE COMPILAÇÃO (Fig. V.27)

Este módulo é executado ao término da compilação da aplicação Cripta e, a partir dos arquivos que contêm o programa Cripta, o Plano de Crítica (opcional) e as mensagens de erro, emite o relatório de compilação intercalando as mensagens nos registros do programa fonte, sublinhando os símbolos e indicando a mensagem associada a cada erro como mostrado na figura V.18.

V.2 - RECURSOS NECESSÁRIOS

Hardware:

- Processador IBM da linha 43XX ou 30XX
- 1 unidade de disco
- 1 impressora
- 1 unidade de fita magnética

Software:

- Sistema operacional: OS/VS ou VM/XA
- Compilador: PLI optimizing compiler

V.3 - CONSTRUÇÃO DO SISTEMA

A partir do projeto Cripta e dos recursos descritos anteriormente foram implementados os principais módulos do sistema: Analisador gerente das estruturas do programa, analisador léxico, analisador dos dicionários, rotina de indicação de erro, rotina de gravação do código gerado, geração do código "default" do esqueleto do programa, analisador sintático das "procs" quebra, critica e consiste, analisador e gerador de código para expressões aritméticas com dois operandos, e analisador léxico do Plano de Critica.

As áreas de comunicação entre os módulos foram declaradas como áreas externas aos módulos e especificadas em "books" a serem incluidos nos programas. As áreas das tabelas de cada Dicionário da Aplicação analisado são alocadas dinamicamente em tempo de execução, otimizando, assim, a área necessária.

Os trechos de código PLI do programa, a ser gerado, utilizado como "default" na geração do programa, foram especificados num arquivo auxiliar e, durante a execução da geração de código, estes são acessados por uma chave correspondente a cada trecho.

A gravação do programa gerado é feita em três arquivos de trabalho: "DECL", "PGM1" e "PGM2". O arquivo "DECL" é usado para se gravar o código PLI gerado correspondente às declarações de arquivos e variáveis a serem usados no programa gerado. O arquivo "PGM1" é usado para gravação do

código PLI gerado correspondente aos procedimentos do programa. E, por último, o arquivo "PGM2" é usado para os casos especiais de geração de código a ser incluido antes de um trecho de código PLI, já gerado e gravado, no arquivo "PGM1". A indicação de onde deve ser inserido este código é feita pela indicação do número do registro do arquivo "PGM1" correspondente.

Assim, o programa gerado final é montado a partir da concatenação do arquivo "DECL" com o arquivo resultante de um "balanced-line" entre os arquivos "PGM1" e "PGM2".

CAPÍTULO VI

CONCLUSÕES E FUTURAS PESQUISAS

O objtivo deste trabalho é fornecer um conjunto de ferramentas para o desenvolvimento de sistemas de apuração de dados.

Para geração da proposta destas ferramentas foi realizada uma pesquisa das características destes tipos de sistemas e as ferramentas disponíveis. A proposta desenvolvida está baseada numa reunião das características mais vantajosas das ferramentas estudadas e dos requisitos observados para o atendimento a estes tipos de sistemas.

Com a implementação dos módulos principais deste sistema verificamos a viabilidade do desenvolvimento destes tipos de ferramentas.

O sistema está projetado e modulado de forma a facilitar a inclusão de novas funções à linguagem Cripta e novas características ao sistema, possibilitando assim refinamentos para o sistema proposto, verificados a partir das necessidades observadas no transcorrer do uso destas ferramentas.

Duas abordagens que poderiam ser propostas para este projeto, mas que não foram analisadas nesta versão são o

uso de uma base Dicionários da Aplicação e a consistência das regras de crítica, especificadas nos Planos de Crítica.

Para a primeira proposta poderia ser analisada a possibilidade da geração dos Dicionários da Aplicação a partir, ou em conjunto, com a Base de Metadados da empresa.

Para a consistência das regras de crítica, a partir do momento em que seja definido um algoritmo para consistência destas regras, através do uso de lógica de predicado, é fácil para o sistema proposto gerar, a partir do Plano de Crítica especificado, uma entrada para execução deste algorítmo.

Assim, através de refinamentos sucessivos, estas ferramentas propostas podem se tornar cada vez mais eficientes e produtivas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CAMPOS, ALDA, "O progresso de software está mesmo emperrado?", Data News, n° 389, pp. 20-20, 1987

[2] GUIMARÃES, GILBERTO, "Produtividade, meta a ser conquistada", Data News, n° 389, pp. 25-27, 1987

[3] BOEHM, B. W., Software Engineering Economics, New Jersey, Prentice-Hall, 1a edição, 1981

[4] GANE, C. e SARSON, T., Análise Estruturada de Sistemas, Rio de Janeiro, Livros Técnicos e Científicos, 1a edição, 1983

[5] DE MARCO, T., Controlling Software Projects, New York, Yourdon Press, 1a edição, 1982

[6] U.S. BUREAU OF THE CENSUS, Concor data editing package, User's Guide

[7] UNITED NATIONS, Unedit - A generalized editing software package for Census and survey data, User's Guide, Versão 3.1, 1980

[8] SILVA, ANTONIO CLAUDIO C. M., Atlas - Sistema de crítica, imputação e tabulação de dados, Manual do usuário, IBGE - Diretoria de Informática, Rio de Janeiro, 1983

[9] UNIVERSIDADE FEDERAL DE VIÇOSA - Central de processamento de dados, SISQUE - Sistema para consistência e preparação de dados, Manual do Usuário, Versão 1.0, 1985

[10] HUNGARIAN CENTRAL STATISTICAL OFFICE, Aero - Systems philosophy, User's Guide, Budapest, 1981

[11] GRAVES, RONALD B., CAN_EDIT - A specification driven editing system for the canadien census of population and housing, User's Guide, Statistics Canada

[12] -INTERNACIONAL-, "Ferramentas de produtividade e suas promessas de Nirvana", Data News, n° 386, pp 28-38, 1987

[13] ROCHA, ANA REGINA CAVALCANTI, Análise e Projero Estruturado de Sistemas, Rio de Janeiro, Editora Campus, 1a edição, 1987

[14] ULLMAN, JEFFREY D. e AHO. V. ALFRED, Principles of compiler design, Massachusetts, 1977

ANEXO A

FORMULÁRIOS PARA DESCRIÇÃO DO DICIONÁRIO

São usados cinco formulário para especificação de cada um dos dicionários da aplicação Cripta, apresentados nas figuras A.1 a A.5.

DESCRIÇÃO DO ARQUIVO DE ENTRADA

CHAVE	NOME DO ARQUIVO	FORMATO	TAMANHO DO REGISTRO	NTIPOSREG	VARIÁVEL CONTADORA DE OCORRÍNCIAS	VARIÁVEL DE OCORRÍNCIA	
A0							

DESCRIÇÃO DOS TIPOS DE REGISTROS

CHAVE	TIPO DO REGISTRO	TAMANHO DO REGISTRO	NÚMERO MÍNIMO DE REGISTROS	NÚMERO MÁNIMO DE REGISTROS	COMENTÁRIO
R0					
R0					
R0					
R0					
R0					
R0					
R0					
R0					
R0					

FIGURA A.1 - Formulário para descrição do arquivo

DESCRIÇÃO DAS CARACTERÍSTICAS FÍSICAS DAS VARIÁVEIS

CHAVE	NÚMERO DA VARIÁVEL	NOME DA VARIÁVEL	POSIÇÃO INICIAL	TAMANHO	CARACTERES DECIMAIS	INDICADOR DE MARGENS	CÓDIGO DE RESPOSTAS	VARIÁVEL DE BASE	TIPO DE REGISTRO
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									
V1									

FIGURA A.2 - Formulário para descrição das características físicas das variáveis

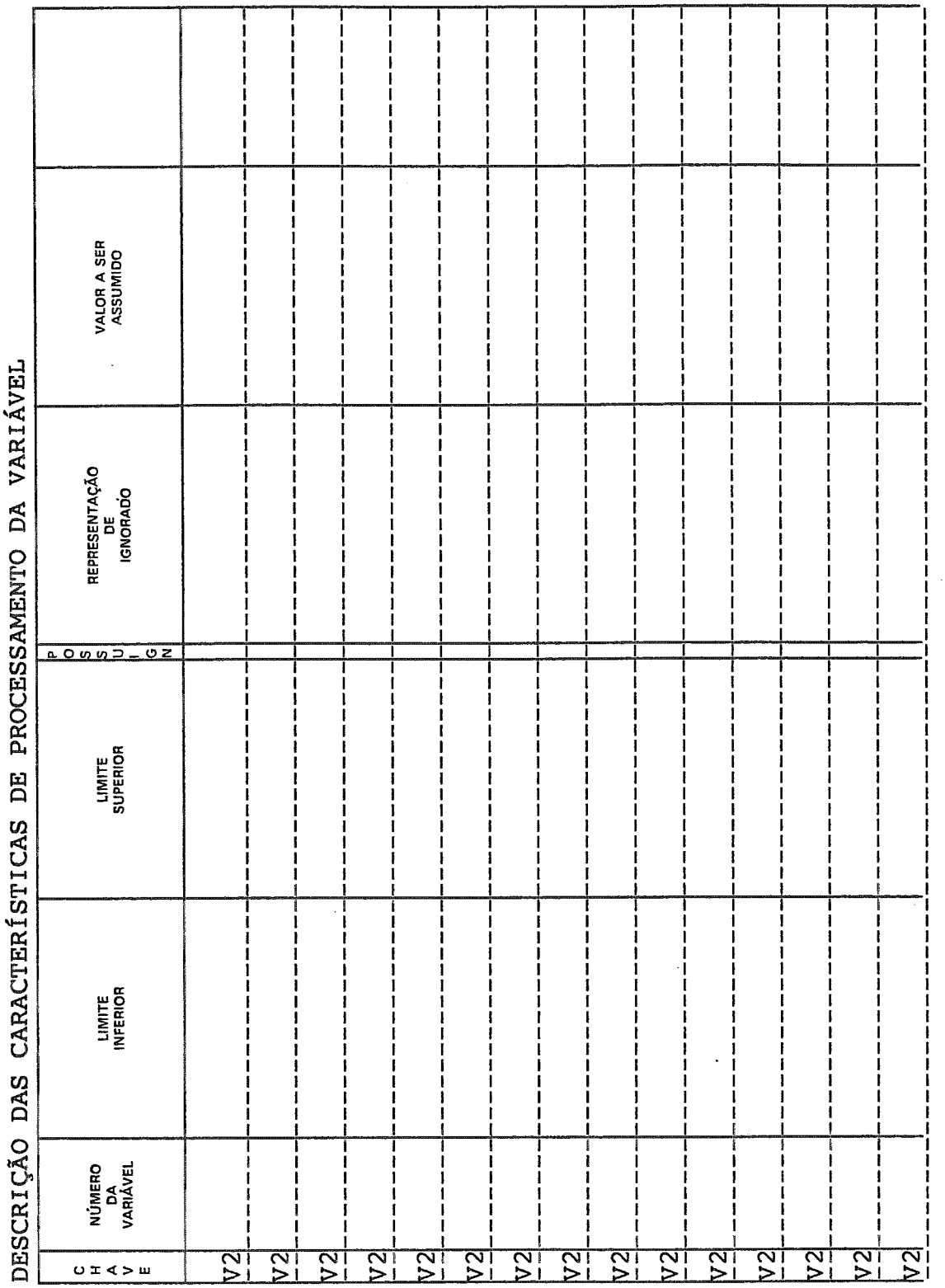


FIGURA A.3 - Formulário para descrição das características de processamento das variáveis

DESCRIÇÃO DAS CATEGORIAS

CHAVE	NÚMERO DA VARIÁVEL	CÓDIGO DA CATEGORIA	NOME DA CATEGORIA	COMENTÁRIO
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				
V3				

FIGURA A.4 - Formulário para descrição das categorias das variáveis

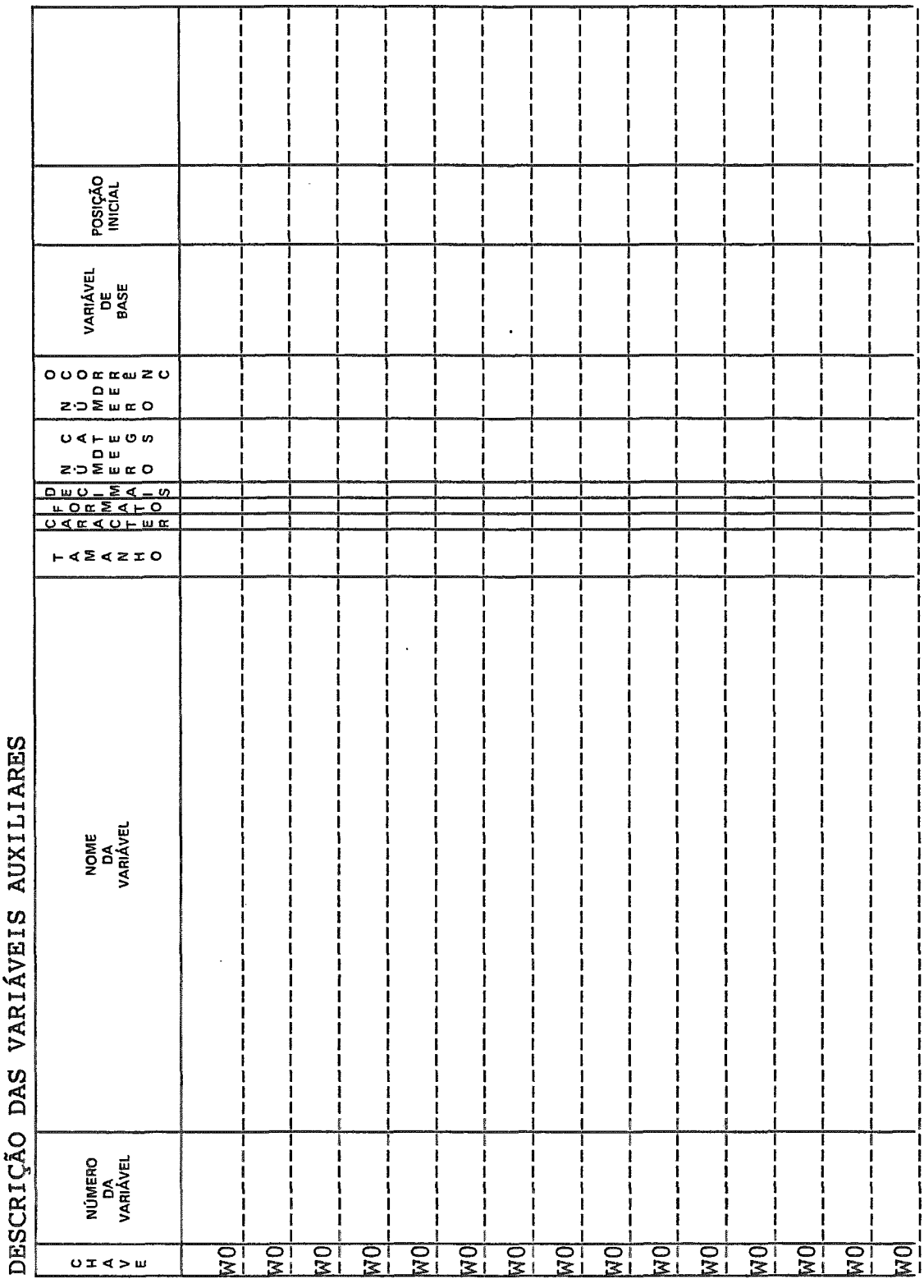


FIGURA A.5 - Formulário para descrição das variáveis auxiliares

ANEXO B

MANUAL DA LINGUAGEM "CRIPTA"

A linguagem Cripta é uma ferramenta que compõe o sistema Cripta, para especificação dos procedimentos da aplicação.

A partir de especificação do Dicionário da Aplicação, através da ferramenta DICAPLIC, que contém a descrição dos arquivos a serem processados, da especificação, opcionalmente de um Plano de Crítica, através da ferramenta PCAUT, e da especificação dos procedimentos em linguagem Cripta, o sistema gera um programa em linguagem PLI com a seguinte estrutura:

```
DECLARAÇÕES
INICIALIZAÇÃO
LEITURA DO REGISTRO DO ARQUIVO DE ENTRADA
DO WHILE ( not FIM DO ARQUIVO DE ENTRADA)
   EXECUTA PROCEDIMENTOS PARA O REGISTRO
   GRAVAÇÃO DO REGISTRO NO ARQUIVO DE SAÍDA (OPCIONAL)
   LEITURA DO REGISTRO DO ARQUIVO DE ENTRADA
END
FINALIZAÇÃO
```

Este manual se encontra dividido em três grandes partes. A primeira descreve a estrutura da linguagem, como os tipos de variáveis, constantes, e operadores permitidos, e os blocos estruturais que compõem esta linguagem.

A segunda parte descreve as instruções da linguagem, isto é, o conjunto de comandos permitidos da linguagem.

A terceira parte descreve os procedimentos automatizados da linguagem, isto é, os que possuem funções de processamento já pré-definidos.

B.1 - ESTRUTURA DA LINGUAGEM

A linguagem "CRIPTA" tem um formato livre, e a sua estrutura de especificação dos comandos se assemelha ao estilo de pseudo-código.

B.1.1 - CONSTRUTORES DA LINGUAGEM

Os construtores da linguagem são:

VARIÁVEIS : Começam com um símbolo (letra) seguido de até sete dígitos. O símbolo usado no início de uma variável identifica a que arquivo ela pertence.

No módulo de especificação do Dicionário da Aplicação podem ser especificados até 10 arquivos para processamento: um do processamento principal e até 9 arquivos de referência, que são usados de modo paralelo ao arquivo principal.

Assim, temos os seguintes símbolos iniciais de identificação para as variáveis:

V - variável do arquivo principal;

A - variável do arquivo de referência 1;

B - variável do arquivo de referência 2;

C - variável do arquivo de referência 3;

D - variável do arquivo de referência 4;

E - variável do arquivo de referência 5;

F - variável do arquivo de referência 6;

G - variável do arquivo de referência 7;

H - variável do arquivo de referência 8;

I - variável do arquivo de referência 9;

W - variável auxiliar;

S - variável do arquivo de principal que foi salva através de uma instrução SALVE.

CONSTANTES NUMÉRICAS : Formadas por até dezesseis dígitos, sinalizados ou não, incluindo decimais.

IDENTIFICADORES : Formados pelo símbolo inicial % seguido de até 35 caracteres alfanuméricos podendo incluir o caracter especial "_" (underscore).

CONSTANTES LITERAIS : Formadas por um conjunto de caracteres alfanuméricos de até 256 caracteres delimitados por ' (plics) ou " (aspas).

OPERADORES RELACIONAIS :

= , IGUAL , IGUAL A

> , MAIOR , MAIOR QUE

< , MENOR , MENOR QUE

<> , ^= , NAO IGUAL , NAO IGUAL A , DIFERENTE, DIFERENTE DE

^> , NAO MAIOR , NAO MAIOR QUE

^< , NAO MENOR , NAO MENOR QUE

>= , MAIOR OU IGUAL , MAIOR OU IGUAL A

<= , MENOR OU IGUAL , MENOR OU IGUAL A

OPERADORES LÓGICOS :

& , E

| , OU

OPERADORES DE NEGAÇÃO :

^ , NAO

OPERADORES ARITMÉTICOS :

+ , - , * , / , ** , (,)

COMENTÁRIO :

/* Xxxxxxxxxxxxx */

B.1.2 - BLOCOS ESTRUTURAIS DA LINGUAGEM

Um programa Cripta é formado por blocos de estruturas e as instruções da linguagem só podem ser especificadas dentro destes blocos. Existem três blocos de estruturas básicas na linguagem Cripta. A primeira incorpora a estrutura de um programa, a segunda as estruturas de

rotinas, que são procedimentos a serem executados num programa somente quando da execução de uma instrução que a comande e as estruturas de procedimentos a serem executados na sequência em que são especificados no programa.

Um programa Cripta deve ser formado pela sequência de estruturas mostradas na figura B.1. As estruturas especificadas entre parênteses "()" significam que são opcionais num programa, as estruturas entre chaves "[]" significam que pelo menos uma das estruturas declaradas em seu conteúdo é obrigatória no programa e as estruturas seguidas de um símbolo de elevado a asterísco significa que podem ser repetidas.

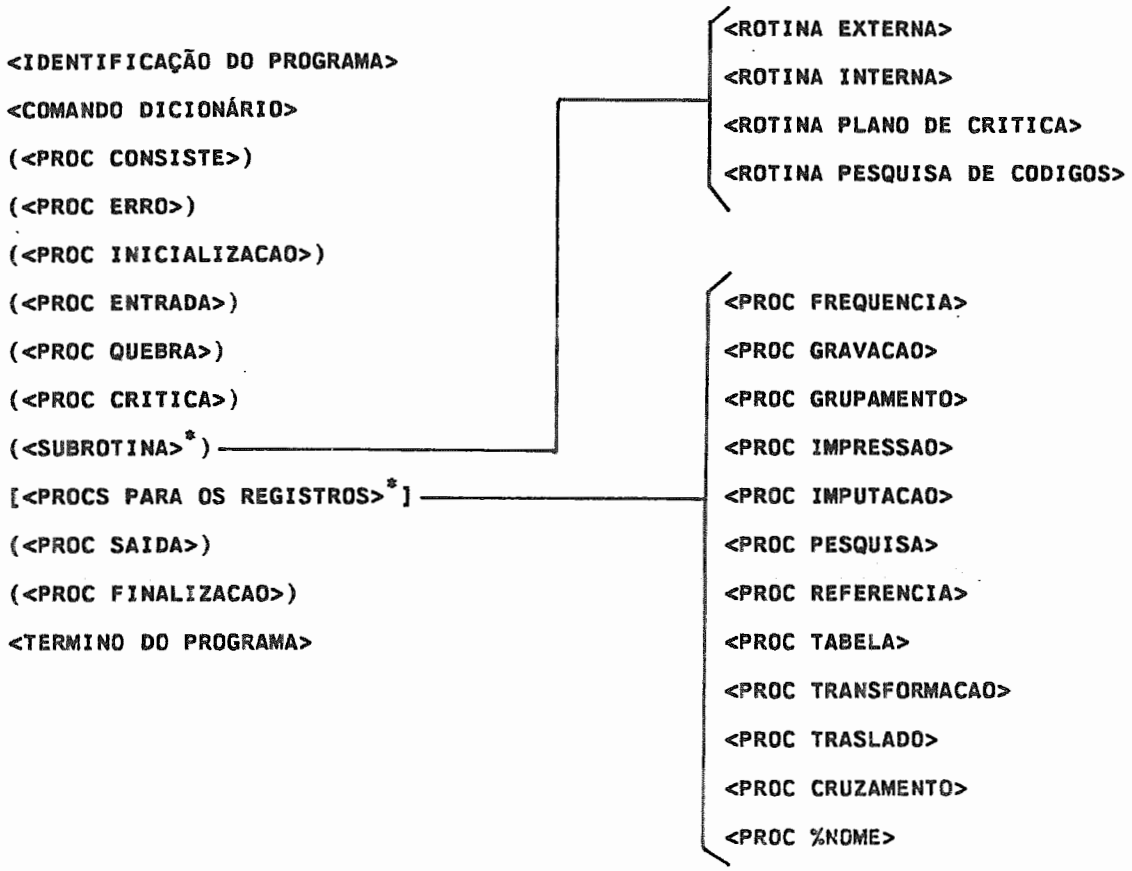


FIGURA B.1 - Esquema da estrutura de um programa Cripta

B.2 - DECLARAÇÃO DE UM PROGRAMA

Um programa consiste de um bloco externo formado pelos comandos de declaração do programa.

```
PROGRAMA "XXXXXXX" ;
        ---
        ---      programa
        ---
FIM PROGRAMA ;
```

onde "XXXXXXX" identifica o nome do programa e deve ser constituído de até 7 caracteres alfanuméricos.

Este nome será usado no código gerado, na declaração da identificação do programa, e também, nos títulos dos relatórios emitidos pelo programa gerado.

B.3 - BLOCOS DE PROCEDIMENTOS

O corpo de um programa "CRIPTA" é formado por blocos de procedimentos delimitados pelos comandos PROC "nome" ; FIM "nome" ; e as intruções da linguagem só podem ser declaradas dentro destes blocos. Assim em cada programa tem de haver pelo menos um bloco de procedimento.

É permitido um aninhamento de blocos de procedimentos, isto é, um bloco de procedimento, pode consistir de um ou mais blocos de procedimentos, neste caso considerados como subprocedimentos, que por sua vez podem consistir de blocos de procedimentos mais internos. Cabe ressaltar que este aninhamento de blocos de procedimentos deve respeitar as regras da gramática da linguagem quanto à hierarquia permitida para especificação de cada um destes blocos.

Existem três tipos de blocos de procedimentos que formam esta linguagem: os blocos de procedimentos que servem apenas para estruturação do programa, servindo assim para sua documentação, consistindo dos comandos:

```
PROC "identificador" ;
        -
        -
        -  instruções da linguagem
        -
        -
FIM "identificador" ;
```

Dentro deste bloco podem ser especificadas instruções simples da linguagem ou outros blocos de procedimentos, desde que respeitada suas hierarquias. Este tipo de bloco de procedimentos pode ser especificado em qualquer ponto do programa, e dentro de qualquer outro bloco que permita especificação de procedimentos da linguagem.

Um outro tipo de bloco de procedimentos que pode ser definido nesta linguagem, são os blocos de procedimentos automatizados, isto é, procedimentos que geram código no programa gerado para funções pré-definidas.

Estes blocos de procedimentos são representados de modo análogo aos blocos de procedimentos de estruturação, definidos anteriormente, sendo identificados pelo nome que tem de ser uma palavra reservada para os nomes de procedimentos automatizados. Alguns destes blocos de procedimentos exigem a especificação de uma lista de parâmetros quando da especificação de seu nome e os tipos de instruções que podem ser especificados dentro de seus blocos varia de procedimento para procedimento. Todas estas características serão descritas adiante, na descrição de cada bloco de procedimentos automatizados.

Uma outra característica destes blocos de procedimentos automatizados é o nível da hierarquia em que podem ser especificados, o que varia também de procedimento para procedimento.

Existe ainda um procedimento com característica especial, nesta linguagem, que é o PROC PLI ; ------ FIM PLI ;.

Este não é um bloco de procedimemto usado somente para estruturação do programa, nem gera código para alguma função pré-definida. O uso deste bloco de procedimentos indica que o seu conteúdo dentro do bloco são instruções de PLI. Quando é encontrado um procedimento deste tipo, seu conteúdo é transcrito diretamente para o programa gerado.

Este tipo de procedimento é usado para podermos especificar algumas situações não previstas pelo sistema, e pode ser usado em qualquer ponto do programa e dentro de qualquer outro bloco de procedimentos que permita especificação de procedimentos da linguagem. Ele não pode ser formado por nenhum outro bloco de procedimentos mais interno, podendo seu conteúdo ser formado apenas por instruções da linguagem PLI.

```
PROC PLI ;
        ---
        ---     instruções da linguagem PLI
        ---
FIM PLI ;
```

B.4 - INSTRUÇÕES DA LINGUAGEM

Serão descritas neste item as instruções da linguagem.

B.4.1 - COMANDOS DE ATRIBUIÇÃO

Um comando de atribuição tem o seguinte formato:

'Variável' = <expressão> ;

onde <expressão> pode ser uma expressão aritmética ou uma função. Usando a notação de Regras de reescrita podemos especificá-la como:

<Expressão> --> (<expressão>)
| <expressão> <op. aritmético> <expressão>
| "variável"
| "constante"
| DIGITO10 ("lista de variáveis")
| DIGITO11 ("lista de variáveis")
| SOMAVAR ("variável" A "variável")
| TOTAL ("VARIAVEL")
| TOTAL ("VARIAVEL" PARA <condição>)
| MENORVALOR ("VARIAVEL")
| MENORVALOR ("VARIAVEL" PARA <condição>)
| MAIORVALOR ("VARIAVEL")
| MAIORVALOR ("VARIAVEL" PARA <condição>)
| FREQUENCIA (<condição>)

As funções DIGITO10 e DIGITO11 calculam o DV módulo 10 e 11 respectivamente, de uma variável ou uma lista de variáveis concatenadas, especificadas como parâmetros da função.

A função SOMAVAR calcula o somatório de uma lista de variáveis especificadas como parâmetros da função. Estas variáveis que compõem a lista devem pertencer ao mesmo registro.

A função TOTAL calcula o somatório vertical de uma variável, isto é, acumula a soma desta variável em todos os registros do mesmo tipo que formam um grupo pelo qual o arquivo esteja sendo processado. Como exemplo podemos citar os registros repetidos do mesmo tipo dentro de um questionário. Para o caso de não se desejar todos os registros que compõem este grupo, pode ser especificada uma condição para seleção dos registros a serem totalizados. Esta condição é especificada de modo análogo à condição especificada nos comandos condicionais.

As funções MAIORVALOR e MENORVALOR calculam o maior valor ou menor valor de uma variável entre os registros que contenham esta variável, dentro de um grupo de registros armazenados pela quebra indicada no processamento. Opcionalmente pode ser especificada uma condição para seleção destes registros.

A função FREQUENCIA fornece o número de registros, dentro do grupo de quebra armazenado, que atendem a condição especificada.

B.4.2 - COMANDOS CONDICIONAIS

Existem dois tipos de comandos condicionais:

```
SE <condição>
  ENTAO
       <Linstruções>
FIM SE ;
```

ou

```
SE <condição>
  ENTAO
       <Linstruções>
  SENAO
       <Linstruções>
FIM SE ;
```

<condição> representa uma expressão condicional e podemos especificá-la como:

```
<Condição> --> "op. Negação" <condição>
          | ( <condição> )
          | <condição> "op. lógico" <condição>
          | "variável" = NUMERICO
          | "variável" = INVALIDO
          | <expressão> "op. relacional" <expressão>
          | <função booleana>
```

| <função lógica> "op. relacional" <expressão>

| <expressão> "op. relacional" <função lógica>

| INTERVALO ("variável", Termo1 , Termo2)

A expressão condicional 'variável' = NUMERICO e 'variável' = INVALIDO fornecem valores booleanos do tipo verdadeiro ou falso. A primeira verifica se o conteúdo da variável é um valor numérico, e a segunda verifica se o conteúdo da variável corresponde a alguma das categorias especificadas para a variável no dicionário.

As funções booleanas fornecem como retorno um valor verdadeiro ou falso e podem ser especificadas como:

<Função booleana> --> PRIMEIRO

| ULTIMO

| CORRENTE

| ANTERIOR

| POSTERIOR

| 'nome ROTPESQ'

A função PRIMEIRO pode ser usada em qualquer bloco de procedimentos que permita especificação de instruções e desde que tenha sido especificado um procedimento QUEBRA no programa. Seu valor de retorno indica se o registro sendo processado correntemente no programa, é o primeiro registro da tabela de quebra.

A função ULTIMO é análoga à PRIMEIRO indicando se o registro sendo processado é o último da tabela de quebra.

As funções CORRENTE, ANTERIOR e POSTERIOR só podem ser usadas dentro de um procedimento PESQUISA.

A função CORRENTE fornece um valor verdadeiro se o registro que está sendo examinado correntemente dentro do loop de varredura da tabela de quebra do procedimento PESQUISA é igual ao que está sendo processado pelo loop geral do programa, ou seja, o mesmo. Em caso contrário seu valor será falso.

A função ANTERIOR fornece um valor verdadeiro, se o registro que está sendo examinado correntemente dentro do loop de varredura da tabela de quebra do procedimento PESQUISA, é o imediatamente anterior na tabela ao registro que está sendo processado pelo loop geral do programa.

A função POSTERIOR é análoga à ANTERIOR sendo que o registro considerado é o imediatamente posterior.

A especificação de um nome de uma rotina de pesquisa, ROTPESQ especificada anteriormente no programa, serve para a chamada desta rotina, e seu valor é verdadeiro quando o código a pesquisar foi encontrado e falso em caso contrário.

A <função lógica> serve para os casos em que desejamos testar uma condição sobre um conjunto de variáveis ou valores e pode ser definida como:

<Função lógica> --> TODAS (Vnnnn, Vnnnn, ...)
"Const.""const."
| QUALQUERUMA (Vnnnn, Vnnnn, ...)
"Const." "const."
| APENASUMA (Vnnnn, Vnnnn, ...)
"Const." "const."

A função TODAS fornece um valor verdadeiro se TODAS as variáveis especificadas na lista de variáveis obedecerem a condição especificada.

Exemplo:

```
SE TODAS (V0010, V0020, V0030) = '1'
  ENTAO
       ----

SE V010 DIF TODAS ( 1 , 2 , 3 )
  ENTAO
       ----
```

A condição será verdadeira somente se o conteúdo da variável V0010 for igual a '1', o da variável V0020 for igual a '1', e o da variável V0030 também for igual a '1'.

No segundo caso a condição será verdadeira se o conteúdo da variável V010 for diferente dos valores 1, 2, e 3.

Esta função equivale à função lógica E (AND).

A função QUALQUERUMA fornece um valor verdadeiro se QUALQUER UMA das variáveis especificadas na lista de variáveis obedecem à condição especificada.

Exemplo:

```
SE QUALQUERUMA (V0010, V0020, V0030) = '1'
  ENTAO
       ----

SE V010 = QUALQUERUMA ( 1 , 2 , 3 )
  ENTAO
       ----
```

A condição será verdadeira para o conjunto de valores das variáveis, onde X representa um valor diferente de '1'.

V0010	V0020	V0030
1	X	X
1	X	1
1	1	X
1	1	1
X	1	X
X	1	1
X	X	1

A condição será falsa apenas para o caso dos valores das variáveis V0010, V0020, V0030 serem todos diferentes de '1'.

No segundo caso a condição será verdadeira, se o conteúdo da variável V010 for igual ao valor 1, ou 2, ou 3.

Esta função equivale à função lógica OU (OR).

A função APENASUMA fornece um valor verdadeiro se SOMENTE uma das variáveis especificadas na lista de variáveis obedecer à condição especificada.

Exemplo:

```
SE APENASUMA (V0010, V0020, V0030) = '1'
  ENTAO
       ----
```

A condição será verdadeira para o conjunto de valores das variáveis, onde X representa um valor diferente de '1'.

V0010	V0020	V0030
1	X	X
X	1	X
X	X	1

e será falsa para o conjunto restante de valores das variáveis.

Esta função equivale à função lógica OU EXCLUSIVO (EXCLUSIVE OR).

A função INTERVALO fornece como resultado um valor booleano verdadeiro ou falso, dependendo da variável especificada pertencer ou não ao intervalo estipulado pelos valores declarados em TERMO1 e TERMO2.

TERMO1 pode ser uma variável ou uma constante e é considerado como o limite inferior do intervalo aberto.

TERMO2 pode ser uma variável ou uma constante e é considerado como o limite superior do intervalo aberto.

Exemplo: SE INTERVALO (V010, 2 , 4)

corresponde à expressão condicional:

SE V010 > 2 E V010 < 4

Se quisermos fechar os limites do intervalo basta colocarmos o sinal de igual antes do termo correspondente desejado, ou em ambos.

Exemplo:

```
SE INTERVALO(V010,=2,4) --> SE V010 >= 2 E V010 < 4
SE INTERVALO(V010,2,=4) --> SE V010 > 2 E V010 <= 4
SE INTERVALO(V010,=2,=4) --> SE V010 >= 2 E V010 <= 4
```

B.4.3 - COMANDOS IMPERATIVOS

DESPREZE ;

Este comando indica que o registro do arquivo de entrada não deve passar para os procedimentos restantes especificados no programa. Isto equivale a um desvio para a leitura de um novo registro do arquivo de entrada.

EXECUTE "identificador" ;

Este comando executa uma chamada para uma rotina cujo nome tenha sido especificado anteriormente no programa como rotina externa, ROTEXT, ou como rotina interna, ROTINT.

LEIA X ;

Este comando executa uma leitura de um registro do arquivo de REFERÊNCIA identificado pelo número X.

IMPUTE "identificação" ;

Este comando executa uma atualização de uma variável a partir de uma matriz de imputação declarada na rotina IMPUTACAO, item B.5.3, identificada pelo campo "identificação" formado por uma constante numérica de até três dígitos.

ATUALIZE "identificação" ;

Este comando faz com que uma matriz de imputação declarada na rotina IMPUTACAO, identificada pelo campo "identificação" formado por uma constante numérica de até três dígitos seja atualizada de acordo com os valores das variáveis correspondentes do registro corrente.

SALVE ("vaiável", "variável", ...) ;

Este comando deve ser usado quando houver necessidade de comparação de variáveis inter-registros, onde para cada variável usada nesta instrução seja criada uma variável pelo sistema automaticamente, guardando seu valor e a identificação do registro que está sendo processado no momento de execução desta instrução.

COPIAPLANO ("string", "string" A "string", ...) ;

Este comando deve ser usado quando o programa Cripta contiver um Plano de Crítica associado. Sua função é incluir no ponto do programa em que for especificada esta instrução as críticas do Plano de Crítica com Referência igual à lista de referências especificadas neste comando. Quando tivermos um intervalo de críticas do Plano de Crítica a serem incluídas.

B.5 - USO DE ROTINAS NA LINGUAGEM CRIPTA

A linguagem Cripta permite quatro tipos de especificação de rotinas para um programa descritas a seguir.

B.5.1 - ROTINAS EXTERNAS - ROTEXT

Às vezes temos a necessidade de termos um trecho do programa declarado à parte, rotina externa, que não precise da passagem dos registros lidos, podendo, inclusive, ser escrito em outra linguagem. Assim, podemos declarar esta rotina como:

```
ROTEXT "identificador" ;
ROTEXT "identificação" (Vnnnn, Vnnnn,.....) ;
```

Os valores a serem passados para esta rotina podem ser declarados como variáveis auxiliares com característica de escopo externo, no Dicionário da Aplicação, ou especificados na lista de parâmetros associada à rotina.

Para execução desta rotina, deve ser usado o comando:

```
EXECUTE "identificador" ;
```

B.5.2 - ROTINAS INTERNAS - ROTINT

Em certos casos, temos um trecho determinado do programa que queremos repetir em vários lugares do programa.

Ao invés de especificá-lo repetidas vezes, podemos, no início do programa especificá-lo dentro de uma rotina interna, constituida de um bloco análogo ao procedimento formada pelos comandos:

```
ROTINT "identificador" ;
        ---
        --- instruções da linguagem
        ---
FIM ROTINT ;
```

Dentro deste bloco podemos podemos especificar instruções e blocos de procedimentos da linguagem desde que a hierarquia de especificação seja respeitada.

A diferença entre um procedimento, (bloco PROC), e uma rotina interna, (bloco ROTINT), é que o procedimento é executado na ordem em que foi especificado, e a rotina interna quando referenciada por uma instrução EXECUTE num ponto qualquer do programa.

B.5.3 - ROTINAS DE PESQUISA DE CÓDIGOS - ROTPESQ

Às vezes temos um conjunto de códigos a serem pesquisados num programa, que por serem numerosos demais para serem declarados como categorias num dicionário das variáveis, ou por poderem sofrer alterações durante o tempo em que o programa esteja em produção, são normalmente armazenados em arquivos (tabelas externas) que são consultados pelo programa de crítica.

Assim podemos criar uma rotina externa, para fazer a pesquisa dos códigos recebidos como parâmetros, e enviar como resposta uma indicação se achou ou não o código pesquisado.

Este tipo de rotina externa deve ser declarada de modo diferente à rotina externa, citada anteriormente, com a forma:

```
ROTPESQ "identificador" ( Vnnnn, Vnnnn ) ;
```

Esta rotina tem que ter pelo menos um parâmetro, representando o código a ser pesquisado, e no máximo mais um parâmetro, representando um código a ser retornado associado ao código pesquisado, caso este tenha sido encontrado.

A rotina externa associada para a pesquisa dos códigos deve ser declarada com os seguintes parâmetros:

PARM1: correspondendo a um "flag" de indicação se o código procurado foi ou não achado.

PARM2: correspondendo ao código a ser pesquisado.

PARM3: este parâmetro só deve ser declarado se na especificação da ROTPESQ no programa Cripta foi feita a declaração do segundo parâmetro correspondente ao código a ser retornado em associação ao código pesquisado.

Uma chamada para esta rotina é referenciada dentro de uma expressão condicional com a especificação de seu nome, funcionando como uma função booleana, sendo seu valor verdadeiro se o código a pesquisar foi encontrado, e falso em caso contrário.

```
PROGRAMA "PGM1" ;
  ---
  ROTPESQ %CODIGOS ( V0050, V0150 ) ;
  ---
      SE %CODIGOS
        ENTAO
           ---
```

B.5.4 - ROTINA ASSOCIADA AO PLANO DE CRÍTICA - ROTPLAN

Esta rotina só poderá ser usada quando o programa seja executado com um Plano de Crítica associado.

A função desta rotina é permitir ao programador a especificação dos procedimentos para uma crítica, pertencente ao Plano de Crítica, que tenha sido marcada como descrita, isto é, o usuário não conseguiu especificá-la através dos comandos permitidos. Assim, o programador deve especificar uma rotina identificada pelo bloco ROTPLAN "referência da crítica associada" que será executada no processamento da crítica correspondente no Plano de Crítica.

A referência da ROTPLAN deve ser formada pelo valor do campo Referência da crítica do Plano de Crítica associado, para o caso do campo que esteja descrito ser o Causa/Procedimento podendo-se, neste caso, usar a instrução EXECUTE EFEITO SIM ou NAO, para execução dos efeitos especificados no Plano de Crítica do campo Efeito.

Se os procedimentos a serem especificados forem de um campo Efeito de um Plano de Crítica, a identificação da ROTPLAN deve ser formada pelo conteúdo do Referência da crítica do Plano de Crítica associada com o sufixo "S" ou "N" conforme o Efeito "SIM" ou "NAO" desejado.

B.5.5 - ROTINA IMPUTAÇÃO

Uma variável de controle pode ser imputada (receber um novo valor) quando não é satisfatória por ser ignorada ou porque está inconsistente com os valores das outras variáveis de controle. Quando o valor desta variável pode ser deduzido a partir de valores específicos das variáveis de controle, podemos usar os procedimentos GRUPAMENTO e TRANSFORMAÇÃO para realizar a atribuição de um novo valor à variável. Porém, existem casos em que os valores das variáveis de controle não podem ser especificados e, para a atribuição de um valor correto à variável a ser imputada, os valores das variáveis de controle são usados para montar uma matriz em que cada célula contém o valor a ser atribuído à variável a imputar, que corresponde ao seu valor encontrado no último registro com os mesmos valores das variáveis de controle.

Para inicialização desta matriz, os valores a imputar devem ser estabelecidos de acordo com os valores obtidos em experiências anteriores.

```
IMPUTACAO "ident" (Viiii, Vcccc, ...., Vcccc) ;
                  'número' 'número' .... 'número'
                  'número' 'número' .... 'número'
                     ---      ---    ....    ---
                  INICIALIZE ;
FIM IMPUTACAO ;
```

Como podemos ter mais de uma variável a imputar, o bloco de procedimento da rotina IMPUTAÇÃO, é usado o parâmetro "ident", formado por um número de 001 a 999, para a identificação da matriz correspondente ao procedimento.

A primeira variável especificada representa a variável a ser imputada, isto é, que receberá um novo valor.

As demais variáveis representam as variáveis de controle e podem ser até um número de quatro variáveis de controle, cada uma representando uma dimensão da matriz.

Dentro do bloco devem ser especificados conjuntos de valores que representarão os valores com que será inicializada a matriz, cada um correspondendo às variáveis especificadas como parâmetros.

Os valores das variáveis de controle devem ser sequenciais, pois cada um deles representa um índice da matriz. Para representarmos um valor para as variáveis de controle que sejam menores ou maiores aos limites especificados, podemos usar o valor INV .

Assim, se especificarmos um procedimento:

```
IMPUTACAO 009 (V005, V053, V027) ;
                        3     1     1
                        2     1     2
                        1     1   INV
                        2     2     2
                        3     2   INV
FIM IMPUTACAO ;
```

Será montada uma matriz com identificação 009 tendo como variável a ser imputada a V005 e como variável de controle a V053 e V027.

		V027		
	V005	1	2	INV
V053	1	3	2	1
	2	-1	2	3
	INV	-1	-1	-1

Se as duas últimas variáveis de controle forem iguais, os seus valores respectivos representam um intervalo.

Opcionalmente, podemos não querer que os valores iniciais da matriz sejam especificados dentro do procedimento, mas sim lidos de um arquivo de matrizes de imputação IMPE e ao final do programa a matriz de imputação seja gravada num arquivo de matrizes de imputação de saída IMPS. Para isto, dentro do bloco de procedimento ao invés de especificarmos os valores para inicialização da matriz, usamos a instrução INICIALIZE.

Quando da execução do procedimento IMPUTAÇÃO, são somente geradas declarações para definição da matriz e estipulação de seus valores iniciais e, por isto, ele é tratado como uma rotina.

No programa restante o uso de uma instrução IMPUTE 999 faz com que seja realizada a atribuição de um valor à variável a ser imputada, de acordo com os valores das variáveis de controle associadas. O uso de uma instrução ATUALIZE faz com que os valores das células da matriz sejam atualizados, de acordo com os valores das variáveis de controle encontrados no registro processado.

B.6 - BLOCOS DE PROCEDIMENTOS AUTOMATIZADOS

Serão descritos aqui os blocos de procedimentos automatizados da linguagem agrupados por suas características e finalidades.

B.6.1 - PROCEDIMENTO PARA CONDICAO DE ERRO - PROC ERRO

Este bloco de procedimentos deve ser usado quando se desejar substituir o procedimento da cláusula "ON-ERROR" do programa PLI gerado pelo sistema, que consiste na impressão do último registro lido.

Se este bloco de procedimento for especificado, todas as variáveis índice que estiverem fora dos limites são indicadas automaticamente.

```
PROC ERRO ;
     LISTA = Vnnnn, Vnnnn,Vnnnn, ..... ;
     PROC PLI ;
        ---
     FIM PLI ;

FIM ERRO ;
```

O comando LISTA serve para se especificar uma lista de variáveis a serem impressas caso haja erro na execução do programa gerado.

O bloco PROC PLI ; ------ FIM PLI ; serve para especificação de instruções a serem efetuados dentro do bloco "ON ERROR" do PLI.

Estes dois comandos são opcionais entre si, mas se declarado o procedimento PROC ERRO, pelo menos um deles tem que ser usado.

B.6.2 - PROCEDIMENTO PARA INICIALIZAÇÃO DE VARIÁVEIS - PROC INICIALIZACAO

Este procedimento serve para se especificar as instruções a serem executados no programa gerado uma única vez antes de entrar no loop de execução dos registros do arquivo do programa.

```
PROC INICIALIZAÇÃO ;
     -----
     -----
FIM INICIALIZACAO ;
```

As instruções permitidas neste bloco são o comando de atribuição e o bloco de procedimentos PROC PLI.

B.6.3 - PROCEDIMENTO PARA LEITURA DO ARQUIVO DE ENTRADA - PROC INICIALIZACAO

O bloco de procedimento entrada é usado para alterarmos a geração automática de leitura do arquivo de entrada, que consiste da leitura sequencial do registro. Este bloco de procedimentos pode ser declarado de duas formas.

```
PROC ENTRADA ;
      ---
      ---  instruções "CRIPTA" para seleção de
      ---  registros com comandos DESPREZE
      ---
FIM ENTRADA ;
```

Este procedimento é usado para indicar que os registros do arquivo de entrada serão lidos um a um e passarão pelo procedimento de seleção de registros especificados neste bloco.

Caso seja executado um comando DESPREZE para o registro, ele será desprezado, isto é, não será processado pelo resto do programa e será lido um novo registro do arquivo de entrada, até que seja aceito ou se chegue ao final do arquivo.

```
PROC ENTRADA (PLI) ;

      ---
      ---  comandos PLI
      ---

FIM ENTRADA ;
```

Este procedimento é usado para se indicar que a leitura do arquivo de entrada não deve ser gerada automaticamente e será definida dentro deste bloco, em código PLI. Um exemplo para uso deste tipo de procedimento é a leitura de um arquivo VSAM.

B.6.4 - PROCEDIMENTOS DE CONTROLE DE EXECUÇÃO

B.6.4.1 - PROC QUEBRA

O bloco de procedimento QUEBRA é usado para especificarmos que o arquivo de entrada não será processado registro a registro, mas um grupo de registros de cada vez. Serve para os casos em que desejamos fazer acumulações de dados para grupos de registros ou comparar os valores de um registro com outro do mesmo grupo.

```
PROC QUEBRA ;
   CHAVE = Vnnnn, .....,Vnnnn ;
   TAMANHO = 99999 ;
   CALCULE TOTAL (Vnnnn EM Wnnnn) ;
   CALCULE TOTAL (Vnnnn EM Wnnnn) PARA "condição" ;
   CALCULE TOTAL (Vnnnn EM Wnnnn) PARA "condição" ;
                  COM FREQUENCIA EM Wnnnn ;
   CALCULE FREQUENCIA EM Wnnnn "condição" ;
   CALCULE MAIORVALOR (Vnnnn EM Wnnnn) ;
   CALCULE MAIORVALOR (Vnnnn EM Wnnnn) PARA "condição" ;
   CALCULE MENORVALOR (Vnnnn EM Wnnnn) ;
   CALCULE MENORVALOR (Vnnnn EM Wnnnn) PARA "condição" ;
FIM QUEBRA ;
```

Quando este procedimento é especificado, os registros são lidos e guardados numa tabela em memória até que haja uma quebra da chave que identifica a quebra especificada no comando CHAVE que pode ser formada por um conjunto de até 5 variáveis pertencentes ao registro de entrada. O tamanho máximo de registros a serem guardados na tabela em memória é especificado no comando TAMANHO = número de registros.

Às vezes existe a necessidade de calcularmos o total de algumas variáveis dentro de cada quebra. Para isso, basta especificarmos o comando CALCULE TOTAL com um conjunto de pares de variáveis Vnnnn EM Wnnnn onde a primeira especifica a variável do registro de entrada a ser acumulada e a segunda especifica a variável auxiliar que receberá a acumulação.

Se esta acumulação é para ser efetuada para todos os registros da quebra, basta apenas especificarmos esta parte do comando. Mas podem existir casos em que só desejamos esta acumulação para certos registros que obedeçam a certas condições. Para isto basta especificarmos a condição de seleção dos registros. Se for necessário armazenar o número de registros que foram selecionados por esta condição basta completar o comando especificando a variável auxiliar que receberá este total.

Em certos casos precisamos apenas de uma frequência do número de registros dentro de uma quebra que obedeça a certas condições. Assim, basta especificarmos um comando de CALCULE FREQUENCIA especificando a variável a receber a frequência dos registros que obedeçam à condição especificada.

Pode ocorrer, também, a necessidade de sabermos o menor ou o maior valor de uma variável dentro de um conjunto de registros. Para isto, basta especificarmos o comando CALCULE MAIORVALOR ou CALCULE MENORVALOR, especificando a variável auxiliar a receber este valor e, caso necessário, a condição para os registros a serem testados.

B.6.4.2 - PROC PESQUISA

Este bloco de procedimento só pode ser usado se tiver sido especificado um procedimento QUEBRA no programa, e

serve para se fazer uma pesquisa dos registros armazenados na tabela de QUEBRA.

```
PROC PESQUISA ;
     -----
     -----
FIM PESQUISA ;
```

Quando o programa entra num bloco de procedimento PESQUISA, ele guarda o registro que está sendo processado correntemente e entra num loop de processamento de registro a registro da tabela de QUEBRA. As variáveis associadas ao registro do arquivo de entrada são agora associadas ao registro da tabela.

Este loop é executado até que sejam processados todos os registros da tabela ou seja executada uma instrução RESTAURE que faz com que o loop termine sem chegar ao último registro da tabela. Se esta instrução não for especificada ou não for executada, o loop termina automaticamente quando processado o último registro da tabela.

Algumas vezes precisamos verificar os demais registros contidos numa tabela, mas só alguns determinados registros que obedeçam a certas condições. Neste caso podemos especificar o procedimento PESQUISA juntamente com as condições para seleção dos registros a serem processados da tabela de QUEBRA.

```
PROC PESQUISA ( "condição" ) ;
      -----
      -----
FIM PESQUISA ;
```

Pode ocorrer também a necessidade de querermos pesquisar dentro da tabela de QUEBRA, não um grupo de registros, mas apenas o registro imediatamente anterior ou posterior, ao registro que está sendo processado pelo loop principal do programa. Neste caso podemos especificar:

```
PROC PESQUISA (ANTERIOR) ;
      -----
      -----
FIM PESQUISA ;

PROC PESQUISA (POSTERIOR) ;
      -----
      -----
FIM PESQUISA ;
```

Neste tipo de procedimento as instruções declaradas dentro do bloco são executadas uma única vez, não tendo sentido, assim, o uso da instrução RESTAURE para término do loop, como também das funções CORRENTE, ANTERIOR e POSTERIOR.

Dentro de uma PROC PESQUISA às vezes temos necessidade de compararmos uma variável do registro que está sendo

processado pelo loop principal do programa com uma variável do registro da Tabela de Quebra sendo pesquisado ou compararmos valores de variáveis de registros diferentes dentro da tabela de Quebra.

Sempre que quisermos fazer a comparação de variáveis do registro sendo processado pelo programa com um registro da tabela de Quebra, devemos antes de entrar no procedimento PESQUISA usar a instrução SALVE (Vnnnn, Vnnnn, ...) onde especificamos as variáveis que serão comparadas dentro do procedimento PESQUISA com o registro da tabela de Quebra.

Se nos referirmos simplesmente ao nome da variável o programa vai associar ao registro que está sendo lido da tabela de Quebra. Para diferenciarmos as variáveis especificamos as variáveis que foram salvas pelo comando SALVE usando ao invés do prefixo 'V', o prefixo 'S'.

B.6.5 - PROCEDIMENTO PARA LEITURA DE ARQUIVOS PARALELOS - PROC REFERENCIA

Às vezes num processamento de um programa precisamos de ler mais de um arquivo, além do arquivo de entrada ARQE. Como exemplo podemos citar o caso, de uma verificação entre dois arquivos.

Para estes casos podemos especificar estes arquivos como arquivos de Referência no Dicionário da Aplicação. O controle da leitura destes arquivos pode ser comandada

através de instruções LEIA (item xxxxxx), que lê o próximo registro de um arquivo de Referência, ou através do bloco de procedimentos descrito a seguir.

```
PROC REFERENCIA N  ;

        Vcccc = Vrrrr ;
          --      --
          --      --

FIM REFERENCIA ;
```

"N" identifica o arquivo de Referência a ser lido.

Cada par Vcccc = Vrrrr identifica uma condição de controle para a leitura do arquivo. Vcccc representa a variável de controle, que pode ser uma variável do arquivo de entrada, uma variável auxiliar ou uma constante.

O código gerado para este bloco de procedimento fará com que o arquivo de Referência seja automaticamente lido enquanto os valores das variáveis de controle sejam menores aos das variáveis do arquivo Referência, isto é, quando o loop de leitura terminar, os valores das variáveis do arquivo de Referência serão sempre iguais ou maiores aos valores das variáveis de controle.

Quando ocorre o fim de arquivo de Referência, são movidos 99999999 para todas as suas variáveis.

B.6.6 - PROCEDIMENTOS PARA CRIAÇÃO DE DADOS

B.6.6.1 - PROC GRUPAMENTO

Este procedimento é usado para se fazer a atribuição de novos valores a uma variável dependendo dos intervalos de valores de uma outra variável. Este tipo de procedimento serve para recodificarmos uma variável quantitativa como renda, aluguel, idade, etc, numa variável categorizada cujos códigos indicam uma classe, grupo ou faixa.

```
PROC GRUPAMENTO ( Vnnnn ,  Vnnnn ) ;
                 'número' 'número'
                 'número' 'número'
                    ---      ---
FIM GRUPAMENTO ;
```

A primeira variável especificada representa a variável para a qual serão atribuídos os novos valores dependendo dos intervalos de valores da segunda variável especificada.

Dentro do bloco do PROC GRUPAMENTO são especificados pares de valores numéricos, o primeiro representa o valor a atribuir à variável a criar, primeira variável especificada, e o segundo, o limite superior fechado da segunda variável para que seja feita a atribuição à primeira variável.

```
EX.:  PROC GRUPAMENTO (V0214, V0204) ;
                              1      1000
                              2     10000
      FIM GRUPAMENTO ;
```

Quando da execução deste procedimento, se a variável V0204 contiver um valor branco, a variável V0214 recebe branco; se a variável V0214 contiver um valor de 0 a 1000, a variável V0214 recebe o valor 1, e se a variável V0204 contém um valor de 1001 a 10000, a variável V0214 recebe o valor 2.

B.6.6.2 - PROC TRANSFORMAÇÃO

Este procedimento serve para se atribuir novos valores a uma variável dependendo dos valores de uma ou mais variáveis de controle.

```
PROC TRANSFORMACAO (Vnnnn, Vcccc, Vcccc, ....) ;
                    número número número ..... ;
                    número número número ..... ;
                      ---    ---    ---  ..... ;
                      ---    ---    ---  ..... ;
                    INVALIDO = "constnumérica" ;
FIM TRANSFORMACAO ;
```

A primeira variável especificada como parâmetro da

transformação especifica a variável que receberá um valor dependendo dos valores das demais variáveis de controle. As variáveis seguintes especificam as variáveis cujos valores servirão de controle para atribuição do valor especificado à primeira variável. Podem ser especificados até 4 variáveis de controle.

Dentro do bloco de transformação devem ser especificados conjuntos de números que corresponderão às variáveis especificadas. Assim, o número de cada conjunto deve corresponder ao número de variáveis especificadas.

```
EX.:  PROC TRANSFORMACAO (V221, V201, V202) ;
                          1     1     1
                          2    10    20
                          3     5    35
      FIM TRANSFORMACAO ;
```

A variável V221 receberá o valor '1' se as variáveis V201 e V202 contiverem o valor 1; o valor 2 se V201 e V202 contiverem os valores 10 e 20, respectivamente, e o valor 3 se V201 e V202 contiverem 5 e 35.

Se os valores das variáveis V201 e V202 contiverem valores não correspondentes a nenhuma combinação especificada para seus valores a variável V221 continuará com o valor igual ao que tinha antes de entrar neste procedimento. Para alterarmos isto, basta especificarmos o comando INVALIDO = 'número' que estabelece um valor a ser atribuído à variável que deverá ser transformada para o

caso das variáveis de controle não terem valores especificados.

B.6.7 - PROCEDIMENTOS PARA FREQUÊNCIA DE VARIÁVEIS - PROC FREQUENCIA

Este procedimento é usado para se montar uma tabela com as frequências de variáveis de registros processados.

```
PROC FREQUENCIA ;
        SAIDAS = IMPRIMIR / GRAVAR ;
        RECORRIDA = Vnnnn, ......, Vnnnn ;
        PESO = Vnnnn ;
        OPCAO = SOMA, MEDIA, DESVIO ;
      VARIAVEIS = Vnnnn, Vnnnn A Vnnnn, ... ;
FIM FREQUENCIA ;
```

A tabela de frequências montada por este procedimento, pode ser impressa ou gravada num arquivo, dependendo da opcão de saída especificada, IMPRIMIR indicando que a tabela deve ser impressa em relatório, e GRAVAR para que a tabela seja gravada em arquivo.

Normalmente a tabela é impressa ao final do programa, mas se quisermos que seja impressa por quebras, basta especificarmos a instrução RECORRIDA, onde podemos especificar uma lista de até 10 variáveis que controlarão as quebras de impressão da tabela.

Se quisermos que as frequências sejam ponderadas, podemos usar a instrução PESO que especificará a variável de peso a ser usada.

Se for desejado que para as variáveis quantitativas sejam calculados a soma, média e desvio padrão, basta especificarmos uma instrução MEDIA.

Após a especificação destas opções para o procedimento frequência, devem ser especificadas as variáveis para as quais serão calculadas as frequências. Para isto usamos a instrução VARIAVEIS que consiste de uma lista de variáveis. Para definirmos um intervalo de variáveis para as quais serão calculadas as frequências, podemos definir Viiii A Vffff, onde Viiii especifica a variável inicial e Vffff, a variável final.

```
EX.:  PROC FREQUENCIA (IMPRIMIR) ;
          RECORRIDA = V0010 ;
          VARIAVEIS = V005, V015 A V020, V030 ;
      FIM FREQUENCIA ;
```

Para cada quebra da variável V0010 será impressa a tabela de frequências das variáveis V005, V015, V016, V017, V018, V019, V020 e V030.

B.6.8 - PROCEDIMENTO PARA EMISSÃO DE RELATÓRIOS DE CRÍTICA - PROC CRITICA

O procedimento CRITICA é usado para se especificar um

relatório de crítica a ser impresso durante a execução do programa.

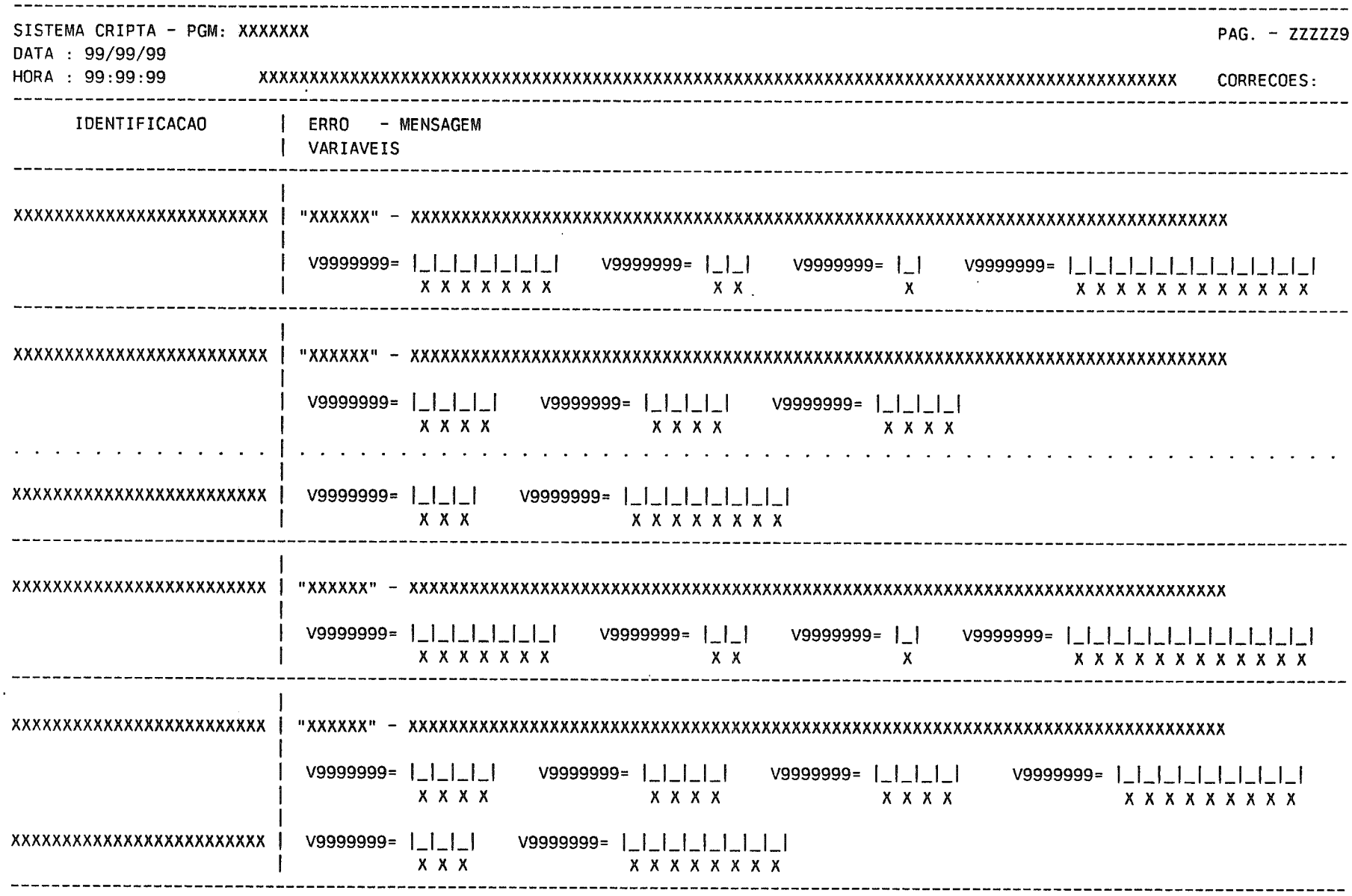
Neste bloco de procedimento são declaradas as variáveis de identificação dos registros a serem criticados, o título do relatório a ser impresso e a tabela de erros que podem ser impressos, cada um contendo o código do erro, a mensagem de erro e uma lista de variáveis associadas ao erro.

O relatório impresso tem um formato padrão, como apresentado na figura B.2, para permitir que as correções sejam especificadas na própria listagem, e as correções transcritas da própria listagem, num formato padrão para a entrada do programa de atualização.

A impressão de um erro é comandada pela execução de instruções ERRO = "nome do erro" nos blocos de procedimentos posteriores a este.

```
PROC CRITICA ;
   IDENTIFICAÇÃO = Vnnnn, .....,Vnnnn ;
   TITULO = "termtit" + "termtit" + ..."termtit" ;
   FREQUENCIA = SIM/NAO ;
   SAIDAS = IMPRIMIR/GRAVAR ;
   DDNAME ="string" ;
   ERRO = "XXXXXX": "mensagem de erro"(Vnnnn,...,Vnnnn) ;
FIM CRITICA ;
```

O comando IDENTIFICAÇÃO especifica uma lista de até 10 variáveis que identificam o registro que contém erro e para



```
SISTEMA CRIPTA - PGM: XXXXXXX                                                                                                    PAG. - ZZZZZ9
DATA : 99/99/99
HORA : 99:99:99          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX    CORRECOES:

      IDENTIFICACAO        |  ERRO   - MENSAGEM
                           |  VARIAVEIS

                           |
XXXXXXXXXXXXXXXXXXXXXXXXXX |  "XXXXXX" - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                           |
                           |  V9999999= |_|_|_|_|_|_|_|    V9999999= |_|_|    V9999999= |_|    V9999999= |_|_|_|_|_|_|_|_|_|_|_|_|_|
                           |             X X X X X X X              X X             X              X X X X X X X X X X X X X
-------------------------------------------------------------------------------------------------------------------------------------------
                           |
XXXXXXXXXXXXXXXXXXXXXXXXXX |  "XXXXXX" - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                           |
                           |  V9999999= |_|_|_|_|    V9999999= |_|_|_|_|    V9999999= |_|_|_|_|
                           |             X X X X              X X X X              X X X X
. . . . . . . . . . . . . .| . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                           |
XXXXXXXXXXXXXXXXXXXXXXXXXX |  V9999999= |_|_|_|    V9999999= |_|_|_|_|_|_|_|_|
                           |             X X X              X X X X X X X X
-------------------------------------------------------------------------------------------------------------------------------------------
                           |
XXXXXXXXXXXXXXXXXXXXXXXXXX |  "XXXXXX" - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                           |
                           |  V9999999= |_|_|_|_|_|_|_|    V9999999= |_|_|    V9999999= |_|    V9999999= |_|_|_|_|_|_|_|_|_|_|_|_|_|
                           |             X X X X X X X              X X             X              X X X X X X X X X X X X X
-------------------------------------------------------------------------------------------------------------------------------------------
                           |
XXXXXXXXXXXXXXXXXXXXXXXXXX |  "XXXXXX" - XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                           |
                           |  V9999999= |_|_|_|_|    V9999999= |_|_|_|_|    V9999999= |_|_|_|_|    V9999999= |_|_|_|_|_|_|_|_|_|
                           |             X X X X              X X X X              X X X X              X X X X X X X X X
                           |
XXXXXXXXXXXXXXXXXXXXXXXXXX |  V9999999= |_|_|_|    V9999999= |_|_|_|_|_|_|_|_|
                           |             X X X              X X X X X X X X
-------------------------------------------------------------------------------------------------------------------------------------------
```

FIGURA B.2 - Relatório de crítica emitido pelo programa gerado pelo sistema Cripta

o qual as variáveis associadas ao erro serão impressas.

O comando TITULO fornece o nome do relatório a ser colocado no cabeçalho de cada página. O título pode ser composto por "strings" e valores de variáveis, sendo usado o operador "+" para significar a concatenação destes termos para formação do título.

O comando FREQUENCIA serve para especificar se ao final da impressão do relatório de crítica devem ser impressas as frequências de cada erro ocorrido na impressão. Este comando é opcional e o default é NÃO.

O comando SAIDA é opcional e, se especificado, informa se o relatório deve ser impresso ou gravado num arquivo. O "default" assumido para este parâmetro é a impressão do relatório.

O comando DDNAME é opcional e, se especificado, informa o DDNAME do arquivo no qual será impresso ou gravado o relatório. O DDNAME do arquivo default é RELCRIT caso a saída seja de impressão ou ARQCRIT para o caso de gravação.

A seguir deve ser especificada a tabela de erros que poderão ser impressos no relatório com a execução do comando ERRO 'nome do erro'.

Para cada erro deve ser especificado:

```
ERRO = "nome do erro":  "Mensagem"( Vnnnn, ....., Vnnnn) ;
```

Este comando especifica um erro identificado pelo código definido em "nome do erro", com a mensagem de erro

associada a ele, definida em "mensagem", e com uma lista de variáveis a serem impressas associadas a ele. O código do erro é formado por uma "string" de até 6 caracteres, e a mensagem de erro pode ser formada por um conjunto de termos, da mesma forma que o título do relatório, sendo composta por "strings" e variáveis usando-se o operador "+" para concatenação dos termos.

B.6.9 - PROCEDIMENTO PARA REFERÊNCIA NUM PROGRAMA

É comum em procedimentos de crítica termos a necessidade de saber se uma determinada condição anterior ocorreu ou não.

Estes casos são normalmente resolvidos com o uso de uma variável auxiliar funcionando como "flag", no entanto, a linguagem Cripta oferece duas vantagens para tratamento destes casos descritas a seguir.

B.6.9.1 - PROCEDIMENTO PARA OCORRÊNCIA DE INDICAÇÃO DE ERROS

Uma das condições que desejamos fazer referência, é a ocorrência da indicação de um determinado erro. Na linguagem Cripta, a indicação de um erro é determinada pela execução do comando ERRO ="XXXXXX" onde "XXXXXX" especifica o seu respectivo código. Assim, quando desejamos saber se

houve ou não a ocorrência deste erro no programa basta usarmos o comando condicional SE "XXXXXX" , onde "XXXXXX" representa o código do erro. Esta expressão condicional fornece um valor verdadeiro se o erro "XXXXXX" foi indicado para o registro que está sendo correntemente processado pelo programa.

Ex: Queremos fazer uma crítica de um campo, variável V0050, somente se um outro campo, variável V0030, estiver preenchida corretamente.

```
    ----
SE V0030 = BRANCO OU V0030 = 0
   ENTAO
        ERRO = "E10" ;
FIM SE ;
   ----
SE NAO "E10"
  ENTAO
       ----
```

B.6.9.2 - PROCEDIMENTO MARCA

Às vezes temos também a necessidade de saber não apenas se uma determinada condição ocorreu, mas se uma ou mais de um determinado conjunto de condições ocorreu ou não. Como exemplo, podemos citar o caso de querermos saber num determinado questionário, se o campo de CGC está

preenchido corretamente ou não, o que envolve um conjunto de críticas a seus subcampos, como raiz, sufixo e dvs.

Para facilitar este tipo de processamento, a linguagem CRIPTA permite a especificação de um processo de Referência, PROC MARCA "identificador" onde dentro deste bloco podemos especificar as críticas necessárias com as respectivas indicações de erros. Fora deste processo podemos usar o comando condicional, SE "identificador", que fornecerá um valor verdadeiro se, pelo menos, um dos erros especificados dentro do processo PROC MARCA "identificação" executado(impresso) para o registro que está sendo processado correntemente pelo programa, e falso em caso contrário.

```
PROC MARCA "identificador" ;
     -----
     -----
FIM MARCA ;
```

Se a condição a ser referenciada não é a indicação de um erro, mas qualquer outra condição, podemos especificar a PROC MARCA com com opcão MARQUE indicando que a condição a ser referenciada será indicada pela instrução MARQUE.

```
PROC MARCA "identificador";
    -----
      MARQUE ;
    -----
FIM MARCA ;
```

O uso deste tipo de bloco de procedimento evita a necessidade da declaração de uma variável auxiliar de "flag" para a condição, uma vez que o sistema a gera automáticamente sendo referenciada pelo "identificador" especificado.

B.6.10 - PROCEDIMENTOS PARA IMPRESSÃO DE RELATÓRIOS - PROC IMPRESSAO

Este procedimento é usado para se imprimir uma lista de variáveis pertencentes aos registros processados pelo programa.

```
PROC IMPRESSAO ;
     TITULO = "string" ;
     RECORRIDA = Vnnnn, Vnnnn, ..... ;
     NOME = SIM/NAO ;
     ERRO =   Vnnnn ;
     DDNAME = "string" ;
     VARIAVEIS = Vnnnn,Vnnnn TOTAL,..., Vnnnn ;
FIM IMPRESSAO ;
```

O comando VARIAVEIS especifica a lista de variáveis a serem impressas. Se quisermos que para algumas variáveis sejam calculados os totais, basta colocarmos em seguida à variável especificada, a palavra reservada TOTAL.

Dentro do bloco de procedimento devemos especificar o

título a ser impresso em cada página do relatório. A impressão é efetuada página a página, isto é, uma página é impressa somente quando já está completa, cheia, ou ao final do programa. Se quisermos estipular uma quebra para esta impressão, devemos usar a instrução RECORRIDA = "lista de variáveis", onde podem ser especificadas até 10 variáveis que controlarão a quebra.

No formato default do relatório de impressão são impressos os nomes das variáveis. Se não quisermos que sejam impressos, basta usarmos a instrução NOME = NAO;.

A instrução DDNAME é opcional e, se especificada, informa o DDNAME do arquivo no qual será feito a impressão. Se não especificada, é assumido o DDNAME default RELIMP.

B.6.11 - PROCEDIMENTOS PARA GRAVAÇÃO DE ARQUIVOS AUXILIARES

B.6.11.1 - PROC GRAVAÇÃO

Este procedimento é utilizado para se copiar registros do arquivo de entrada sendo processado para um arquivo de gravação de trabalho.

Este procedimento pode ser usado quando se deseja criticar um arquivo de entrada e gravar-se um arquivo com os registros errados, a ser impresso posteriormente por um programa específico.

Para se gravar um código de erro num registro a ser

gravado, deve-se definir uma variável no registro de entrada no dicionário com característica "E".

```
PROC GRAVACAO ;
      DDNAME = "string" ;
FIM GRAVACAO ;
```

B.6.11.2 - PROC TRASLADO

Este procedimento é utilizado para se gravar num arquivo de trabalho uma lista de variáveis dos registros sendo processados ou variáveis auxiliares.

```
PROC TRASLADO ;
     FORMATO =   FB/VB ;
     TAMANHO = "constnumérica" ;
     DDNAME = "string" ;
     VARIAVEIS= Vnnnn, Vnnnn, ...., Vnnnn ;
FIM TRASLADO ;
```

A instrução FORMATO informa o formato dos registros a serem gravados "FB" para registros fixos e "VB" para registros variáveis.

A instrução TAMANHO informa o tamanho em bytes dos registros a serem gravados.

A instrução DDNAME é opcional, e se informada, especifica o DDNAME do arquivo a ser gravado. O DDNAME

assumido como "default" é ARQTRAS.

As variáveis a serem gravadas neste arquivo são fornecidas na instrução VARIAVEIS e podem ser qualquer tipo de variável.

B.6.12 - PROCEDIMENTOS PARA TABULAÇÃO DE DADOS

B.6.12.1 - PROC CRUZAMENTO

Este procedimento é usado para se calcular uma matriz onde cada linha corresponde a uma categoria de uma variável e cada coluna corresponde a uma categoria de outra variável. Cada célula da matriz contém o número de registros selecionados, cujo valor na variável que indica a linha é igual ao código da linha, e cujo valor da variável que indica a coluna é igual ao código da coluna.

```
PROC CRUZAMENTO "constnumerica" ;
     LINHA = Vnnnn ;
     COLUNA = Vnnnn ;
     POR = Vnnnn ;
     TITULO = "string" ;
     RECORRIDA = Vnnnn, Vnnnn, ....., Vnnnn ;
     PESO = Vnnnn ;
     INVALIDO =   SIM/NAO ;
     DDNAME = "STRING" ;
     SAIDA = IMPRIMIR/GRAVAR ;
FIM CRUZAMENTO ;
```

A constante numérica que identifica o procedimento serve para a identificação das matrizes gravadas.

A instrução LINHA informa a variável cujas categorias formarão as linhas da matriz.

A instrução COLUNA informa a variável cujas categorias formarão as colunas da matriz.

Às vezes queremos especificar um cruzamento com além das duas variáveis que formam as linhas e colunas de uma matriz, uma outra variável categorizada que definiria uma matriz, com as linhas e colunas especificadas para o procedimento, para cada uma das categorias desta terceira variável. Para isto basta especificarmos a instrução POR definindo esta terceira variável.

A instrução TITULO tem que ser especificada se a opção para impressão das matrizes foi IMPRIMIR, e informa o TITULO a ser impresso no cabeçalho das matrizes impressas.

As matrizes calculadas, por default são impressas ao final do programa se a matriz foi atualizada pelo menos uma vez. Se desejado, podemos usar a instrução RECORRIDA para especificarmos uma lista de variáveis que determinarão as quebras para impressão das matrizes.

A instrução PESO é opcional e, se usada, especifica que a frequência a ser computada no procedimento cruzamento deve ser uma frequência ponderada tendo a variável especificada nesta instrução como variável de peso para cada registro.

A instrução SAIDA especifica se as matrizes geradas devem ser impressas ou gravadas. Se esta instrução não for

especificada, as matrizes serão impressas e gravadas.

A instrução INVALIDO é opcional, e serve para indicar se a linha e coluna correspondentes aos valores de categorias inválidas da matriz, geradas automaticamente, devem ou não ser impressas. O default de processamento é imprimir estas linhas e colunas.

A instrução DDNAME é opcional, e se especificada, informa o nome do arquivo no qual serão impressas ou gravadas as matrizes geradas no procedimento. O DDNAME assumido como "default" é o "ARQTAB".

B.6.12.2 - PROC QUANTIFICADO

Este procedimento é análogo ao procedimento CRUZAMENTO, com a diferença que ao invés de serem acumuladas as frequências em cada célula da matriz, são acumulados os os valores de uma terceira variável quantificada.

```
PROC QUANTIFICADO "constnumerica" ;
     LINHA = Vnnnn ;
     COLUNA = Vnnnn ;
     SOMA = Vnnnn ;
     POR = Vnnnn ;
     TITULO = "string" ;
     RECORRIDA = Vnnnn, Vnnnn, ....., Vnnnn ;
     PESO = Vnnnn ;
     INVALIDO =   SIM/NAO ;
     DDNAME = "STRING" ;
     SAIDA = IMPRIMIR/GRAVAR ;
FIM QUANTIFICADO ;
```

As instruções dentro deste procedimento tem as mesmas funções do procedimento CRUZAMENTO, com exceção da instrução SOMA, que especifica a variável cujo valor será somado a cada célula da matriz correspondente.

B.6.12.3 - PROC TABELA

Este procedimento é utilizado para se especificar uma tabela de uma variável categorizada por até 150 variáveis quantitativas. O número de linhas da tabela é igual ao número de categorias da variável categorizada mais um para valores inválidos desta variável. O número de colunas é igual ao número de variáveis quantitativas especificadas.

```
PROC TABELA "constnumerica" ;
     LINHA = Vnnnn ;
     COLUNAS = Vnnnn, Vnnnn, ...., Vnnnn ;
     POR = Vnnnn ;
     TITULO = "string" ;
     RECORRIDA = Vnnnn, Vnnnn, ....., Vnnnn ;
     INVALIDO =   SIM/NAO ;
     DDNAME = "STRING" ;
     SAIDA = IMPRIMIR/GRAVAR ;
FIM QUANTIFICADO ;
```

As instruções tem as mesmas funções que nos procedimentos CRUZAMENTO e QUANTIFICADO.

B.6.13 - PROCEDIMENTO PARA GRAVACAO DO ARQUIVO ATUALIZADO - PROC SAIDA

Este procedimento é usado para alterarmos a geração automática de gravação do arquivo de saída e pode ser usado de dois modos.

```
PROC SAIDA ;
   ---
   --- instruções para seleção dos registros a
   --- serem gravados, com uso de comandos REJEITE
   ---
FIM SAIDA ;
```

A execução de um comando REJEITE especificado no interior deste bloco faz com que o registro corrente não seja gravado. Em caso contrário os registros serão gravados sequencialmente.

Um outro modo de usarmos este procedimento é :

```
PROC SAIDA (PLI) ;
    ---
    ---  comandos PLI
    ---
FIM SAIDA ;
```

que indica que a gravação do arquivo de saída será definido totalmente neste bloco de procedimento em código PLI.

B.6.14 - PROCEDIMENTO PARA FINALIZAÇÃO DO PROGRAMA - PROC FINALIZAÇÃO

Este procedimento serve para se especificar as instruções a serem executadas no programa gerado uma única vez depois de sair do loop de execução dos registros do arquivo de entrada do programa.

```
PROC FINALIZAÇÃO ;

      ----

      ----  instruções da linguagem especificando os

      ----  procedimentos a serem realizados podendo

      ----  conter o bloco PROC PLI ---- FIM PROC;

      ----

FIM FINALIZACAO ;
```

ANEXO C

EXEMPLO DA ESPECIFICAÇÃO DE UMA APLICAÇÃO

A figura C.1 apresenta um questionário de uma pesquisa hipotética de domicílios.

Este questionário é composto pelos dados de identificação do questionário, o tipo de entrevista realizada e a relação dos moradores do domicílio com os dados referentes a cada um.

Após preenchidos, estes questionários são digitados num arquivo com a seguinte estrutura:

- tamanho do registro de 80 posições;

- três tipos de registros conforme os quadros do questionário, tendo como chave os campos:
  - . UF
  - . MUNICÍPIO
  - . CÓDIGO DO MUNICÍPIO

- o registro tipo "01" é formado pelos campos:
  - . CHAVE
  - . NÚMERO DE MORADORES

PESQUISA DE DOMICILIOS

01 IDENTIFICAÇÃO DO QUESTIONÁRIO

1) UF	2) MUNIC	3) CÓDIGO DO DOMICÍLIO	4) NÚMERO DE MORADORES
⌊__⌋	⌊____⌋	⌊_____⌋	⌊__⌋

02 SITUAÇÃO FINAL DA ENTREVISTA

01 ☐ COMPLETA 02 ☐ INCOMPLETA 03 ☐ NÃO REALIZADA

03 MORADORES DO DOMICÍLIO

Num. de ordem	Nome	Relação com o chefe do domicílio	Nível de instrução	Sexo 1-Masc 2-Fem	Salário
(01)	(02)	(03)	(04)	(05)	(06)
01	.....	☐	☐	☐	⌊_____⌋
02	.....	☐	☐	☐	⌊_____⌋
03	.....	☐	☐	☐	⌊_____⌋
04	.....	☐	☐	☐	⌊_____⌋
05	.....	☐	☐	☐	⌊_____⌋
06	.....	☐	☐	☐	⌊_____⌋
07	.....	☐	☐	☐	⌊_____⌋
08	.....	☐	☐	☐	⌊_____⌋
09	.....	☐	☐	☐	⌊_____⌋
10	.....	☐	☐	☐	⌊_____⌋
11	.....	☐	☐	☐	⌊_____⌋
12	.....	☐	☐	☐	⌊_____⌋
13	.....	☐	☐	☐	⌊_____⌋
14	.....	☐	☐	☐	⌊_____⌋
15	.....	☐	☐	☐	⌊_____⌋

CÓDIGOS

RELAÇÃO COM O CHEFE DO DOMICÍLIO	NÍVEL DE INSTRUÇÃO	OBSERVAÇÕES
1- Chefe do domicílio 2- Cônjuge 3- Filho 4- Emp. doméstico 5- Parente do emp. doméstico	1- Sem instrução 2- Pré-escolar 3- Primário 4- 1. grau 5- 2. grau 6- Superior	- -

FIGURA C.1 - Questionário da pesquisa de domicílios

- o registro tipo "02" é formado pelos campos:

  . CHAVE

  . SITUAÇÃO FINAL DA ENTREVISTA

- o registro tipo "03" é formado pelos campos:

  . CHAVE

  . NOME DO MORADOR

  . RELAÇÃO COM O CHEFE DO DOMICÍLIO

  . NÍVEL DE INSTRUÇÃO

  . SEXO

  . SALÁRIO

As figuras C.2, C.3, C.4 e C.5 aprentam a descrição deste arquivo usando-se a ferramenta para especificação do Dicionário da Aplicação, DICAPLIC.

A identificação das variáveis é feita pelo número do bloco do questionário e o item que representam.

Para este questionário podemos definir um conjunto de críticas de compatibilidades a serem efetuadas como por exemplo:

- o campo de Número de moradores tem de ser igual ao número de moradores informados;

- o campo Situação final da entrevista tem de ser um dos códigos especificados no questionário;

DESCRIÇÃO DO ARQUIVO DE ENTRADA

CHAVE	NOME DO ARQUIVO	FORMATO	TAMANHO DO REGISTRO	NTIPOSREG	VARIÁVEL CONTADORA DE OCORRÍNCIAS	VARIÁVEL DE OCORRÍNCIA	
A0	ARQUIVO DA PESQUISA DE DOMICILIOS	FB	80	3			

DESCRIÇÃO DOS TIPOS DE REGISTROS

CHAVE	TIPO DO REGISTRO	TAMANHO DO REGISTRO	NÚMERO MÍNIMO DE REGISTROS	NÚMERO MÁNIMO DE REGISTROS	COMENTÁRIO
R0	01	80	1	1	IDENTIFICACAO DO QUESTIONARIO
R0	02	80	1	1	SITUACAO FINAL DA ENTREVISTA
R0	03	80	1	15	MORADO DO DOMICILIO
R0					
R0					
R0					
R0					
R0					
R0					

FIGURA C.2 - Descrição do arquivo da pesquisa de domicílios

DESCRIÇÃO DAS CARACTERÍSTICAS FÍSICAS DAS VARIÁVEIS

CHAVE	NÚMERO DA VARIÁVEL	NOME DA VARIÁVEL	POSIÇÃO INICIAL	TAMANHO	CARACTER.	FORMATO	DECIMAIS	NÚMERO DE CATEGS	NÚMERO DE OCORRENC	VARIÁVEL DE BASE	TIPO REG A QUE PERTENCE	
V1	0010	TIPO DO REGISTRO	1	2	T	N					CHAVE	
V1	0011	SEQ DO	3	2		N					CHAVE	
V1	0101	UF	5	2		N					CHAVE	
V1	0102	MUNIC	7	5		N					CHAVE	
V1	0103	COD. DOMICILIO	12	7		N					CHAVE	
V1	0104	NUM. MORADORES	19	2		N					01	
V1	0200	SIT FINAL DA ENTREVISTA	19	2		N		3			02	
V1	0301	NUM ORDEM MORADOR	19	2		N					03	
V1	0302	NOME	21	22		L					03	
V1	0303	REL C/ CHEFE DOMICILIO	51	1		N		5			03	
V1	0304	NIVEL INSTRUCAO	52	1		N		6			03	
V1	0305	SEXO	53	1		N		2			03	
V1	0306	SALARIO	54	8		N	2				03	
V1												
V1												

FIGURA C.3 - Descrição das características físicas das variáveis do arquivo da pesquisa de domicílios

DESCRIÇÃO DAS CARACTERÍSTICAS DE PROCESSAMENTO DA VARIÁVEL

CHAVE	NÚMERO DA VARIÁVEL	LIMITE INFERIOR	LIMITE SUPERIOR	POSSUI IGN	REPRESENTAÇÃO DE IGNORADO	VALOR A SER ASSUMIDO	
V2	0104	1	15				
V2	0301	1	15				
V2	0306			S	999999,99	999999,99	
V2							
V2							
V2							
V2							
V2							
V2							
V2							
V2							
V2							
V2							
V2							
V2							
V2							

FIGURA C.4 - Descrição das características de processamento das variáveis do arquivo da pesquisa de domicílios

DESCRIÇÃO DAS CATEGORIAS

CHAVE	NÚMERO DA VARIÁVEL	CÓDIGO DA CATEGORIA	NOME DA CATEGORIA	COMENTÁRIO
V3	0200	01	COMPLETA	
V3	0200	02	INCOMPLETA	
V3	0200	03	NAO REALIZADA	
V3	0303	1	CHEFE DO DOMICILIO	
V3	0303	2	CONJUGE	
V3	0303	3	FILHO	
V3	0303	4	EMP DOMESTICO	
V3	0303	5	PARENTE EMP DOMESTICO	
V3				
V3				
V3				
V3				
V3				
V3				
V3				

FIGURA C.5 - Descrição das categorias das variáveis do arquivo da pesquisa de domicílios

- se o campo Relação do morador for igual a cônjuge, então, os valores do campo Sexo deste morador e do morador com o campo de Relação do morador igual a chefe do domicílio têm de ser diferentes;

- se o campo Relação do morador for igual a parente do empregado doméstico, então deve existir pelo menos um morador com código igual a empregado doméstico;

A figura C.6 apresenta a especificação do Plano de Crítica, PCAUT, para as críticas apresentadas.

A figura C.7 apresenta o programa Cripta especificado para execução desta aplicação de aplicação das críticas definidas. Este programa será executada associadamente com a especificação do dicionário e o plano de crítica. Pode ser notado, neste programa, que a crítica "E301", apresentada como descrita no Plano de Crítica, é especificada na linguagem de modo a ser executada em substituição à especificação correspondente do Plano de Crítica.

A figura C.8 apresenta o relatório de crítica emitido pelo programa gerado a partir desta especificação.

REFERÊNCIA	DESCR	CAUSA/PROCEDIMENTO	DESCR	EFEITO
E101		"NUM MORADORES INCOMP C/INFORMADOS" SE V010 DIF MAIORVALOR(V0301);		V0104; ERRO;
E201		"SIT FINAL ENTREVISTA INVALIDA" SE V0200 = INVALIDA;		V0200 ERRO;
E301	*	SE RELACAO DO MORADOR IGUAL A CONJUGE, ENTAO OS VALORES DO CAMPO SEXO DESTE MORADOR E DO MORADOR CHEFE DO DOMICILIO TEM DE SER DIFERENTES;		
E302		"PARENTE EMP.DOMEST. SEM EMP.DOMIC" SE V0303 = 5 E FREQUENCIA(V0303 = 4) = 0;		V0301,V0303; ERRO;

FIGURA C.6 - Plano de Crítica para a pesquisa de domicílios

```
"PROGRAMA PESQDOM";

   PROC QUEBRA;
       CHAVE = V0101,V0102,V0103;
       TAMANHO = 50;
   FIM QUEBRA;

   PROC CRITICA;
       IDENTIFICACAO=V0101,V0102,V0103,V0010,V0011;
       TITULO = "CRITICA DO QUESTIONARIO DE DOMICILIOS";
       ERRO = "E301":"CONJUGE INCOMPATIVEL COM CHEFE DO DOMICILIO"
                     (V0301,V0302,V0303,V0305,
                      S0301,S0302,S0303,S0305);
   FIM CRITICA;

   ROTPLAN "E301";
      SE V0303 = 2
        ENTAO
             SALVE V0301,V0302,V0303;

             PROC PESQUISA (V0010 = 3 E V0303 = 1);
                 SE S0305 = V0305
                   ENTAO
                        ERRO = "E301";
                 FIM SE;
             FIM PESQUISA;
      FIM SE;
   FIM ROTPLAN;

   PROC %CRITDOM;
       CONFORME V0010; /* TIPO DO REGISTRO */
            1 : COPIAPLANO("E101");
            2 : COPIAPLANO("E201");
            3 : COPIAPLANO("E301","E302";)
       FIM CONFORME;
   FIM %CRITDOM;

FIM PROGRAMA;
```

FIGURA C.7 - Programa CRIPTA para a pesquisa de domicílios

```
------------------------------------------------------------------------------------------------------------------
SISTEMA CRIPTA - PGM: PESQDOM                                                                       PAG. -       1
DATA : 28/03/89
HORA : 15:24:35                                 CRITICA DO QUESTIONARIODE DOMICILIO                 CORRECOES:
------------------------------------------------------------------------------------------------------------------
     IDENTIFICAO       | ERRO   - MENSAGEM
                       | VARIAVEIS
------------------------------------------------------------------------------------------------------------------
                       |
234150282647350100     | "E101"  - NUM MORADORES INCOMP. C/ INFORMADOS
                       |
                       | V0000104= |_|_|
                       |            0 3
------------------------------------------------------------------------------------------------------------------
                           |
234150262647350301     | "E301"   - CONJUNGE INCOMPATIVEL COM CHEFE DO DOMICILIO
                       |
                       | V0000301= |_|_|    V0000302= |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|   V0000303= |_|    V0000305= |_|
                       |            0 1                 J O A O   S I L V A                                        1                 1
 . . . . . . . . . . . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
                       |
234150262647350302     | V0000301= |_|_|    V0000302= |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|   V0000303= |_|    V0000305= |_|
                       |            0 2                 M A R I A   F I L O M E N A   S I L V A                    1                 1
------------------------------------------------------------------------------------------------------------------
                       |
523647534563480200     | "E201"   - SITUACAO FINAL ENTREVISTA INVALIDA
                       |
                       | V0000200= |_|_|
                       |            0 5
------------------------------------------------------------------------------------------------------------------
                           |
523647534563480304     | "E302"   - PARENTE EMP.DOMEST. SEM EMP.DOMIC.
                       |
                       | V0000301= |_|_|    V0000303= |_|
                       |            0 4                5
                           |
333647534563480311     | "E302"   - PARENTE EMP.DOMEST. SEM EMP.DOMIC.
                       | V0000301= |_|_|    V0000303= |_|
                       |            1 1                5
------------------------------------------------------------------------------------------------------------------
```

FIGURA C.8 - Relatório de crítica gerado para a crítica da pesquisa de domicílios

ANEXO D

DIAGRAMA MODULAR HIERÁRQUICO DO SISTEMA

As figuras D.1 a D.49 apresentam a estrutura modular hierárquica do sistema proposto.

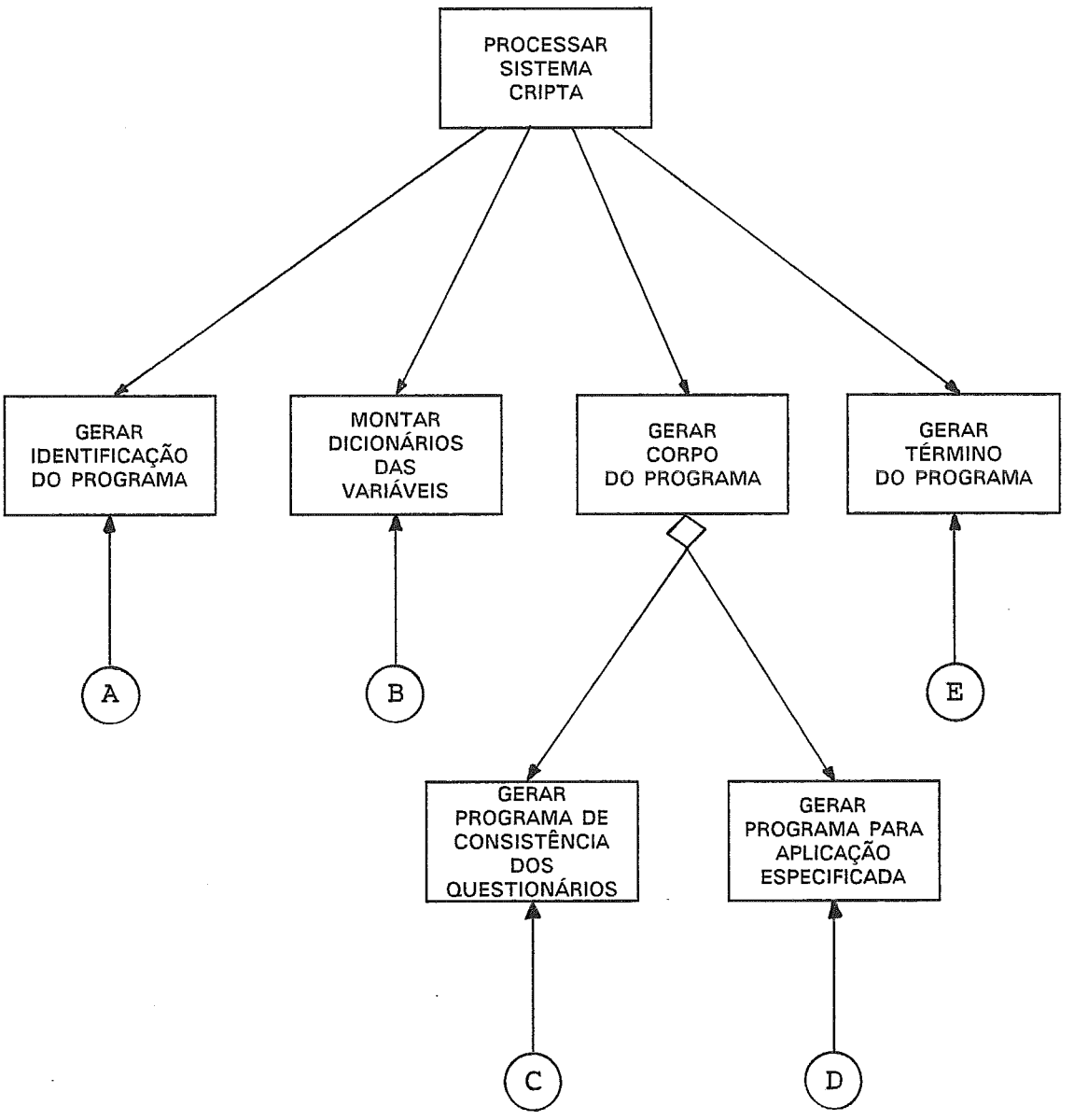


FIGURA D.1 - Estrutura modular do Sistema Cripta (nível 0)

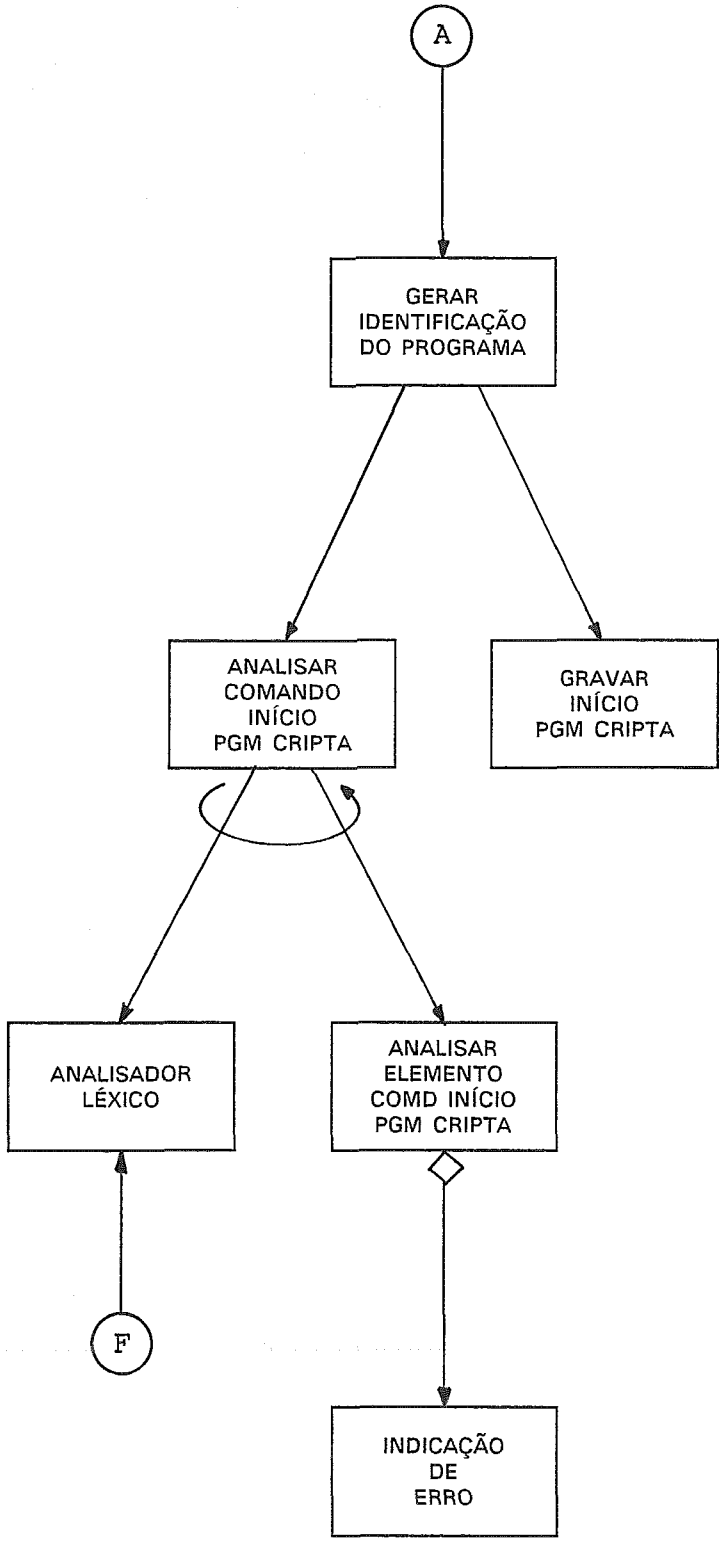


FIGURA D.2 - Estrutura modular "GERAR IDENTIFICAÇÃO DO PROGRAMA" (nível 1)

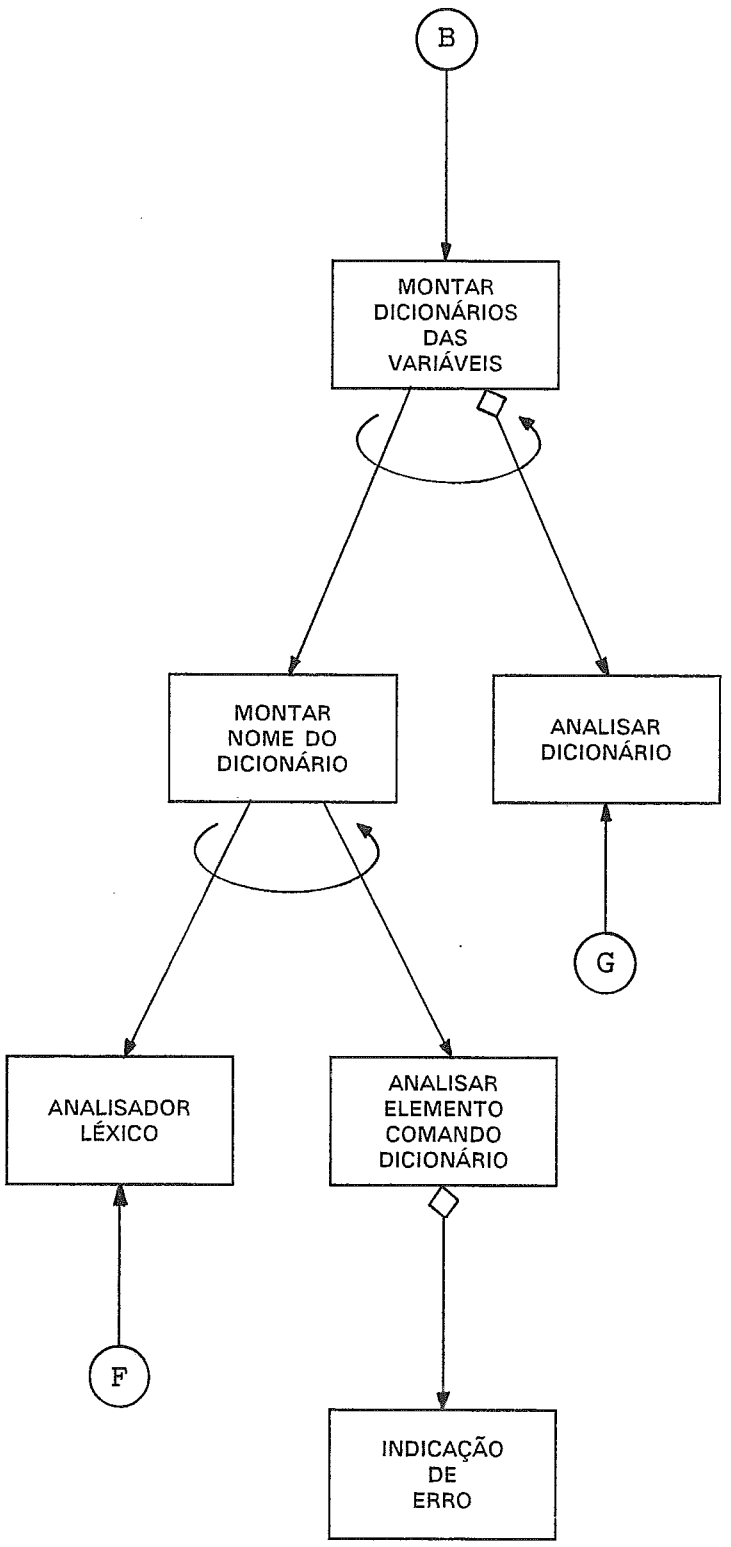


FIGURA D.3 - Estrutura modular "MONTAR DICIONÁRIOS DAS VARIÁVEIS" (nível 1)

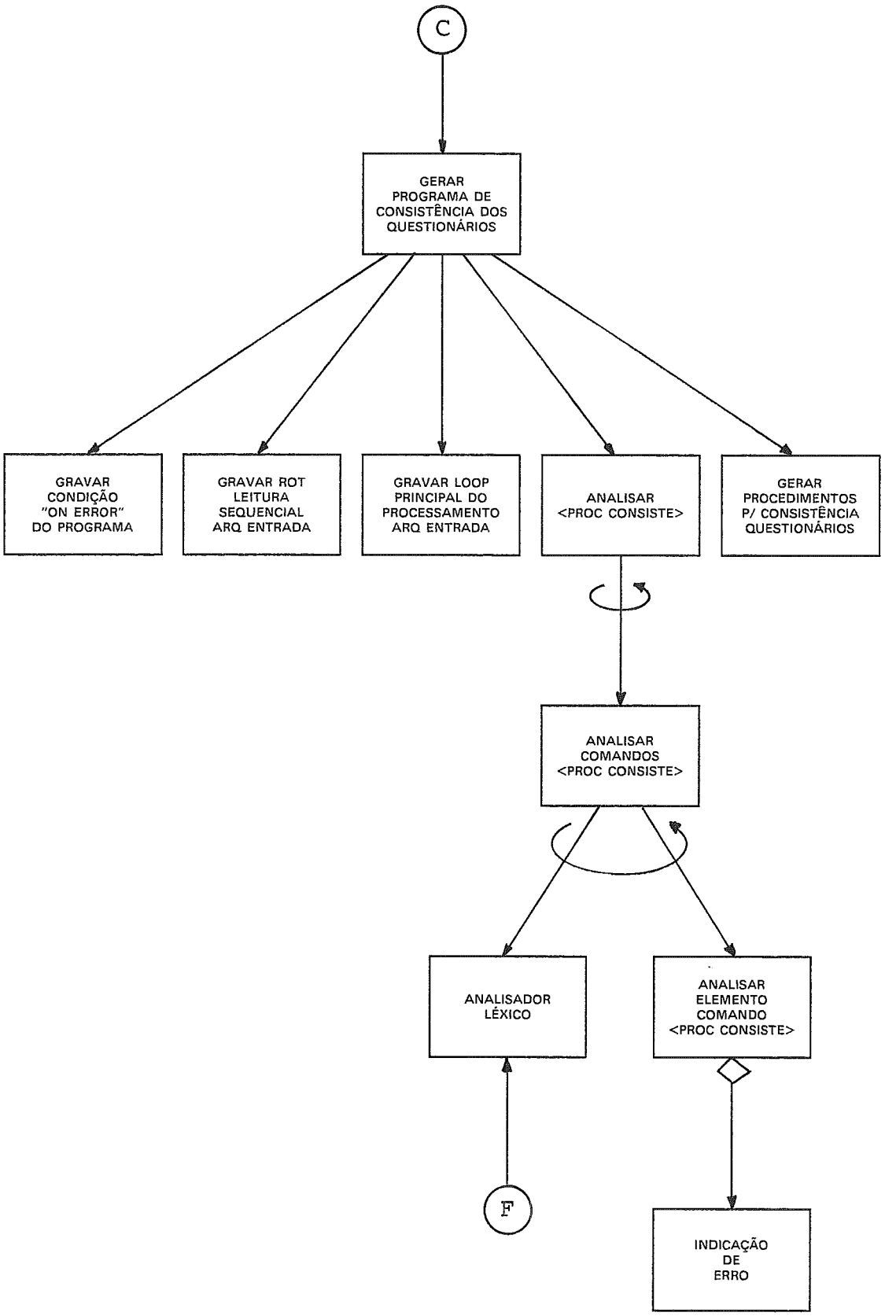


FIGURA D.4 - Estrutura modular "GERAR PROGRAMA DE CONSISTÊNCIA DOS QUESTIONÁRIOS" (nível 1)

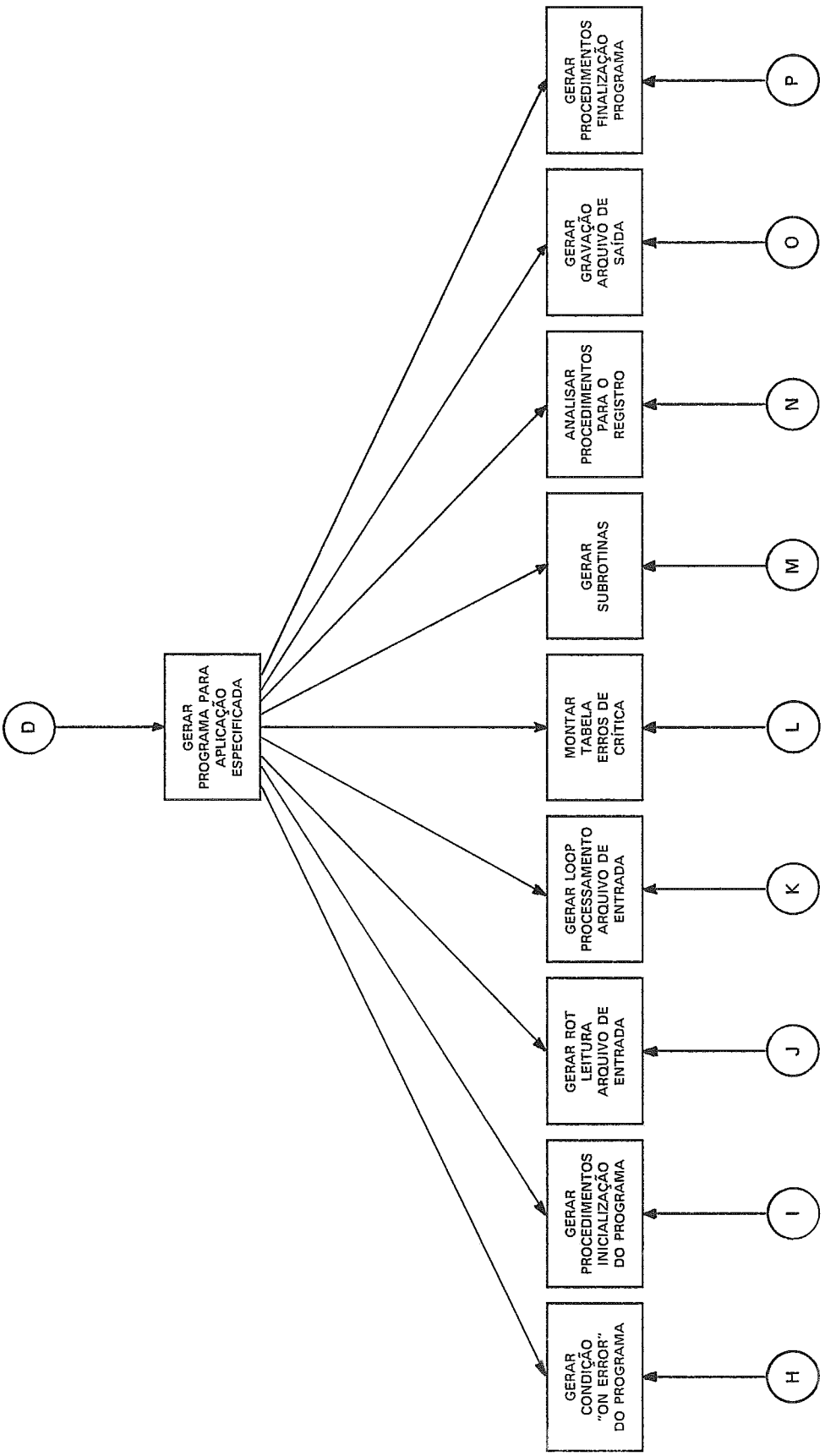


FIGURA D.5 - Estrutura modular "GERAR PROGRAMA PARA APLICAÇÃO ESPECIFICADA" (nível 1)

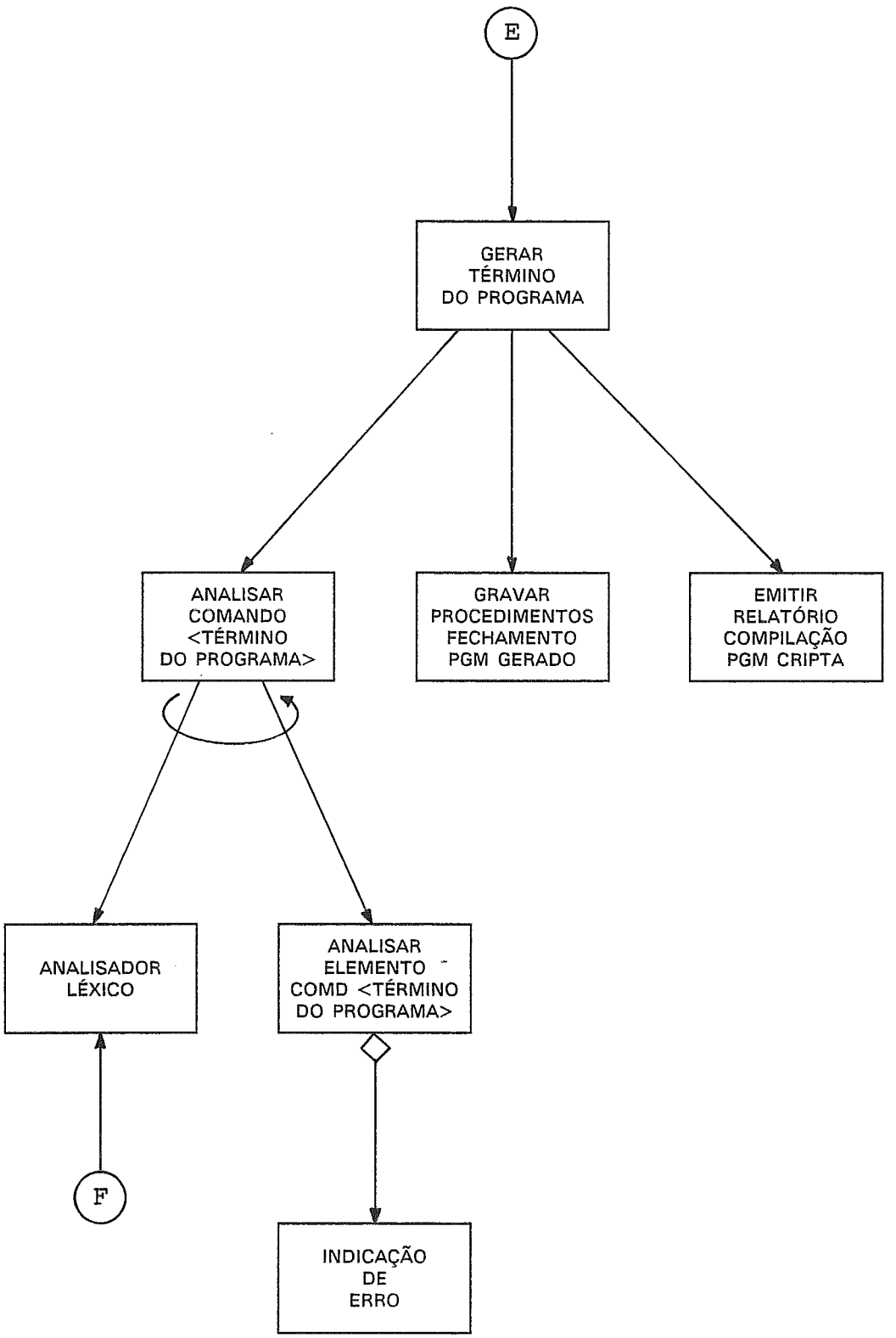


FIGURA D.6 - Estrutura modular "GERAR TÉRMINO DO PROGRAMA" (nível 1)

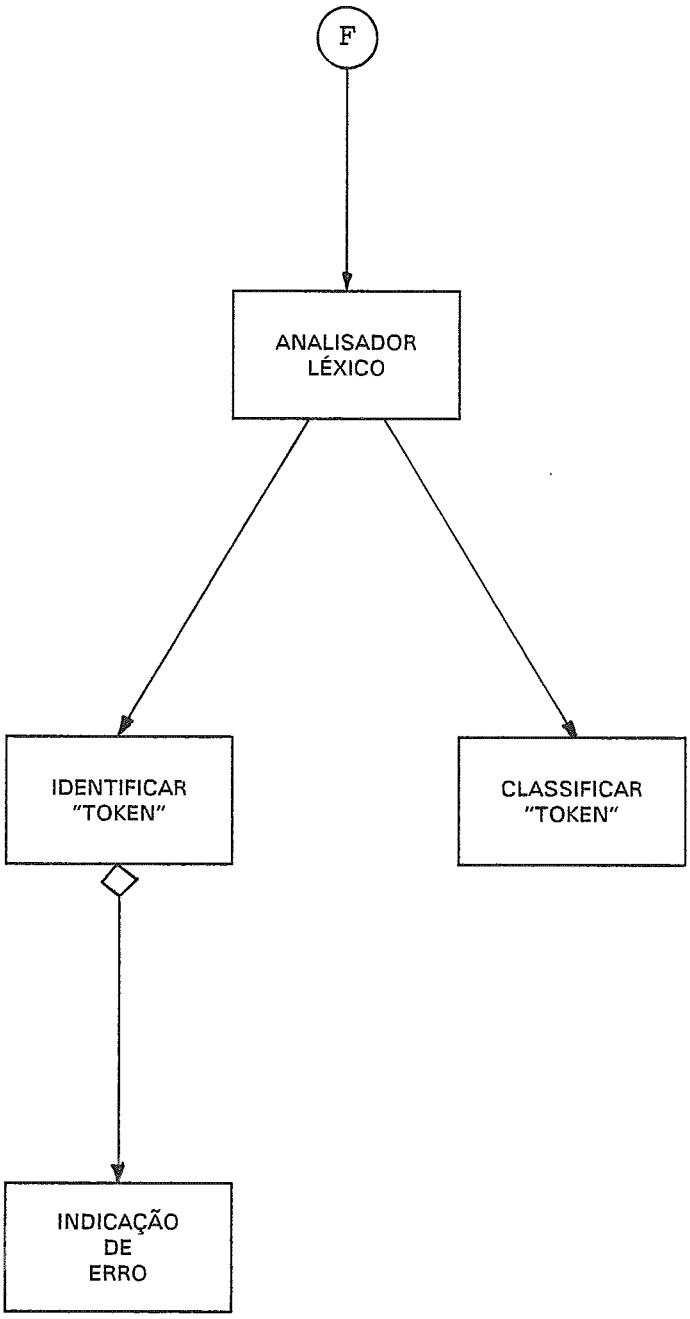


FIGURA D.7 - Estrutura modular "ANALISADOR LÉXICO"
(nível 2)

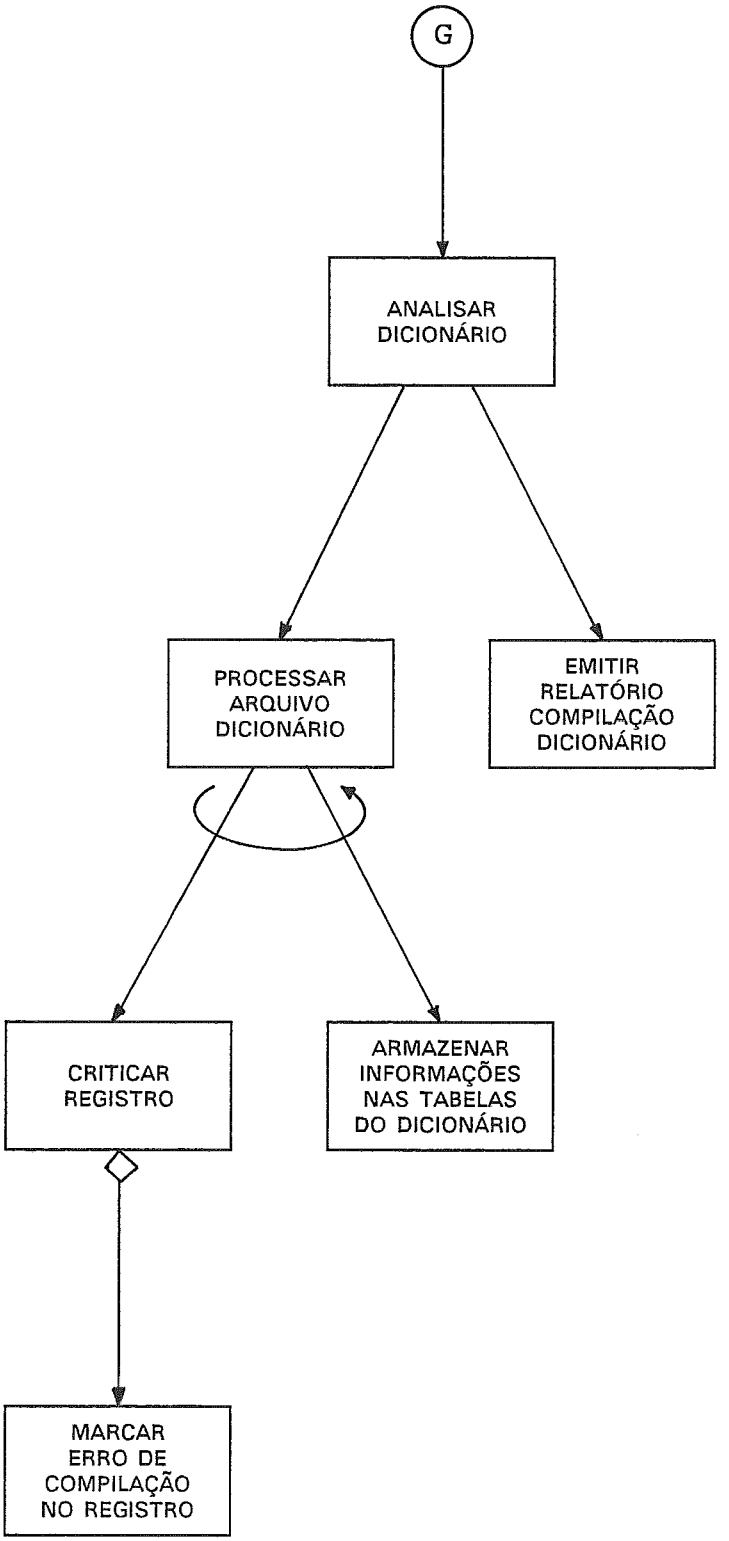


FIGURA D.8 - Estrutura modular "ANALISAR DICIONÁRIO" (nível 2)

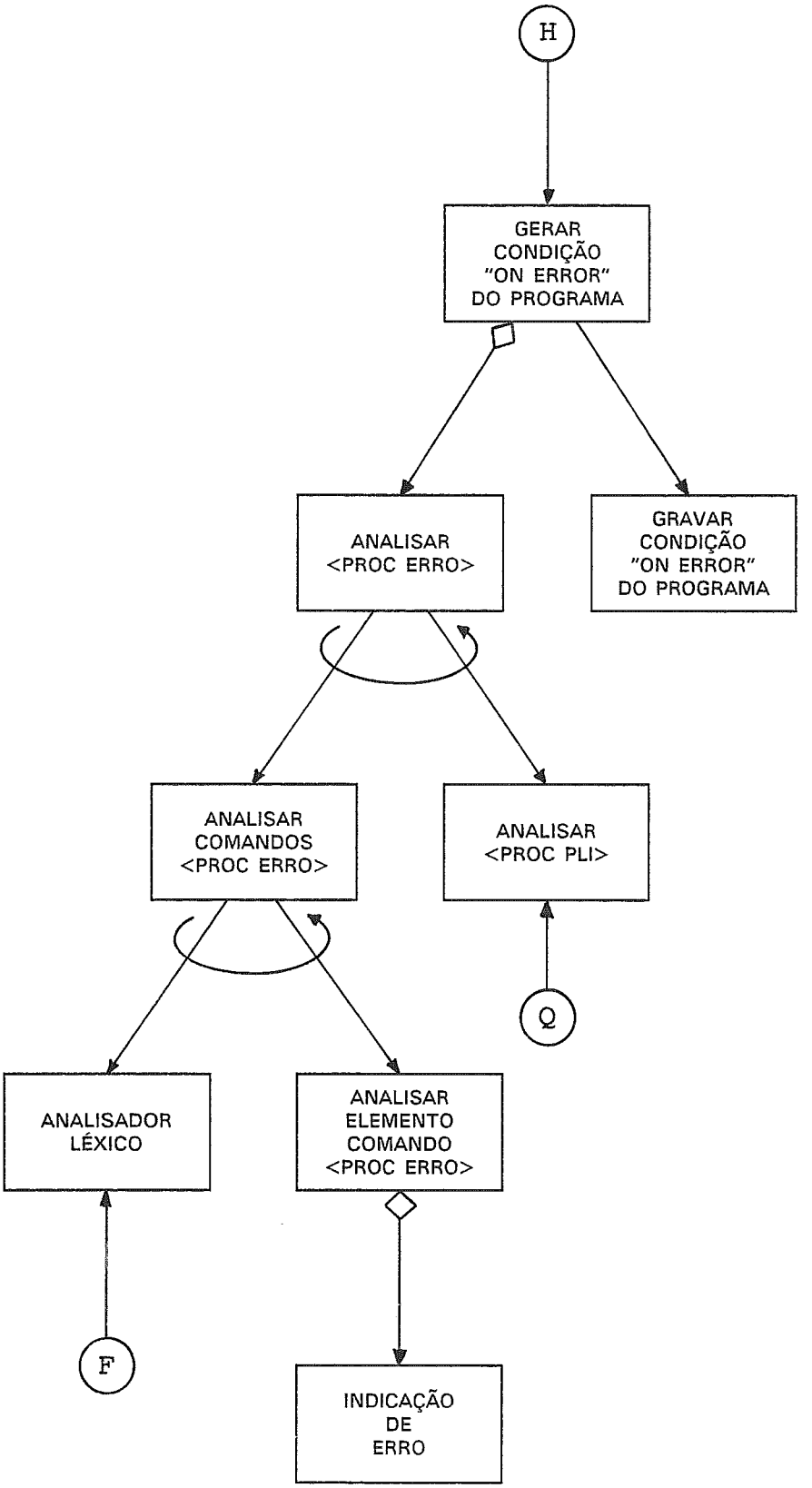


FIGURA D.9 - Estrutura modular "GERAR CONDIÇÃO "ON ERROR" DO PROGRAMA" (nível 2)

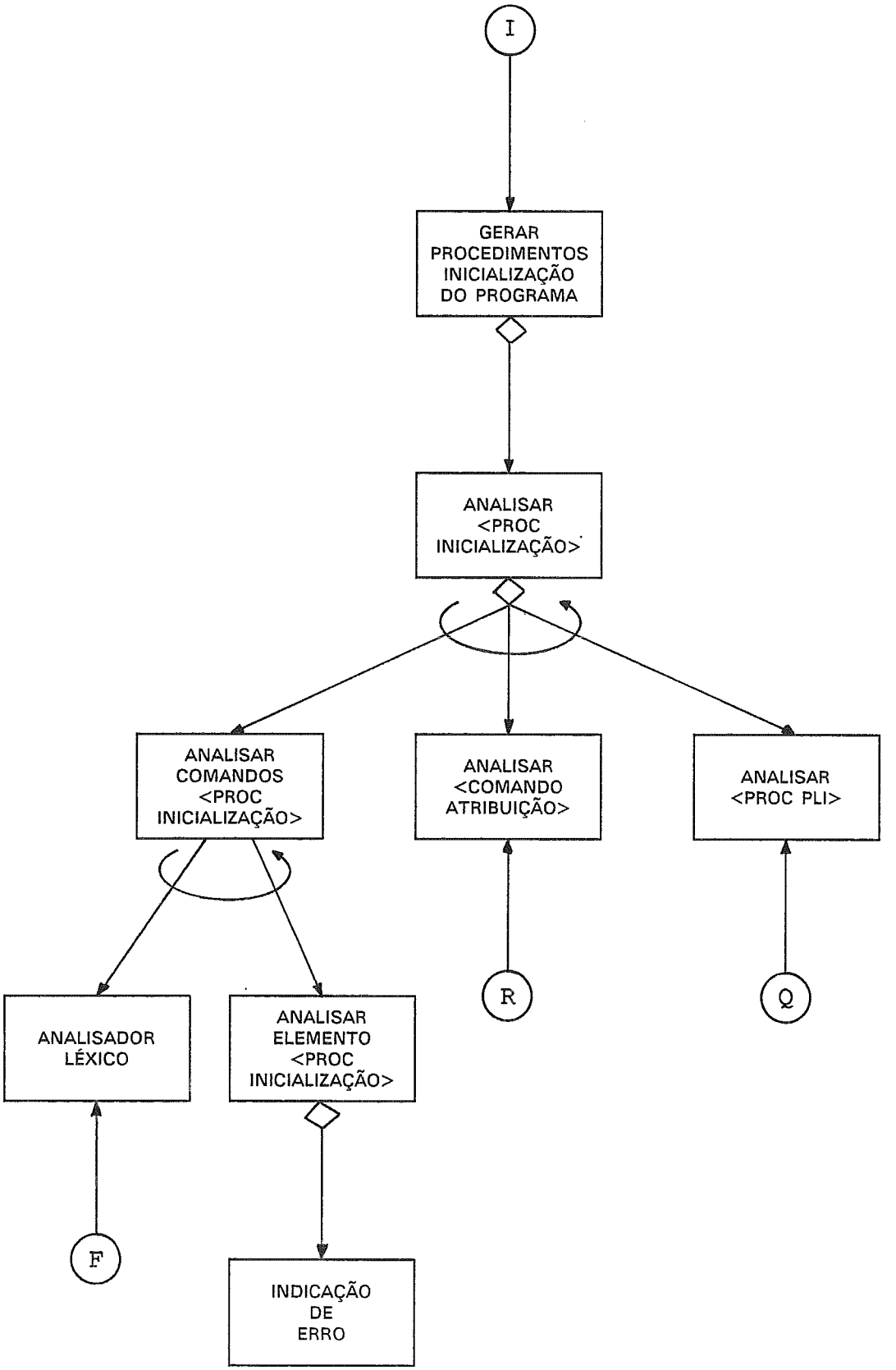


FIGURA D.10 - Estrutura modular "GERAR PROCEDIMENTOS DE INICIALIZAÇÃO DO PROGRAMA" (nível 2)

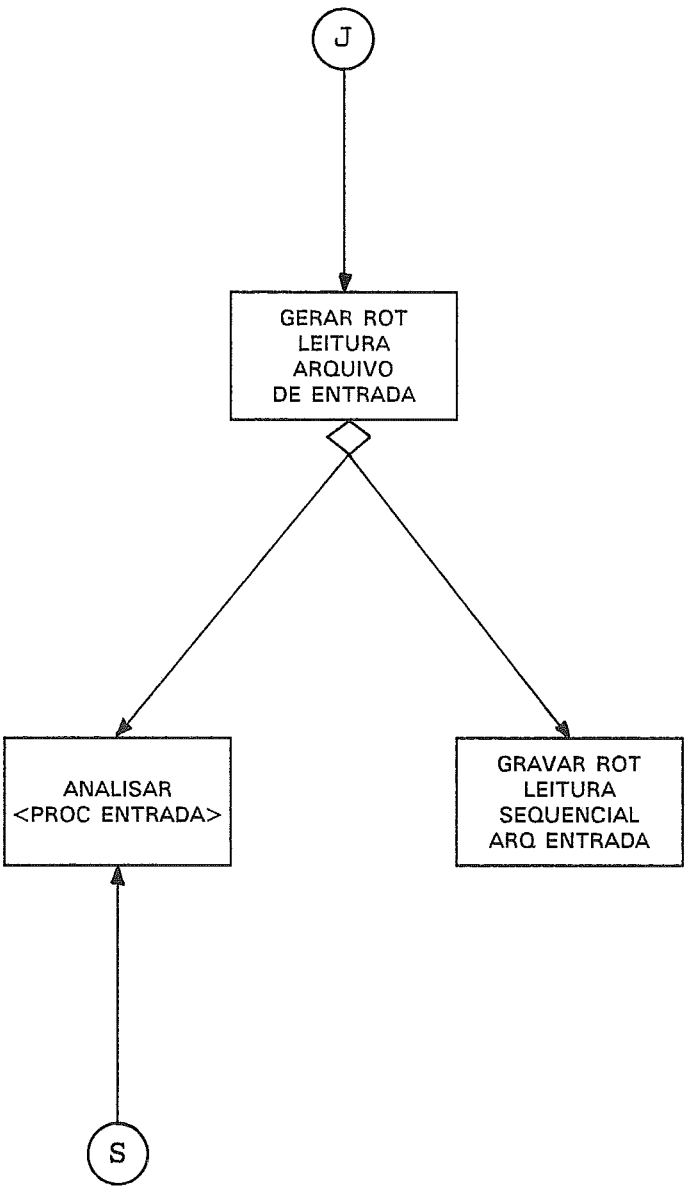


FIGURA D.11 - Estrutura modular "GERAR ROT LEITURA DO ARQUIVO DE ENTRADA" (nível 2)

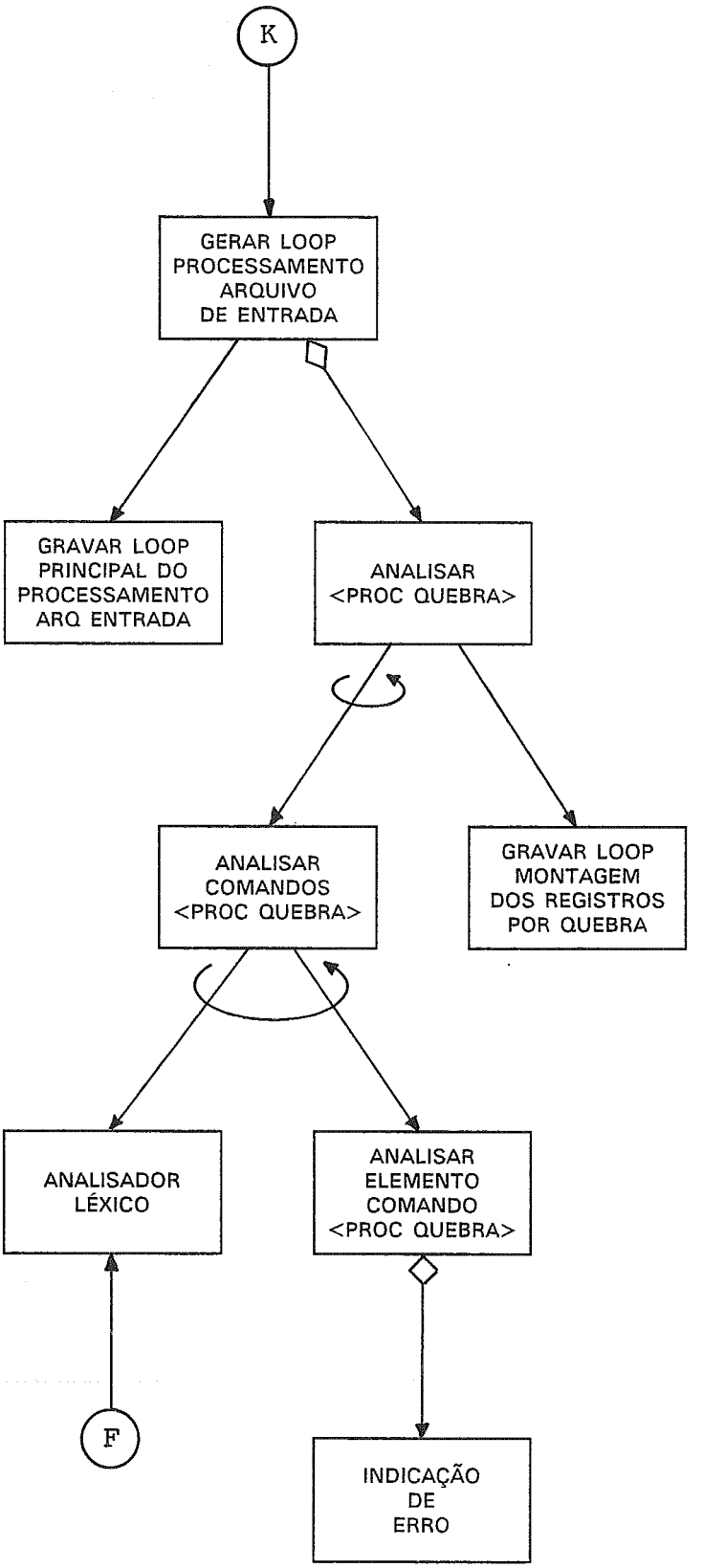


FIGURA D.12 - Estrutura modular "GERAR LOOP PROCESSAMENTO DO ARQUIVO DE ENTRADA" (nível 2)

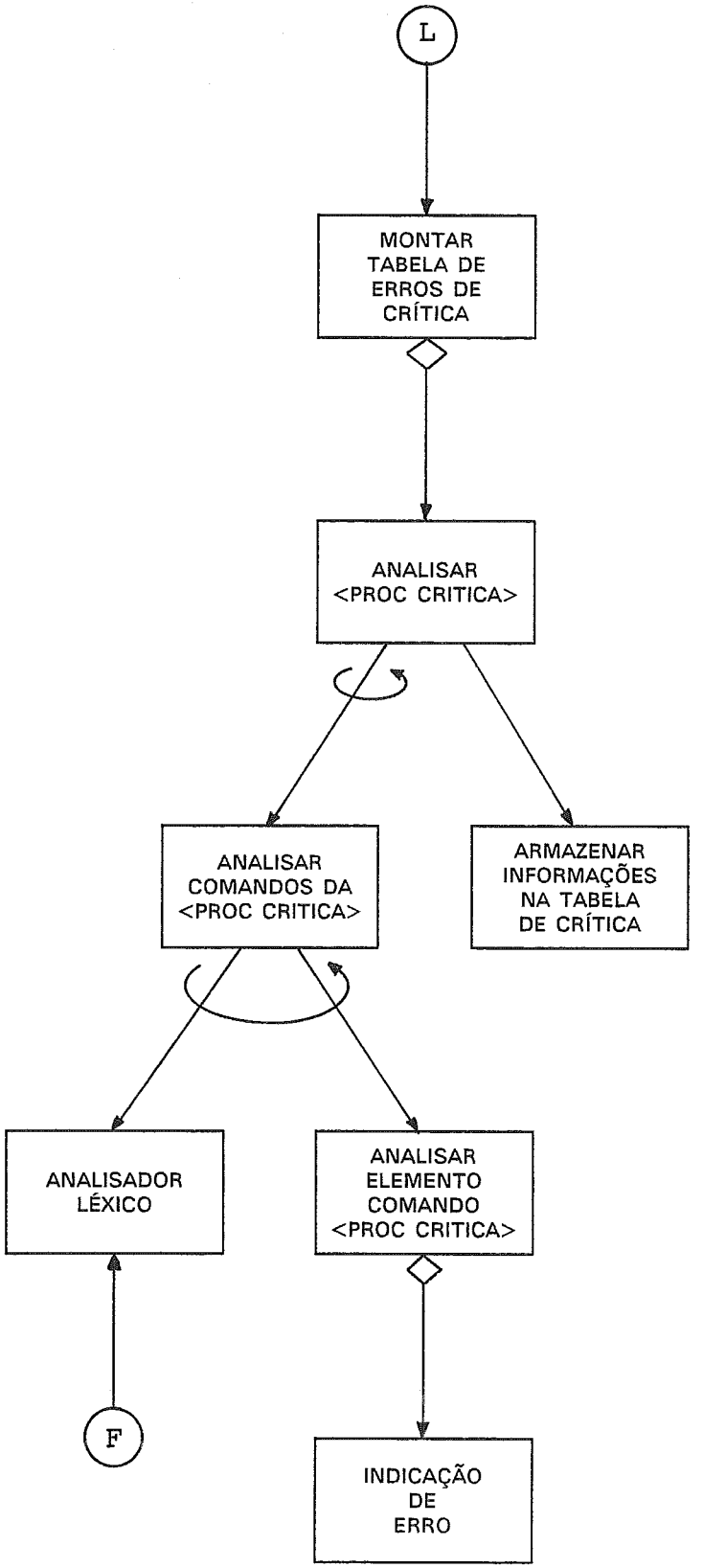


FIGURA D.13 - Estrutura modular "MONTAR TABELA DE ERROS DE CRÍTICA" (nível 2)

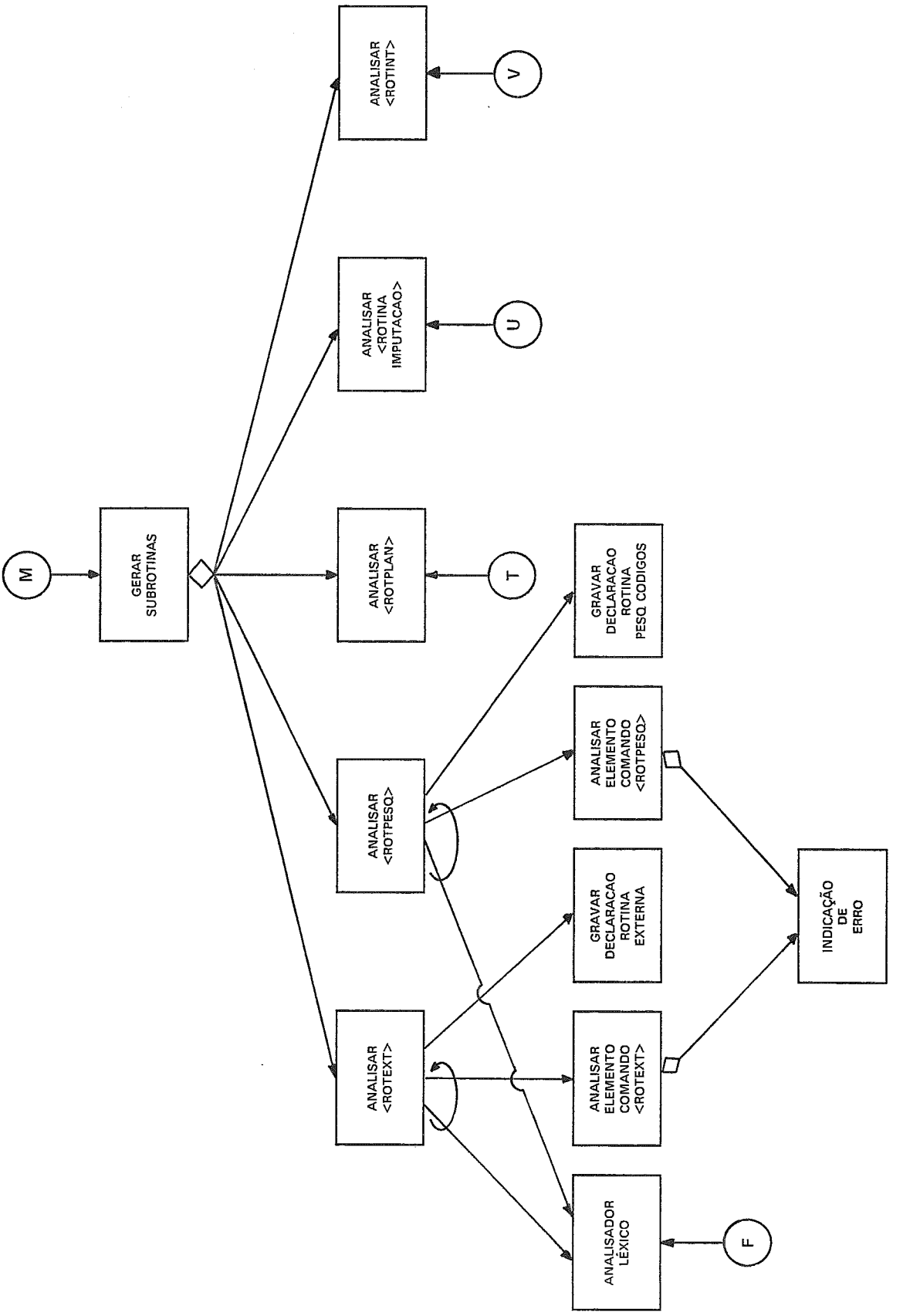


FIGURA D.14 - Estrutura modular "GERAR SUBROTINAS" (nível 2)

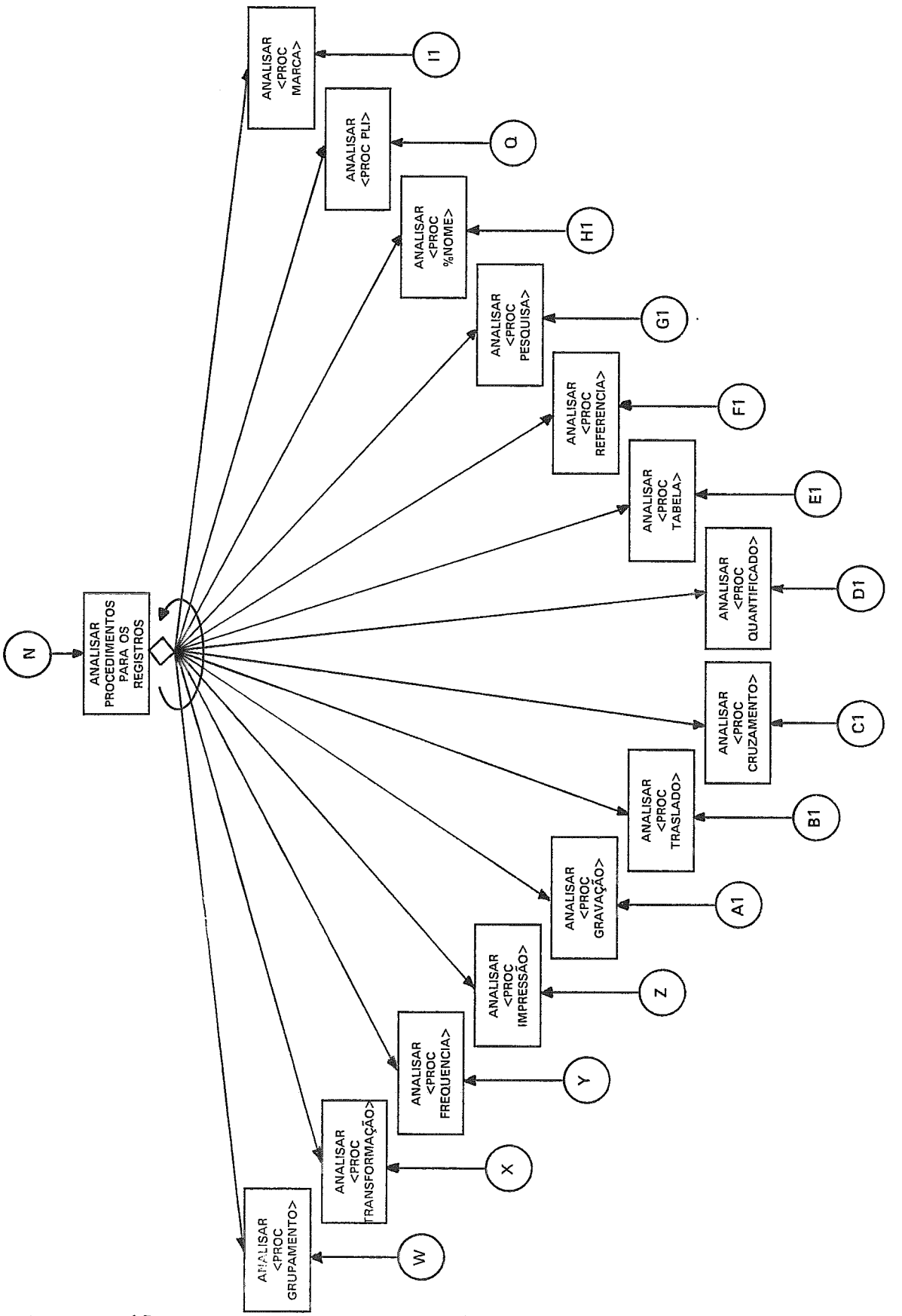


FIGURA D.15 - Estrutura modular "ANALISAR PROCEDIMENTOS PARA OS REGISTROS" (nível 2)

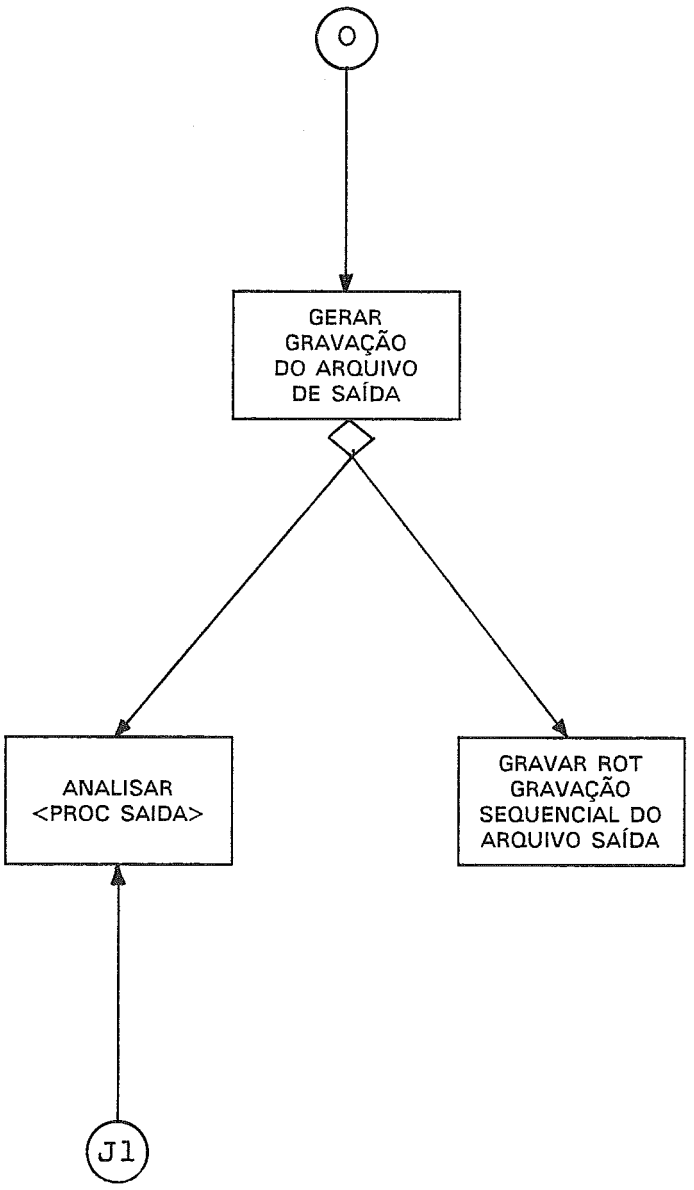


FIGURA D.16 - Estrutura modular "GERAR GRAVAÇÃO DO ARQUIVO DE SAÍDA(nível 2)

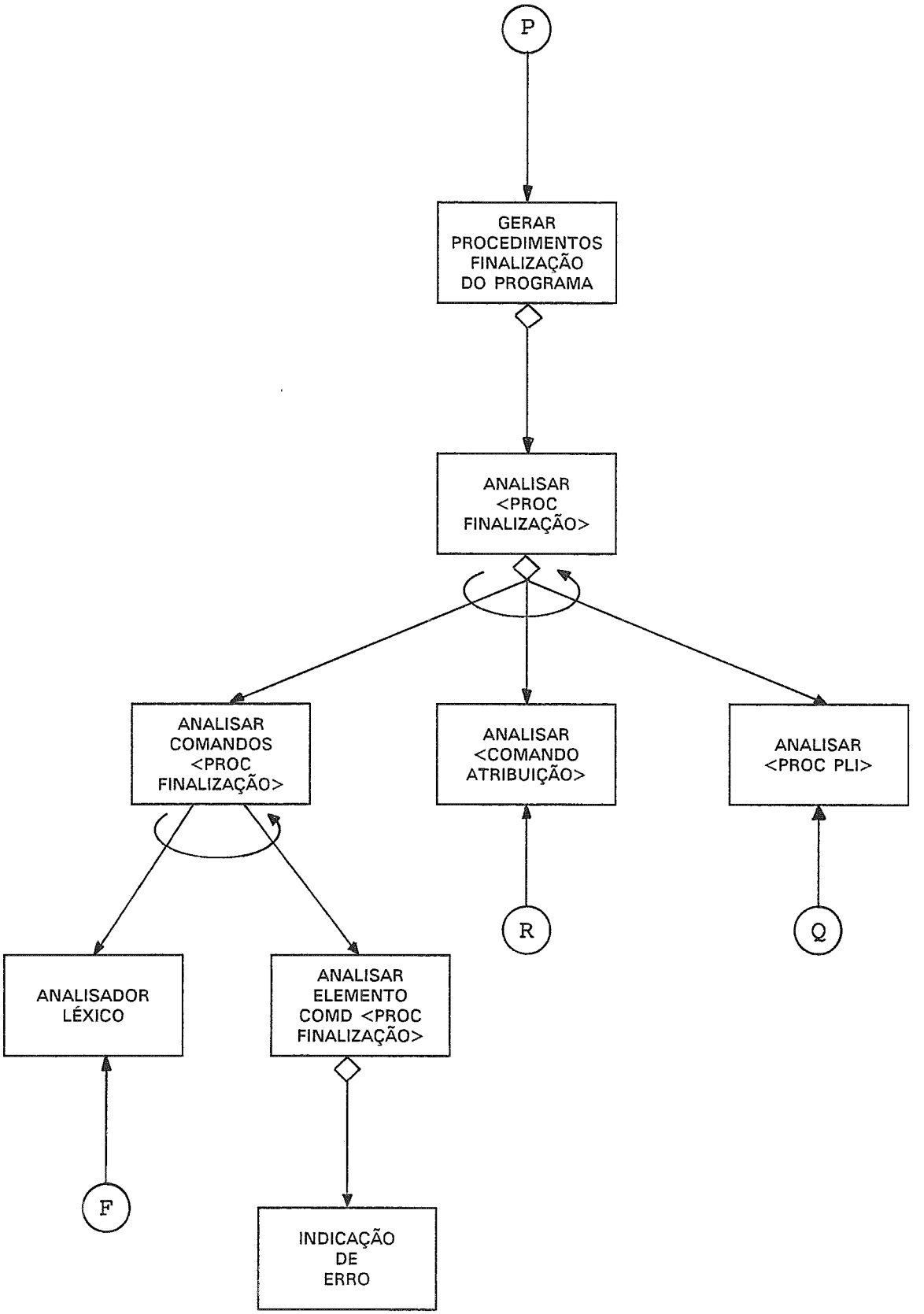


FIGURA D.17 - Estrutura modular "GERAR PROCEDIMENTOS DE FINALIZAÇÃO DO PROGRAMA" (nível 2)

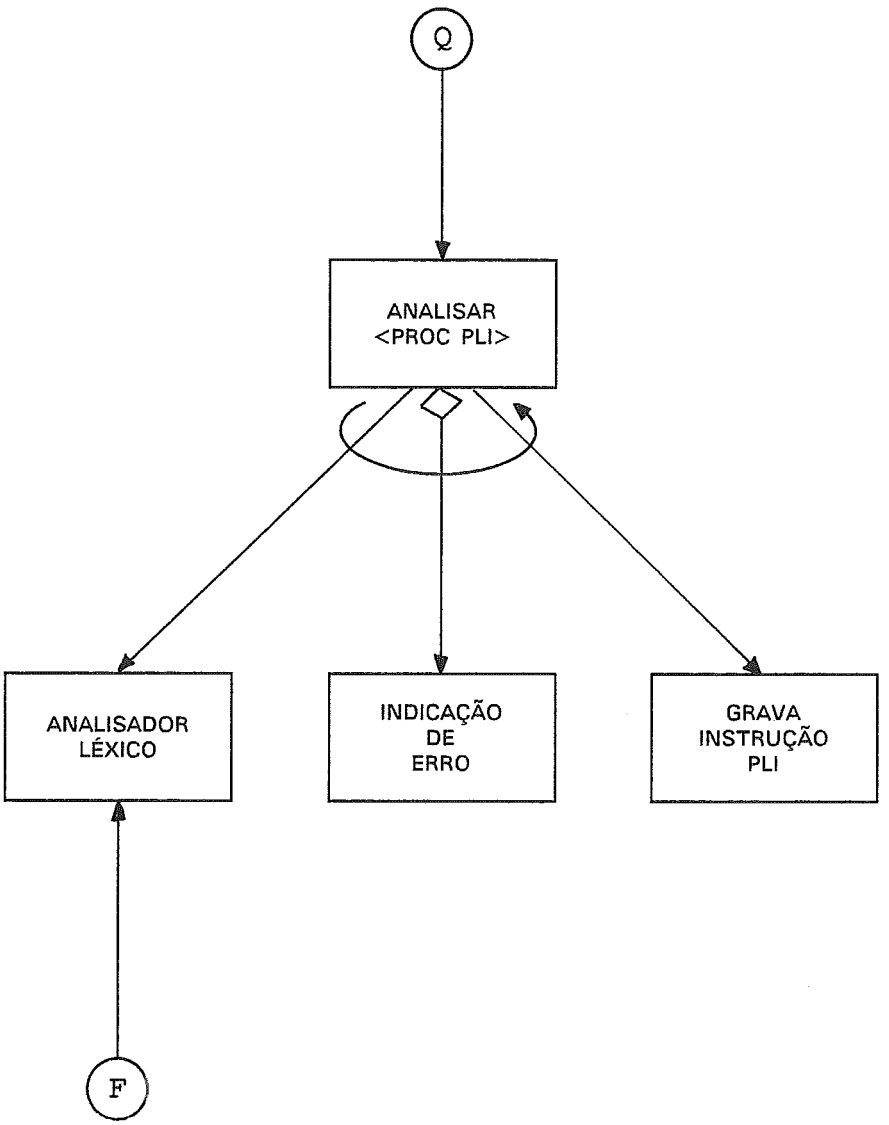


FIGURA D.18 - Estrutura modular "ANALISAR <PROC PLI>"

(nível 2)

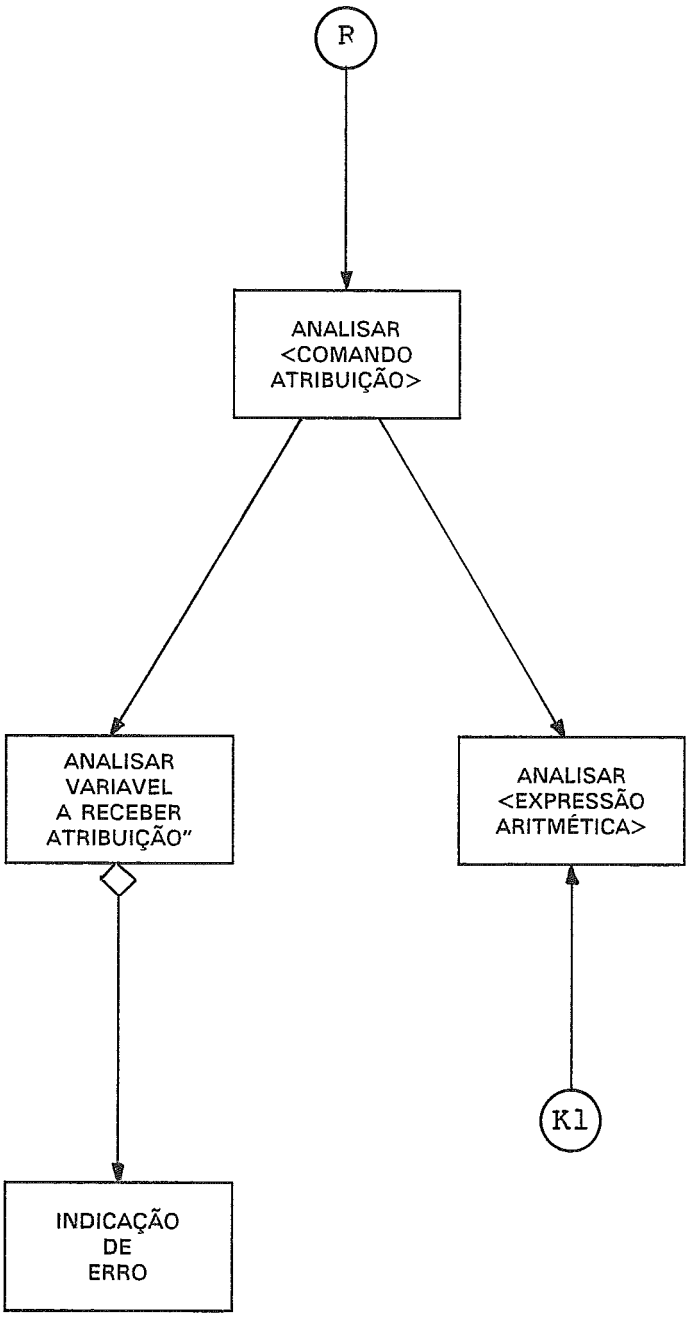


FIGURA D.19 - Estrutura modular "ANALISAR <COMANDO ATRIBUIÇÃO>" (nível 3)

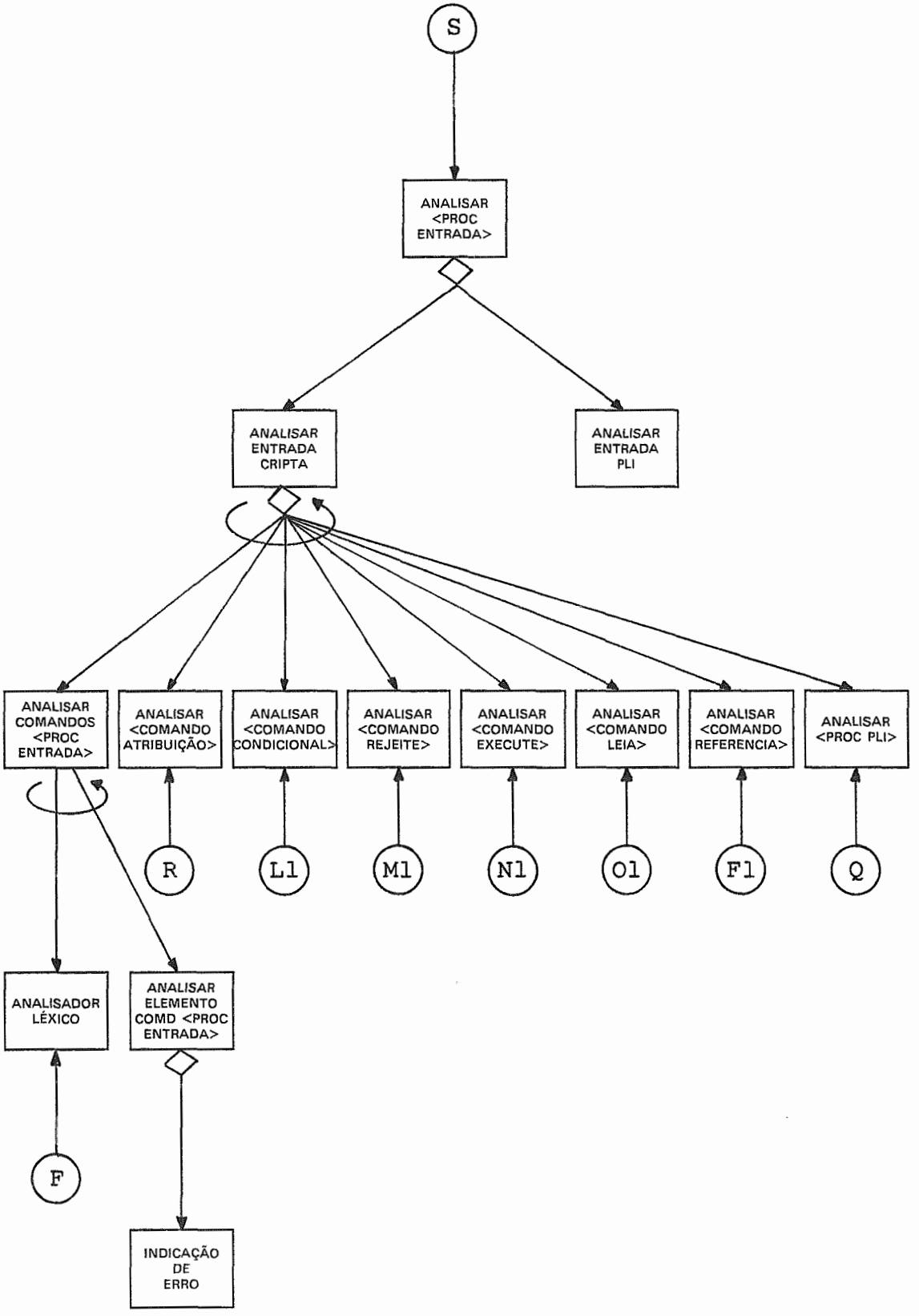


FIGURA D.20 - Estrutura modular "ANALISAR <PROC ENTRADA>" (nível 3)

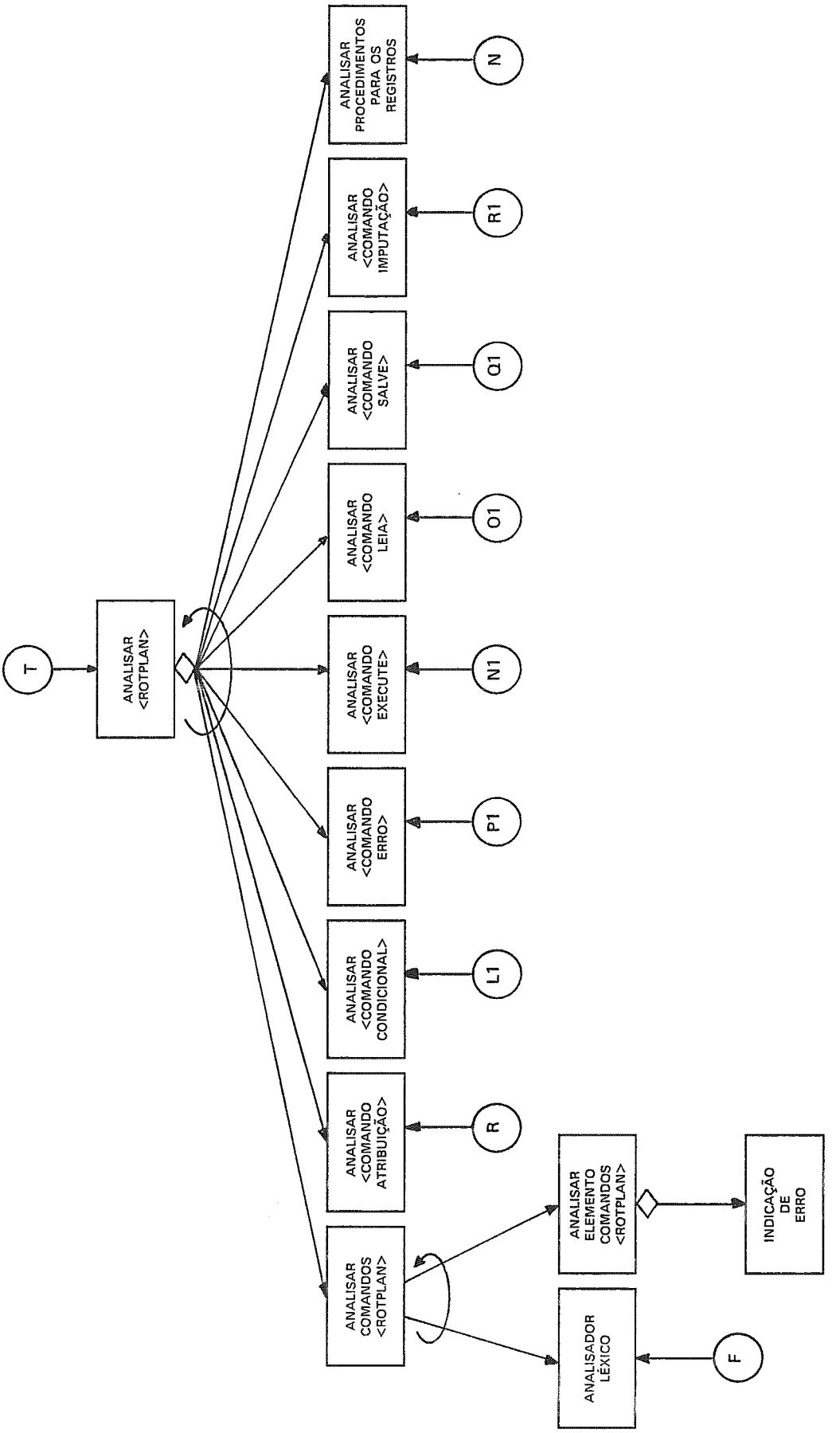


FIGURA D.21 - Estrutura modular "ANALISAR <ROTPLAN>" (nível 3)

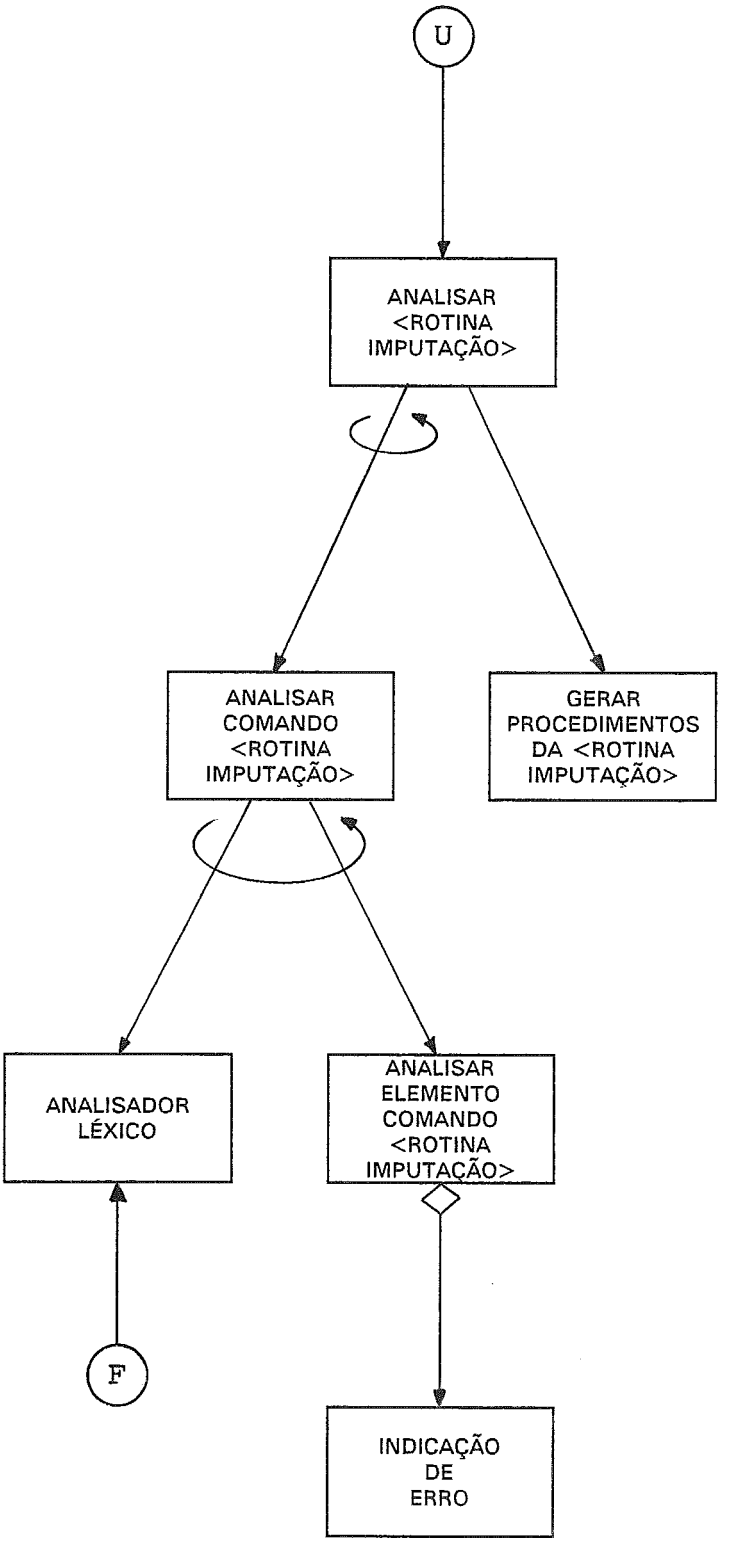


FIGURA D.22 - Estrutura modular "ANALISAR <ROTINA IMPUTAÇÃO>" (nível 3)

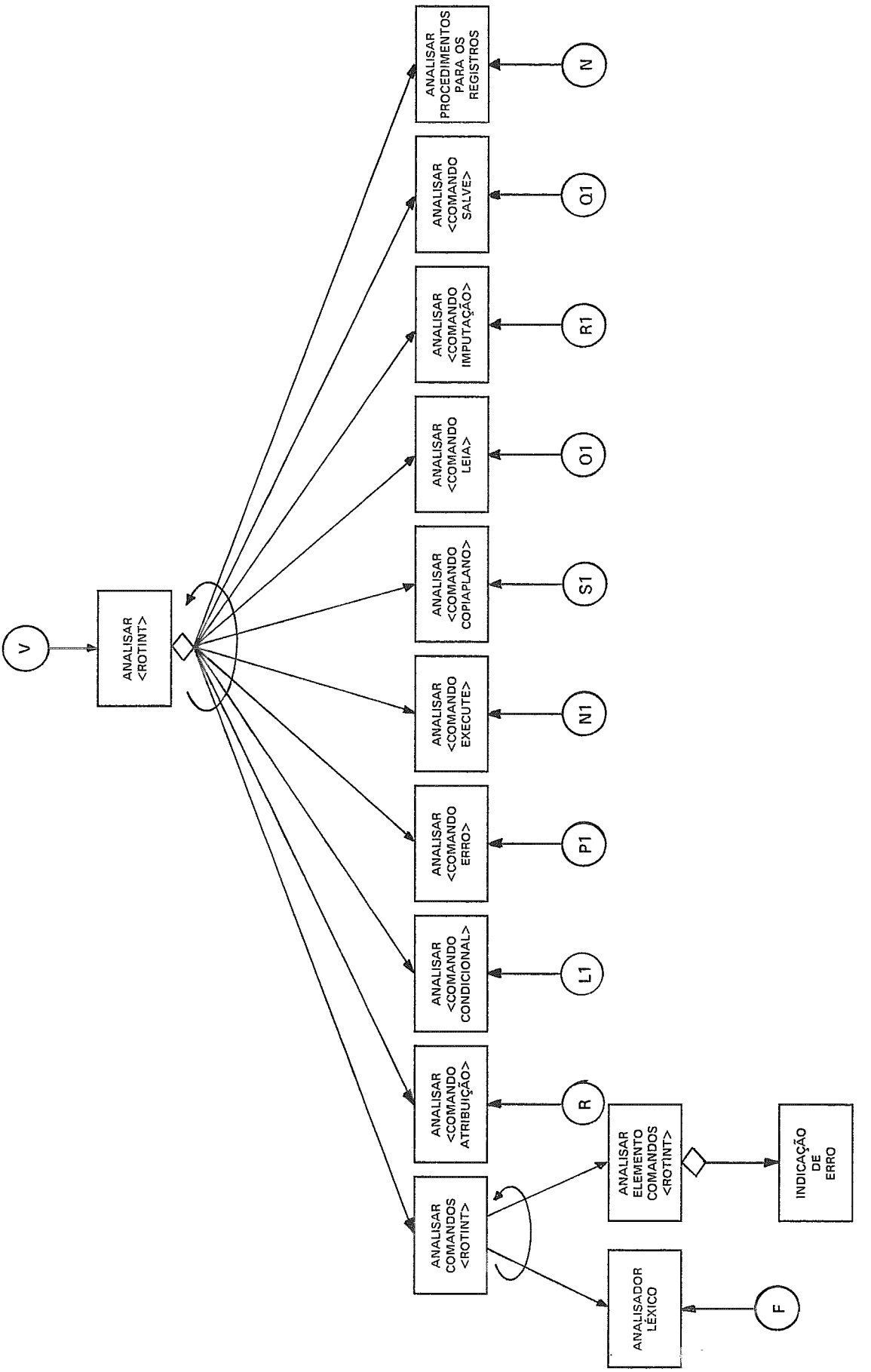


FIGURA D.23 - Estrutura modular "ANALISAR <ROTINT>" (nível 3)

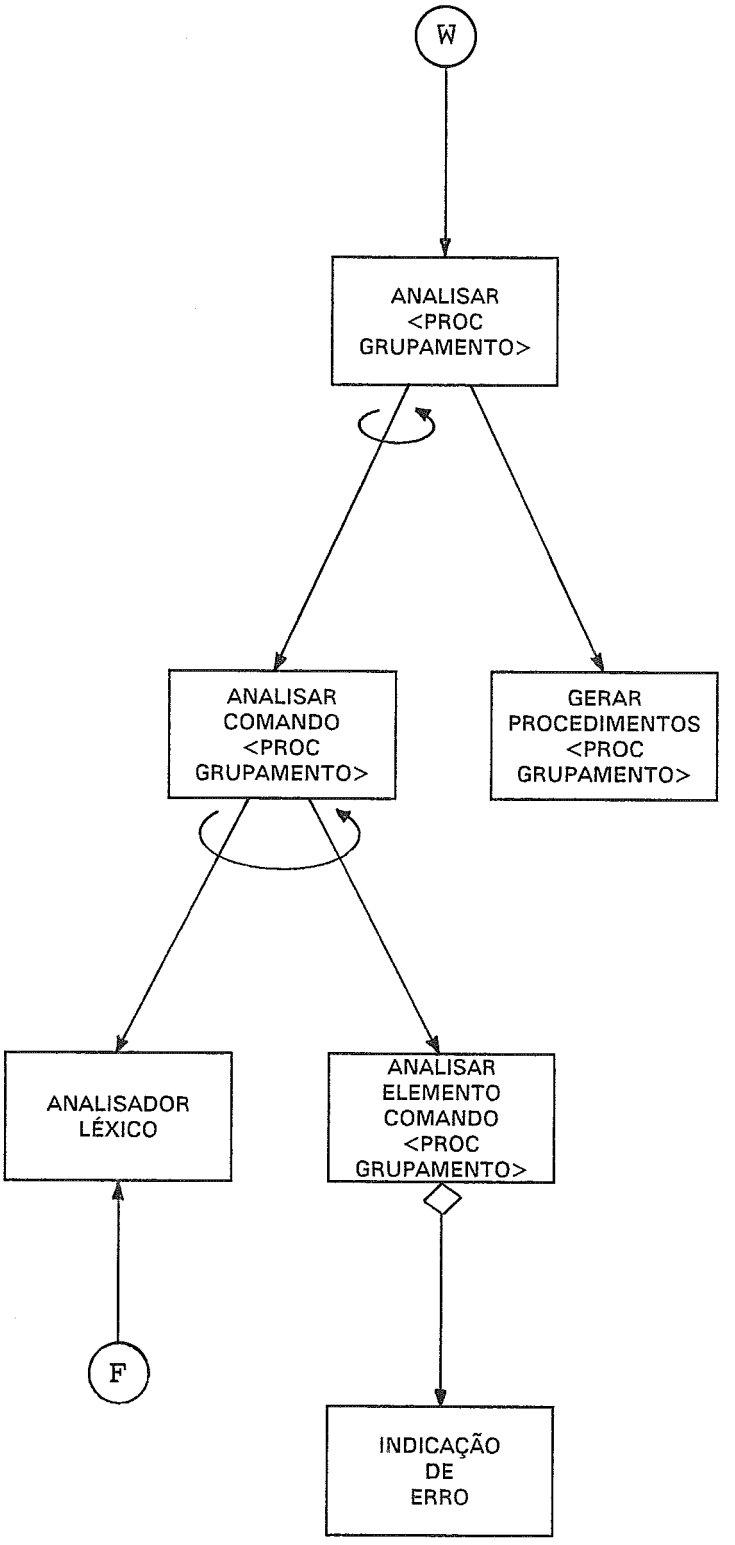


FIGURA D.24 - Estrutura modular "ANALISAR <PROC GRUPAMENTO>" (nível 3)

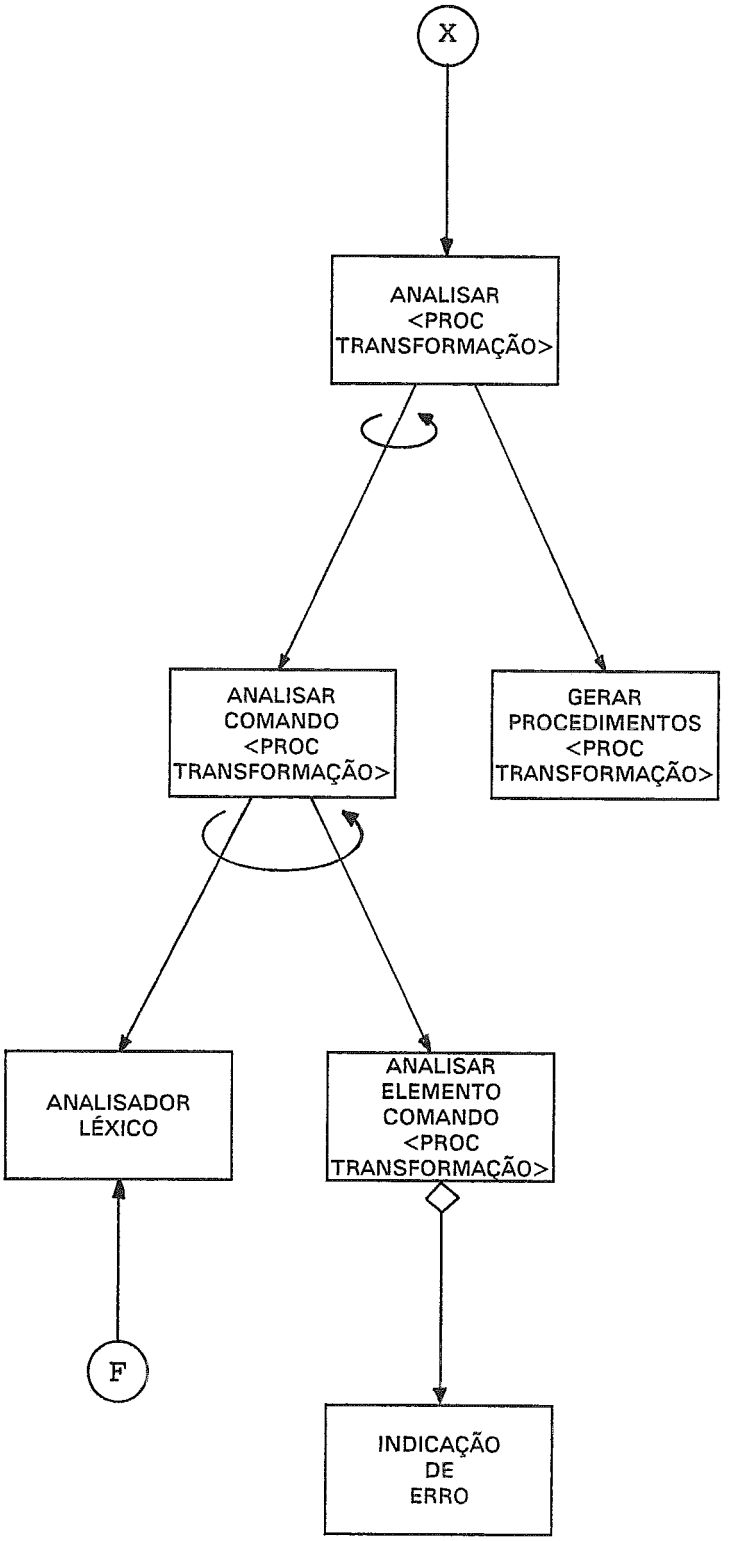


FIGURA D.25 - Estrutura modular "ANALISAR <PROC TRANSFORMAÇÃO>" (nível 3)

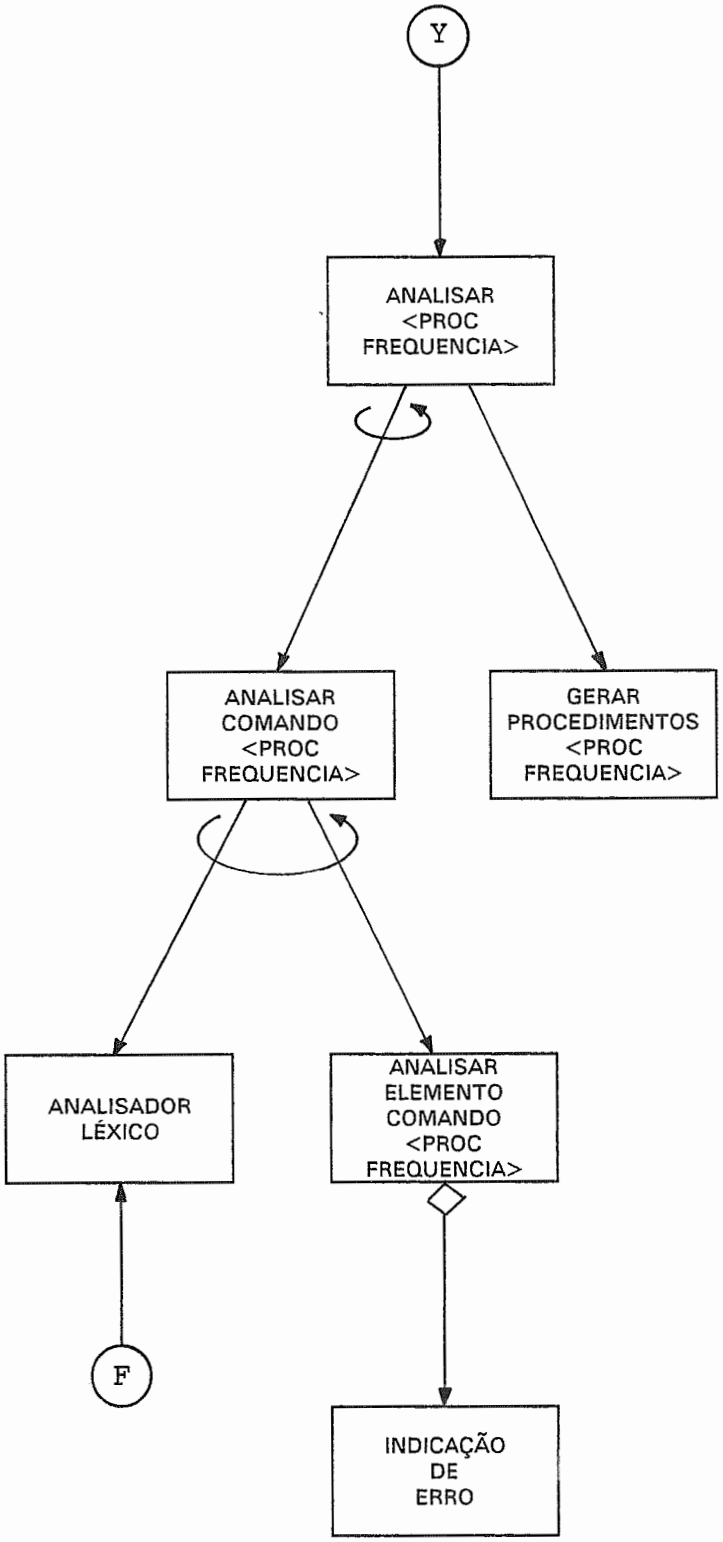


FIGURA D.26 - Estrutura modular "ANALISAR <PROC FREQUENCIA>" (nível 3)

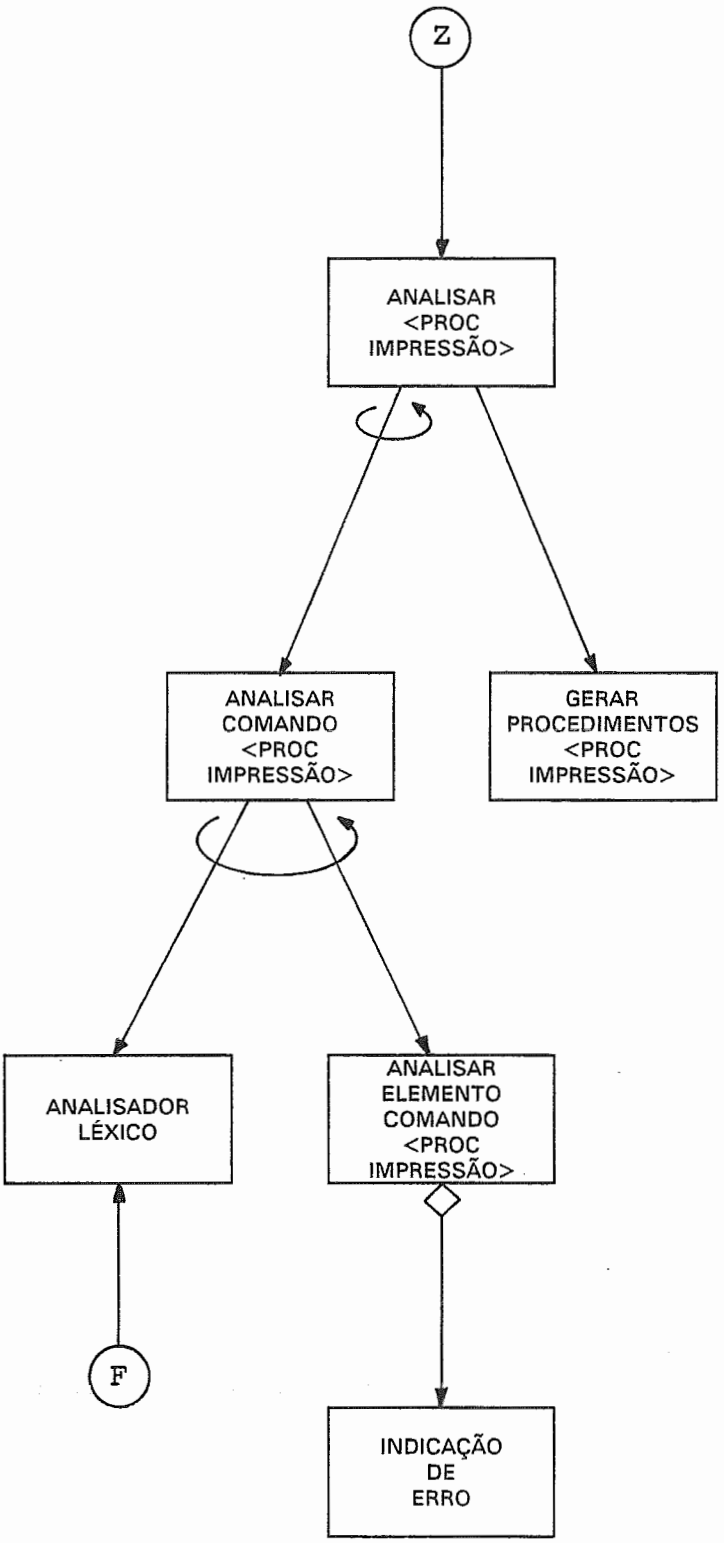


FIGURA D.27 - Estrutura modular "ANALISAR <PROC IMPRESSÃO>" (nível 3)

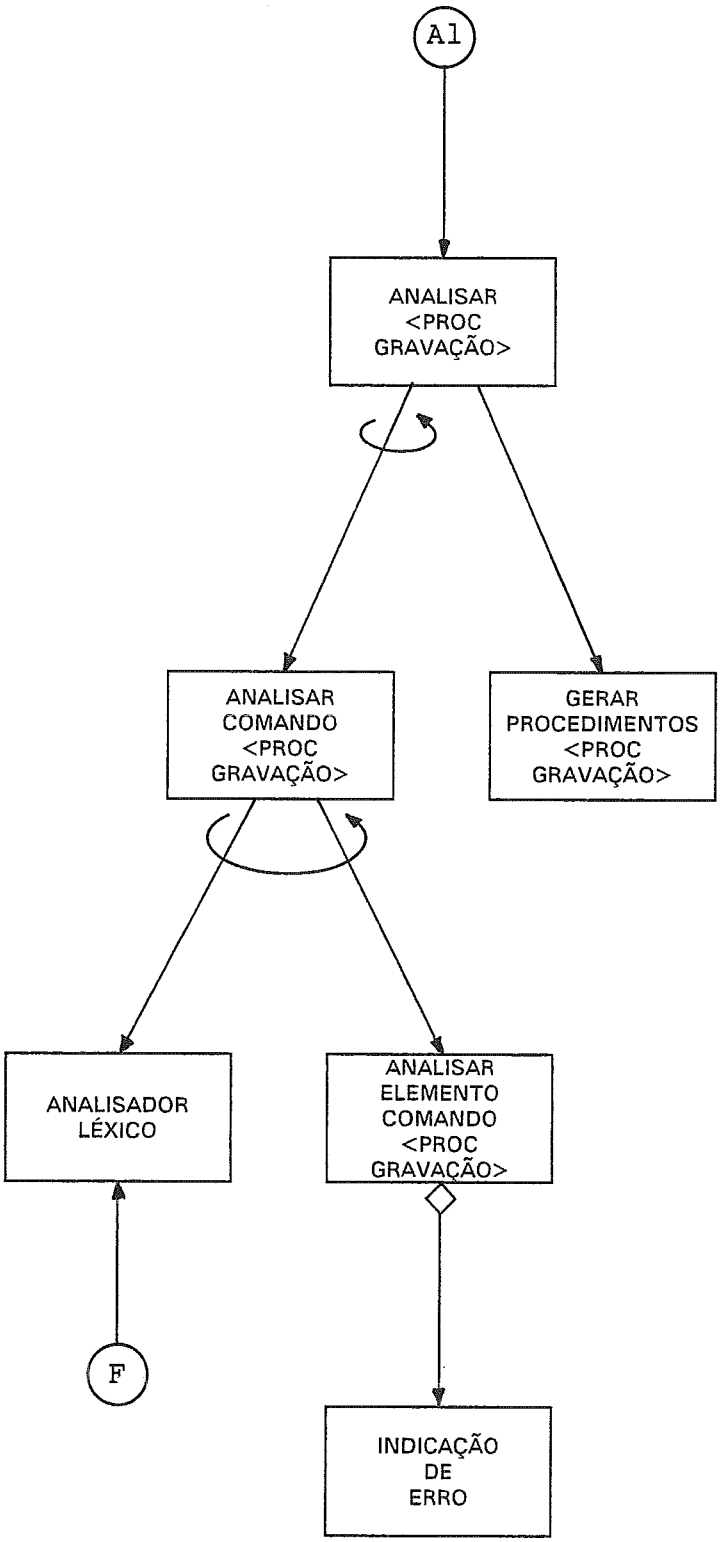


FIGURA D.28 - Estrutura modular "ANALISAR <PROC GRAVAÇÃO>" (nível 3)

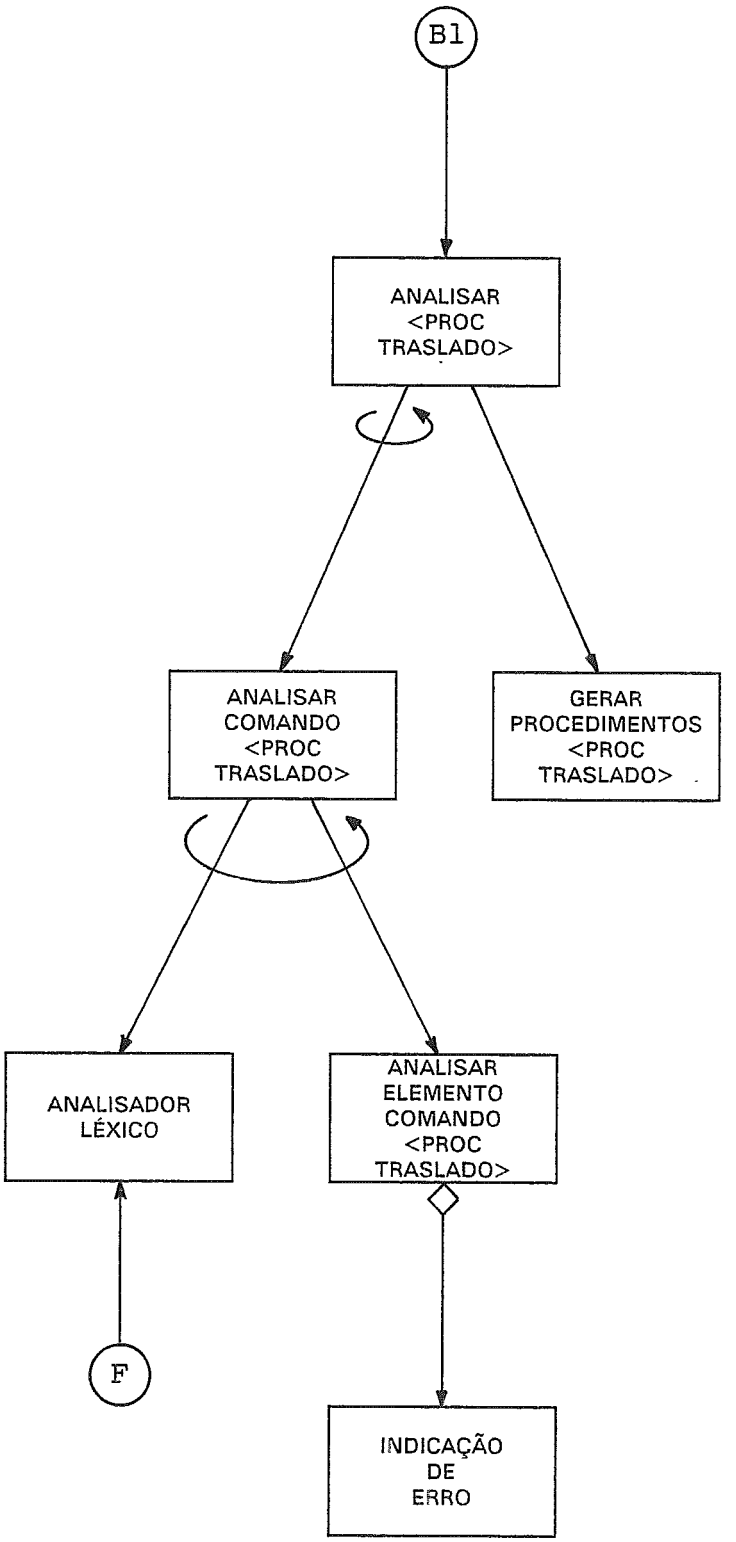


FIGURA D.29 - Estrutura modular "ANALISAR <PROC TRASLADO>" (nível 3)

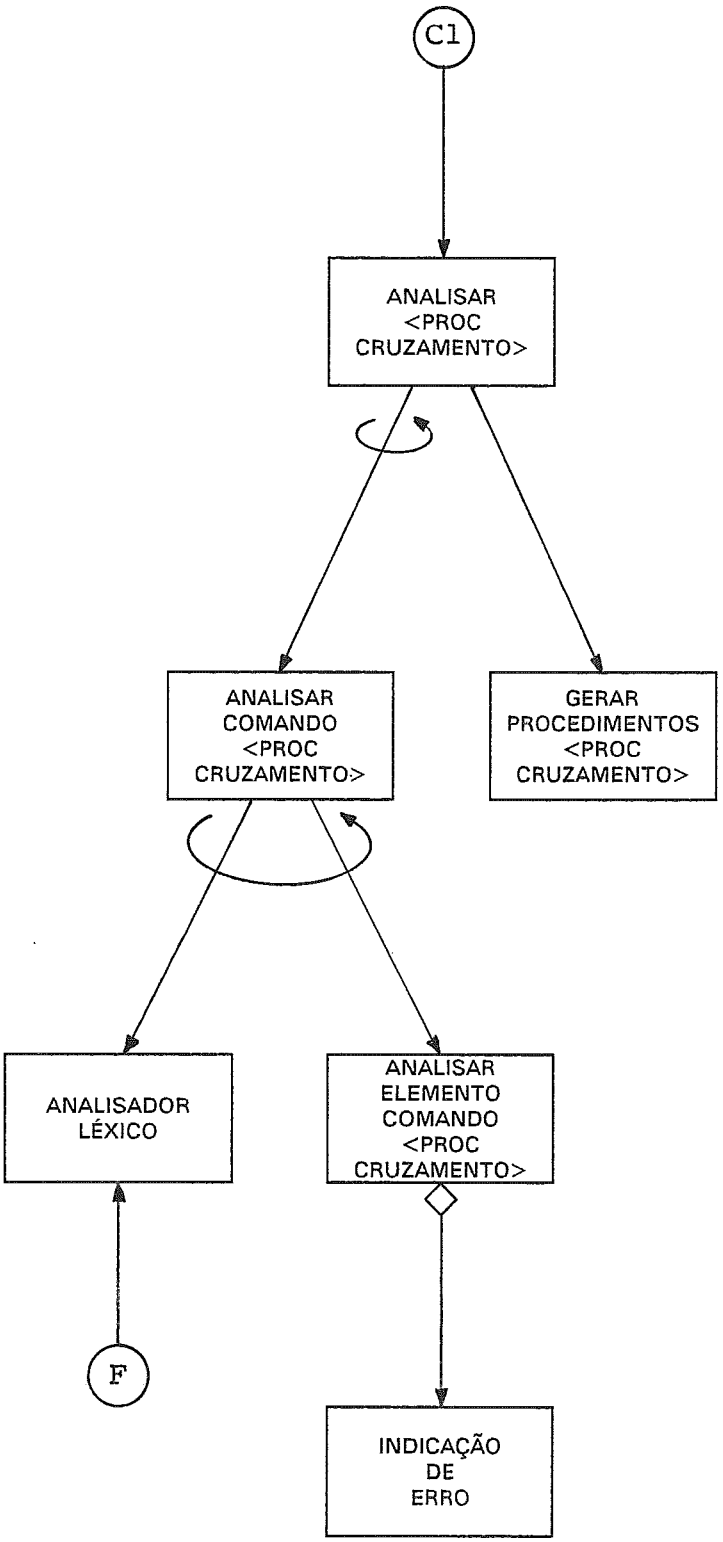


FIGURA D.30 - Estrutura modular "ANALISAR <PROC CRUZAMENTO>" (nível 3)

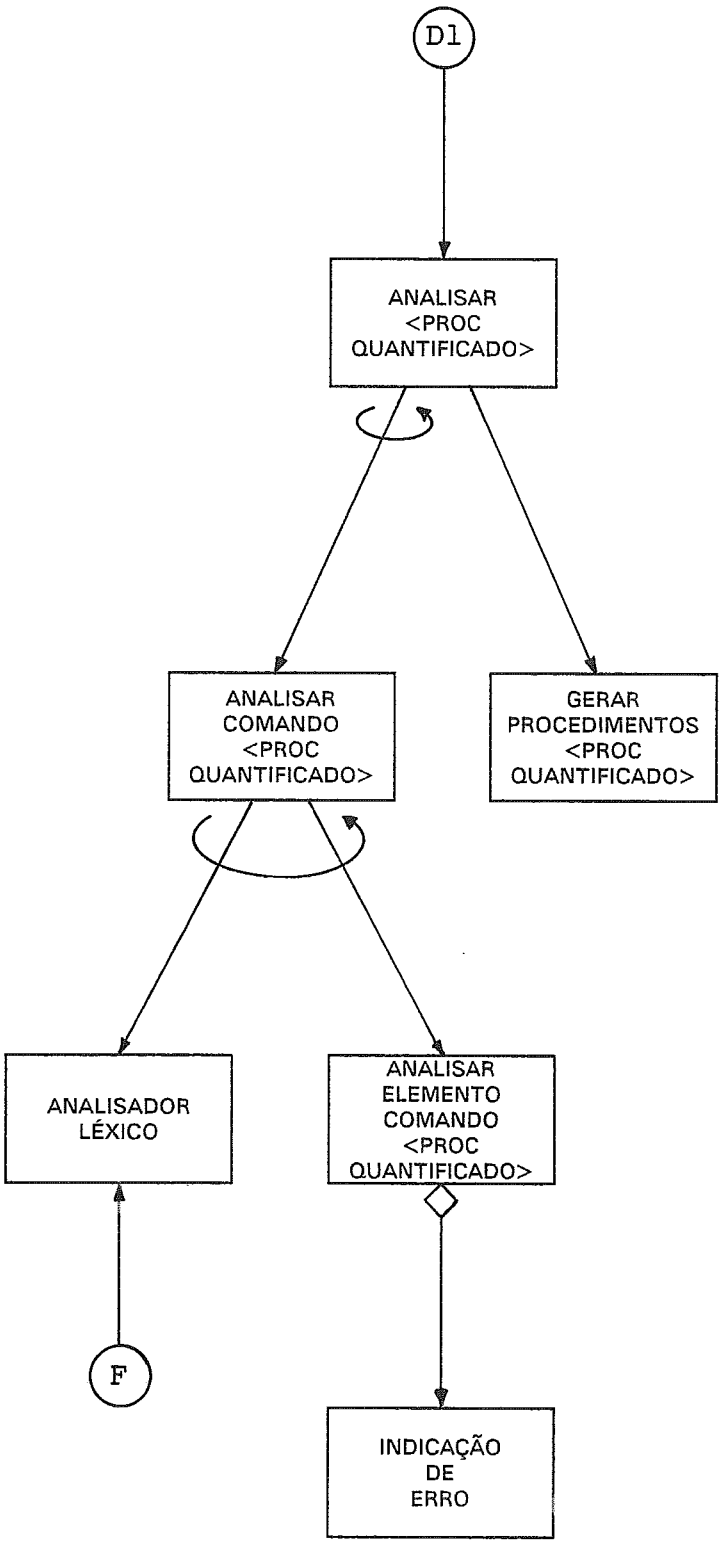


FIGURA D.31 - Estrutura modular "ANALISAR <PROC QUANTIFICADO>" (nível 3)

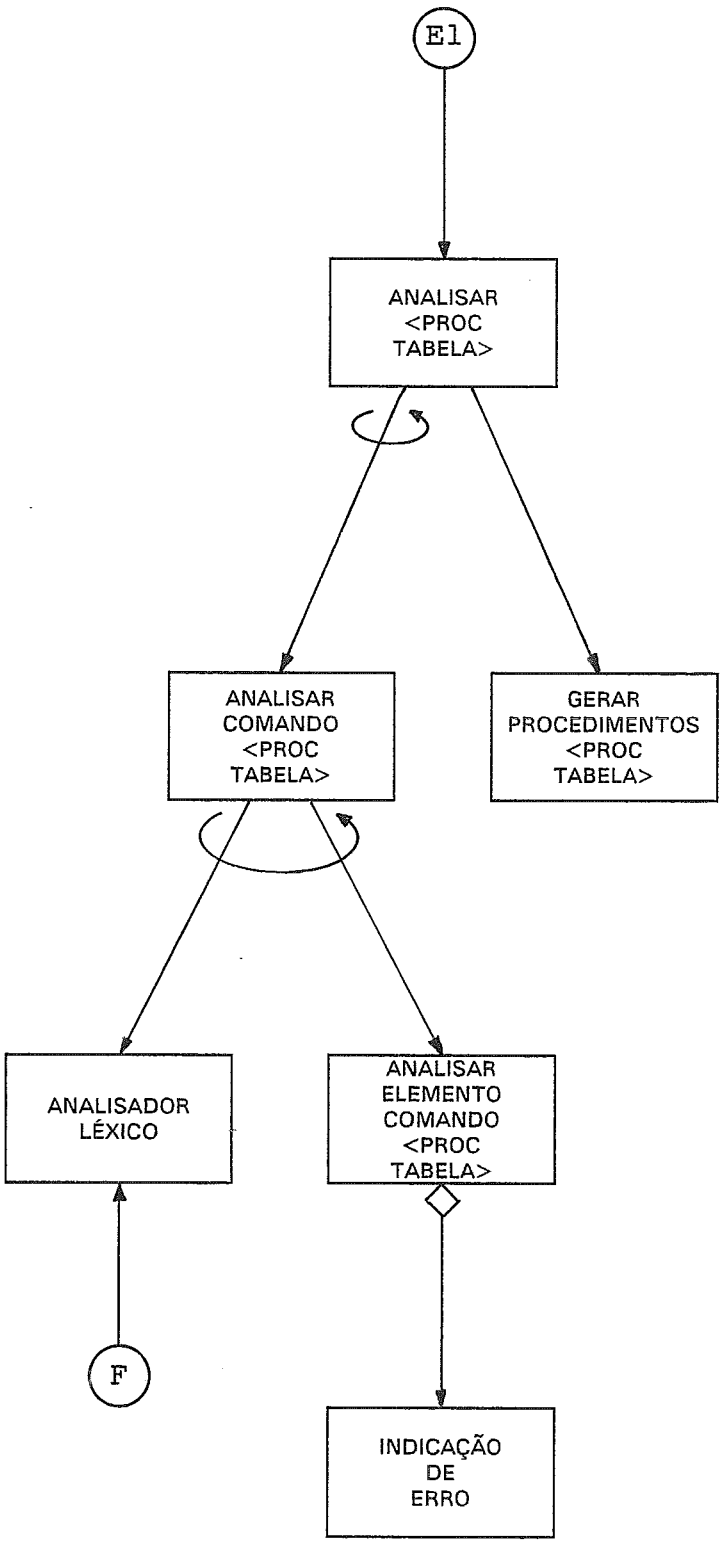


FIGURA D.32 - Estrutura modular "ANALISAR <PROC TABELA>" (nível 3)

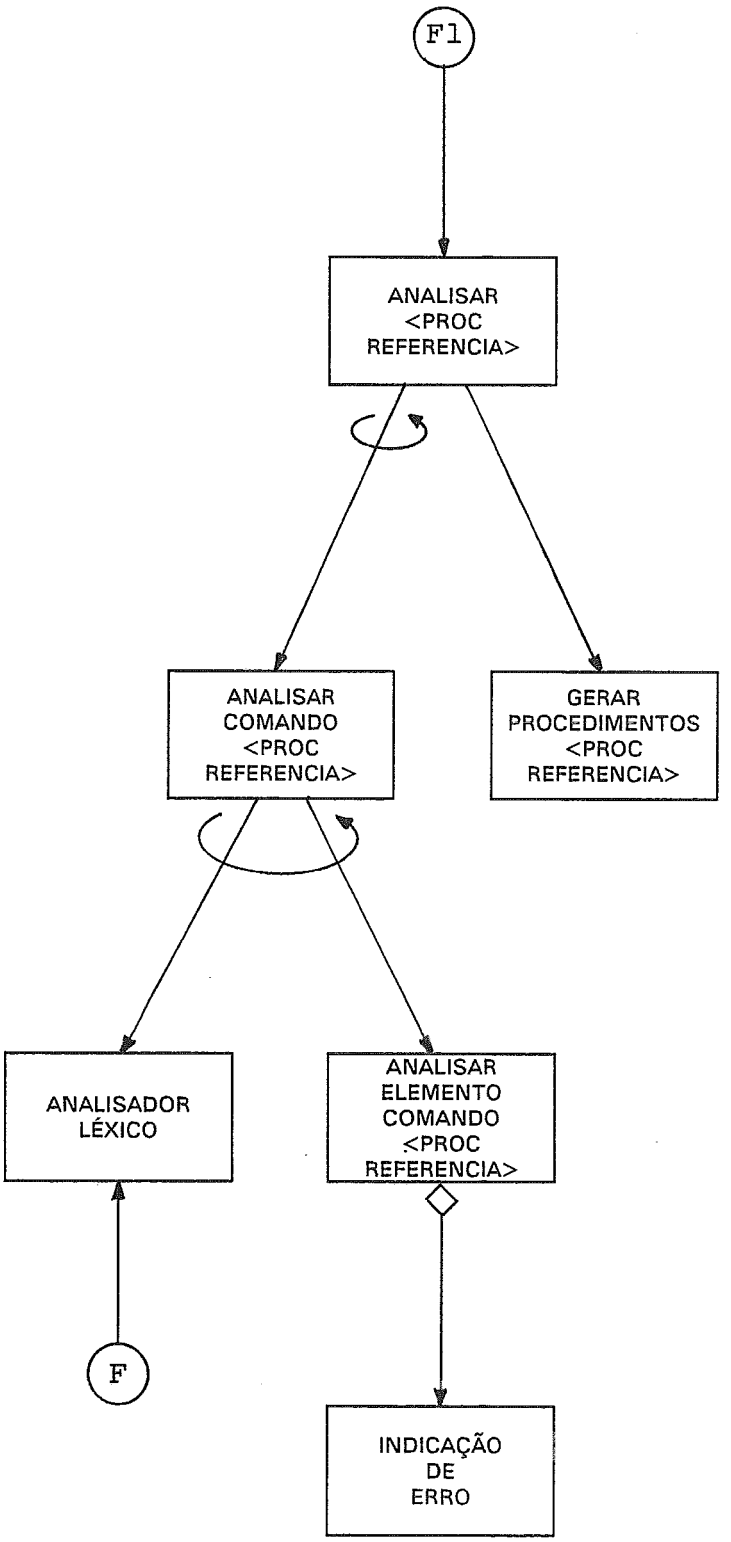


FIGURA D.33 - Estrutura modular "ANALISAR <PROC REFERENCIA>" (nível 3)

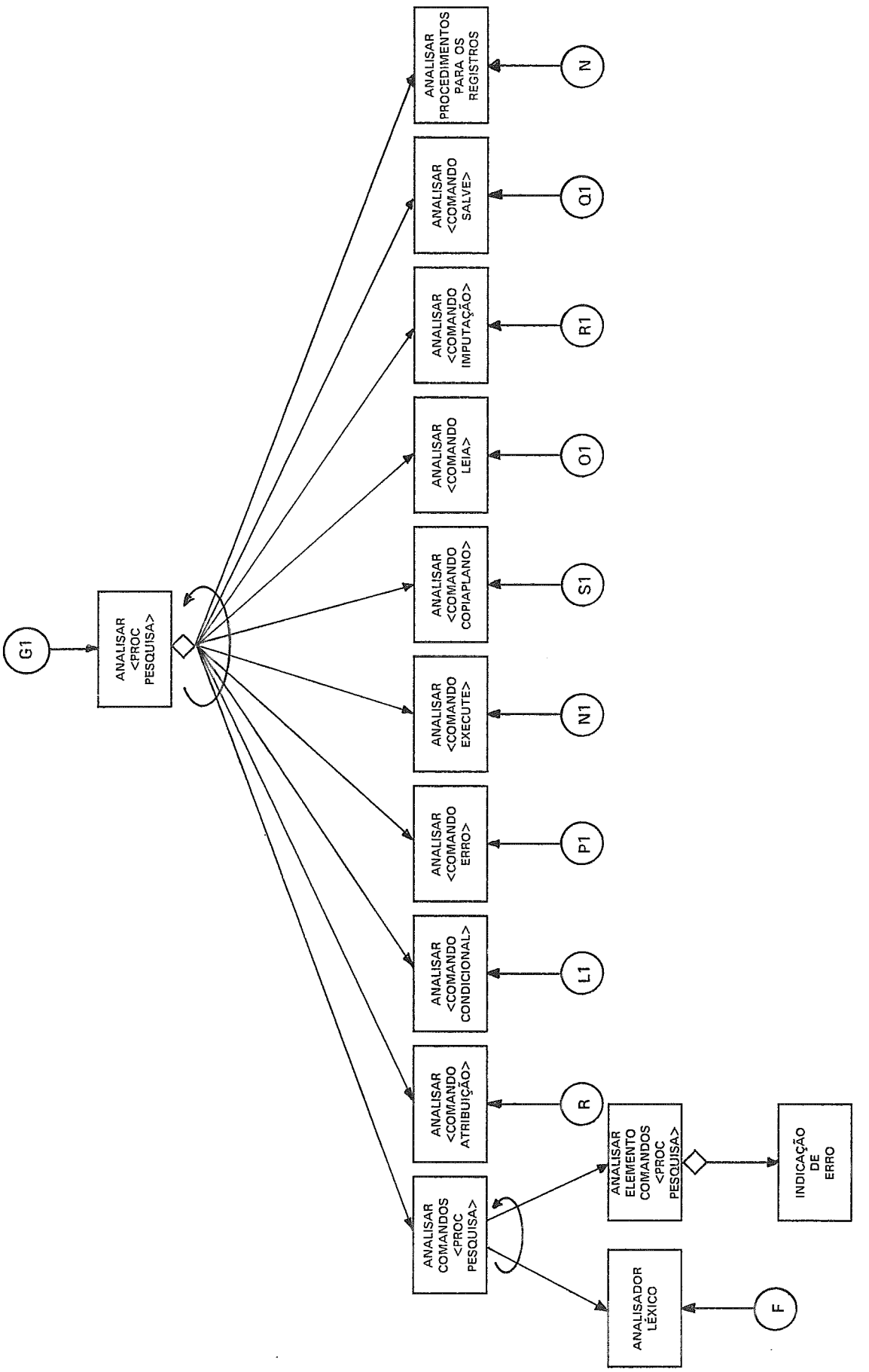


FIGURA D.34 - Estrutura modular "ANALISAR <PROC PESQUISA>" (nível 3)

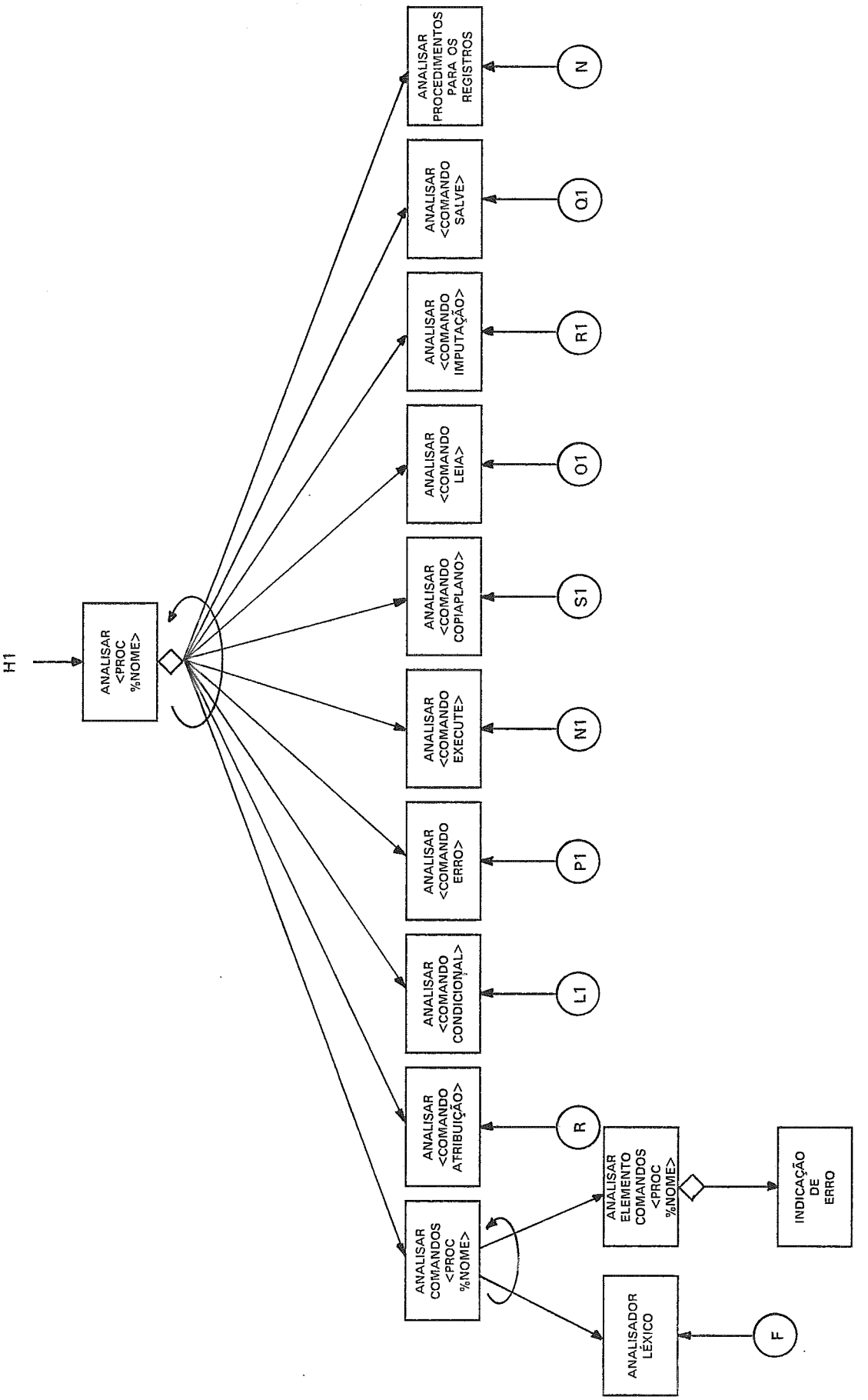


FIGURA D.35 - Estrutura modular "ANALISAR <PROC %NOME>" (nível 3)

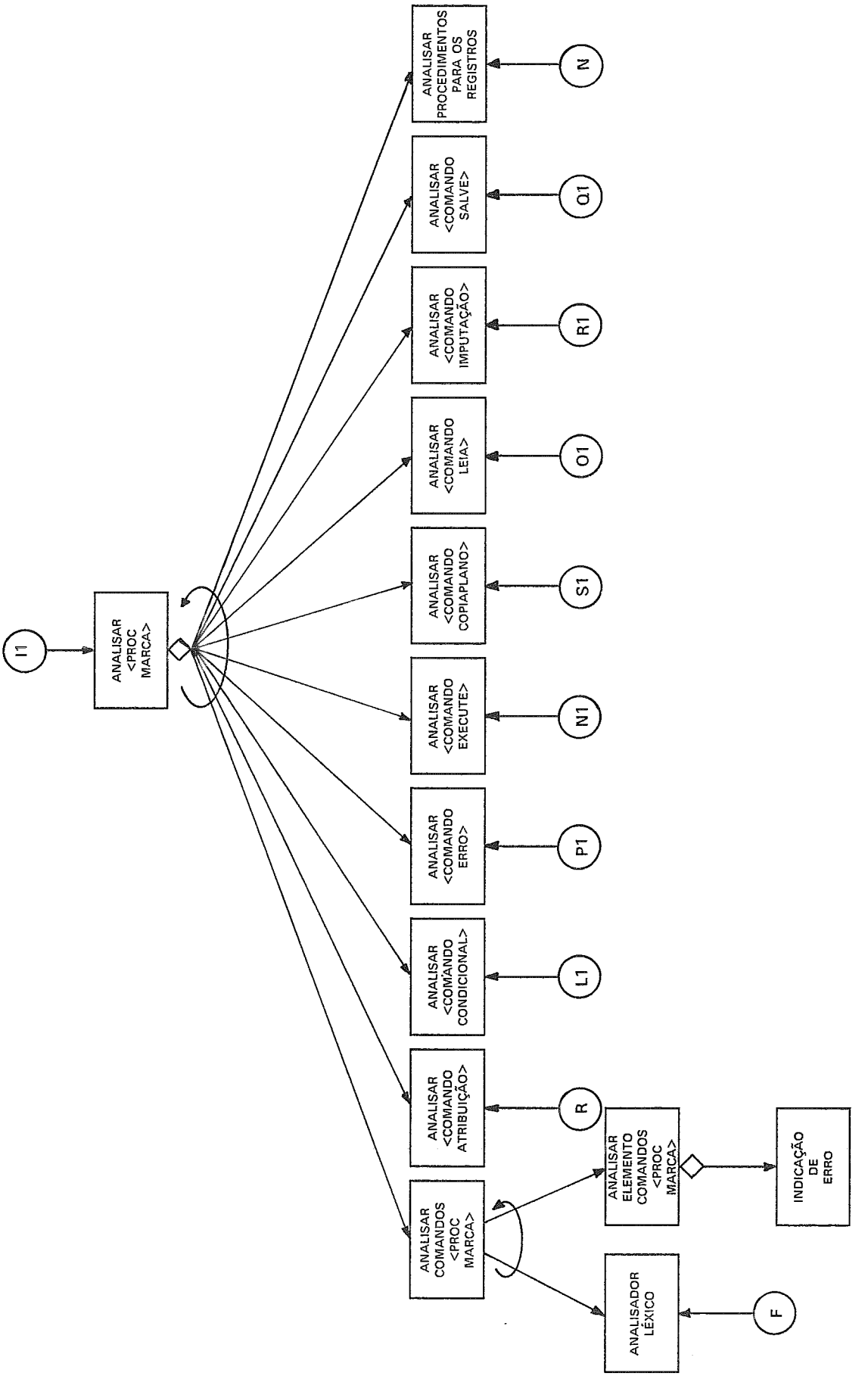


FIGURA D.36 - Estrutura modular "ANALISAR <PROC MARCA>" (nível 3)

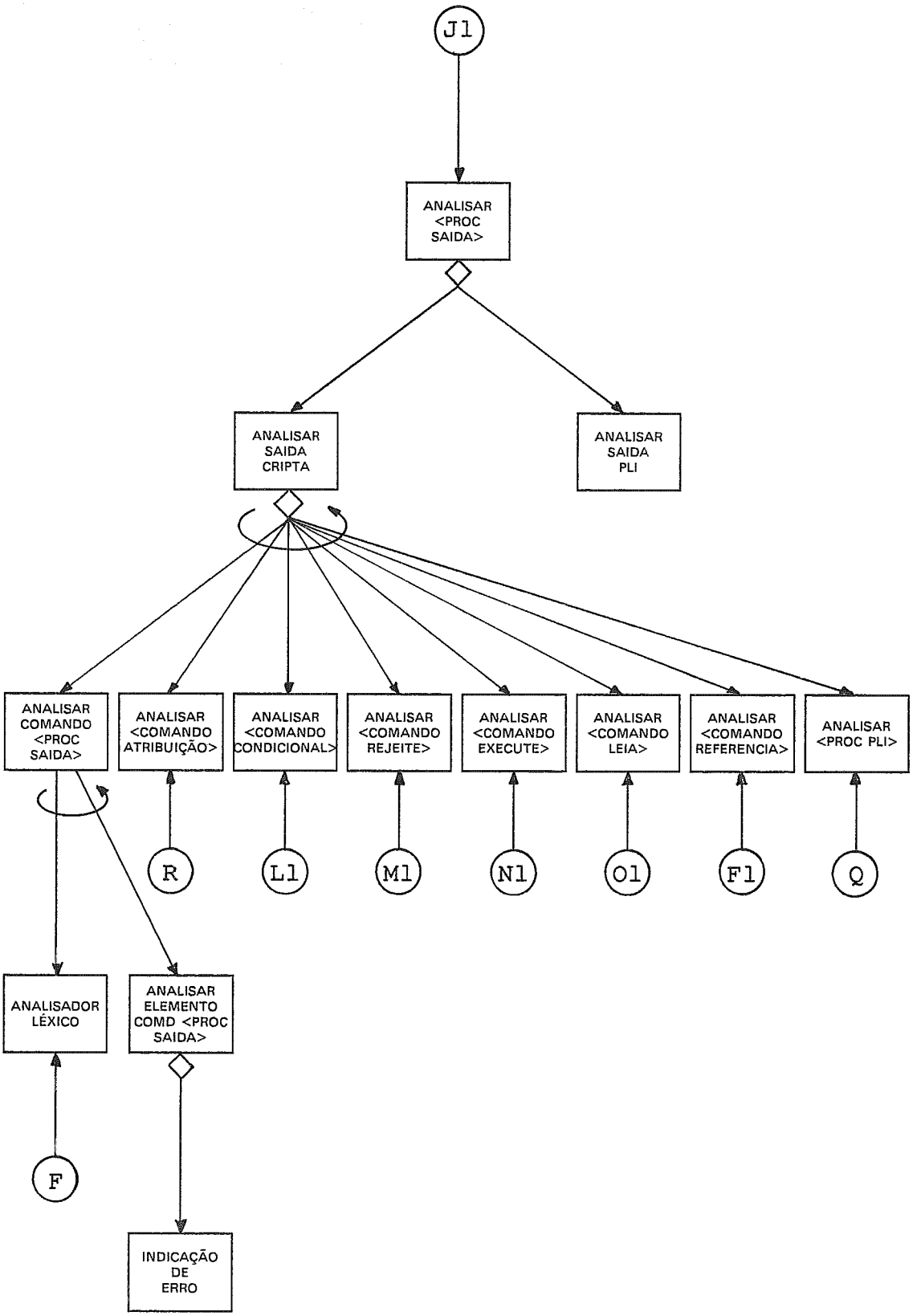


FIGURA D.37 - Estrutura modular "ANALISAR <PROC SAIDA>" (nível 3)

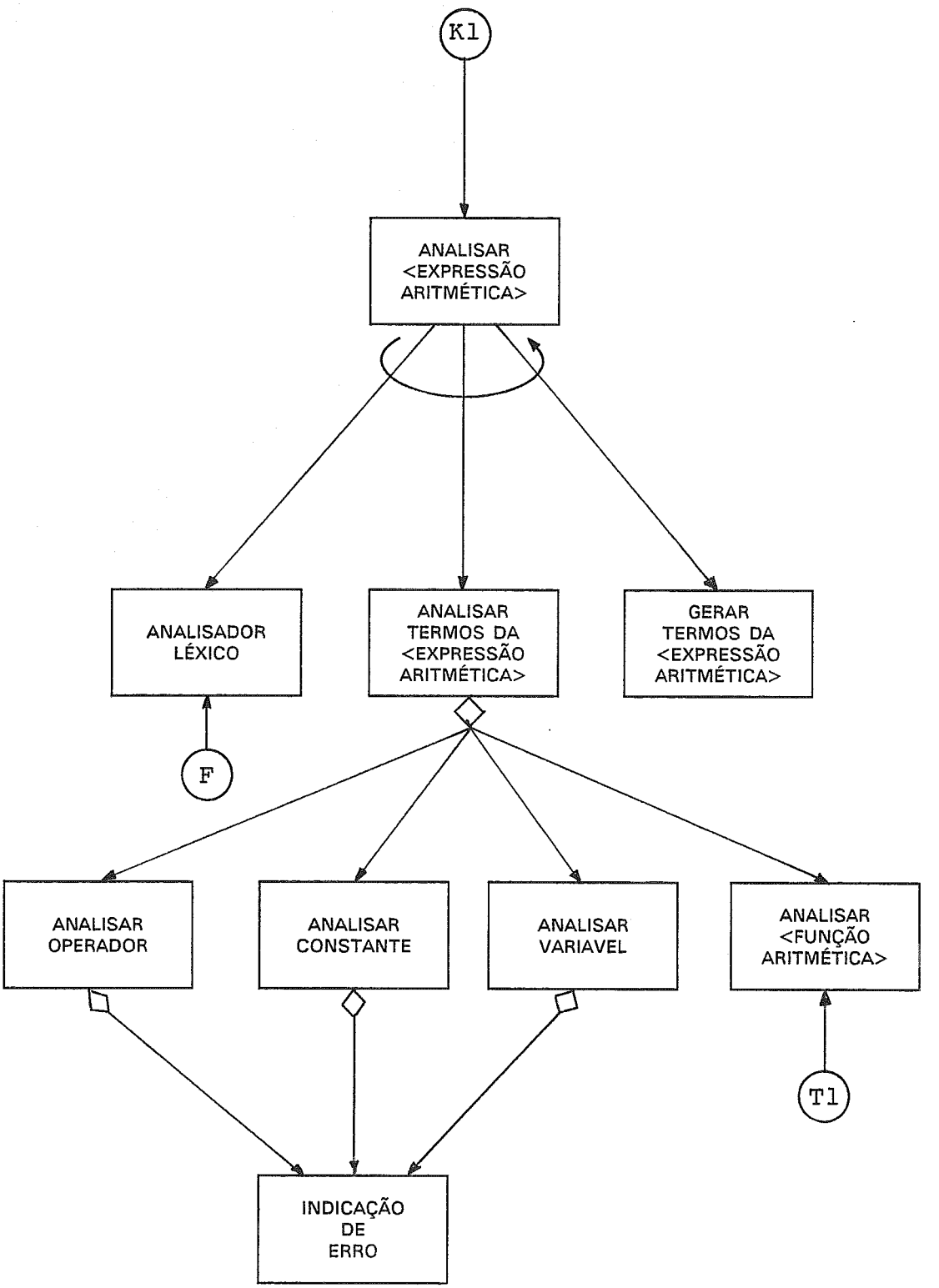


FIGURA D.38 - Estrutura modular "ANALISAR <EXPRESSÃO ARITMÉTICA>" (nível 4)

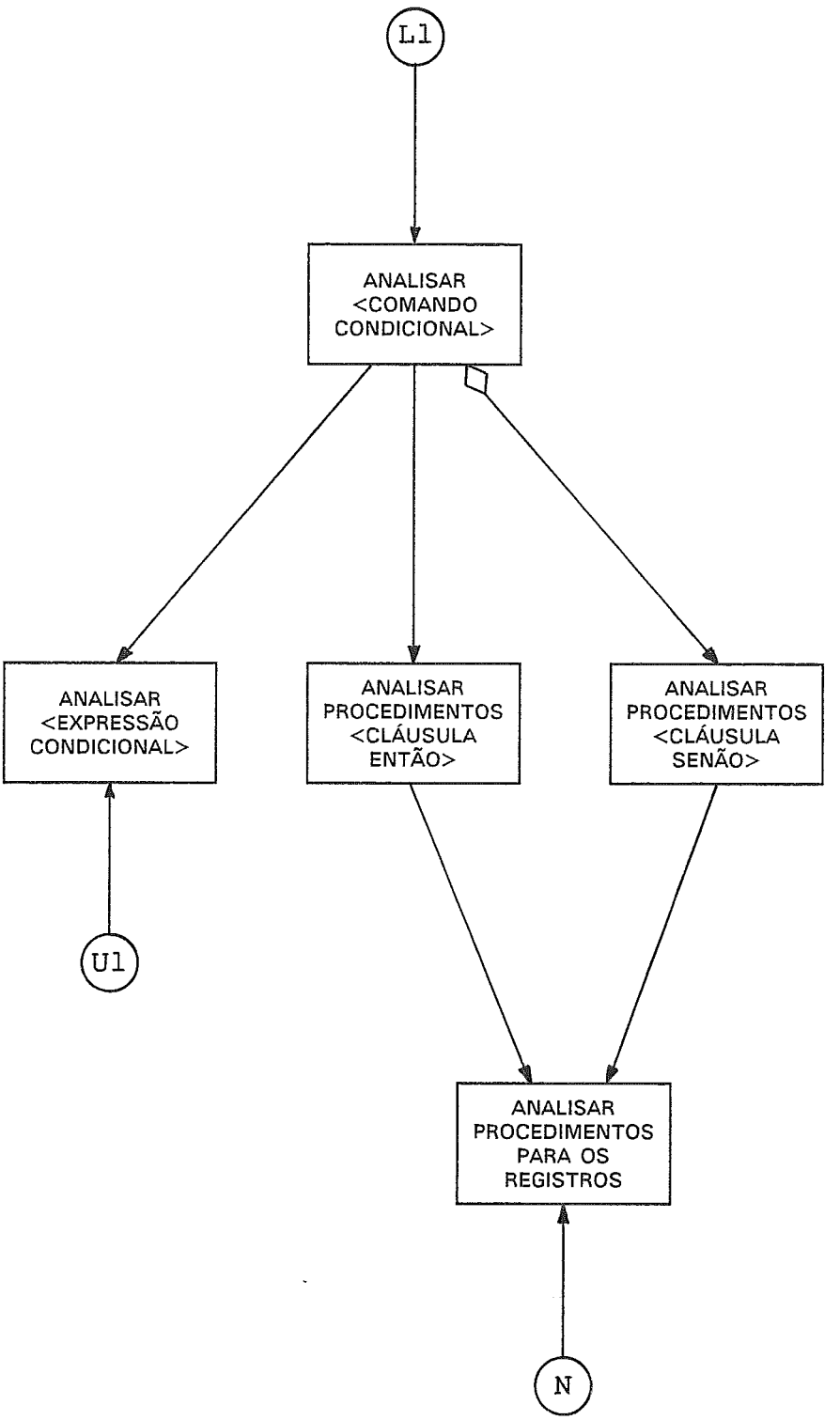


FIGURA D.39 - Estrutura modular "ANALISAR <COMANDO CONDICIONAL>" (nível 4)

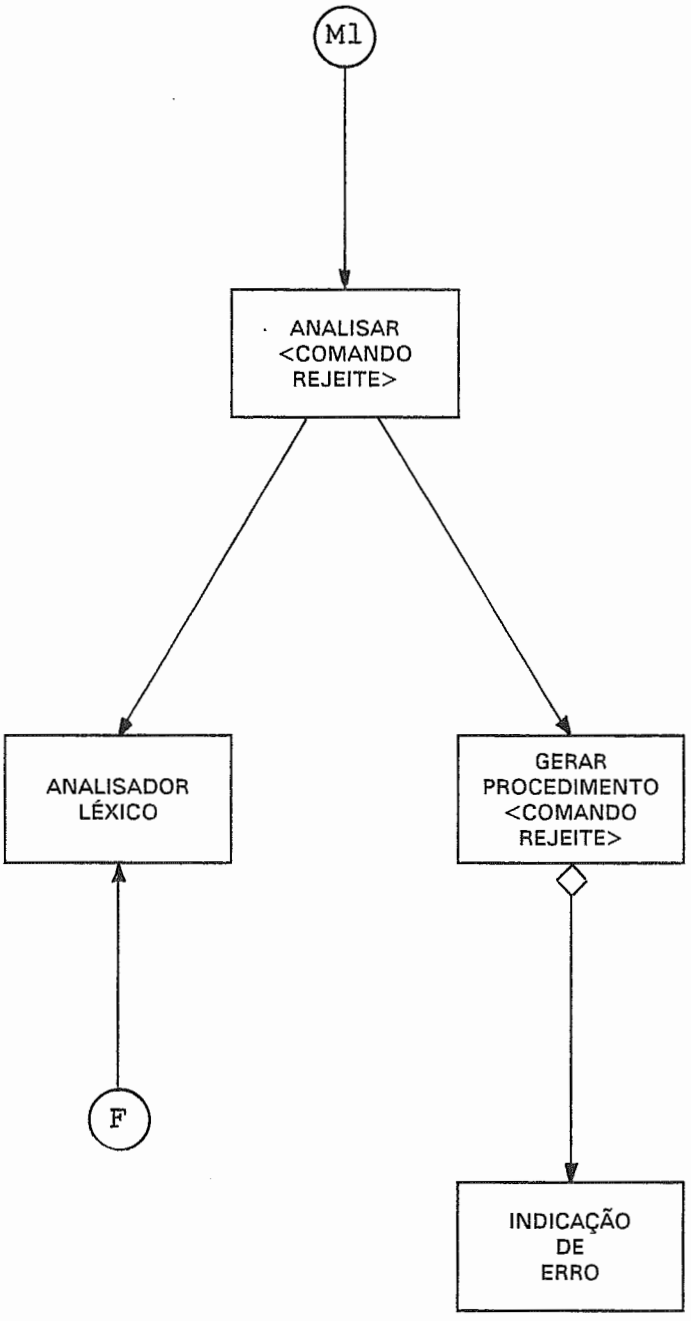


FIGURA D.40 - Estrutura modular "ANALISAR <COMANDO REJEITE>" (nível 4)

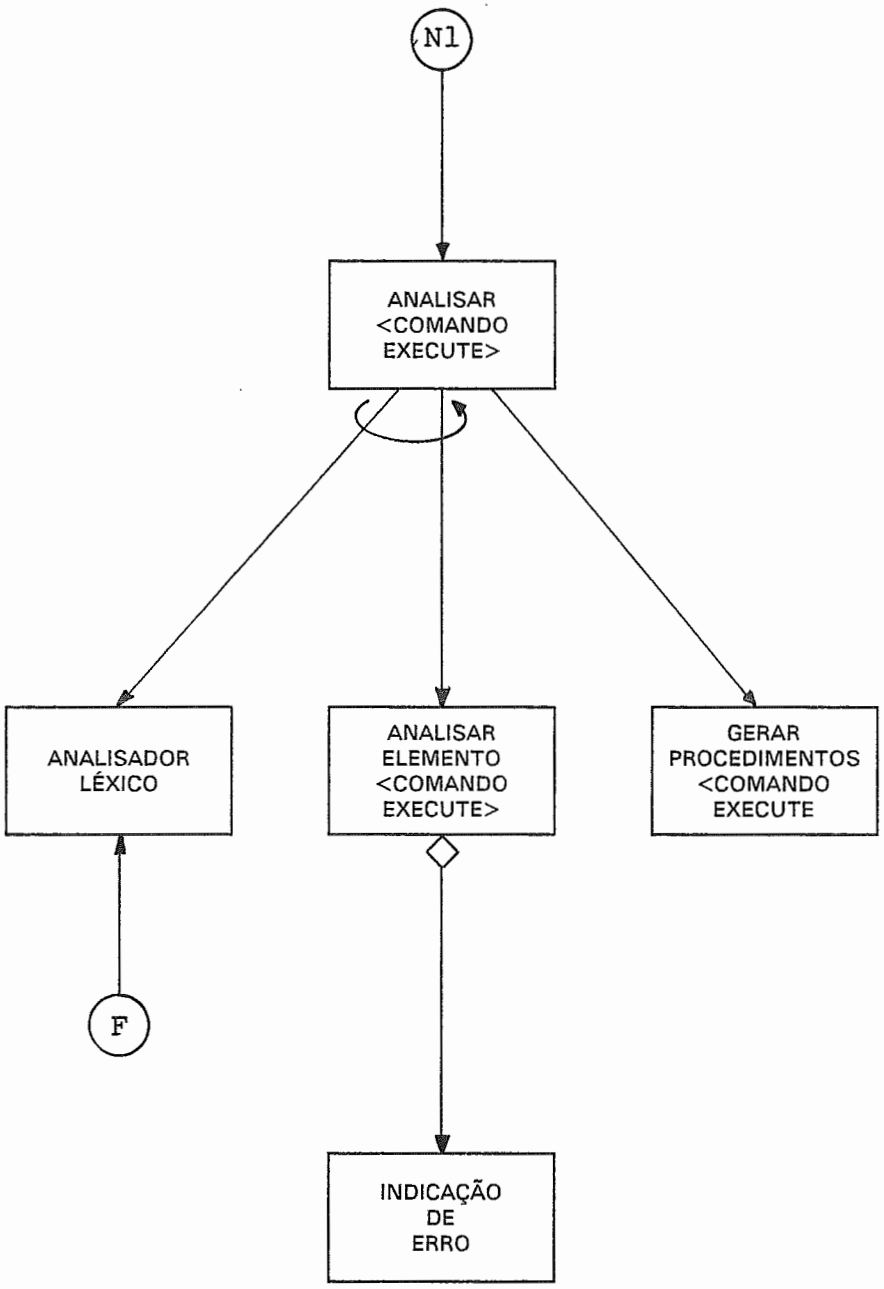


FIGURA D.41 - Estrutura modular "ANALISAR <COMANDO EXECUTE>" (nível 4)

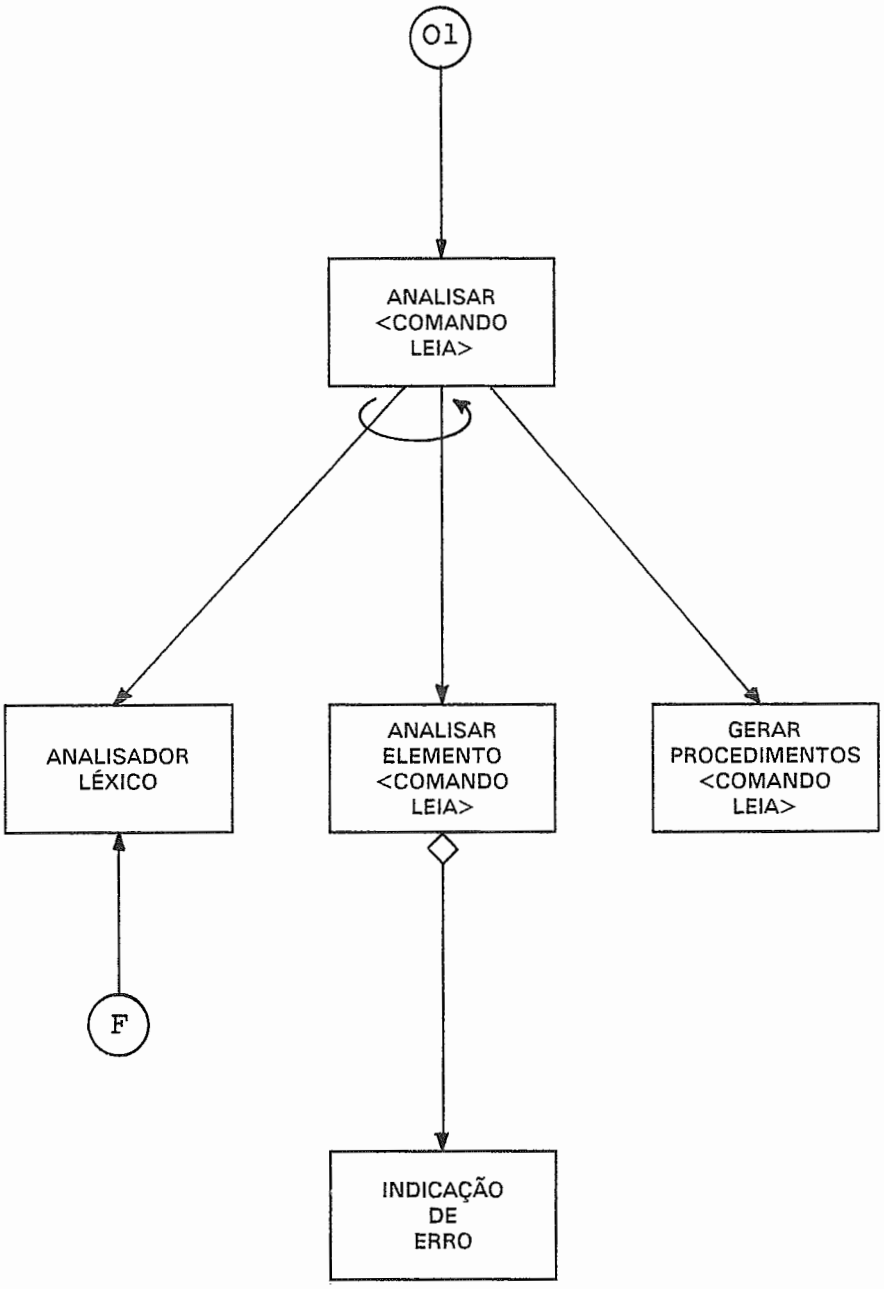


FIGURA D.42 - Estrutura modular "ANALISAR <COMANDO LEIA>"
(nível 4)

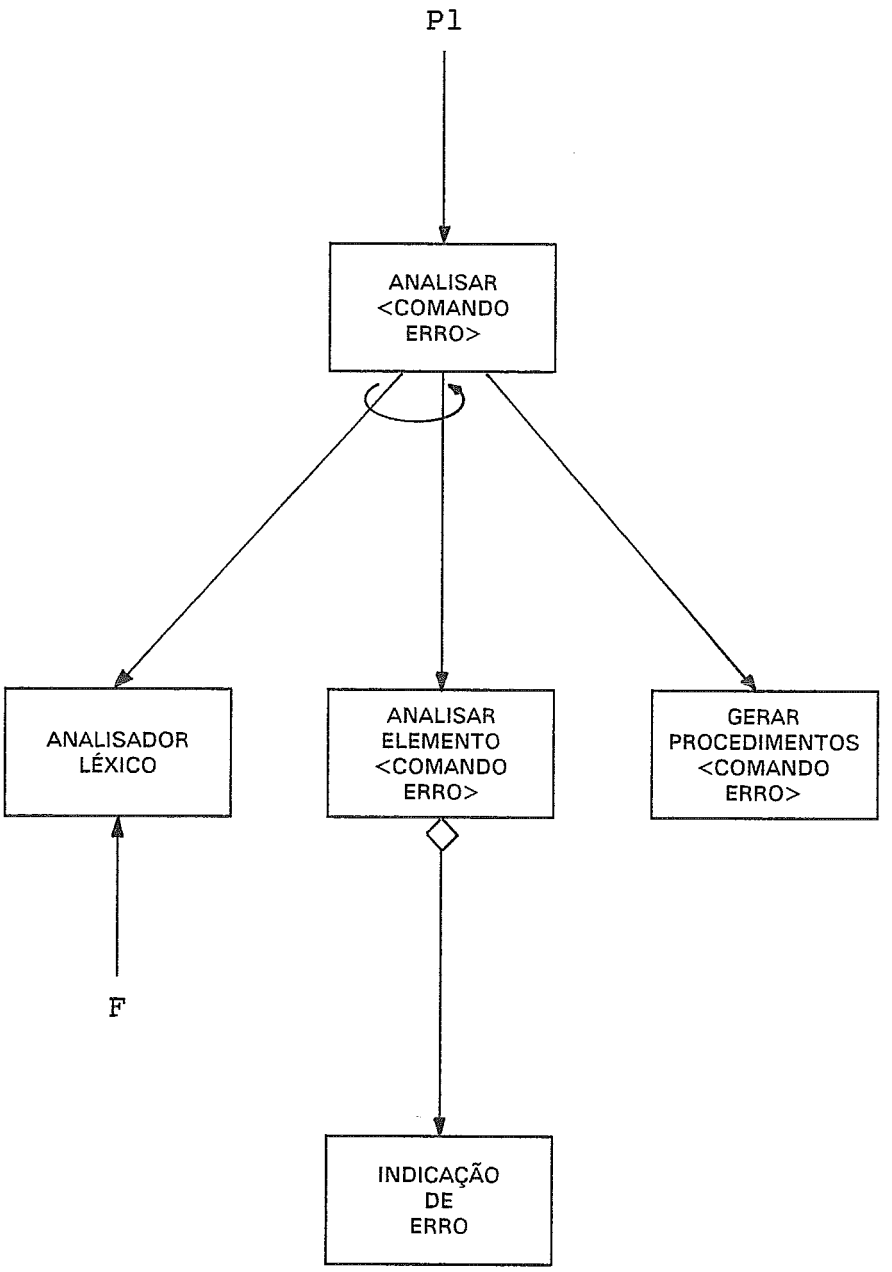


FIGURA D.43 - Estrutura modular "ANALISAR <COMANDO ERRO>"
(nível 4)

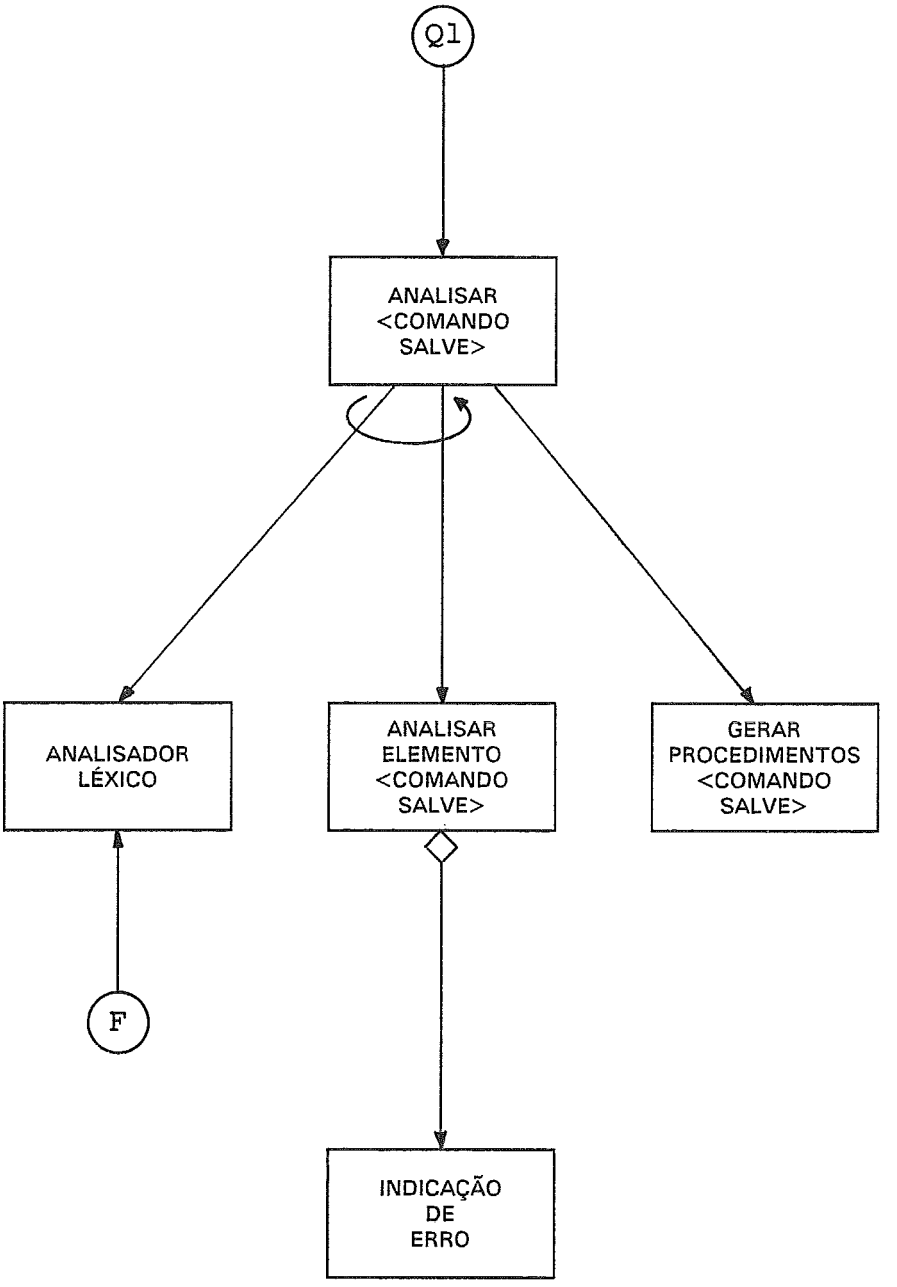


FIGURA D.44 - Estrutura modular "ANALISAR <COMANDO SALVE>" (nível 4)

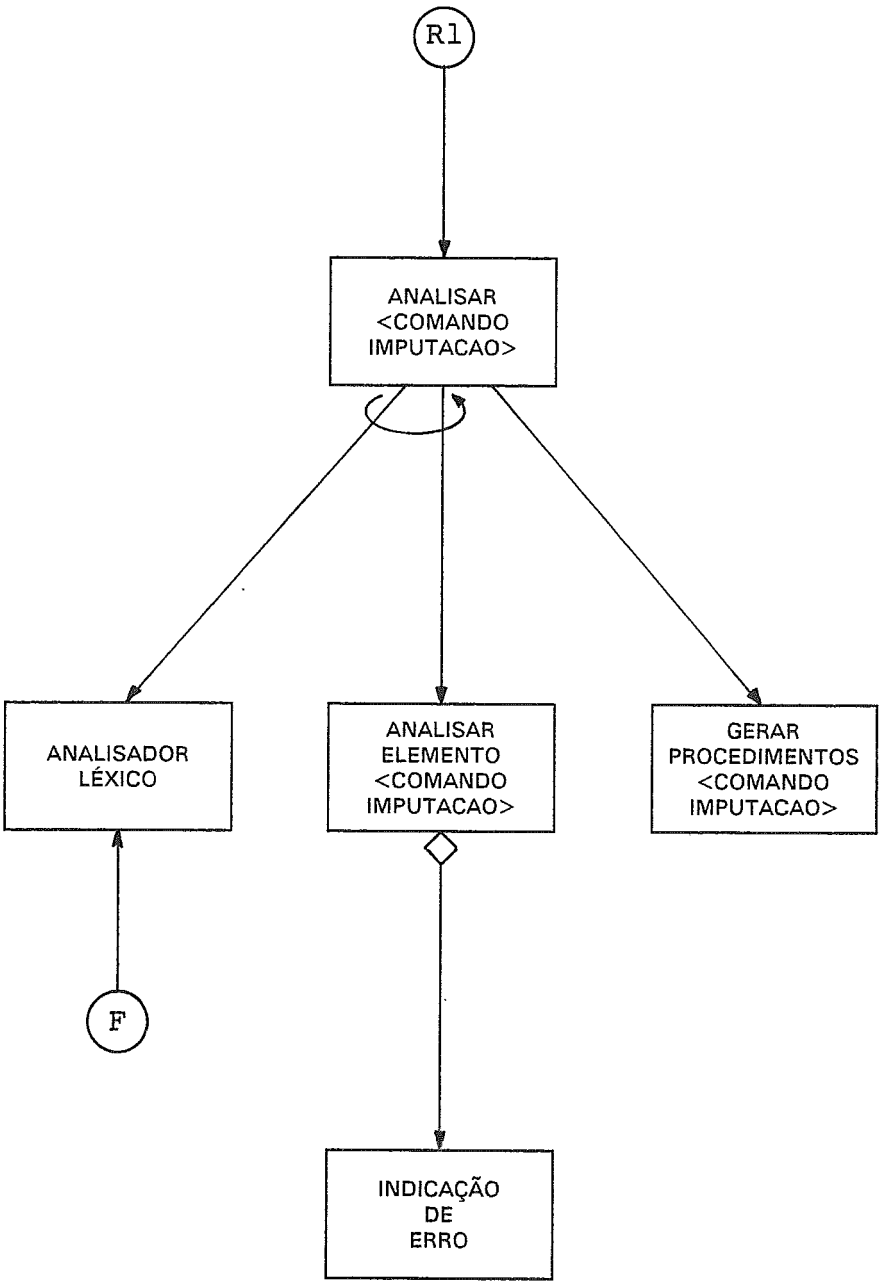


FIGURA D.45 - Estrutura modular "ANALISAR <COMANDO IMPUTAÇÃO (nível 4)

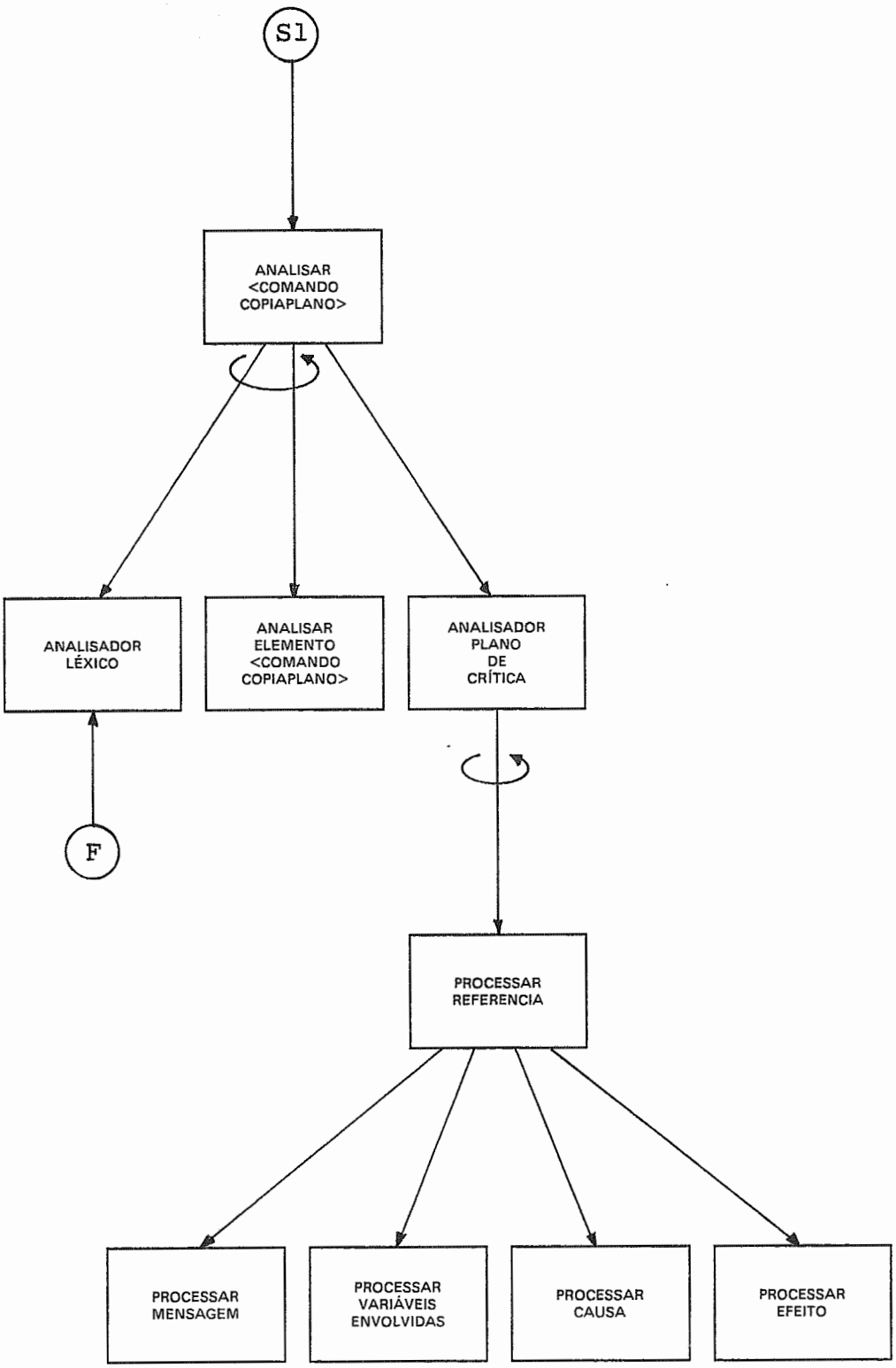


FIGURA D.46 - Estrutura modular "ANALISAR <COMANDO COPIA-PLANO>" (nível 5)

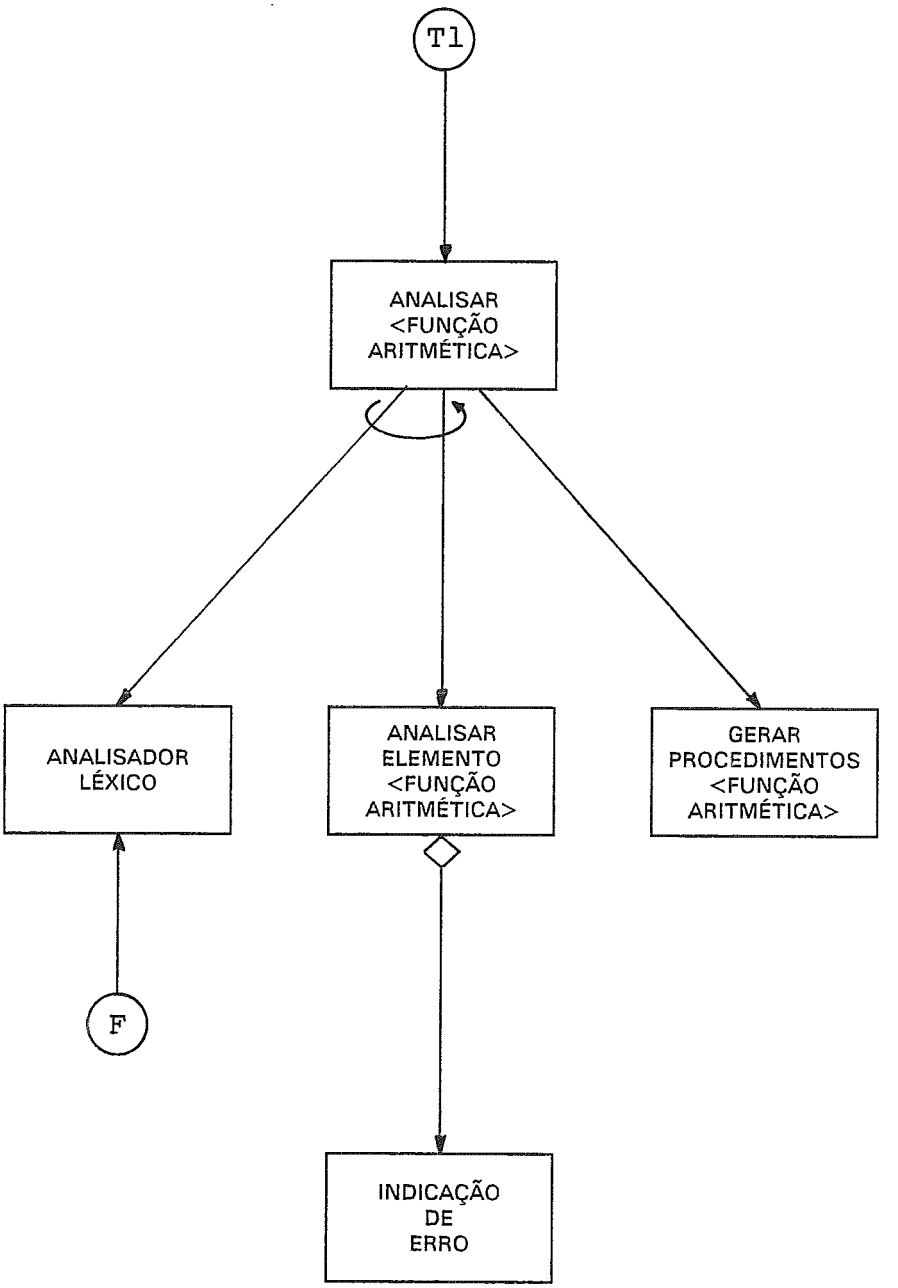


FIGURA D.47 - Estrutura modular "ANALISAR <FUNÇÃO ARITMÉTICA> (nível 5)

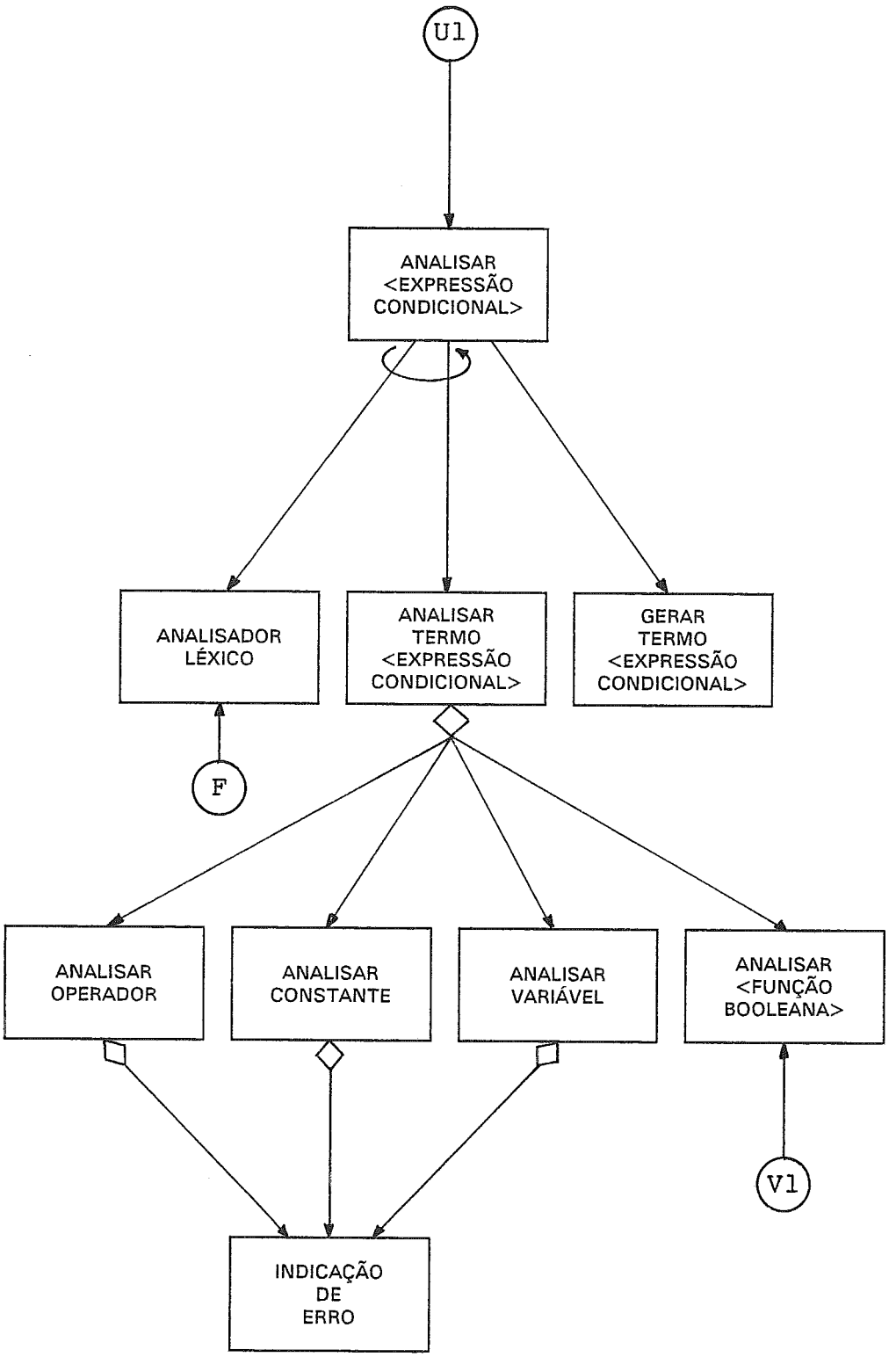


FIGURA D.48 - Estrutura modular "ANALISAR <EXPRESSÃO CONDICIONAL>" (nível 5)

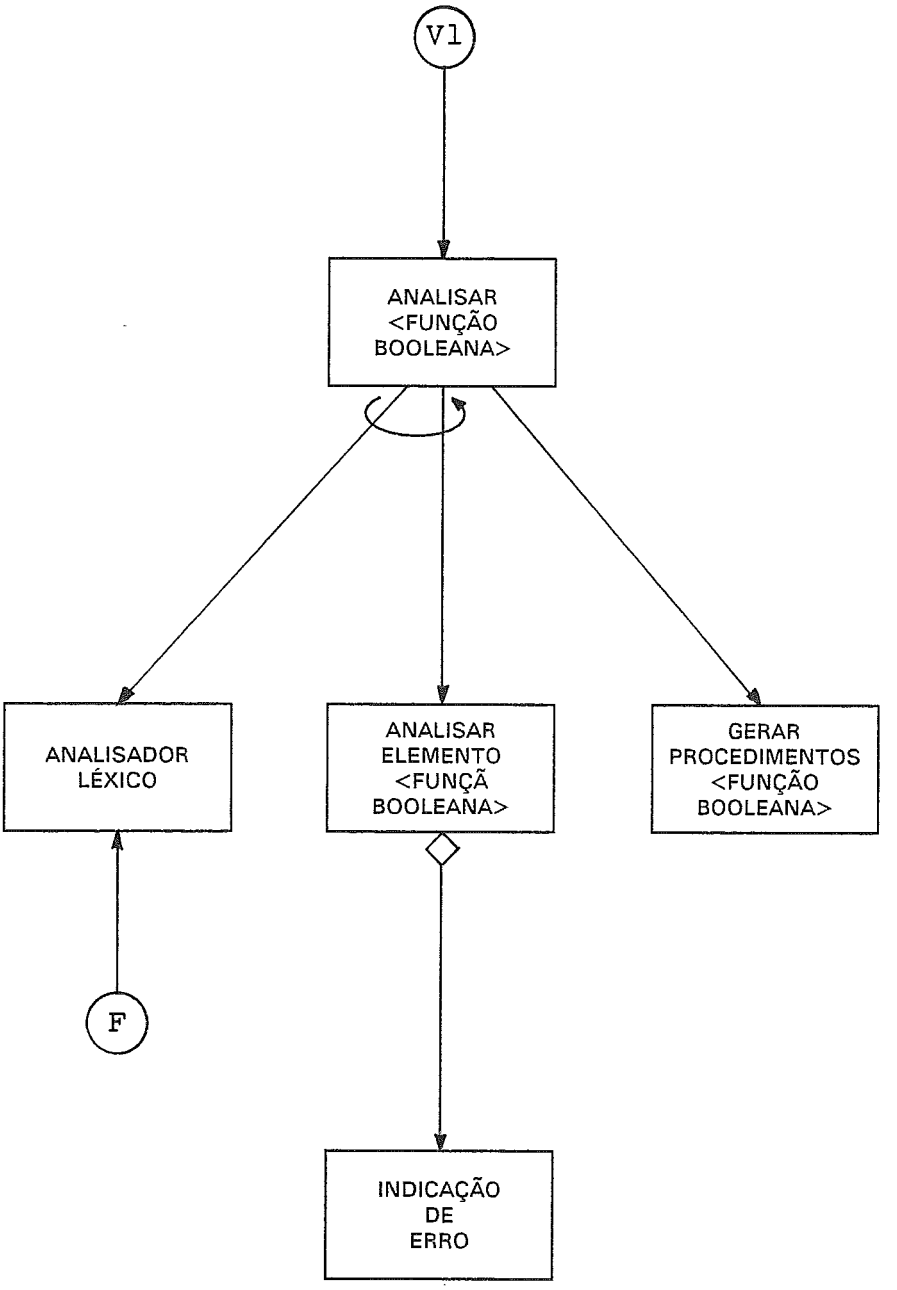


FIGURA D.49 - Estrutura modular "ANALISAR <FUNÇÃO BOOLEANA>"
(nível 4)