Conteúdo

Introdução

O Pascal do CDS/ISIS é uma linguagem de programação projetada para desenvolver aplicações do CDS/ISIS que requeiram funções que não estão prontamente disponíveis no pacote padrão. Para poder usar o Pascal do CDS/ISIS deve-se estar familiarizado com a linguagem de programação Pascal e, obviamente, com o CDS/ISIS. Não ha necessidade porém, de qualquer conhecimento detalhado da natureza interna do CDS/ISIS, tais como estruturas de arquivos e formatos de registros. A especificidade do Pascal do CDS/ISIS é, na verdade, sua biblioteca de procedimentos pré-definidos, que fornece acesso a maioria das funções do CDS/ISIS de uma maneira simples e adequada. Por exemplo: com uma simples chamada a função MENU, pode-se exibir qualquer menu de sistema do CDS/ISIS ou qualquer outro menu que se tenha projetado para uma aplicação especifica e obter a escolha feita pelo usuário; semelhantemente, as funções CREAT e MODIFY permitem criar ou editar um registro interativamente, usando os recursos de entrada de dados do CDS/ISIS na sua totalidade.

Adicionalmente, além de fornecer uma interface poderosa e de alto nivel, ao "software" CDS/ISIS, a biblioteca do Pascal do CDS/ISIS tornará seus programas independentes da versão do sistema que se estiver usando e proporcionará a segurança de uso de funções bem testadas.

O "Guia do usuário", na página 1, descreve os procedimentos para compilar e rodar os programas em Pascal do CDS/ISIS; a prória linguagem, que é um subconjunto do Pascal padrão, é descrita em "Especificações da Linguagem", na página 5; "A biblioteca do Pascal do CDS/ISIS", na página 12, fornece uma descrição completa da biblioteca do Pascal do CDS/ISIS, e a seção "Programas-amostra", na página 33, descreve os exemplos de programas fornecidos nos disquetes de distribuição.

Seção 1

Guia do usuário

A. Pascal do CDS/ISIS

O Pascal do CDS/ISIS é uma parte integrante do CDS/ISIS e consiste de um compilador, um interpretador e uma biblioteca. O compilador produz um pseudo-código o qual é executado pelo interpretador. Por ser o código executável independe da máquina, os programas de aplicação escritos em Pascal do CDS/ISIS são totalmente compatíveis com toda a gama de computadores aceitos pelo CDS/ISIS (isto é, uma aplicação desenvolvida em um microcomputador IBM PC rodará, sem alterações, em um computador VAX). Observe-se, porém, que alguns procedimentos da biblioteca, só estão disponíveis em determinados computadores. Este fato é notado na descrição dos procedimentos relevantes. Se a compatibilidade for essencial, deve-se evitar usar estes procedimentos e escolher métodos alternativos para obter o resultado pretendido.

Pode-se projetar programas para rodar no modo "stand-alone" (que fornece um modo de operação equivalente ao modo "batch") ou como rotinas do usuário em certas funções do CDS/ISIS. As rotinas do usuário proporcionam um meio poderoso de aumentar a funcionalidade do CDS/ISIS.

B. Compilação e execução de um programa

Para inserir um programa pode-se usar qualquer editor de texto com que se esteja familiarizado (o EDLIN por exemplo). Note-se que o compilador não aceita caracteres de tabulação (código ASCII 9). O programa deve ser armazenado em um arquivo com a extensão PAS, no diretório de programas definido no parâmetro 1 do SISPAR.PAR (ver a seção "Instalação do sistema" no *Manual de Referência do CDS/ISIS*).

Para compilar um programa, deve-se selecionar a opção A do menu principal de serviços do CDS/ISIS, xXISI. Aparecerá, então, uma mensagem pedindo que se selecione uma opção do seguinte submenu:

Compilar(C) Executar(R) Sair(Q)

Deve-se selecionar a opção como segue:

C para compilar um programa. Aparecerá, então, uma mensagem para entrar com o nome do programa. Deve-se entrar o nome do programa (sem a extensão). Pode-se sufixar o nome do programa com o parâmetro de compilação de listagem, como segue:

/L para obter um arquivo em disco contendo uma listagem do programa fonte e do código compilado. Por "default" nenhuma listagem será fornecida, a menos que um erro de compilação seja detectado.

O parâmetro pode ser inserido em maiúcula ou minúcula, por exemplo: myprog/l.

R para executar um programa. Como na opção C, aparece uma mensagem para entrar o nome do programa (a menos que se

esteja executando um programa que acabou de ser compilado).

Q para sair e retornar ao menu principal xXISI.

A opção C é usada para compilar programas "batch" e rotinas do usuário. Para executar um programa "batch", deve-se selecionar a opção R. Note-se que esta opção é destinada à execução de programas "batch". Não se deve usá-la para executar rotinas do usuário. A execução de uma rotina do usuário é controlada pelo Programa CDS/ISIS ou pela função à qual foi destinada (ver "Projeção e integração de rotina do usuário", página 8).

C. Arquivos produzidos pelo Pascal do CDS/ISIS

Dois arquivos são produzidos pelo compilador (xxxxx é o nome do programa):

xxxxxx.LST que contém a listagem do programa fonte e uma listagem simbólica do código compilado. Note-se que este arquivo só será produzido se se utilizar o parâmetro de compilação /L, ou se um erro de compilação for detectado.

xxxxxx.PCD que contém o código executável.

Se quaisquer erros de sintaxe forem detectados durante a compilação, o arquivo de listagem conterá uma mensagem de erro apropriada, marcando o ponto onde o erro foi encontrado. A linha de programa contendo o erro, bem como a mensagem, também são mostrados na tela. Um programa que tenha produzido erros de compilação não será executado. O interpretador produzirá uma mensagem de erro se se tentar fazê-lo.

Note-se que os arquivos PCD são armazenados no diretório de programas definido no parâmetro 1 do SYSPAR.PAR, enquanto que os arquivos LST são armazenados no diretório de arquivos de trabalho, definido no parâmetro 4 do SYSPAR.PAR (ver a seção "Instalação do sistema" no *Manual de Referência do CDS/ISIS*).

D. Mensagens de erro do compilador

Mensagens de erro produzidas pelo compilador são listadas abaixo (note-se que quando um erro é encontrado, várias mensagens de erro podem ser produzidas; neste caso, a primeira delas é a significativa):

. PROGRAMA esperado
. Identificador esperado
. ; esperado
. . esperado
. := esperado
. Programa grande demais (o compilador não pode manipular o programa, tente reduzir o seu tamanho, por exemplo agrupando códigos repetidos em "procedures" ou "functions"; ou dividi-lo em dois programas, se possível)

. END esperado (também pode ser causado ao chamar uma "function" como se fosse uma "procedure")

THEN esperado

. DO esperado
. BEGIN esperado
. : esperado
. Limite de "array" ou CASE "label" inválido
. Identificador já declarado
. Identificador desconhecido
. Constantes ou identificadores em excesso
. Tipo desconhecido. Apenas os tipos REAL e STRING são permitidos
. Procedimentos READ e WRITE requerem uma lista de argumentos
.) esperado
. (esperado
. Estouro na tabela de endereços
. Apenas INP pode ser especificado para a função EOF
. UNTIL esperado
. Valor constante do tipo REAL inválido
. Operandos da expressão são de tipos diferentes
. Lados esquerdo e direito da mensagem de atribuição são de tipos diferentes
. Número de argumentos ou o tipo de um ou mais argumentos não combina com a declaração da "procedure" ou da "function"
. Instrução em código P desconhecida (indica um erro no compilador)
. OF esperado
. Limites de "array" inválidos ou faltando. Limites de "array" devem ficar entre [ ]
. Limites de "array" não separados por ..
. OF faltando na declaração de "array"
. Tipo de "array" não suportado. Apenas "arrays" do tipo REAL são permitidos
. Limites de "array" inválidos. O limite inferior deve ser 1 e o limite superior deve ser maior do que o limite inferior
. [esperado
. Variável não é um "array"
. Indice de "array" tem que ser uma expressão REAL
.] esperado
. Operador booleano esperado (AND, OR, NOT)
. Variável do comando FOR não pode ser um "array" ou uma variável STRING
. TO ou DOWNTO esperado
. Espressão REAL esperada
. "Procedures" ou "functions" em excesso
. Variável FOR deve ser uma variável local
. Atributo PROGRAM inválido

E. Erros de execução

. Estouro na pilha
. Ponteiro da pilha inválido (*)
. Instrução em código P desconhecida (*)
. Chamada de "procedure" inválida (*)
. Programa muito grande
. Programa continha erros de compilação
. Chamada desconhecida de "procedure" predefinida (*)
. Nome de componente inválido na chamada da função POSTING (nome de componentes válidos são: MFN, TAG, OCC, CNT)
. Excesso de variáveis STRING
. Nome de arquivo inválido no procedimento ASSIGN (nomes de arquivos válidos são: INP, OUT)

- Leitura após fim de arquivo em INPUT ou INP
- Nivel máximo (16) de aninhamento de "procedures" ultrapassado
- Procedimento UPDATE chamada sem leitura prévia (RECORD ou NEWREC)
- Parâmetro inválido na chamada da função PATH (verificar se o primeiro parâmetro é SYS ou DBN, e se o segundo parâmetro está dentro da faixa correspondente)

Note que os erros marcados com (*) indicam um mau funcionamento do compilador ou do interpretador.

Seção 2

Especificações da Linguagem

Este capítulo não se destina a fornecer uma descrição completa da linguagem Pascal, mas apenas indicar as restrições do Pascal do CDS/ISIS em relação ao Pascal padrão, ou em relação às variações do mesmo. Para uma descrição completa da linguagem, deve-se remeter a um manual de programação em Pascal apropriado.

A. Constantes, Tipos e Variáveis

Declarações CONST e TYPE não são permitidas.

Apenas 3 tipos pré-definidos podem ser usados:

REAL — Valores numéricos reais. Constantes do tipo Real podem ser inteiros ou números decimais contendo sinais opcionais. Notação exponencial não é permitida.

ARRAY [1..n] OF REAL
Os limites do "array" devem ser constantes inteiras e o limite inferior deve ser 1. Constantes de "array" ou "arrays" não são permitidas.

STRING — Cadeias de caracteres de tamanho variável. Cadeias nulas e constantes (por exemplo '') são permitidas.

Todas as variáveis devem ser declaradas em uma cláusula VAR antes de serem usadas.

B. Expressões

Expressões aritméticas, booleanas e de cadeia de caracteres seguem a sintaxe do PASCAL padrão (note-se que as expressões booleanas são permitidas em comandos condicionais, embora variáveis do tipo BOOLEAN não possam ser declaradas).

Os seguintes operadores são permitidos:

Unário	+ -
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Concatenação	| (operador de cadeia de caracteres)
Booleano	AND OR NOT
Relacional	= <> <= >= < > (note-se que, para que possam ser consideradas iguais, duas cadeias de caracteres devem ter o mesmo tamanho)

C. Comandos

Os seguintes comandos estão implementados (a menos que uma restrição específica seja mencionada, todos os comandos implementados seguem a sintaxe do PASCAL padrão):

Comandos de atribuição: Os lados esquerdo e direito do comando de atribuição devem ser do mesmo tipo.

Comando IF

Comando CASE: O seletor CASE pode ser tanto uma expressão REAL ou STRING. Quando espressões de cadeia de caracteres são usadas, as etiquetas CASE correspondentes podem ser cadeias de caracteres ou caracteres únicos. Porém, para ser combinada apropriadamente, a expressão deve produzir uma cadeia de caracteres que tenha o mesmo tamanho que as constantes da etiqueta CASE. Constantes de etiqueta CASE múltiplas não são permitidas.

Comando FOR: A variável de controle deve ser uma variável local, não podendo ser um parâmetro ou um componente de um "array".

Comando WHILE

Comando REPEAT ... UNTIL

D. "Procedures" e "Functions"

Ambas as "functions" REAL e STRING são permitidas. "Functions" ARRAY não são permitidas.

Todos os parâmetros das "procedures" e das "functions" do tipo REAL são passados por *valor*, enquanto que os parâmetros do tipo STRING são passados por *referência*. O prefixo do parâmetro do tipo VAR, porém, não deve ser declarado. Embora o compilador não emita uma mensagem se uma atribuição for feita a um parâmetro formal passado por valor, o valor do parâmetro atual correspondente na "procedure" chamada não será afetado.

Da mesma maneira, o compilador não verifica se o parâmetro atual, correspondendo a um parâmetro formal passado por referência, é uma expressão, e a "procedure" atribui-lhe um valor. É de responsabilidade do usuário assegurar que, sempre que uma "procedure" atribuir um valor a um parâmetro formal passado por referência, o parâmetro atual correspondente seja sempre uma única variável.

Por exemplo:

```
Procedure A;
Var x: Real;
    c: String;

Procedure B(n: Real; s: string);
Begin
  n:=5;
  s:='.';
End;

Begin
x:=1; c:='Water';
B(x,c);
{ o valor de x neste ponto ainda é 1 }

 { enquanto que o valor de c é 'Water.' }
```

```
End;
```

Note-se também que o compilador não verifica se a uma função é realmente atribuido um valor dentro do corpo da função. Se um valor não for atribuido antes que a função termine, o valor retornado será imprevisivel.

"Functions" e "procedures" externas não são permitidas.

E. Ambito dos identificadores

O ambito de um identificador de uma função ou de uma variável é definido da mesma maneira que no PASCAL padrão (por exemplo, uma variável declarada em uma "procedure" é local em relação aquela "procedure" e desconhecida em quaisquer "procedures" externas).

F. Arquivos-Texto

Além dos arquivos padrões INPUT e OUTPUT, dois arquivos-texto adicionais predefinidos podem ser usados: INP (para entrada) e OUT (para saida). INPUT e OUTPUT são sempre atribuidos aos dispositivos padrão de entrada e saida. Um comando RESET ou REWRITE é automaticamente emitido para todos esses arquivos. INP e OUT são atribuidos por "default" aos dispositivos padrão de entrada e de saida respectivamente, mas podem ser re-atribuidos através do procedimento ASSIGN predefinido (ver a seguir).

Os procedimentos padrão READ, READLN, WRITE, WRITELN têm a mesma sintaxe do PASCAL padrão. Nos comandos WRITE e WRITELN a opção de tamanho de campo é permitida apenas para expressões do tipo REAL. Neste caso, as seguintes regras se aplicam:

```
WRITE[LN] ( [OUT,] exp1 [: exp2 [: exp3]] );
```

onde: *exp1* é o valor a ser escrito, *exp2* é o tamanho do campo e *exp3* é a precisão.

Se for apresentado apenas *exp2* o valor é primeiro convertido em inteiro (o valor escrito é na verdade TRUNC(*exp1*):*exp2*). Se ambos *exp2* e *exp3* forem apresentados, o valor é escrito na notação decimal. Se nem *exp2* ou *exp3* estiverem presentes, o valor será escrito na notação exponencial cientifica.

A função EOF pode ser usada na entrada padrão ou no arquivo INP. Como no Pascal padrão, ela retorna um valor booleano e a sua sintaxe é a seguinte:

```
EOF [(INP)]
```

A função EOLN não é permitida.

G. Comentários

Comentários são permitidos e devem ser delimitados por chaves (caracteres { e }). Os delimitadores alternativos (* *) não são permitidos.

H. Projeção e integração de rotinas do usuário

Uma rotina do usuário é projetada para uma função especifica do CDS/ISIS, a qual deve ser declarada como um atributo no comando PROGRAM. Diferentemente dos programas "batch", as rotinas do usuário podem declarar parâmetros de nível de programa que serão passadas para o programa CDS/ISIS sempre que a rotina for ativada. Estes parâmetros são fixados para cada tipo de rotina do usuário e são verificados pelo compilador.

As rotinas do usuário podem ser escritas para as funções do CDS/ISIS descritas abaixo:

1. Rotinas de Menu

Uma rotina de menu é ativada cada vez que se selecionar uma opção especifica de um menu do sistema. Uma opção do menu pode ser associada a uma rotina de menu, atribuindo-se aquela opção o código de ação E (ver a seção "O Editor de Menu" no *Manual de Referência do CDS/ISIS*).

Rotinas de MENU possuem um parâmetro e são declaradas como segue:

```
Program xxxxx (s1: string) [MENU];
```

O parâmetro s1 é ao mesmo tempo um parâmetro de entrada e de saída:

na entrada: s1 é uma cadeia de um caractere contendo o identificador de opção que fez com que o programa fosse chamado;

na saída: deve-se atribuir a s1 um dos seguintes parâmetros:

espaço — para fazer com que o menu corrente seja mostrado novamente;

s — para provocar a seleção automática da opção s do menu corrente (o menu não é mostrado novamente e o controle é passado diretamente para o programa "chamador" como se tivesse sido selecionada a opção s do menu); neste caso, s deve ser o identificador *externo* (isto é, aquele mostrado no menu) de uma opção válida do menu corrente. Se s não for uma opção válida, um bip será produzido e o menu será mostrado. Note-se que s deve ser diferente daquele passado em s1, do contrário será produzido um "loop" infinito;

.s — para provocar a execução direta da opção s; neste caso, s deve ser o identificador *interno* de uma opção válida do menu corrente. Isto é útil sempre que se quer fazer algum trabalho preliminar *antes* de executar a opção padrão (ver exemplo dado abaixo).

Note-se que se o programa *digitar automaticamente* uma cadeia de caracteres, (ver procedimento AUTOTYPE a seguir), a ação indicada pelo parâmetro de saída s1 é executada *antes* da cadeia ser digitada automaticamente.

O exemplo seguinte ilustra como se poderia controlar o acesso a entrada de dados na base de dados CDS por meio de uma senha.

```
Program CHKPWD(s: string) [menu];

var dbname: string;

Function CHECK: string;
var cnt: real;
    pw: string;
begin
clear;
cursor(10,10); write('Enter password ');
cnt:=0;
repeat
attr(' ',5,10,25,6); readln(pw);
cnt:=cnt+1; { count number of trials }
if pw<>'x12y' then
   begin cursor(12,25); writeln(chr(7),'Invalid password'); end;
until (pw='x12y') or (cnt>2); { allow 2 trials at most }
if pw<>'x12y'
   then begin check:=' '; pw:=inkey; end
   else check:='.E';
end;


begin
dbname:=dbn;
if  dbname=''  then { force user to select data if  none  currently
selected }
   begin
   clearmsg; write('Data base name? '); readln(dbname);
if dbname<>'' then open(dbname);
end;
if dbn=''
   then s:='  '    { if no data base  selected  redisplay menu }
   else if dbn='CDS'
     then  s:=check  {  if  CDS  data  base  selected  then  check
password }
     else s:='.E';   { ok for all other data bases }
end.
```

Quando a rotina de menu acima estiver associada com a opção E do menu principal de serviços xXISI, ela solicitará a senha x12y cada vez que a base de dados CDS for selecionada para entrada de dados.

2. Rotinas de formatação

Uma rotina de formatação é ativada quando um formato de exibição do CDS/ISIS estiver sendo processado. Ela pode ser usada para fornecer dados para a rotina de formatação padrão (ver a seção "A Linguagem de Formatação", no *Manual de Referência do CDS/ISIS*). Uma rotina de formatação é chamada codificando-se o seguinte no seu formato:

```
&nome(formato)
```

onde:

& identifica-o como uma chamada de rotina de formatação;

nome é o nome do programa em Pascal CDS/ISIS a ser executado; e

formato é um formato do CDS/ISIS.

Rotinas de formatação possuem 4 parâmetros e são declaradas como segue:

```
Program xxxx (s1: string: lw,occ: real; s2: string) [FORMAT];
```

onde:

xxxx é o nome do programa (o qual será utilizado para chamar a rotina a partir do formato, como mencionado acima);

s1 é o resultado da execução do parâmetro do formato fornecido quando a rotina é chamada; ele pode ser usado para fornecer um ou mais parâmetros ao programa e é útil no projeto de rotinas de formatação generalizadas; o tamanho máximo desta cadeia é de 255 caracteres;

lw é o tamanho da linha que está em vigor quando a rotina é executada; ela pode ser útil, em alguns casos, para saber o tamanho da linha se a sua rotina imprimir dados de tamanho fixo;

occ é o contador de ocorrências. Ele é zero se a rotina for chamada *de fora* de um grupo repetitivo e maior que zero se a rotina for chamada *de dentro* de um grupo repetitivo. Neste caso, o CDS/ISIS chamará repetidamente sua rotina para cada execução do grupo repetitivo, a primeira vez com occ=1, a segunda occ=2, etc; até que se retorne uma cadeia de caracteres vazia (em s2) para significar que não há mais dados a serem impressos.

s2 é o dado de saída a ser retornado para o programa de formatação do CDS/ISIS. O CDS/ISIS manipulará este dado como se fosse um campo do registro sendo formatado. Note-se que o tamanho de s2 não é restrito.

Se a rotina de formatação for projetada para manipular campos repetitivos, deve-se verificar o parâmetro occ para saber se todas as ocorrências devem ser manipuladas de uma vez (se occ=0, sua rotina será chamada apenas uma vez), ou uma de cada vez. Como alternativa, pode-se estabelecer restrições no uso da rotina; por exemplo: decidir que ela deve sempre ser usada dentro de um grupo repetitivo. Por exemplo: a seguinte rotina de formatação pode ser usada dentro de um grupo repetitivo para numerar as ocorrências de um campo repetitivo (a etiqueta do campo a ser numerado é passado como parâmetro):

```
Program NUMBER(s1: string; lw,occ: real; s2: string) [FORMAT];
begin
if nocc(val(s1))>=occ
   then s2:=encint(occ,3)|'. '
   else s2:='';
end.
```

Um exemplo do uso da rotina de formatação acima (para numerar as ocorrências do campo 70) é dado a seguir:

```
Formato:          (number('70'),v70/)
                     |          |
                     |          |
                     |          |
                     |    +-----+
                     |    |
                     V    V
Saida                1.  Jones, P.
                     2.  Brown, J.
                     3.  McKenzie, K.
```

Seção 3

A Biblioteca do Pascal do CDS/ISIS

A. Visão geral

A biblioteca do Pascal do CDS/ISIS contém uma coleção de procedimentos e funções pré-definidos, que podem ser classificadas em duas grandes categorias: procedimentos gerais e procedimentos do CDS/ISIS. Os seguintes procedimentos e funções pré-definidos estão atualmente disponíveis:

1. Funções e procedimentos gerais

a. Gerenciamento de tela

ATTR	Estabelece o atributo de tela
BOX	Desenha uma caixa
CHATTR	Muda o atributo de tela
CLEARBOX	Limpa a caixa
CLEARDATA	Limpa a área de dados (linhas 1-21)
CLEARLN	Limpa linha
CLEARMSG	Limpa área de mensagens (linhas 22-24)
CLEAR	Limpa a tela
CURSOR	Estabelece a posição do cursor
GETCC	Obtém a posição do cursor (coluna)
GETCL	Obtém a posição do cursor (linha)
PAGE	Seleciona página ativa
SAVESCR	Armazena página ativa

b. Gerenciamento de teclado

AUTOTYPE	Entrada pelo teclado simulado
INKEY	Entrada de um único caractere (com eco)
KBDKEY	Entrada de um único caractere (sem eco)
DEFKEY	Define tecla de função

c. Conversão de tipo

CHR	Converte real para caractere
ENCINT	Converte real para string (inteiro)
ENCREAL	Converte real para string
ORD	Converte caractere para real
VAL	Converte string para real

d. Utilitários de cadeia de caracteres

POSITION	Encontra a posição da cadeia de caracteres
SIZE	Tamanho da cadeia de caracteres
SUBSTR	Subcadeia
UC	Converte para maiúsculas

e. Miscelânea

ASSIGN	Atribui nome do arquivo
DATESTAMP	Obtém data e hora

FILEXIST	Verifica existência do arquivo
EXEC	Executa outro programa

2. Procedimentos e Funções do CDS/ISIS

a. Sistema

LANG	Idioma corrente
MENU	Mostra menu do sistema
MSGTEXT	Obtém mensagens do sistema
MSG	Mostra mensagens do sistema
PATH	Obtém a localização (path) definida no SISPAR.PAR ou dbn.PAR
SETLANG	Estabelece idioma do diálogo

b. Base de dados

CLOSE	Fecha a base de dados
DBN	Nome da base de dados corrente
LOCK	Bloqueia a base de dados (apenas para o VAX)
MAXMFN	Próximo MFN a ser atribuído
OPEN	Abre a base de dados
UNLOCK	Desbloqueia a base de dados (para o VAX apenas)

c. Arquivo-mestre

CREAT	Cria novo registro (interativo)
DELETE	Apaga registro
FIELDN	Encontra número do campo
FIELD	Obtém conteúdo do campo
FLDADD	Adiciona campo
FLDCHA	Muda o campo
FLDDEL	Apaga o campo
FLDMOD	Modifica o campo
FLDREP	Substitui o campo
MODIFY	Modifica o registro (interativo)
NEWREC	Cria novo registro (batch)
NFIELDS	Número de campos no registro
NOCC	Número de ocorrências do campo
RECORD	Obtém registro do arquivo-mestre
UPDATE	Atualiza o registro

d. Arquivo invertido

DELTERM	Elimina termo de dicionário
FIND	Obtém termo do dicionário
NXTPOST	Obtém a próximo "posting"
NXTTERM	Obtém o próximo termo do dicionário
POSTING	Obtém componente do "posting"
UPDIF	Atualiza arquivo invertido

e. Busca

NXTPOS	Obtém MFN do próximo registro recuperado
SEARCH	Executa uma expressão de busca do CDS/ISIS
SETPOS	Obtém o MFN do registro recuperado

f. Formatação

FMTNAME	Nome corrente do formato
FORMAT	Executa formato de exibição
GETFMT	Define formato de exibição
LINES	Número de linhas produzidas pelo FORMAT
NXTLINE	Obtém a próxima linha produzida pelo FORMAT

g. Edição e entrada de dados

DATAENTRY	Edita uma página de planilha
DUMMYREC	Estabelece um registro fictício
EDIT	Edita uma cadeia através do editor de campo do CDS/ISIS
WORKSHEET	Seleciona uma planilha

Uma detalhada descrição de todos os procedimentos e funções predefinidos é dada a seguir (em ordem alfabética). Note-se que os parâmetros VAR são permitidos para funções e procedimentos predefinidos.

B. Descrição funcional

1. Procedimento ASSIGN (s1,s2: string);

Atribui o arquivo padrão INP ou OUT a um arquivo ou dispositivo definidos pelo usuário.

s1 é uma expressão de cadeia de caracteres igual ao INP ou ao OUT.

s2 é uma expressão de cadeia de caracteres que produz um nome de arquivo válido para o sistema operacional em uso (que pode incluir informações sobre o "drive" e/ou sobre o diretório) ou um nome padrão de dispositivo (por exemplo PRN).

É permitido reatribuir um arquivo já atribuído. O arquivo corrente é fechado automaticamente. Em alguns casos um ASSIGN fictício pode ser necessário para forçar o fechamento de OUT antes de reatribuir este arquivo ao INP. Um arquivo atribuído ao INP já deve existir. Em caso de dúvida, deve-se primeiro verificar sua existência por meio da função FILEXIST. Além disso, pode-se obter informação sobre a localização (path) por meio da função PATH para arquivos onde este possa ser dependente do SYSPAR.PAR ou do dbn.PAR.

2. Procedimento ATTR (fill: string; attr, lin, col, len: real);

Limpa a área de tela do tamanho dos caracteres len, começando na linha lin, coluna col. O atributo de tela da área limpa é estabelecido em attr e a área é preenchida com o primeiro caractere de fill. Os valores de attr podem ser quaisquer dos atributos padrão do CDS/ISIS.

-2	Fundo de tela	2	Bold
-1	Area de mensagens	3	Sublinhado
0	Normal	4	Piscante
1	Video reverso	5	Invisivel

O atributo real dependerá dos valores estabelecidos para o CDS/ISIS usando a opção A para o menu de serviços de utilidades do sistema xXM1.

ATTR também irá posicionar o cursor na lin, coluna col.

3. Procedimento AUTOTYPE (s1: string);

Insere a cadeia s1 dentro do "buffer" de entrada interno do teclado. Todas as chamadas subsequentes ao INKEY ou KBDKEY obterão entradas deste "buffer" enquanto houver caracteres disponíveis. Quando o "buffer" de entrada interno estiver vazio, a entrada normal do teclado será retornada.

Se o "buffer" interno já contiver um ou mais caracteres, s1 será inserido *antes* dos mesmos. Portanto, normalmente não se deve usar AUTOTYPE mais de uma vez em um determinado programa. Se os caracteres a serem digitados automaticamente forem coletados em vários pontos do programa, deve-se usar uma cadeia de caracteres para mantê-los até que a sequência final seja obtida, e, então, utilizar o procedimento AUTOTYPE nesta cadeia antes de sair.

Se se usar AUTOTYPE em uma rotina de menu, deve-se notar que a ação indicada pelo parâmetro do programa é executada *antes* da cadeia de caracteres digitada automaticamente.

s1 pode conter códigos de verificação e/ou caracteres de controle codificados para identificar teclas que não estejam produzindo um código ASCII (por exemplo: F1, F2, etc). Os códigos de verificação e caracteres de controle são codificados usando as mesmas regras utilizadas nos parâmetros de definição de teclas no SYSPAR.PAR (ver a seção "Instalação do sistema" no *Manual de Referência do CDS/ISIS*).

Este procedimento é particularmente útil quando se quer submeter uma pesquisa para processamento e o controle ser retornado ao programa sem nenhuma intervenção do usuário. Por exemplo:

```
autotype(' ');
s:=search('adult * education');
x:=inkey;
```

Normalmente, depois de executar uma busca, o CDS/ISIS dá uma pausa para permitir que o operador leia os resultados. Esta pausa, na realidade, é uma INKEY. Tão logo se digite um caractere, o CDS/ISIS, por sua vez, digita-lo-á automaticamente novamente (isto permite inserir uma opção válida de menu, que executará imediatamente sem reexibir o menu). No exemplo acima, não haverá pausa depois de executar a busca 'adult * education' (por causa do AUTOTYPE precedente). O INKEY subsequente é necessário para apagar do "buffer" interno os ' ' digitados automaticamente.

4. Procedimento BOX (lin, col, h, w, frame: real);

Desenha uma janela retangular, com h linhas de altura e w caracteres de largura, começando da linha lin, coluna col. A janela é desenhada com uma linha simples (se frame=1) ou com uma linha dupla (se

frame=2).

Nota de implementação do VAX: em terminais VAX, as janelas são implementadas através do conjunto alternativo de caracteres (G0). Para janelas com uma linha simples, o conjunto gráfico é designado [ESC(0], e para janelas com uma linha dupla o conjunto gráfico alternativo é designado [ESC(2]. Como o último é normalmente uma característica opcional, deve-se verificar se ela é aceita pelo terminal antes de usá-la.

5. Procedimento CHATTR (attr, lin, col, len: real);

Muda o atributo da área da tela começando na linha lin, coluna col, com len caracteres de comprimento para attr. O texto que pode já estar presente na tela nesta área não é apagado. Os valores de attr são os mesmos que para o procedimento ATTR.

Nota de portabilidade: Este procedimento não está disponível nos sistemas VAX ou WANG. Se utilizado nestes computadores ' será ignorado.

6. Função CHR (n: real): string;

Esta função retorna uma cadeia de 1 caractere, contendo o caractere cujo código ASCII é n. Por exemplo, CHR (65) retorna 'A'. Note-se que n deve estar na faixa de 0 - 255; se isto não acontecer o caractere retornado será CHR (n módulo 256).

7. Procedimento CLEAR;

Limpa toda a tela. O atributo de tela é fixado em -2 (fundo de tela) e o cursor é posicionado na linha 1, coluna 1.

8. Procedimento CLEARBOX (lin, col, h, w, attr: real);

Apaga uma janela retangular (lin, col, h, w da mesma forma que para o procedimento BOX). A área apagada é fixado no atributo attr. Os valores de attr são os mesmos que para o procedimento ATTR.

9. Procedimento CLEARDATA;

Apaga a área de dados da tela (linhas 1 a 21). A área apagada é fixada no atributo -2 (fundo de tela) e o cursor é posicionado na linha 1, coluna 1.

10. Procedimento CLEARLN;

Apaga a linha corrente, começando pela posição do cursor.

11. Procedimento CLEARMSG;

Apaga a área de mensagens da tela (linhas 22 a 23). A área apagada é fixada no atributo -1 (atributo da área de mensagens) e o cursor é posicionado na linha 22, coluna 1.

12. Procedimento CLOSE;

Fecha (e desbloqueia se necessário) a base de dados corrente, caso

haja.

13. Função CREATE: real;

Executa a mesma função que a opção de entrada de dados N do menu de serviços de entrada de dados xXE1 (criação de um novo registro). O valor da função é o MFN do registro criado.

Em resposta a função CREATE, o CDS/ISIS seleciona a planilha corrente e mostra a mesma na tela. A entrada de dados normal prossegue até que o operador saia. O registro criado não fica disponível para processamento depois de CREATE. Se necessário, deve-se reler o registro usando a função RECORD. Se se quiser interagir com o operador durante a criação de um registro, deve-se usar a função DATAENTRY.

14. Procedimento CURSOR (lin, col: real);

Posiciona o cursor na linha lin, coluna col. Note-se que ATTR, CHATTR, CLEAR, CLEARDATA, CLEARMSG também posicionam o cursor.

15. Função DATAENTRY (var s1: string): real;

Esta função permite controlar a criação e/ou a edição de um registro e interagir com o operador. Antes de usar esta função deve-se assegurar que um registro se encontra disponível. Isto é feito através do emprego da função RECORD, se se estiver editando um registro existente, ou da função NEWREC, se se estiver criando um registro. Para a edição de registros o CDS/ISIS usará a primeira página da planilha correntemente selecionada (pode-se definir a planilha a ser usada com a função WORKSHEET, se necessário).

DATAENTRY edita uma página de cada vez. Se se necessitar de uma planilha com múltiplas páginas, deve-se definir cada página como uma planilha separada e selecionar a página apropriada antes de chamar DATAENTRY. Devido ao controle adquirido depois de cada página, pode-se selecionar a próxima página dependendo do conteúdo do registro, conforme exemplo seguinte:

```
rc:=record(n);     { read record n }
if rc=0 then
   begin
   rc:=worksheet('a');  { select worksheet a }
   if rc=0 then
      begin
      rc:=dataentry(x); { perform data entry }
      if rc<>0 then check else
      case x of          { check exit option }
      ' ': begin
           t:=field(fieldn(10,1); uc(t);
           case t of { select next worksheet depending on field 10 }
           'M': rc:=worksheet('b');
           'S': rc:=worksheet('c');
           'C': rc:=worksheet('d');
           end;
           end;
```

```
        if (x=' ') and (rc=0) then rc:=dataentry(x);
        if(position(' EXN ',x,1)>0) and (rc=0) then update else check;
        end else writeln('Worksheet not found');
    end else writeln('Record not found');
```

Neste exemplo, a edição de registros começa com a planilha A e continua com as planilhas B, C ou D dependendo do conteúdo do campo 10 (o qual poderia ser, por exemplo, um tipo de indicador de registro).

Note-se que se pode também executar a validação de campo depois de editar uma página.

Na saída o valor da função será atribuído como segue:

0	A edição foi bem sucedida
1	A planilha é inadequada para editar o registro (um campo excede o tamanho especificado na planilha)
2	A edição fez com que o tamanho máximo do registro fosse excedido
3	A planilha não foi encontrada

Se uma condição de erro for verificada, o registro será deixado em sua situação original.

s1 conterá a opção de saída selecionada pelo operador depois de editar a página. É de responsabilidade do programador respeitar este requisito. As opções de saída que podem ser verificadas são listadas abaixo:

espaço	Próxima página (se existir) ou saída e registro de atualização.
B	Página precedente
C	Cancela (sai sem atualização do registro)
T	Cancela e encerra o grupo (caso se esteja oferecendo um grupo de registros para editar)
E	Sai e atualiza o registro (nenhuma página adicional, se existir alguma a ser mostrada)
D	Elimina o registro
N	Sai e atualiza o registro e, então, cria um novo registro

Se uma das opções de saída demandar atualização do registro, deve-se fazê-lo usando o procedimento UPDATE. Note-se que a opção de saída "espaço" pode implicar em atualização se esta foi a última página da planilha.

Enquanto se estiver editando um registro com este método, *não se deve acessar outro registro*, pois isto fará com que as mudanças feitas no registro corrente sejam perdidas. Pode-se, porém, acessar o arquivo invertido, se necessário.

16. Função DATESTAMP: string;

Esta função retorna uma cadeia de 18 caracteres contendo a data e a hora corrente (do sistema operacional), no seguinte formato:

```
MM-DD-AA  HH:MM:SS
```

onde: MM é o mês, DD o dia, e AA o ano; e HH a hora, MM os minutos e SS os segundos (note-se que existem dois espaços entre a data e a hora).

Visto que esta função interage diretamente com o sistema operacional, sua exatidão depende da inserção correta da data e hora do sistema operacional. Em particular naqueles computadores que não preservam automaticamente a data e a hora quando desligados, é essencial que eles sejam restabelecidos manualmente ao seu valor correto cada vez que o computador for ligado.

17. Função DBN: string;

Retorna o nome da base de dados correntemente selecionada ou uma cadeia nula se não existir nenhuma.

18. Procedimento DEFKEY (k: real; s: string);

Define s como o valor da tecla de função k. k é o código de verificação da tecla que está sendo definida. Este procedimento é equivalente a definir a tecla de função no SYSPAR.PAR (ver a seção "Instalação do sistema" no *Manual de Referência do CDS/ISIS*). Por exemplo:

```
defkey(67,substr(datestamp,1,8));
```

fará com que a tecla de função <F9> (código de verificação 67) digite automaticamente a data corrente.

19. Procedimento DELETE;

Marca o registro corrente como logicamente apagado (o registro corrente é o último registro lido através da função RECORD ou estabelecido através da função NEWREC). Uma chamada a DELETE sem uma chamada prévia para RECORD ou NEWREC não é permitida e causará o término do programa.

Depois de DELETE o registro corrente não se encontrará mais disponível.

20. Função DELTERM (t: string): real;

Apaga o termo t do dicionário de arquivo invertido. Note-se que só se pode apagar aqueles termos que não possuem "postings" atuais.

O valor retornado é zero se a operação for executada corretamente e diferente de zero se ela não puder ser executada (por exemplo: tentar apagar um termo que possua um ou mais "postings").

21. Procedimento DUMMYREC;

Como o NEWREC, ele estabelece um registro vazio adequado à entrada de dados. DUMMYREC, porèem, não incrementa o valor máximo do MFN da base de dados e o registro estabelecido não pode ser gravado na base de dados. Na verdade, DUMMYREC não necessita que uma base de dados esteja aberta. Uma tentativa de usar UPDATE ou DELETE em um registro estabelecido com DUMMYREC produzirá um erro de execução.

O propósito do DUMMYREC é facilitar a coleta de dados de uma planilha usando os recursos de entrada de dados do CDS/ISIS de uma maneira semelhante àquela usada pelo CDS/ISIS com planilhas de sistema, tais como a planilha de impressão.

As planilhas a serem usadas juntamente com o DUMMYREC, diferentemente das planilhas reais do sistema, que são criadas com os serviços ISISUTL, devem ser criadas com os serviços ISISDEF. Isto ocorre porque a etiqueta dos campos definidos na planilha identifica cada segmento de dados inserido (os campos de uma planilha de sistema não possuem etiquetas, pois o CDS/ISIS *conhece* os campos pela sua posição relativa). Para definir estas planilhas deve-se usar uma base de dados existente ou uma base de dados estabelecida para este propósito. Porèm, para serem recuperados mesmo quando nenhuma base de dados estiver correntemente selecionada, deve-se atribuir a eles um nome com 'Y' na segunda posição (isto assegurará que eles serão armazenados na localização dos menus - definida no parâmetro 2 do SYSPAR.PAR).

Uma vez que se tenha estabelecido um registro (fictício) através de DUMMYREC, podem-se coletar dados por meio da função DATAENTRY, depois de selecionar a planilha apropriada. Podem-se então recuperar os dados através da função FIELD.

22. Função EDIT (var s: string; max, line, col, wsize, attr: real; fill: string): real;

Esta função permite editar um campo usando o editor de campo do CDS/ISIS. Os parâmetros são descritos a seguir:

s	a cadeia de caracteres a ser editada (depois de EDIT, s conterá a cadeia editada resultante). Note-se que s pode estar inicialmente vazia
max	o tamanho máximo de expansão (até 255 caracteres) permitida a cadeia durante a edição
line, col	a posição da linha e da coluna do canto superior esquerdo da janela de edição
wsize	o tamanho da janela de edição (se wsize<max então o campo rolará)
attr	o atributo de tela da janela de edição
fill	o caractere de preenchimento a ser usado para preencher posições vazias da janela de edição

EDIT apaga a janela de edição e preenche-a com o conteúdo da cadeia de caracteres a ser editada. O alcance total da função do editor de campo do CDS/ISIS pode ser usado durante a edição. Note-se também que a linha 24 é usada para mensagens do modo de edição (insere/substitui).

Na saída o valor da função indica a tecla de término de edição, como segue:

0	ENTER	Edição terminada normalmente
1	F1	A tecla ajuda foi pressionada
2	<TAB>	A tecla <TAB> foi pressionada
3	F2	A cadeia de caracteres foi apagada
4	<ESC>	A tecla ESC foi pressionada
5	<PgDn>	A tecla PgDn foi pressionada

É de responsabilidade do programador interpretar o significado das várias opções de término. Note-se porém, que se a cadeia de caracteres foi modificada, <ESC> restitui a versão *modificada*. Se se quiser implementar o significado padrão desta tecla, isto é, restaurar o campo ao seu valor original e ignorar quaisquer mudanças feitas no mesmo, e deve-se armazenar o valor original antes de chamar EDIT.

23. Função ENCINT (n1, n2: real): string;
Trunca o valor de n1 para um inteiro e o converte em sua representação ASCII com uma precisão mínima de n2 dígitos.

24. Função ENCREAL (n1, n2, n3: real): string;

Converte n1 para a sua representação ASCII. n2 é o menor tamanho do campo e n3 é o número de posições decimais.

25. Procedimento EXEC (s1: string);

Executa o programa s1. Note-se que o controle não é retornado ao programa corrente. EXEC pode apenas ser usado para programas "batch". Se usado em uma saída de usuário, os resultados serão imprevisíveis.

26. Função FIELD (n: real): string;

Retorna o valor do n-ésimo campo no registro corrente. Note-se que n *não* é a etiqueta do campo, mas o *número do campo* requerido obtido através da função FIELDN.

Por exemplo:

```
s:=field(fieldn(10,1));
```

retornará no s o valor da primeira ocorrência do campo 10 ou uma cadeia vazia, se o campo 10 não estiver presente no registro.

27. Função FIELDN (tag, occ: real): real;

Retorna o *número do campo* de um determinado campo no registro corrente. O número do campo é o número ordinal da entrada do diretório correspondente à occ-ésima ocorrência do campo com a

etiqueta tag. O número do campo em um campo não existente é 0.

28. Função FILEXIST (f: string): real;

Esta função retorna 0 se o arquivo f existir. f deve ser uma expressão de cadeia de caracteres produzindo um nome de arquivo válido para o operacional em uso. Pode-se usar a função PATH sempre que a localização do arquivo possa depender do SYSPAR.PAR ou do dbn.PAR. Por exemplo:

```
n:=filexist(path('sys',4)|'meuarq.wrk');
```

retornará 0 se o arquivo meuarq.wrk existir na localização dos arquivos de trabalho definido no parâmetro 4 do SYSPAR.PAR.

29. Função FIND (var s1: string): real;

Busca s1 no dicionário da base de dados corrente e retorna 0 se este existir. Se o termo não existir, s1 conterá na saída, o próximo termo de mais alta ordem ou uma cadeia vazia se o fim do dicionário tiver sido alcançado. Depois de FIND pode-se usar NXTTERM para recuperar termos do dicionário seqüencialmente.

30. Função FLDADD (tag, n: real; s: string): real;

Adiciona um novo campo ao registro corrente. tag é a etiqueta do campo e s é o conteúdo do campo a ser adicionado. O campo é adicionado antes do n-ésimo campo correntemente existente no registro (pode-se especificar n=1 se a ordem não for importante ou n=NFIELDS+1 para adicionar uma nova ocorrência de um campo repetitivo depois da última ocorrência correntemente presente).

O valor retornado é 0 se a operação tiver sido executada com sucesso, ou 1 se não tiver sido possível executá-la (por exemplo: falta de espaço no registro).

31. Função FLDCHA (n: real; s1, s2: string): real;

Substitui s1 por s2 no n-ésimo campo do registro corrente (n deve ser obtido com a função FIELDN). O valor retornado é 0 se a operação tiver sido executada corretamente ou 1 se não tiver sido possível executá-la (por exemplo: s1 não foi encontrado).

32. Função FLDDEL (n: real): real;

Elimina o n-ésimo campo do registro corrente (n deve ser obtido com a função FIELDN). O valor retornado será 0 se a operação tiver sido executada com êxito ou 1 se não tiver sido possível executá-la (por exemplo, o campo não existe).

33. Função FLDMOD (n1, n2, n3: real; s: string): real;

Elimina a parte do n1-ésimo campo no registro corrente (n1 deve ser obtido com a função FIELDN), começando na posição n2, de tamanho n3 e o substitui por s1. O valor retornado será 0 se a operação tiver sido executada com êxito ou 1 se não tiver sido possível executá-la (por

exemplo, o campo não existe). Um exemplo é dado abaixo:

```
rc:=fldmod(fieldn(10,1),5,8,'xxx');

                Conteúdo do campo 10

      antes de fldmod              depois de fldmod

  International Corporations      Intexxxl  Corporation
```

34. Função FLDREP (n: real; s: string): real;

Substitui o conteúdo do n-ésimo campo no registro corrente por s (n deve ser obtido com a função FIELDN). O valor retornado será 0 se a operação tiver sido executada com êxito ou 1 se não tiver sido possível executá-la (por exemplo: o campo não existe).

35. Função FMTNAME: string;

Esta função retorna o nome do formato de exibição corrente. Os seguintes nomes especiais também podem ser retornados:

''	uma cadeia de caracteres nula indica que nenhum formato está disponível (por exemplo, se ainda não tiver sido aberta nenhuma base de dados)
*	o formato corrente foi definido pelo usuário durante a sessão corrente (ele, portanto, não existe como um arquivo .PFT)
ALL	o formato corrente é o formato embutido

36. Função FORMAT (lw: real): real;

Formata o registro corrente de acordo com o formato correntemente selecionado usando um tamanho de linha de lw caracteres. O valor retornado será 0 se a operação tiver sido executada com êxito ou um código de erro de formato se um erro de formato tiver sido encontrado (o código retornado é um daqueles listados na seção "A Linguagem de Formatação" no *Manual de Referência do CDS/ISIS*.

Observe-se que esta função simplesmente executa o formato: as linhas produzidas são armazenadas em uma área de trabalho. Pode-se usar a função LINES para determinar quantas linhas foram produzidas e a função NXTLINE para obter o conteúdo de cada linha.

37. Função GETCC: real;

Retorna a posição da coluna corrente do cursor.

38. Função GETCL: real;

Retorna a posição da linha corrente do cursor.

39. Procedimento GETFMT (s: string);

Define o formato corrente de exibição. s pode ser um formato atual na linguagem de formatação do CDS/ISIS ou o nome de um formato pré-definido. No último caso o nome deve ser precedido por um símbolo @.

Por exemplo:

```
GETFMT ('v24/v56(0,4)');
GETFMT ('@xxx');
```

Note-se que o formato definido com este procedimento substitui o formato *corrente* e permanece ativo mesmo depois do término do programa. Pode-se usar a função FMTNAME para armazenar o nome do formato corrente, de forma que se possa restaurá-lo, antes de sair, se for necessário.

40. Função INKEY: string;

Retorna o próximo caractere digitado pelo teclado (ou disponível no "buffer" AUTOTYPE interno). Se o caractere for atualmente digitado no teclado, ele será ecoado na tela, enquanto que caracteres digitados automaticamente nunca são ecoados.

A cadeia retornada é sempre uma cadeia de 1 caractere. Note-se que a tecla <CR> retorna um espaço. Depois desta função o cursor ' esempre posicionado na coluna 1 da linha seguinte.

41. Função KBDKEY (var c: string): real;

Retorna em c o próximo caractere digitado no teclado (ou disponível no "buffer" AUTOTYPE interno). Se c for um caractere ASCII, o valor da função será 0, do contrário, ele será o *código de verificação* da tecla pressionada ou digitada automaticamente. Na maioria dos casos, se o valor da função for maior que 0 então c será estabelecido igual a CHR(0). Em alguns casos, o valor da função e o ORD(c) são maiores que zero. Alguns exemplos são oferecidos a seguir:

Tecla pressionada	valor de c	valor da função
A	A	0
<F2>	CHR(0)	60
<ESC>	CHR(27)	1

Pode-se usar o seguinte programa-amostra para determinar os valores exatos retornados por esta função (pressionar x para sair do programa):

```
program SCANC;


var c: string;
    sc: real;

begin
repeat
write('Press a key ');
sc:=kbdkey(c);
writeln('Scan code: ',sc:3,' ASCII code: ',ord(c):3,' char: ',c);
until(c='x') or (c='X');
end.
```

42. Função LANG: string;

Retorna uma cadeia de 1 caractere contendo o código do idioma selecionado.

43. Função LINES: real;

Retorna o número de linhas produzidas pela função FORMAT. Note-se que o tempo de execução desta função é proporcional ao número de caracteres produzidos pelo formato. Deve-se usá-la, portanto, apenas quando necessário (ver também a função NXTLINE).

44. Função LOCK: real;

Esta função solicita um bloqueio exclusivo de gravação para a base de dados correntemente selecionada. Se o bloqueio for concedido o valor da função será 0. Para se obter controle exclusivo com êxito, não deve haver nenhum outro usuário executando uma operação de atualização sobre a base de dados no momento em que o bloqueio for requisitado. Se o bloqueio for concedido, nenhum outro usuário terá permissão para executar qualquer operação de atualização, até que o bloqueio seja removido. Uma vez concedido, um bloqueio permanece ativo até que: (a) um procedimento UNLOCK (ver abaixo) seja executado na *mesma* base de dados; (b) a base de dados seja (implícita ou explicitamente) fechada; ou (c) a sessão tenha se encerrado.

Deve-se trancar uma base de dados sempre que o seu programa executar quaisquer operações de saída ou de atualização do arquivo-mestre ou sobre o arquivo invertido, isto é, se se executarem quaisquer dos seguintes procedimentos: NEWREC, UPDATE, DELETE, CREAT, MODIFY, DELTERM ou UPDIF. Uma tentativa de gravar em uma base de dados sem antes emitir LOCK (ou ignorar um pedido de bloqueio mal sucedido), terminará a sessão anormalmente.

Nota de portabilidade: A função LOCK, como descrita acima, só está implementada para sistemas VAX. Em IBM-PC's ou sistemas WANG, onde o acesso para gravação simultânea a uma base de dados não é permitido, esta função sempre retorna 0 (isto é, o acesso para gravação é sempre concedido). Se o programa se destina a trabalhar tanto no VAX quanto nos PC's, deve-se emitir LOCK sempre que apropriado. Do contrário, sua aplicação não será portável.

45. Função MAXMFN: real;

Retorna o próximo MFN a ser atribuído na base de dados corrente.

46. Função MENU (s: string): string;

Mostra o menu de sistema s e retorna a seleção do usuário. A cadeia de caracteres retornada é sempre uma cadeia de 1 caractere. A cadeia de caracteres s deve conter o nome do menu a ser mostrado (não o nome do arquivo), e este nome deve adaptar-se as convenções dos nomes de menus do CDS/ISIS, isto é, o primeiro caractere deve ser o código do idioma e o segundo deve ser X, por exemplo: PXABC.

Note-se que o CDS/ISIS selecionará automaticamente a versão do idioma do menu correspondente à linguagem correntemente estabelecida. E de

responsabilidade do programador assegurar que todas as versões de idioma necessárias do menu tenham sido previamente definidas.

47. Função MODIFY (n: real): real;

Executa a mesma função que a opção E do menu de serviços de entrada de dados xXE1. n é o MFN do registro a ser editado. O valor retornado será 0 se a operação tiver sido executada corretamente ou diferente de zero 0 se não tiver sido possível executá-la.

48. Procedimento MSG (n: real);

Mostra a mensagem n do CDS/ISIS (no idioma correntemente selecionado). A mensagem é mostrada começando na posição corrente do cursor, e depois de mostrar a mensagem do cursor será posicionado imediatamente depois do texto da mensagem.

Se n não corresponder a um número de mensagem válido, então o CDS/ISIS exibirá MSG-n [text not found] (N.T.: texto não encontrado).

49. Função MSGTEXT (n: real): string;

Retorna o texto da mensagem n do CDS/ISIS (no idioma correntemente selecionado). Se n não corresponder a um número de mensagem válido, uma cadeia vazia é retornada.

50. Função NEWREC: real;

Inicializa um novo registro vazio e retorna o seu MFN. Podem-se, então, inserir dados no registro, através da função FLDADD e escrevê-lo no arquivo-mestre usando o procedimento UPDATE.

O registro estabelecido através do NEWREC torna-se o registro *corrente*, isto é, todos os procedimentos direcionados para o registro e as funções executadas subseqüentemente, até e inclusive o UPDATE ou o DELETE, funcionarão nesse registro.

51. Função NFIELDS: real;

Retorna o número de campos no registro corrente. Note-se que cada ocorrência de um campo repetitivo é contada como um campo. Por exemplo, se o registro contiver uma ocorrêcia dos campos 10 e 20 e 3 ocorrências do campo 30, então NFIELDS retorna 5.

52. Função NOCC (n: real): real;

Retorna o número de ocorrências do campo com etiqueta n no registro corrente (ou 0 se o campo não existir).

53. Função NXTLINE (var lin: string): real;

Retorna em lin a próxima linha produzida pela função FORMAT. O valor da função será 0 se uma linha estiver disponível em lin. Um valor diferente de zero será retornado se nenhuma linha a mais estiver disponível. Quando chamada imediatamente depois de FORMAT, NXTLINE retorna a primeira linha; cada chamada subseqüente retornará a próxima linha. O valor da função deve ser verificado para determinar

se lin contém dados válidos.

O exemplo seguinte ilustra o uso desta função:

```
var rc,mfn,lrc,frc: real;
    lin: string;
- - - - - - - - - -
rc:=record(mfn);
if rc=0 then
   begin
   frc:=format(79);
   if frc=0 then
      begin
      lrc:=nxtline(lin);
      while lrc=0 do begin writeln(lin); lrc:=nxtline(lin); end;
      end
   else writeln('Format error ',frc:1);
   end;
```

Note-se que o exemplo a seguir produziria os mesmos resultados, embora ele tivess sido levemente menos eficiente, especialmente se o formato tivesse produzido muitas linhas (por causa da função LINES):

```
var rc,mfn,lrc,frc,i:real;
    lin: string;
- - - - - -
rc:=record(mfn);
if rc=0 then
   begin
   frc:=format(79);
   if frc=0 then
for i:=1 to lines do begin lrc:=nxtline(lin); writeln(lin); end
else writeln('Format error ',frc:1);
end;
```

54. Função NXTPOS (n: real): real;

Retorna o MFN do próximo n-ésimo registro recuperado, relativo ao registro corrente. NXTPOS pode ser usado, depois de SETPOS ou de um NXTPOS prévio, para obter o MFN da do próximo registro no conjunto corrente. O conjunto corrente é o determinado pelo último SETPOS emitido. n pode ser maior que 1. Por exemplo: NXTPOS(3) fornecerá o MFN do terceiro registro recuperado depois do registro corrente. Valores negativos de n não são permitidos.

NXTPOS retornará 0 quando nenhum (ou nenhum outro) registro estiver disponível.

55. Função NXTPOST: real;

Depois de FIND ou NXTTERM, posiciona para o próximo "posting" para o termo. O "posting" pode ser subseqüentemente analisada com a função POSTING. Um valor negativo será retornado se mais nenhum "posting" estiver disponível.

56. Função NXTTERM: string;

Retorna o próximo termo do dicionário. NXTTERM pode ser usado depois de FIND ou outro NXTTERM para ler o dicionário seqüencialmente. Um valor nulo será retornado quando mais nenhum termo estiver disponível (isto é, o fim do dicionário tiver sido alcançado).

O uso do NXTTERM sem um FIND prévio produz resultados imprevisíveis.

57. Procedimento OPEN (s: string);

Abre (isto é, torna disponível para processamento) a base de dados s. Depois de executado este procedimento, a base de dados anterior, caso exista, não estará mais disponível. A execução deste procedimento equivale a selecionar a opção C (mudar de base de dados) no menu principal de serviços xXISI. Se uma outra base de dados já estiver selecionada ao ser executado este procedimento, a base de dados antiga será automaticamente fechada (e desbloqueada, se necessário).

Nota de portabilidade: No sistema VAX, a base de dados é aberta no modo "read-only" (somente para leitura). Se se pretender executar operações de gravação na base de dados que se está abrindo, deve-se também emitir uma função LOCK.

58. Função ORD (s: string): real;

Esta função retorna o código ASCII do primeiro caractere de s. Por exemplo, o valor de ORD('A') é 65.

59. Procedimento PAGE (n: real);

Seleciona a página n (1 <= n <= 4) como a página ativa. A página deve ter sido previamente armazenada com SAVESCR, do contrário o resultado será imprevisível.

Nota de portabilidade: Este procedimento não está disponível nos sistemas VAX ou WANG. Se usado nestes computadores ele será ignorado.

60. Função PATH (s1: string; n: real): string;

O valor desta função depende do valor de s1, como segue:

s1='SYS' o valor retornado será o valor do parâmetro n definido no SYSPAR.PAR (neste caso 1 <= n <= 5);

s1='DBN' o valor retornado será o valor do parâmetro n definido no dbn.PAR (neste caso 1 <= n <= 10). Note-se que, neste caso, o valor retornado será confiável apenas se a base de dados estiver correntemente selecionada. Se não, o valor retornado será a localização definida no parâmetro 5 do SYSPAR.PAR.

Por exemplo, se se quisesse ler a FDT da base de dados corrente, poder-se-ia usar o seguinte procedimento ASSIGN:

```
Assign('inp',path('dbn',10)|dbn|'.fdt');
```

61. Função POSITION (s1, s2: string; n: real): real;

Busca a cadeia de caracteres s2 na cadeia s1, começando pela posição n e retorna sua posição, ou 0 se s2 não ocorrer dentro de s1.

Por exemplo, position('abcd','c',1) retorna 3.

62. Função POSTING (s1: string): real;

Retorna o valor da componente "posting" indicada por s1. O "posting" corrente deve ter sido previamente estabelecido por NXTPOST. A componente retornada é indicada por s1 como segue:

MFN número do arquivo-mestre

TAG identificador do campo

OCC número da ocorrência

CNT número de sequência

63. Função RECORD (n: real): real;

Recupera o registro do arquivo-mestre com MFN=n. O registro recuperado torna-se o *registro corrente*, isto é, todos os procedimentos e funções orientados a registro subseqüentemente executados operarão sobre este registro. O valor retornado indica se o registro foi recuperado com êxito, como segue:

-1 Fim do arquivo

0 O registro foi recuperado

1 O registro foi marcado para ser apagado

2 O registro foi fisicamente apagado

Nenhuma operação orientada a registro deverá ser executada se o valor retornado for -1. Se o valor retornado for 1, o conteúdo do registro no momento em que ele foi marcado para eliminação estará disponivel para inspeção e o sinal de eliminação será removido. Se se quizer deixar o registro marcado para eliminação, não se deve grava-lo novamente. Alternativamente, pode-se reativar o registro gravando-o novamente por meio do procedimento UPDATE (depois de modificá-lo, se necessário).

Se o valor retornado for, 2 o CDS/ISIS fornecerá um registro vazio, o qual se pode preencher com a função FLDADD e grava-lo no arquivo-mestre usando o procedimento UPDATE. Este registro reterá seu MFN original. Se não se escrever o registro de volta, ele permanecerá eliminado.

64. Procedimento SAVESCR (n: real);

Armazena a tela corrente no "buffer" interno n (1 <= n <= 4). A tela não será afetada por esta operação. Uma tela armazenada poderá ser restaurada usando-se o procedimento PAGE.

Note-se que uma tela armazenada só fica disponível para restauração dentro do programa que a armazenou. Não se deve esperar que uma tela armazenada por um programa fique disponível para outro programa ou para uma subsequente re-execução do programa que a armazenou.

Nota de portabilidade: Este procedimento não está disponível nos sistemas VAX ou WANG. Se usado nestes computadores, será ignorado.

65. Função SEARCH (s: string): real;

Executa uma busca do CDS/ISIS na base de dados correntemente selecionada usando a expressão de busca definida por s e retorna o número da expressão atribuído a expressão de busca. Executar esta função equivale a selecionar a opção S dos serviços ISISRET.

Observe-se que, se a expressão em s contiver um erro de sintaxe, o usuário poderáa editá-lo interativamente. Se s for uma cadeia nula ('') é solicitado ao usuário que insira a expressão de busca.

66. Função SETLANG (s: string): real;

Estabelece o idioma corrente de diálogo. s é uma cadeia de 1 caractere contendo o código do idioma a ser estabelecido. Se se usar esta função, deve-se assegurar-se de que todos os menus, planilhas e arquivos de mensagens necessários para o idioma selecionado existam.

A função retornará 0 se o idioma indicado tiver sido selecionado com êxito, ou 1 se não tiver sido possível estabelecer o idioma (se nenhum arquivo de mensagens tiver sido definido). Se o valor retornado for 1, o idioma corrente será indefinido. Deve-se, portanto, reinstalar um idioma válido antes de continuar. Por exemplo:

```
Program SETLNG;
var l: string;
begin
l:=lang;          { armazena o código do idioma corrente }
if setlang ('q')=1
   then  setlang(1);  { restabelece o idioma armazenado se 'q' for
                        indefinido }
end.
```

67. Função SETPOS (n1, n2: real): real;

O valor retornado por esta função depende do valor de n2 como segue:

n2<>0 Retorna o MFN do próximo n2-ésimo registro recuperado pela n1-ésima busca (neste caso n1 é o número da expressão de busca retornado por um SEARCH prévio). n2 pode ser maior que 1. Por exemplo: SETPOS (2,3) fornecerá o MFN do terceiro registro recuperado pela expressão 2. Para uma performance otimizada, registros subsequentes devem ser obtidos através de NXTPOS. Valores negativos de n2 não são permitidos.

n2=0 Retorna o número de registros recuperados ("hits") pela n1-ésima busca.

Em ambos os casos, se n1 for fixado em zero, a expressão de busca considerada é a última expressão submetida.

68. Função SIZE (s: string): real;

Retorna o valor corrente da cadeia de caracters s.

69. Função SUBSTR (s: string; n1, n2: real): string;

Retorna a subcadeia de caracteres de s, começando na posição n1 com n2 de tamanho máximo. Uma cadeia nula será retornada se n1 estiver fora dos limites de s. Se n1+n2-1 for maior que SIZE(s) apenas os últimos n1+SIZE(s)-1 caracteres de s serão retornados.

70. Procedimento UC (var s: string);

Converte a cadeia s para maiúscula. A conversão será executada de acordo com a tabela do sistema ISISUC.TAB.

71. Procedimento UNLOCK;

Remove um bloqueio da base de dados estabelecido por uma chamada prèvia a função LOCK. Uma vez que a função LOCK requer controle exclusivo de gravação sobre a base de dados, deve-se remover o bloqueio tão logo ele não seja mais necessário. Observe-se que o procedimento UNLOCK não precisa ser emitido pelo mesmo programa que executou a função LOCK.

Nota de portabilidade: A função UNLOCK só é implementada nos sistemas VAX. Em sistemas IBM-PC ou WANG, onde o acesso de gravação simultâneo a uma base de dados não é permitido, este procedimento não tem efeito. Porêm, se o programa se destinar a trabalhar tanto em sistemas VAX quanto nos PCs, então deve-se emitir UNLOCK sempre que for apropriado para assegurar a portabilidade.

72. Procedimento UPDATE;

Regrava o registro corrente no arquivo-mestre (o registro corrente é o último registro lido através da função RECORD ou estabelecido através da função NEWREC). Uma chamada para UPDATE sem uma chamada prèvia para RECORD ou NEWREC é incorreto e causará o término do programa.

73. Procedimento UPDIF;

Executa uma atualização do arquivo invertido (que é equivalente a selecionar a opção U do menu xXG1 dos serviços ISISINV).

74. Função VAL (s: string): real;

Converte s para REAL. Se s não contiver um valor numérico válido o valor retornado será zero.

75. Função WORKSHEET (s: string): real;

Seleciona a planilha de entrada de dados s. Todas as operações de

entrada de dados subseqüentes usarão esta planilha. O valor retornado será zero se a planilha tiver sido recuperada corretamente e diferente de zero se a planilha não tiver sido encontrada. A cadeia de caracteres s deve conter o *nome* da planilha, não o nome do arquivo. Executar esta função é equivalente a selecionar opção W do menu xXE1 dos serviços ISISENT.

Note-se que, uma vez que uma planilha tenha sido selecionada, ela permanecerá ativa mesmo depois de terminado o programa.

Seção 4

Programas-amostra

Esta seção descreve os programas-amostra fornecidos no disquete de EXEMPLOS. E aconselhável executar de fato os programas (eles podem ser executados usando-se a opção A do menu principal) a medida em que se estuda a listagem.

A. Programa KEYB

Este programa ("batch") desenha na tela a parte numérica do teclado do IBM-PC/XT e ilustra a aplicação dos procedimentos BOX e CLEARBOX.

Para simplificar o algoritimo, o programa assume que existem 20 teclas numeradas como segue:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

porém, como as teclas 16 e 17 são longas, elas são manipuladas de modo especial.

Os "arrays" l e c contêm a posição linha/coluna do canto superior esquerdo de cada tecla. Estas são inicializadas com base nos valores de l1 e c1, os quais determinam a posição do desenho na tela. As variáveis de cadeias de caracteres top e bot contêm os rótulos a serem mostrados no meio e na parte inferior da tecla correspondente. Uma vez que os "arrays" de cadeia de caracteres não são permitidos pelo Pascal do CDS/ISIS, eles são simulados aqui usando-se uma única variável de cadeia, contendo subcadeias de tamanho fixo: top contêm 20 cadeias de 4 caracteres e bot contêm 20 cadeias de 1 caractere.

O teclado é desenhado duas vezes, tecla por tecla, usando o procedimento DRAWKEY: a primeira vez com atributo 0 e a segunda com atributo 1. O efeito visual obtido é que, depois de serem desenhadas, as teclas mudarão de cor.

Note-se que o procedimento DRAWKEY duplica a altura da tecla 16 e a largura da tecla 17, e não desenha as teclas 18 e 20, pois estas correspondem a segunda metade das teclas longas 16 e 17.

Note-se como os rótulos corretos das teclas são extraídos de top e bot usando a função SUBSTR.

B. Programa DISPL

Este programa ilustra o uso dos procedimentos de formatação. Ele mostra, seqüencialmente, os registros das bases de dados CDS. Quatro campos são mostrados para cada registro: o MFN, o titulo (campo 24), os autores (campo 70) e as palavras-chave (campo 69). Cada campo é encerrado numa moldura. Depois de mostrar um registro o programa emite mensagem 'N[ext record] Q[uit]' e termina quando a resposta é Q ou o Gion do arquivo

mestre é alcançado (rc<0).

O procedimento DISPLAY limpa as janelas, define o formato para cada campo e então mostra-o usando o procedimento WRT.

C. Programa TEXT

Este programa executa uma busca simples de texto livre numa base de dados. Primeiro pede-se que se insira a etiqueta do campo e em seguida a cadeia de caracteres para ser buscada. A busca é feita usando-se a função POSITION. Como tanto a cadeia a ser buscada quanto o conteúdo do campo são convertidos para maiúsculas, antes de serem combinados, a busca independe do uso de maiúsculas ou minúsculas. A medida em que um registro satisfizer a busca, ele será mostrado e pode-se, então, continuar ou sair. O programa faz uma pausa a cada 100 registros consecutivos que não satisfaçam os critérios de busca.

C. Programa THES

Este programa fornece um recurso completo (em um único idioma) de gerenciamento de Tesauros. Ele é projetado como uma saída de menu (mas também pode ser executado como um programa independente). Ele assume que uma base de dados de Tesauros chamada THES existe.

As seguintes variáveis globais são usadas:

maxt — o número máximo de relações definidas

rel — uma cadeia de caracteres contendo o nome (de 3 caracteres) das relações

invrel — uma cadeia de caracteres contendo o nome (de 3 caracteres) das relações inversas

tag — um "array" contendo a etiqueta de cada relação

term — o termo corrente do tesauros em processamento

q — armazena os termos de busca selecionados através da opção Q (ver a seguir)

Submenus e "prompts" são mostrados na área de mensagens. As opções principais do programa vêm descritas a seguir:

T — permite selecionar um termo: uma janela vazia é mostrada no alto da tela onde o termo deve ser inserido; se existir, será exibido. Caso contrário, a opção L será selecionada automaticamente; o programa terminará se nenhum termo for selecionado.

L — mostra a lista de termos do tesauro em ordem alfabética começando pelo termo corrente; pode-se avançar páinas pressionando P, mover-se termo a termo com "Enter" (para frente) ou B (para tras), ou selecionar as opções S, C, T ou X;

S — seleciona um determinado termo e mostra o registro correspondente (ver a descrição do procedimento FINDTERM);

A adiciona uma relação a um termo (ver ADDREL);

C cria um novo termo do tesauro (ver CREATERM);

D dependendo da posição do cursor, apaga um termo (ver DELTERM) ou uma relação (ver DELREL);

Q seleciona um termo para busca, se outros termos tiverem sido previamente selecionados, um operador '+' (OR lógico) será inserido;

? mostra o conjunto corrente de termos de busca selecionados com a opção Q, se existirem; estes são mostrados em uma janela do tipo "pop-up";

X sai; se termos de busca tiverem sido selecionados, a expressão de busca será mostrada no modo de edição e então executada, desde que uma base de dados tenha sido selecionada quando THES foi chamada.

O programa usa um número de procedimentos e funções internos:

FUC: converte uma cadeia de caracteres para maiúsculas;

ERRMSG: sinaliza com um "beep" e mostra uma mensagem de erro na área de mensagens, e, então, espera que uma tecla seja pressionada;

DISPLT: apaga a área de dados e mostra a janela de termos; se chamada a partir da opção S ela também mostrara o MFN do termo; a tela é armazenado no "buffer" 1;

DISPLAY: mostra até maxl relações de um termo, começando de determinada etiqueta e ocorrência; o índice da etiqueta e a ocorrência para cada relação mostrada é gravado nos "arrays" dt e doc, respectivamente, e nl é atribuído ao número de relações exibidas;

DECIDE: permite visualizar as relações de um termo e selecionar uma das opções do programa descritas acima; se a opção selecionada for S, o termo selecionado será armazenado em term;

FINDTERM: busca um termo e mostra-o se ele existir; atribui mfn ao MFN do registro correspondente e atribui action a opção selecionada através de DECIDE;

FLDUC: retorna um dado campo do registro convertido para maiúscula;

CHKREL: verifica se uma relação já existe;

UPDINVF: atualiza o arquivo invertido;

CREATERM: cria um novo termo do tesauro (desde que ele ainda não exista): um novo registro é criado e o arquivo invertido é atualizado imediatamente;

ADDREL: adiciona uma relação a um termo; verifica se o termo relacionado já está definido no tesauro e se uma relação para o mesmo termo já não existe; a relação inversa, se necessária, também é adicionada ao registro correspondente ao termo relacionado;

DELREL: elimina uma relação, e, se necessário, a relação inversa correspondente;

DELTRM: elimina um termo, desde que ele não possua relações; o arquivo invertido também é atualizado;

SHOWDICT: mostra a lista alfabética dos termos do tesauro e permite selecionar uma das opções C, S, T ou X; pode-se avançar as páginas pressionando P, ou mover-se termo a termo com "Enter" (para frente) ou B (para trás).

```
Program KEYB;

{ Draws the numeric keypad of the IBM PC keyboard }

var l: array[1..20] of real;
    c: array[1..20] of real;
    l1,c1,i,j: real;
    top,bot: string;


Procedure DRAWKEY(i,a: real);

{ Draws a key with i=key number, a=attribute }

Var h,w: real;

Begin
  if (i<>18) and (i<>20) then
     begin
       if i=16
          then begin h:=6; w:=6; end
          else if i=17
                  then begin h:=3; w:=12; end
                  else begin h:=3; w:=6; end;
       if a<>0 then clearbox(l[i],c[i],h,w,a);
       box(l[i],c[i],h,w,1);
       cursor(l[i]+1,c[i]+1); write(substr(top,(i-1)*4+1,4));
       if substr(bot,i,1)<>' ' then
          begin cursor(l[i]+2,c[i]+1); write(substr(bot,i,1)); end;
     end;
end;

Begin
top:='Esc Num ScrLSysRHome ^  PgUpPrtS <-      ->     End  v  PgDn    Ins
Del     ';
bot:='    789*456-123+0 . ';
l1:=3; c1:=30;
clear;
for i:=1 to 5 do for j:=1 to 4 do l[(i-1)*4+j]:=l1+(i-1)*3;
for i:=1 to 5 do for j:=1 to 4 do c[(i-1)*4+j]:=c1+(j-1)*6;

for i:=1 to 20 do drawkey(i,0);

for i:=1 to 20 do
begin
  drawkey(i,1);
  for j:=1 to 200 do; { wait loop }
end;

cursor(22,1);

end.
```

```
Program DISPL;
VAR I,J,K,N,RC,L: REAL;
    F,LIN: STRING;

PROCEDURE WRT(I,NL,LW: REAL);
VAR L,LL,RC: REAL;
BEGIN
RC:=FORMAT(LW);
RC:=NXTLINE(LIN); L:=I;
LL:=L+NL-1;
WHILE RC=0 DO
 BEGIN
 CURSOR(L,2);
 IF L>LL THEN RC:=1
        ELSE BEGIN WRITELN(LIN); RC:=NXTLINE(LIN); END;
 L:=L+1;
 END;
END;

Procedure Display;
Begin
CURSOR(2,74); WRITE(I:4);

CLEARBOX(2,2,1,28,2);
GETFMT('MDL,(V70/)');
WRT(2,1,28);

CLEARBOX(5,2,4,78,1);
GETFMT('MDL,V24/');
WRT(5,4,78);

CLEARBOX(11,2,2,78,1);
GETFMT('MDL,V70+|; |/');
WRT(11,2,78);

CLEARBOX(15,2,6,78,1);
GETFMT('MDL,V69/');
WRT(15,6,78);

End;


BEGIN
OPEN('CDS');
I:=0;
CLEAR;
BOX(1,1,3,30,2); BOX(1,71,3,10,1);
BOX(4,1,6,80,1);
CURSOR(4,1);WRITELN('Title: ');
BOX(10,1,4,80,1);
CURSOR(10,1);WRITELN('Authors: ');
BOX(14,1,8,80,2);
CURSOR(14,1);WRITELN('Keywords: ');

REPEAT
I:=I+1;
```

```
RC:=RECORD(I);
if rc=0 then    { Record exists }
   begin
   Display;
   CURSOR(23,1); WRITE('N[ext record]  Q[uit]'); F:=INKEY; uc(f);
   end;

UNTIL (F='Q') or (rc<0);
END.
```

```
Program TEXT(S1: string);

{ Performs a simple text search }

var tc,tag,rc,n,i,o,j,k,tot,nr,y: real;
    dbname,fld,s,r: string;

Begin
clear;
dbname:=dbn;
if dbname='' then
   begin
   msg(1); readln(dbname);
   open(dbname);
   end;
write('Data base name: ',dbname);
cursor(2,1); write('Tag to be searched?     ');
cursor(3,1); write('String to be searched? ');

repeat
box(5,2,16,78,2);
r:='';
cursor(2,24); write('      '); cursor(2,24);
readln(s);
if s<>'' then
   begin
   tag:=val(s);
   cursor(3,24); write('                                        '); cursor(3,24); readln(s);
   uc(s);
   i:=0; y:=0; nr:=0; tot:=0; tc:=0;
   cursor(22,1); writeln('Records retrieved:           ');
   repeat
   i:=i+1;   rc:=record(i);
   tc:=tc+1;
   cursor(5,5); writeln(i:1);
   if rc=0 then
      begin
      nr:=nr+1; n:=nocc(tag); o:=0;
      while o<n do
      begin
      o:=o+1;
      j:=fieldn(tag,o); fld:=field(j); uc(fld); k:=position(fld,s,1);
      if k<>0 then
         begin
         Y:=Y+1;
         k:=format(70); o:=6;
         clearbox(6,3,14,76,1);
         while (nxtline(fld)=0) and (o<19) do
           begin cursor(o,5); writeln(fld); o:=o+1; end;
         cursor(22,20); write(y:1,'  '); r:=inkey; uc(r);
         clearbox(6,3,14,76,0);
         tc:=0;
         end;
      end;
      end;
  if tc=100 then

   begin
```

```
      cursor(23,1);
      write('<CR> - continue for 100 more records   X - Exit --> ');
      r:=inkey; uc(r);
      tc:=0;
      end;
   until (rc<0) or (r='X');
   end;
until (s='') or (r='X');
s1:=' ';
end.
```

```
Program THES(option: string) [menu];

var dt: array[1..15] of real;
    doc: array[1..15] of real;
    tag: array[1..10] of real;    { tag of relation }
    maxt: real;                   { max no. of tags (upper bound of tag) }
    maxl: real;                   { max no. of lines (upper bound of dt,doc) }
    rel,invrel: string;           { Relation indicators }
    it,io: real;                  { current tag/occ }
    nl: real;                     { lines on this page }
    cl: real;                     { current line }
    term: string;                 { current term }
    q: string;                    { query }
    dbname: string;               { current data base }
    mfn: real;                    { current mfn (in THES data base) }
    s,action,ft: string;
    i,k,kl,lq,rc: real;


Function FUC(s: string): string;

{-------------------------------------------------------------------------}
{ Converts string s to upper case }
{-------------------------------------------------------------------------}

var us: string;
begin
us:=s; uc(us);
fuc:=us;
end;


Procedure ERRMSG(t: string);

{-------------------------------------------------------------------------}
{ Display error message t and pause                                       }
{-------------------------------------------------------------------------}

var s: string;
begin
clearmsg; writeln(chr(7),t);
write('Press ENTER to continue'); s:=inkey;
end;

Procedure DISPLT;

{-------------------------------------------------------------------------}
{ Display top term box }
{-------------------------------------------------------------------------}

begin
cleardata;
box(1,1,3,32,2); clearbox(2,2,1,30,2);
cursor(2,2); write(fuc(term));
if action='S' then
   begin
box(1,74,3,7,1);
```

```
   cursor(1,76); write('MFN'); cursor(2,75); write(mfn:5);
   end;
savescr(1);
end;


Procedure DISPLAY(t,o: real);

{-------------------------------------------------------------------------}
{ Display term relations starting from tag[t], occurrence o }
{-------------------------------------------------------------------------}

var rc,fn: real;

begin
nl:=0;
if t=1
   then begin
        displt;
        it:=1; io:=1;
        end
   else begin
        clearbox(5,1,15,80,0);
        it:=t; io:=o;
        end;
while (it<=maxt) and (nl<=maxl) do
  begin
  repeat
    fn:=fieldn(tag[it],io);
    if fn=0 then begin it:=it+1; io:=1; end;
  until (fn>0) or (it>maxt);
  if fn>0 then
     begin
     nl:=nl+1; dt[nl]:=it; doc[nl]:=io; io:=io+1;
     cursor(nl+4,1);
     write('_ ',substr(rel,(it-1)*3+1,3),' ',field(fn));
     end;
  end;
end;


Function DECIDE(l: real): string;

{-------------------------------------------------------------------------}
{ Read action code (<CR>,B,F and P are handled here; othere codes returned}
{-------------------------------------------------------------------------}

var s: string;
    sc: real;
begin
cl:=l;
if nl>0 then
  begin
  clearmsg;
  writeln('Y Next  B[ack]   F[irst]   P[age]   S[elect]  T[erm select]
  Q[uery]');

    write ('?[display query]  A[dd relation]     D[elete]  C[reate term]
```

```
  X[exit]');
  repeat
  if cl<1 then cl:=1;
  if cl>nl then cl:=nl;
  cursor(cl+4,1);
  sc:=kbdkey(s); uc(s);
  if s=chr(13) then s:=' ';
  case s of
  ' ': if cl>=nl then cl:=1 else cl:=cl+1;
  'B': cl:=cl-1;
  'F': begin display(1,1); cl:=1; end;
  'P': begin
       display(dt[nl],doc[nl]);
       cl:=1;
       end;
  end;
  until position('?ACDLMQSTX',s,1)>0;
  end;
decide:=s;
if s='S' then term:=field(fieldn(tag[dt[cl]],doc[cl]));
end;


Function FINDTERM(term: string): real;

{-------------------------------------------------------------------------}
{ Search and display selected term                                         }
{       Return 0 if term exists (action contains a valid action code)     }
{              1 if term does not exist (action is not set)               }
{-------------------------------------------------------------------------}

var rc: real;
    t: string;
begin
t:=fuc(term);
rc:=find(t);
findterm:=rc;
if rc=0 then
   if nxtpost<0
      then findterm:=1
      else begin
           mfn:=posting('MFN');
           rc:=record(mfn);
           findterm:=rc;
           if rc=0 then
              begin
              display(1,1);
              action:=decide(0);
              end;
           end;
end;

Function FLDUC(k: real): string;

{-------------------------------------------------------------------------}
{ Returns k-th field of record converted to upper case                    }
```

```
var f: string;
begin
f:=field(k); uc(f);
flduc:=f;
end;


Function CHKREL(t: string): real;

{-------------------------------------------------------------------------}
{ Check if a relation already exists                                      }
{-------------------------------------------------------------------------}

var i,n: real;
begin
n:=nfields; i:=1;
while (i<=n) and (flduc(i)<>t) do i:=i+1;
if i>n then chkrel:=0
       else chkrel:=i;
end;


Procedure UPDINVF;

{-------------------------------------------------------------------------}
{ Update inverted file (screen is clear because FST is displayed)         }
{-------------------------------------------------------------------------}

begin
cleardata;
updif;
end;


Procedure CREATERM;

{-------------------------------------------------------------------------}
{ Create new thesaurus term                                               }
{-------------------------------------------------------------------------}

var tuc: string;
    rc,np: real;
begin
term:=''; clearmsg;
displt;
clearmsg; write('Enter new term');
rc:=edit(term,30,2,2,30,1,' ');
if term<>'' then
  begin
    tuc:=term; uc(tuc); rc:=find(tuc); np:=-1;
    if rc=0 then np:=nxtpost;
    if (rc=0) and (np>0)
       then errmsg('Term already exists')
       else begin
            mfn:=newrec;

            rc:= Fldadd ( Tag [ 1 ], 1, Term );
```

```
            update; updinvf;
            action:='S';
            end;
   end
   else action:='T';
end;


Procedure ADDREL;

{-------------------------------------------------------------------------}
{ Add new relation to a term                                              }
{-------------------------------------------------------------------------}

var r,rt,rtu: string;
    rc,i,rtag: real;

Function ADDIT: real;
var tt,ir: string;
    n,k: real;
    relmfn: real;

Procedure RELADD;
var rc: real;
begin
n:=nocc(rtag); k:=1;
while (k<=n) and (flduc(fieldn(rtag,k))<rtu) do k:=k+1;
rc:=fldadd(rtag,k+1,rt); update;
end;

begin
if (find(rtu)<>0) and (substr(r,1,2)<>'SN')
   then begin
        addit:=1;
        errmsg('Related term does not exist');
        end
   else
if (chkrel(rtu)<>0) and (substr(r,1,2)<>'SN')
   then begin
        addit:=1;
        errmsg('Relation already exists');
        end
   else
begin
rtag:=tag[(rtag-1)/3+1];
reladd;
ir:=substr(invrel,(rtag-1)*3+1,3);
if ir<>'   ' then
   begin
   k:=nxtpost; relmfn:=posting('MFN');
   rtag:=tag[(position(rel,ir,1)-1)/3+1];
   rt:=field(fieldn(tag[1],1)); rtu:=rt; uc(rtu);
   k:=record(relmfn);
   reladd;
   end;
k:=record(mfn);

 addit:=0;
```

```
end;
end;

begin
box(18,10,3,5,1); box(18,14,3,52,1);
cursor(19,1); write('Relation');
r:=''; rt:='';
repeat
clearbox(19,15,1,50,1);
clearmsg; write('Enter relation code: ');
for i:=2 to maxt do write(substr(rel,(i-1)*3+1,3),' ');
clearbox(19,11,1,3,1); rc:=edit(r,3,19,11,3,1,' '); uc(r);
rtag:=position(rel,r,1);
if rtag=0 then write(chr(7));
until (r='') or (rtag>0);
repeat
i:=0;
if rtag>0 then
   begin
   clearmsg;
   rc:=edit(rt,30,19,16,30,1,' '); rtu:=rt; uc(rtu);
   if rtu<>'' then i:=addit;
   end;
until i=0;
action:='S';
end;


Procedure DELREL;

{-------------------------------------------------------------------------------}
{ Delete a relation                                                             }
{-------------------------------------------------------------------------------}

var rtag,rc,k,relmfn: real;
    rt,rtu,ir: string;
begin
rtag:=fieldn(dt[cl],doc[cl]);
rt:=field(rtag); rtu:=rt; uc(rtu);
rc:=flddel(rtag);
update;
ir:=substr(invrel,(dt[cl]-1)*3+1,3);
if ir<>'   ' then
   begin
   rc:=find(rtu);
   if rc=0 then
      begin
      k:=nxtpost;
      if k>=0 then
         begin
         relmfn:=posting('MFN');
         rtag:=tag[(position(rel,ir,1)-1)/3+1];
         rt:=field(fieldn(tag[1],1));
         rtu:=rt; uc(rtu);
         rc:=record(relmfn);
         if rc=0 then


          begin
```

```
            k:=chkrel(rtu);
            if k>0 then
               begin
               rc:=flddel(k);
               update;
               end;
            end;
         end;
      end;
   end;
k:=record(mfn);
action:='S';
end;


Procedure DELTRM;

{-------------------------------------------------------------------------------}
{ Delete a thesaurus term                                                       }
{-------------------------------------------------------------------------------}

begin
if nfields>1
   then begin
        errmsg('Cannot delete term with relations. Delete all relations
        first.');
        action:='S';
        end
   else begin
        rc:=flddel(1);
        update; updinvf;
        action:='T';
        end;
end;


Procedure SHOWDICT;

{-------------------------------------------------------------------------------}
{ List dictionary                                                               }
{-------------------------------------------------------------------------------}

var i,ii,k,sc: real;
    tp: array[1..16] of real;
    ts: array[1..16] of real;
    pg,ft: string;

begin
ft:=term;
repeat
pg:=''; i:=1; sc:=find(ft);
repeat
tp[i]:=size(pg)+1; ts[i]:=size(ft);
pg:=pg|ft;
ft:=nxtterm; i:=i+1;
until (i=17) or (ft='');

  i:=i-1;
```

```
for k:=1 to i do
    begin cursor(k+4,5); writeln('_ ',substr(pg,tp[k],ts[k])); end;
k:=1;
repeat
ii:=k;
chattr(1,k+4,5,30); term:=substr(pg,tp[k],ts[k]);
sc:=kbdkey(action); uc(action);
if action=chr(13) then k:=k+1 else
if action='B' then if k>1 then k:=k-1;
chattr(0,ii+4,5,30);
until (position('CPSTX', action,1)>0) or (k>i);
page(1);
until (position('CSTX',action,1)>0) or (term='');
end;



{------------------------- Body of program THES -------------------------}

begin

maxt:=7;                                    { Number of defined relations }
rel:=    '   SN USEUF BT NT RT ';           { Name of relations }
invrel:='      UF USENT BT RT ';            { Name of inverse relation }
for i:=1 to maxt do tag[i]:=i;              { Tag of relation }

maxl:=15; q:='';
dbname:=dbn; { save currently selected data base }
if dbname<>'THES' then open('THES');
clear;
if maxmfn=1 then action:='C' else action:='T';

repeat

case action of

'T': { Term selection }

     begin
     clearmsg;
     write('Select term');
     term:=''; displt;
     cursor(2,2); readln(term);
     if term='' then action:='X' else
     if (substr(term,size(term),1)='$') or (findterm(term)<>0)
        then action:='L';
     end;

'L': { List of thesaurus terms }

     begin
     uc(term);
     rc:=find(term);
     page(1);
     clearmsg;
     writeln('Y [Next]        B[previous]       P[age]        S[elect]');
      write  ('C[reate term]    T[erm select]     X[exit]');
```

```
      savescr(1);
      showdict;
      if term='' then action:='L';
      end;

 'S': { Display term relations }

      begin
      rc:=findterm(term);
      if rc<>0 then action:='L';
      end;

 'A': { Add a relation }

      addrel;

 'C': { Create a new term }

      createrm;

 'D': { Delete a term or a relation }

      if cl=1 then deltrm else delrel;

 'Q': { Select term for searching }

      begin
      s:=field(fieldn(tag[dt[cl]],doc[cl]));
      if size(s)+size(q)+3>255
         then begin
              write('§');
              action:='?';
              end
          else begin
              if q<>'' then q:=q|' + ';
              q:=q|s;
              action:=decide(cl+1);
              end;
        end;

   '?': { Display current query }

        begin
        savescr(2);
        box(16,8,6,66,2); clearbox(17,9,4,64,1);
        cursor(17,9); lq:=size(q);
        if lq=0 then write('§No search terms currently selected') else
           begin
           k:=1; kl:=17;
           repeat
           if lq>64 then i:=64 else i:=lq;
           writeln(substr(q,k,i));
           k:=k+i; lq:=lq-i;
           kl:=kl+1; cursor(kl,9);
           until lq=0;
           end;

        clearmsg; write('Press any key to continue');
```

```
      s:=inkey;
      page(2);
      action:=decide(cl+1);
      end;

end;
until action='X';

if dbname<>'THES' then
   begin
   open(dbname);
   if size(q)>0 then
      begin
      clear;
      clearmsg; write('Edit search expresssion or press Enter');
      rc:=edit(q,254,2,1,254,0,' ');
      if size(q)>0 then rc:=search(q);
      end;
   end;

end.
```