

# Natural Fundamentals

***CONSIST***

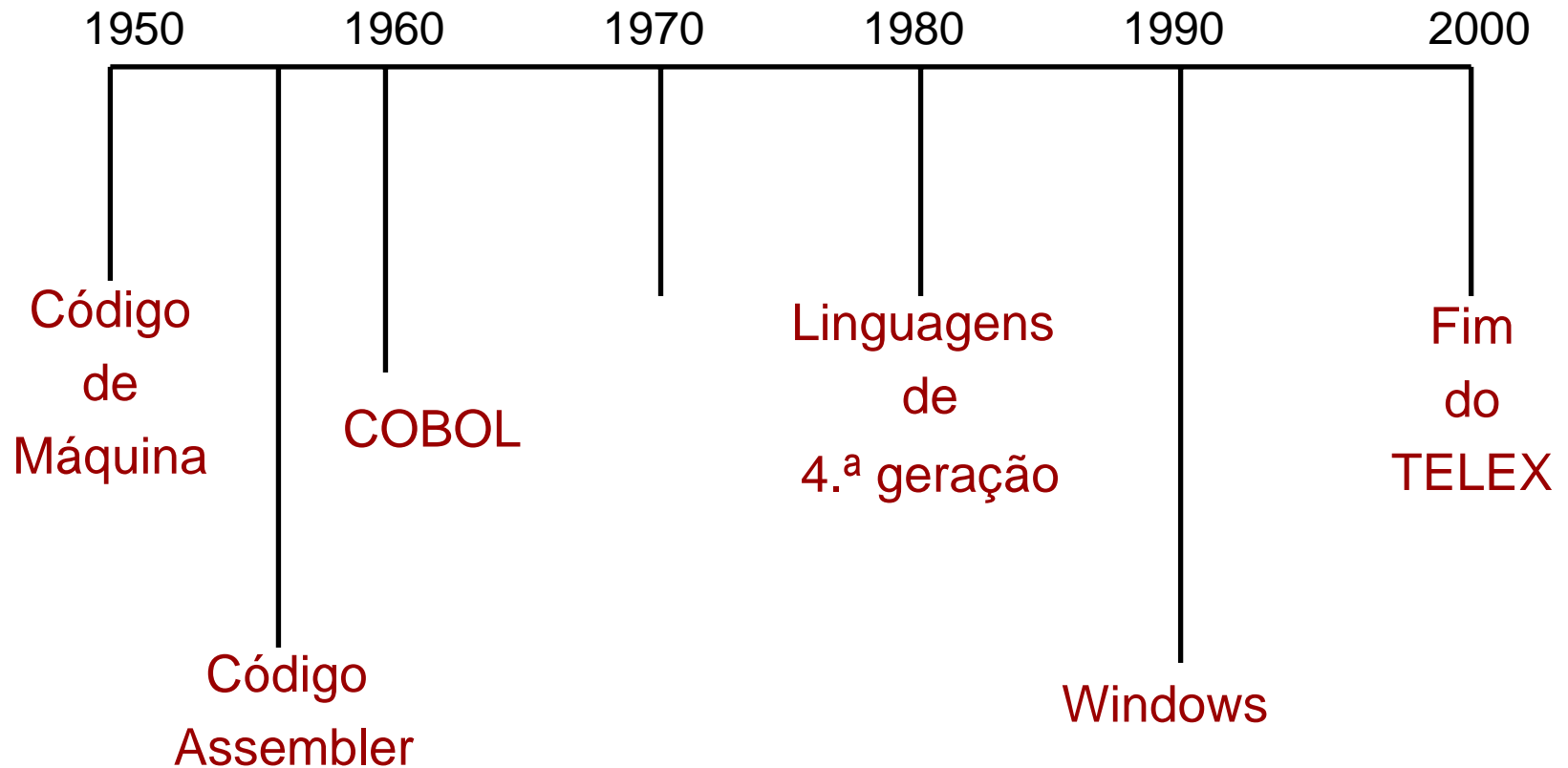
*Business Information Technology*

<b>Módulo I - Introdução ao Natural .....</b>	<b>3</b>
<b>Módulo II - Área de Dados .....</b>	<b>28</b>
<b>Módulo III - Funções Programáticas .....</b>	<b>78</b>
<b>Módulo IV – Mapas .....</b>	<b>193</b>
<b>Módulo V - Acesso a Base de Dados .....</b>	<b>267</b>
<b>Módulo VI – Relatórios .....</b>	<b>333</b>
<b>Módulo VII - Mapas de Help e Helprotinas .....</b>	<b>386</b>
<b>Módulo VIII - Atualização a Base de Dados .....</b>	<b>399</b>
<b>Módulo IX - Processamento Batch .....</b>	<b>413</b>
<b>Apêndice A - Editor do Mainframe .....</b>	<b>427</b>
<b>Exercícios .....</b>	<b>472</b>

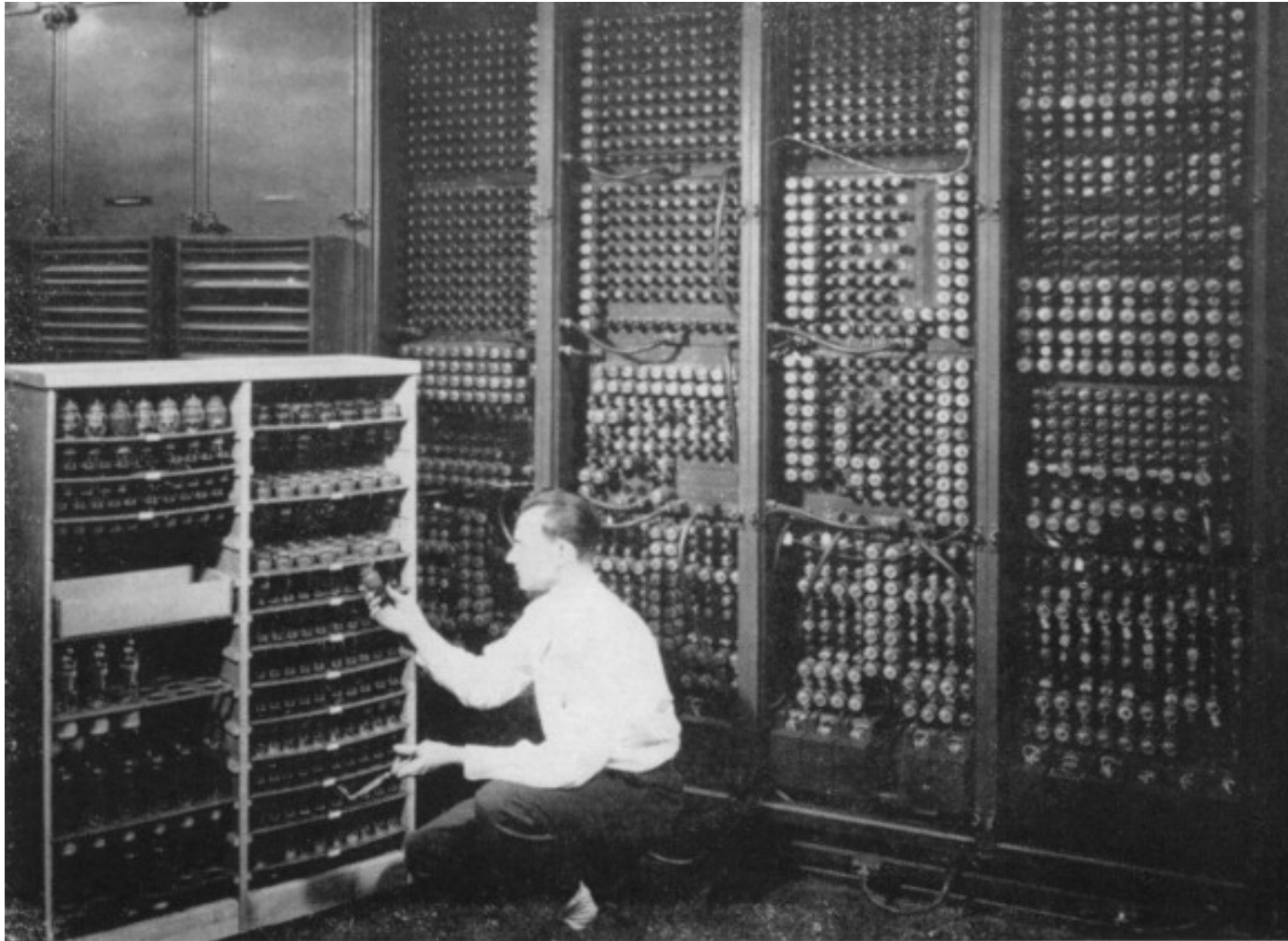
## Unidade A - Introdução à Linguagem de 4.<sup>a</sup> geração e ao Natural

- Introdução à Linguagem de 4.<sup>a</sup> Geração
- O que torna o Natural tão atraente
- Bibliotecas
- Ambiente operacional Natural
- User Work Areas

# Introdução à tecnologia de 4.<sup>a</sup> geração



# Introdução à tecnologia de 4.ª geração



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

As empresas dependentes das linguagens de alto nível enfrentavam alguns problemas, tais como:

- O desenvolvimento de uma aplicação levava muito tempo para ser concluída, as vezes, anos;
- As modificações e manutenções nos aplicativos eram demoradas e muito caras;
- O processo de depuração de erros era extremamente tedioso.

As linguagens de 4.<sup>a</sup> geração foram desenvolvidas para simplificar a computação.

Em vez dos códigos de máquina, dos códigos assembler e das linguagens de alto nível do passado, foi adotada para as 4GL comandos mais textuais em inglês, com poucas instruções, de fácil manutenção e capaz de acessar o banco de dados diretamente.

O Natural é um ambiente de desenvolvimento para produção orientada de tecnologia de 4.<sup>a</sup> geração e um poderoso ambiente portátil de execução.

A SAGA oferece outros produtos que, junto com Natural funcionam como um completo sistema de informação de software. Entre eles:

- ADABAS C;
- ADABAS D (Relacional);
- Natural Engineering Workbench;
- Natural for Windows;



# Linguagem de 4.<sup>a</sup> geração e Natural

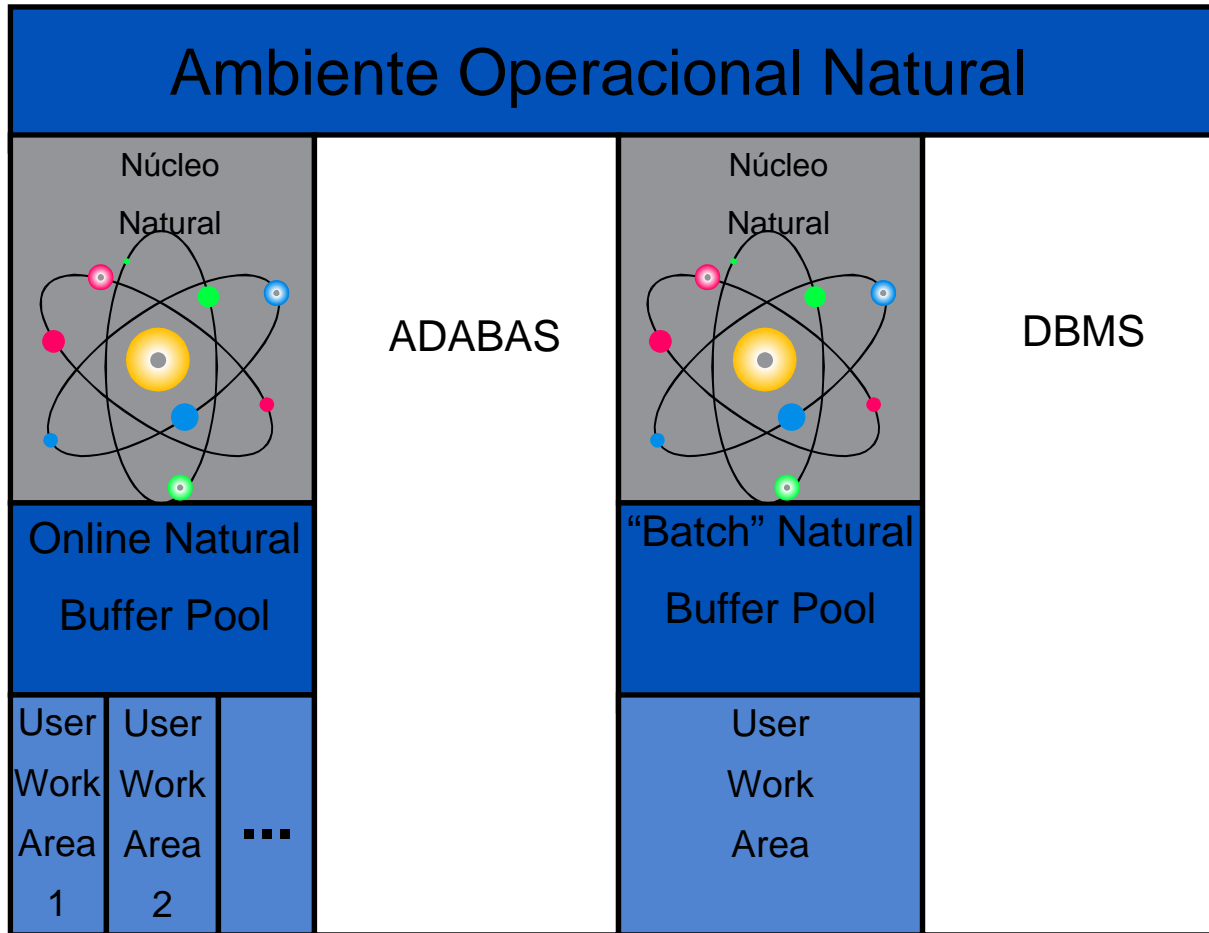
- Construct (Gerador de Aplicações);
- Predict (Documetação – Dicionário de Dados);
- Entire Connection (Emulação e Transferência de dados);
- Super Natural (Data Query);
- EntireX (integrador de aplicações);
- Natural Web (internet / intranet).

Onde o Natural armazena as aplicações criadas pelo usuário?

- Após a criação ou modificação de um objeto, você deve armazená-lo numa biblioteca, caso contrário, você perderá todo seu trabalho ao finalizar a sessão.
- Ao armazenar um objeto, ambos, o código fonte e o código compilado ficam armazenados na mesma biblioteca.
- Cada vez que você acessa o Natural você é colocado numa biblioteca “*default*”. Essa biblioteca depende de como o seu sistema foi parametrizado.

Onde o Natural armazena as bibliotecas?

- As bibliotecas do Natural são armazenadas numa estrutura do banco de dados ou diretórios (dependendo do ambiente operacional). FUSER ou FNAT.
- Você só pode trabalhar em uma biblioteca por vez. Para acessar outra biblioteca basta entrar com o comando LOGON na linha de comando direto e, em seguida, o nome da biblioteca que você deseja acessar.
- Como convenção, o nome da biblioteca deve ter até oito caracteres, sendo o primeiro, alfanumérico.



O Natural funciona tanto em modo online como em modo “batch”, mas para executar um programa em “batch” é preciso submeter um “job” que chame o Natural.

Há cinco componentes principais usados pelo ambiente operacional Natural:

- Núcleo Natural;
- Natural Buffer Pool
- As áreas de trabalho do usuário (user work areas)
- Arquivos para roll-ins/roll-outs das threads (mainframe) e
- Systems Files

## User Work Areas

Cada vez que você aciona o Natural, uma área é alocada para você. Trata-se de uma área de armazenamento temporária composta de vários buffers.

Cada buffer é alocado para um propósito específico. Há um, por exemplo, que armazena o código fonte do seu objeto Natural. Há um outro que, durante a execução de um programa, retêm os valores dos dados para os campos que estão sendo utilizados.

Os nomes dos buffers em sua área de trabalho irão variar em função da plataforma utilizada (MVS, Open VMS, Windows ou Unix).

## Unidade B - O que envolve a construção de aplicações Natural

- Tipos de objetos Natural e Editores
- Modo Estruturado vs. Modo Report
- Modularização

Há três grupos principais de objetos Natural:

- **Objetos Programáticos:** Programas (P), Subprogramas (N), Subrotinas (S), Helprotinas (H) e Copycode (C);
- **Área de Dados:** Global (G), Parameter (A) e Local (L);
- **Mapas:** Mapa (M) e Mapa de Help (H).

Cada um desses grupos têm o seu próprio editor, que é aberto quando você escolhe o tipo de objeto com o qual você irá trabalhar.



- **Modo Report**

É útil para a criação de programas simples e pequenos, que não envolvem dados complexos e/ou lógica de programação.

- **Modo Estruturado**

É usado em aplicações complexas, fornecendo uma estrutura clara e bem-definida do programa.

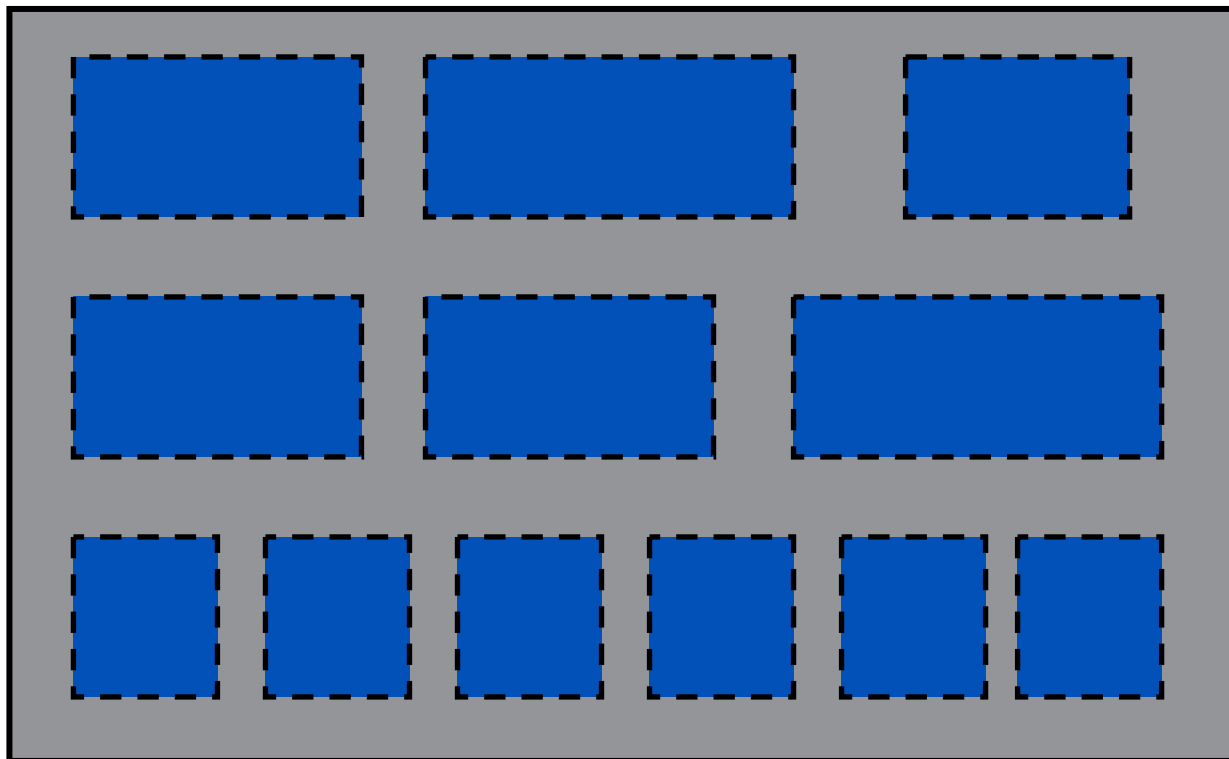
- Esses programas são mais fáceis de serem lidos, facilitando também sua manutenção;
- Todos os campos usados no programa são definidos numa única localização.

O principal motivo para o uso de módulos nas aplicações é devido à facilidade da manutenção. Os vários sistemas de aplicações com que você trabalha possui diferentes exigências.

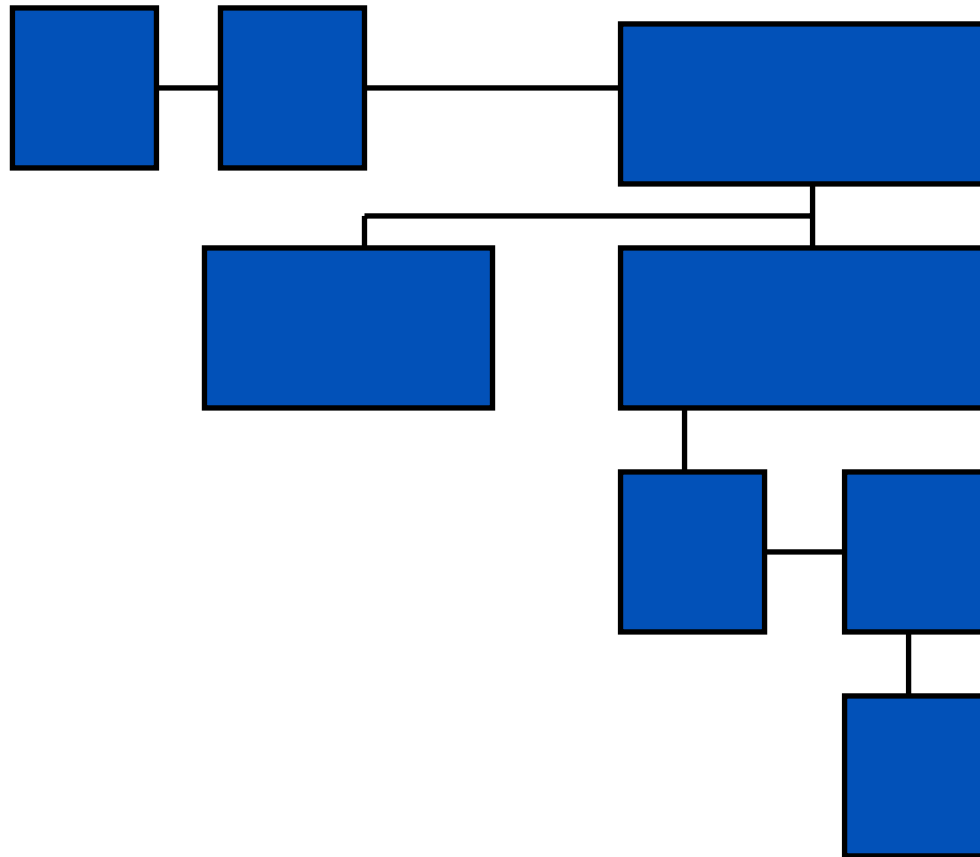
Em vez de tentar programar todas essas funções em um único e grande programa, é mais fácil agrupar essas funções em pequenos módulos. Isto exige um planejamento antecipado.

## Modelo não-modular

Um objeto



## Modelo modular



## Unidade C - Visão Geral do Natural

- Instruções e Comandos do Natural
- Visão Geral

O Natural responde a dois tipos de declarações primárias: as instruções e os comandos. As instruções são usadas para codificar o seu programa, e os comandos para gerenciar o ambiente.

- **Instruções**

Dividem-se em cinco grupos funcionais: Definição de dados, acesso aos dados, manipulação de dados, modificação de dados e exibição de dados.

- **Comandos**

Executam funções da sessão e podem ser emitidos a partir da linha de comando direto ou dos prompts. Há dois grupos de comandos: Comandos do sistema (ex.: SAVE) e Comandos de terminal (ex.: %T).

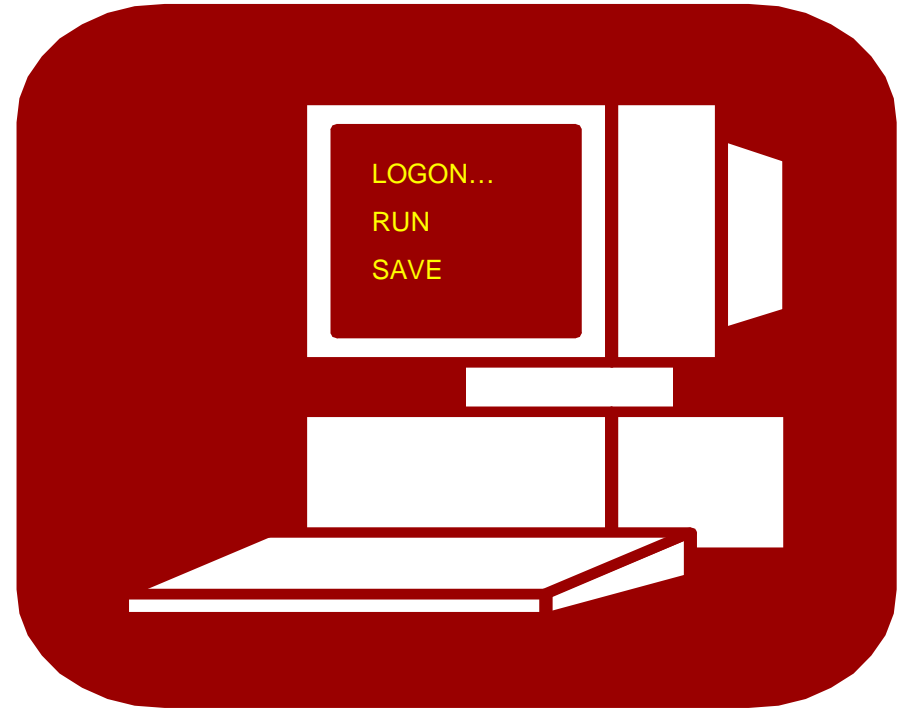
# Instruções e Comandos Natural

## Instruções



Define programas Natural

## Comandos



Gerencia o ambiente Natural  
Executa funções de sessão

## Unidade D - O Help do Natural

- Help online
- Documentação



O Help online do Natural fornece informações sobre:

- instruções;
- variáveis de sistema;
- funções do sistema;
- comandos;
- editores;
- utilitários;
- parâmetros da sessão;
- comandos de terminal;
- mensagens do sistemas e mensagens definidas pelo usuário.

## Help Online

Para acessar o help basta digitar a palavra 'help' na linha de comando ou '?', ou ainda, pressionar a tecla chave para essa função (geralmente PF1). No caso do Natural Windows, para acessar o help basta clicar na barra de menus o item Help.

Se você souber exatamente que tipo de auxílio solicitar, poderá dirigir-se diretamente para a tela de help específica seguida do parâmetro que foi solicitado. Por exemplo:

<b>Este comando...</b>	<b>Faz isto...</b>
<b>HELP CHECK</b>	Explica o comando Natural 'CHECK'
<b>HELP USER 1234</b>	Exibe a mensagem de usuário completa do erro n.º 1234
<b>HELP 0082</b>	Exibe a mensagem Natural completa n.º 0082
<b>HELP LAST</b>	Exibe o texto do última comando

Há vários manuais de documentação sobre o Natural e eles podem variar de acordo com a plataforma que você está utilizando. Os seguintes manuais podem ser interessantes:

- Statements Manual;
- Reference Manual;
- User's Guide e
- Programmer's Guide.

Além disso, a SAGA disponibiliza via CD-ROM uma documentação eletrônica contendo todos os manuais citados acima e suas respectivas atualizações.

### Unidade A - Visão Geral dos Tipos de Dados Disponíveis

- Área de Dados
- Variáveis do Sistema e Definidas pelo Usuário
- Dados do Banco
- Escolhendo a Área de Dados

# Data Areas

Área de Dados	Função
<i>Global Data Area</i> (GDA)	Define os dados que podem ser compartilhados por múltiplos objetos na aplicação.
<i>Parameter Data Area</i> (PDA)	Referencia os dados que estão definidos tanto na GDA quanto na área de dados local (LDA).
<i>Local Data Area</i> (LDA)	Define os dados que podem ser usados somente pelo objeto.
<i>Application-Independent Variables (AIVs)</i>	São variáveis independentes de qualquer estrutura de dados específica. Alguns a utilizam para permitir o acesso de subprogramas à área de dados global indiretamente.
<i>Context</i>	Usadas com o Natural RPC para variáveis disponíveis a múltiplos subprogramas remotos com uma conversação.

### Área de Dados Interna vs. Externa

As áreas de dados podem ser criadas internamente nas linhas de código do seu programa, ou externamente, e acessadas quando forem necessárias.

Você pode ter múltiplas *PDA*s e *LDA*s, mas somente uma *GDA* por aplicação.

# Data Areas

```
0010 *****
0020 * EXEMPLO DE DEFINICAO DA AREA DE DADOS GLOBAL, PARAMETER, *
0030 * LOCAL EXTERNA E LOCAL INTERNA *
0040 * NOME DO PROGRMA : PEXEMPLO *
0050 *****
0060 DEFINE DATA
0070 GLOBAL USING MAINGLOB
0080 PARAMETER
0090 1 #PARM1 (A10)
0100 1 #PARM2 (A20)
0110 LOCAL USING LDA1
0120 LOCAL
0130 1 PEOPLE VIEW OF EMPLOYEES
0140 2 PERSONNEL-ID
0150 2 NAME
0160 1 #DATA (A6)
0170 END-DEFINE
```

### **Área de Dados Global**

O objetivo da área de dados global é compartilhar dados entre os objetos Natural. Os objetos que podem compartilhar dados globais são:

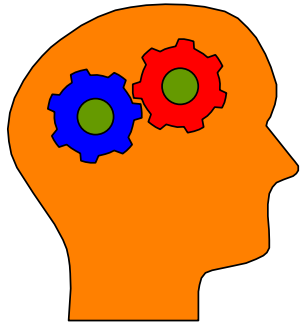
- Programas;
- Subrotinas e
- *Helprotinas*.



### **Vantagens no uso da *Global Data Area***

A vantagem de usar a GDA é que você não precisa definir a passagem de parâmetros do objeto chamador para o chamado.

Além disso, ela fornece dados para vários programas com muito menos espaço que as áreas de dados individuais dos programas (usa-se a área ESIZE).



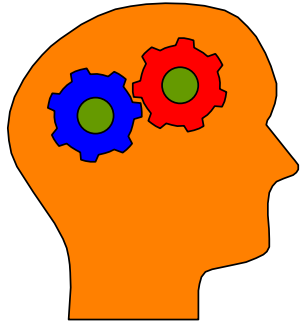
### Lembre-se!

- As *GDA*s só podem ser criadas como estruturas externas;
- A *GDA* deve ser o objeto mais antigo da aplicação. Caso você precise criar uma *GDA* numa aplicação existente ou modificá-la, todos os demais objetos deverão ser recompilados;
- Deve haver apenas uma *GDA* por aplicação;
- Múltiplas *GDA*s podem ser usadas dentro de uma aplicação somente quando usada conjuntamente com subprogramas.
- Programas, subrotinas e rotinas de *help* podem acessar a *GDA*.

### ***Parameter Data Area***

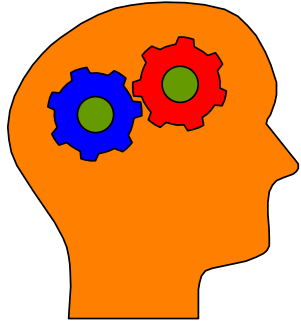
Uma área de *parameter* é usada para definir elementos de dados que um subprograma, subrotina ou uma *help rotina* usa para receber e retornar dados para um módulo chamador. Ela pode ser definida internamente no programa ou ser um módulo externo. Os objetos que podem usar uma PDA são:

- Subprogramas;
- Helproteínas e
- Subrotinas Externas.



### Lembre-se!

- A *PDA* deve definir todos os campos que estão sendo passados para ela;
- Os campos devem ser definidos na seqüência exata, com os mesmos formatos e tamanhos que foram definidos no objeto chamador (a não ser que se use a cláusula *by value*). Os nomes dos campos podem ser diferentes no objeto chamador. Isto permite ao objeto ser usado por diferentes aplicações;
- *User views* não podem ser definidas na *PDA*.



### Lembre-se!

- Os parâmetros são passados para um subprograma/*routine* por referência (conforme seus endereços). A opção *BY VALUE*, significa que os valores atuais dos parâmetros são passados e não apenas o endereço, assim, formato e tamanho não precisam coincidir;

Exemplo: Define data parameter

```
#NAME (A20) BY VALUE
```

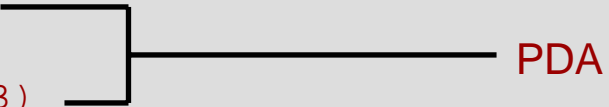
- A cláusula *BY VALUE RESULT* faz com que os parâmetros sejam passados por valor em ambas as direções (no envio e no recebimento).

# Data Areas

```
0010 *****
0020 * EXEMPLO DO USO DA PARAMETER DATA AREA *
0030 * NOME DO PROGRMA : PDAEXEMP *
0040 *****
0050 DEFINE DATA
0060 LOCAL
0070 1 EMP VIEW OF EMPLOYEES
0080 2 PERSONNEL-ID
0090 2 NAME
0100 2 INCOME
0110 3 SALARY (1)
0120 2 DEPT
0130 1 #TAX (P6.3)
0140 END-DEFINE
0150 *
0160 READ EMP BY NAME
0170 CALLNAT 'SUBPEXEM' SALARY (1) #TAX /* Passando parâmetros para a PDA
0180 DISPLAY NAME PERSONNEL-ID SALARY (1) #TAX DEPT
0190 END-READ
0200 END
```

# Data Areas

```
0010 *****
0020 *  EXEMPLO DE UM SUBPROGRAMA QUE RECEBE PARAMETROS      *
0030 *  NOME DO SUBPROGRAMA: SUBPEXEM                        *
0040 *****
0050 DEFINE DATA
0060 PARAMETER
0070 1 #SALARY (P9)
0080 1 #TAX    (P6.3)
0090 END-DEFINE
0100 *
0110 COMPUTE #TAX = #SALARY * .045
0120 END
```



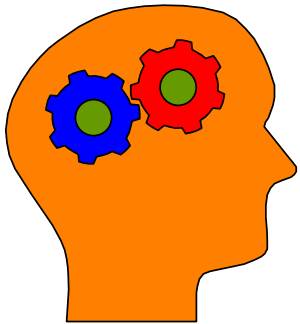
PDA

### Local Data Area

A área de dados local define os campos que serão utilizados por um objeto específico. Elas podem ser definidas internamente ou como um módulo externo. Diferentes objetos podem usar as definições de uma *LDA*, mas não podem compartilhar os dados em tempo de execução.

Em tempo de execução os valores dos dados são retidos num *buffer* da sua área de trabalho.





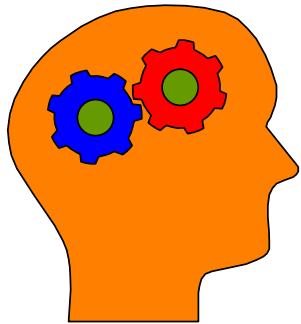
### Lembre-se!

- Em tempo de execução, os dados locais são usados somente pelo objeto que define a *LDA*. Dois objetos programáticos podem compartilhar as definições de uma *LDA*, mas não podem compartilhar os dados;
- Defina somente aqueles campos que você vai usar no seu programa. Se você define mais campos do que realmente precisa, acaba gastando o espaço de seu *buffer*.

Se você precisa definir campos diferentes dos encontrados na DDM, defina-os como *user-defined variables*. Há três importantes razões para o uso desse tipo de campo:

1. Exibir informações geradas pelo usuário;
2. Para armazenagem intermediária de dados;
3. Para criar contadores.

Toda variável definida pelo usuário deve ter nome e formato.



### Lembre-se!

- Como os demais campos, você deve criar as *user-defined variables* dentro da estrutura da instrução *DEFINE DATA*, ou então numa área de dados externa;
- Defina primeiro o formato de um campo e depois o tamanho;
- Costuma-se iniciar as variáveis definidas pelo usuário com o caracter (#);
- Os nomes dos campos podem ter entre 1-32 caracteres.

# Variáveis definidas pelo usuários

Formato	Significado	Tamanhos Permitidos
<b>A</b>	Alfanumérico	1-253
<b>N</b>	Numérico (descompactado)	1-29 ou 1-27 (plataforma específica)
<b>P</b>	Numérico (compactado)	1-29 ou 1-27 (plataforma específica)
<b>I</b>	Inteiro	1, 2 ou 4
<b>F</b>	Ponto Flutuante	4 ou 8
<b>B</b>	Binário	1-126
<b>C</b>	Atributo de Controle	(2)*
<b>D</b>	Data	(6)* (Armazenada como compactada de 4)
<b>T</b>	Hora	(12)* (Armazenada como compactada de 7)
<b>L</b>	Lógico	(1)*

\* Os números entre parênteses indicam valores que você não pode alterar, portanto, para essas variáveis, não é preciso definir o tamanho.

## Variáveis do Sistema

As variáveis do sistema contém as informações atualizadas do sistema, tais como: o nome da biblioteca corrente, do usuário e do terminal, etc.

A variável de sistema é facilmente reconhecida pois o primeiro caracter é um asterisco(\*).

### Data e Hora

Essas variáveis contém a data e a hora em vários formatos. Você pode exibí-las através das declarações *DISPLAY*, *WRITE* e *MOVE*. Veja o exemplo:

```
WRITE *DATE *TIME
```

```
01/01/96 12:00:00.9
```

## Variáveis do Sistema

Nome	Conteúdo
*NUMBER	Número de registros
*COUNTER	Número de vezes que o loop foi executado
*PAGE-NUMBER	Valor atualizado do número da página
*LIBRARY-ID	Nome da biblioteca Natural
*PROGRAM	Nome do objeto Natural
*USER	Identificação do usuário
*LANGUAGE	Idioma corrente (1 – Inglês) (11 ou 38 – Português)
*CURSOR	Posição do cursor
*ERROR-NR	Número de erro Natural

Nome	Formato/Tamanho	Formato dos Conteúdos
*DATN	N8	YYYYMMDD
*DATE	A8	DD/MM/YY
*DATI	A8	YY-MM-DD
*DATJ	A5	YYDDD
*DATX	D	Formato interno
*TIME	A10	HH:MM:SS.T
*TIMN	N7	HHMMSST
*TIMX	T	Formato interno

Para cada variável de sistema DATA corresponde outra que inclui o ano com 4 dígitos, exemplo: \*DAT4E, \*DAT4I, \*DAT4J.

### Arquivo Físico vs. Arquivo Lógico

Os dados são armazenados no seu banco em arquivos físicos (tabelas). Para acessar esses arquivos com uma das velhas linguagens de programação, você deve escrever rotinas de acesso, checar os *return codes* e extrair os dados antes de processá-los.

Com o Natural esse processo tedioso foi eliminado. Ele pode usar arquivos lógicos e criar esse acesso para você.



### **Data Definition Module (DDM)**

Trata-se de uma visão lógica do arquivo físico. A DDM define os campos do banco que vão ser usados no programa. Os campos de uma DDM podem compreender todos os campos definidos no arquivo do banco, ou apenas um subconjunto desses campos.

Dependendo do seu DBMS, seu sistema pode ter mais que uma DDM por arquivo.

O uso de DDMs também promove independência dos dados nos programas.

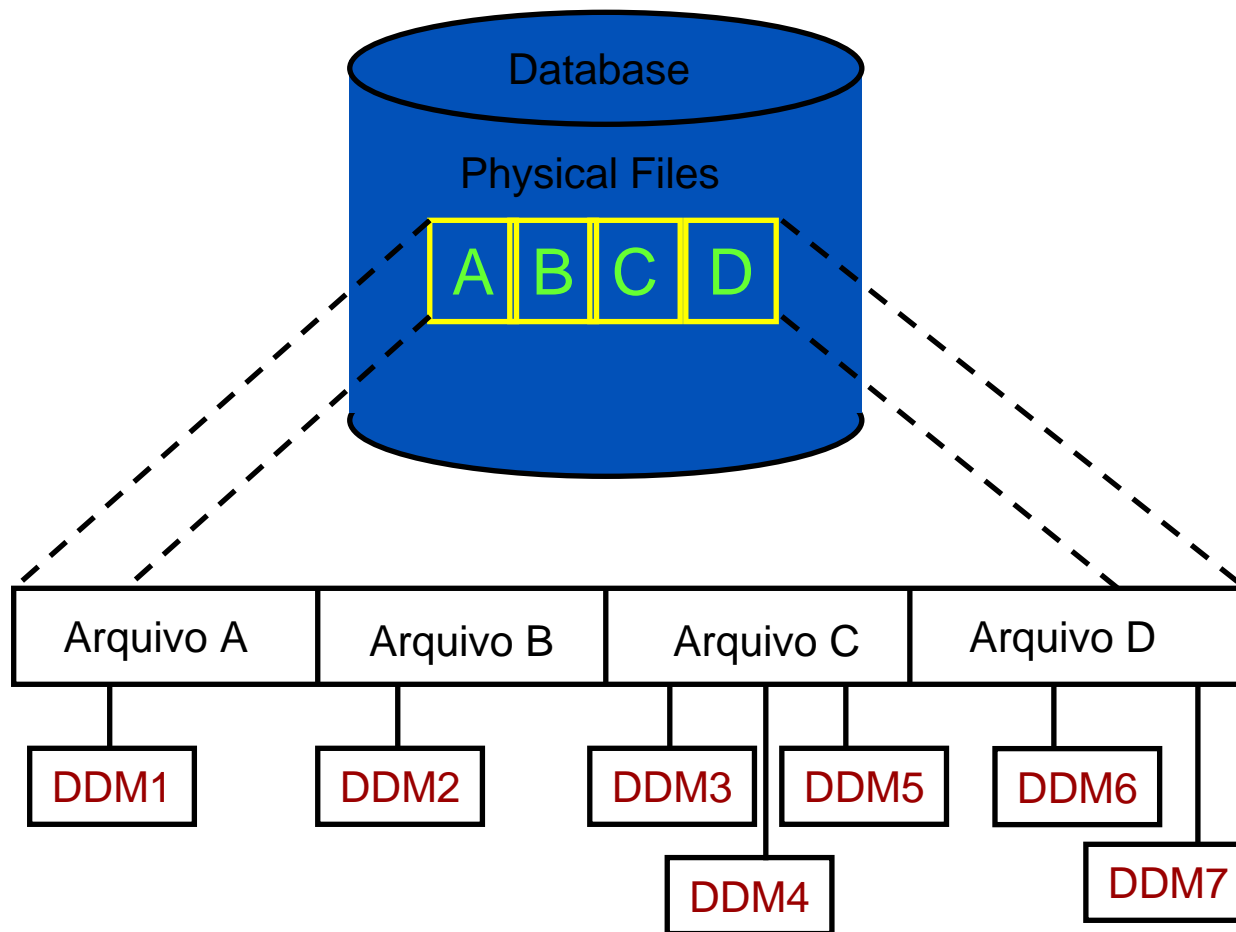
### Data Definition Module (DDM)

As seguintes informações estão armazenadas na DDM para cada campo:

- Tipo de campo (grupo, elementar, MU e/ou PE);
- Nível do campo na estrutura;
- Nome do campo no banco;
- Nome do campo no Natural;
- Formato;
- Tamanho;

- Supressão de campos no ADABAS;
- Situação do campo Descritor/Chave;
- Observações e/ou comentários.

# Arquivos do Banco de Dados e DDMs



### O que são *Programmatic user Views*?

Trata-se de um subconjunto da DDM, que irá definir os campos que seu objeto Natural irá usar.

### Instrução Define Data

Você define sua *programmatic user view* na área de dados dentro da instrução DEFINE DATA. Para verificar a sintaxe apropriada para essa instrução, veja o exemplo:

```
DEFINE DATA LOCAL
```

```
    1 cars view of vehicles /* cars é a user view e vehicles é a DDM
```

```
END-DEFINE
```

## Escolhendo a Área de Dados

Para escolher corretamente a Área de Dados devemos levar em consideração a função e o propósito e cada uma delas. Observe a tabela abaixo:

Interna ou Externa ?

Se a área de dados contém...	Então use...
Muitos campos e/ou precisam estar centralizados.	Externa
Definições de campos a serem compartilhados.	Externa
Poucos campos e/ou é usado somente por um objeto Natural.	Interna

GDA, PDA ou LDA?

Se os dados...	O tipo mais indicado é...
São usados por vários objetos Natural.	GDA ou PDA
São usados por apenas um objeto Natural.	LDA
São passados por um objeto chamador.	PDA

## Unidade B - Definição de Dados

- Área de Dados Interna
- Área de Dados Externa
- Movendo Definições de Dados
- DBMS
- Modelos de Banco de Dados

## Área de Dados Interna

Você usa o editor de programa para criar a área de dados interna. Quando você codifica o seu programa, é necessário definir dentro da declaração *DEFINE DATA* todos os campos que pretende utilizar.

```
DEFINE DATA
GLOBAL USING GDA2
LOCAL USING LDA3
LOCAL
1 CARS VIEW OF VEHICLES
  2 MAKE
  2 MODEL
  2 YEAR
  2 COLOR
1 #NAME (A20)
END-DEFINE
```



Área de Dados Interna



A edição de máscara permite que você mude o formato de exibição de um campo sem mudar o formato e tamanho do próprio campo. As principais razões para usar uma máscara de edição são:

- Mudar a aparência da informação que vai ser exibida;
- Remover dados desnecessários de um campo;
- Economizar espaço de armazenamento.

# Máscaras de Edição

Máscaras de Edição	Exemplo A		Exemplo B	
	Valor	Saída	Valor	Saída
EM=999.99- EM=9(3).9(2)-	36732	367.32	-530	005.30-
EM=ZZZZZ9 EM=Z(5)9(1)	0	0	579	579
EM=X^XXXXX EM=X(1)X(5)	BLUE	B LUE	A19379	A 19379
EM=XXX..XX EM=X(3)..X(2)	BLUE	BLU..E	AAB01	AAB..01

# Máscaras de Edição

Máscaras de Edição	Exemplo A		Exemplo B	
	Valor	Saída	Valor	Saída
EM=HHH EM=H(3)	100	F1F0F0	ABC	C1C2C3
EM=MM/DD/YY	Use *DATU	01/05/96	Use *DATU	02/30/96
EM=MM/DD/YYYY	Use *DAT4U	01/26/1999	Use *DAT4U	02/30/1996
EM=HH.II.SS.T	Use *TIME	08.54.12.7	Use *TIME	14.32.54.3
EM=OFF/ON	TRUE	ON	FALSE	OFF

## Máscaras para campos do tipo DATA

<b>Código</b>	<b>Descrição</b>
<b>DD, ZD</b>	Dia
<b>MM, ZM</b>	Mês
<b>YYYY, YY, ZY</b>	Número da semana
<b>WW, ZW</b>	Ano
<b>JJJ, ZZJ</b>	Dia juliano
<b>N...N, N(n)</b>	Nome do dia
<b>L...L, L(n)</b>	Nome do mês
<b>R</b>	Ano em numerais romanos

## Máscaras para campos do tipo HORA

<b>Código</b>	<b>Descrição</b>
<b>T</b>	Décimos de segundo
<b>SS, ZS</b>	Segundos
<b>II, ZI</b>	Minutos
<b>HH, ZH</b>	Hora
<b>AP</b>	AM/PM

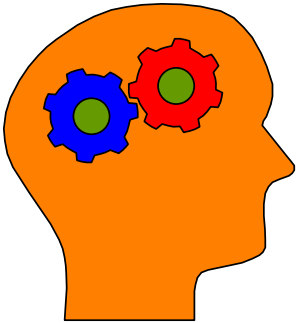
## Exemplo de máscara para o formato **DATA** :

```
DEFINE DATA LOCAL
1 #BIRTH (D) INIT <*DATX>
END-DEFINE
WRITE #BIRTH (EM=NNNNNNNNN', 'LLLLLLLLLLL' `DD`th')
END
```

## Saída:

Monday, September 28th

Valores de inicialização (*INIT*) podem ser atribuídos aos campos da área de dados interna. Ao defini-los você estará sobrescrevendo o valor nulo que é o *default* de um campo. Para campos alfanuméricos, o valor nulo *default* é ' '(branco) , para campos numéricos é (0), para variáveis lógicas', FALSE e para as variáveis de controle, (AD=D).



### Lembre-se!

- Os valores de inicialização para campos alfanuméricos devem estar entre apóstrofes;
- As variáveis de sistema podem ser usadas para inicializar variáveis definidas pelo usuário;
- Valores de inicialização para ocorrências de *array* devem ser separados por vírgula, ou os índices devem ser especificados para definir os valores iniciais de cada ocorrência do *array*.

# Exemplo

```
0010 *****
0020 * ILUSTRA O USO DO EDIT MASK E DOS INIT
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 #COLOR          (A10)   INIT  <'TURQUOISE'>
0070                               (EM=X' 'X' 'X' 'X^X^X^X^X)
0080 1 #SSN            (N9)     (EM=999-99-9999)
0090 1 #MONTH          (A3/12) INIT  <'JAN', 'FEV', 'MAR', 'ABR', 'MAI', 'JUN',
0100                               'JUL', 'AGO', 'SET', 'OUT', 'NOV', 'DEZ'>
0110 1 #COUNTRY-MENU  (9)
0120   2 #SELECT       (N1)     INIT  <1,2,3,4,5,6,7,8,9>
0130   2 #COUNTRY-TEXT (A20)
0140                               INIT  (1) <'AUSTRALIA'>
0150                               (2) <'CANADA'>
0160                               (3) <'INGLATERRA'>
0170                               (4) <'FRANCA'>
0180                               (5) <'ALEMANHA'>
0190                               (6) <'JAPAO'>
0200                               (7) <'ESPANHA'>
0210                               (8) <'ESTADOS UNIDOS'>
0220                               (9) <'IOGUSLAVIA'>
```



# Exemplo

## Continuação

```
0230 1 #DATE          (D)      INIT <*DATX>
0240 1 #TIME          (T)      INIT <*TIMX>
0250 1 #REPEAT        (L)      INIT <TRUE>
0260 END-DEFINE
0270 *
0280 #SSN := 123456789
0290 *
0300 WRITE NOTITLE
0310 / 10T '=' #COLOR (CD=YE)
0320 // 10T '=' #SSN (CD=GR)
0330 // 10T '=' #DATE #TIME (CD=PI)
0340 // 10T '=' #MONTH(1:12)(CD=RE)/
0350 *
0360 DISPLAY NOHDR #COUNTRY-MENU (*)
0370 END
```

# Exemplo

## Saída:

```
#COLOR: T U R Q U O I S E
```

```
#SSN: 123-45-6789
```

```
#DATE: 02-10-02 18:00:50
```

```
#MONTH: JAN FEV MAR ABR MAI JUN JUL AGO SET OUT NOV DEZ
```

- 1 AUSTRALIA
- 2 CANADA
- 3 INGLATERRA
- 4 FRANCA
- 5 ALEMANHA
- 6 JAPAO
- 7 ESPANHA
- 8 ESTADOS UNIDOS
- 9 IOGUSLAVIA

O Natural permite a redefinição de um grupo ou de um único campo da área de dados interna. Uma das razões para redefinir campos é a necessidade de usar trechos de campos, tais como o ano, o mês e/ou o dia de um campo *DATA-ACQ*, por exemplo.

A cláusula para redefinição de campos é *REDEFINE*. Ela irá definir os trechos que compõem o campo a ser redefinido. Esses trechos também podem ser campos definidos pelo usuário ou um *FILLER* (bytes que não têm nenhum significado específico para o objeto em questão).

O *FILLER* é uma opção da cláusula *REDEFINE* para definir o número de bytes que preenchem os campos que estão sendo redefinidos. É importante lembrar que quando se usa essa notação, não é preciso especificar qualquer formato e tamanho entre os parênteses - a notação nX basta.

# Exemplo

```
0010 *****
0020 * ILUSTRA DECLARACAOREDEFINE COM A OPCAO DE FILLER
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 CAR VIEW OF VEHICLES
0070 2 PERSONNEL-ID
0080 2 MAKE
0090 2 DATE-ACQ
0100 2 REDEFINE DATE-ACQ
0110 3 FILLER 4X
0120 3 #MONTH (A2)
0130 3 FILLER 2X
0140 END-DEFINE
0150 *
0160 FIND (1) CAR WITH PERSONNEL-ID = '11100106'
0170 WRITE NOTITLE / 10T '=' MAKE
0180 / 10T '=' DATE-ACQ 5X '=' #MONTH
0190 END-FIND
0200 END
```

# Exemplo

## Saída:

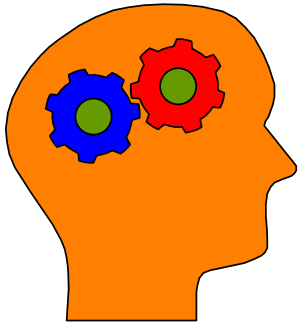
```
MAKE: VW  
DATE-ACQ: 19860115      #MONTH: 01
```

## Área de Dados Externa

No editor da área de dados, você pode definir qualquer tipo de campo, assim como no editor de programa. Se você estiver definindo uma *user-variable*, entre com o número do nível, o nome, formato e tamanho diretamente. Se for usar campos da DDM, você deve usar um comando para “puxar” os campos selecionados. O Natural traz os campos com seus respectivos nomes, tamanhos e formatos automaticamente para você.

Além da DDM, os dados podem ser inseridos na sua área de dados externa usando o comando **.I(xxxxxxxx)** (válido somente para o editor Natural no mainframe), a partir dos seguintes objetos: Mapas, Programas, Subprogramas, Subrotinas e *Helprotinas*.

Dependendo de como o seu sistema está parametrizado, o natural poderá dar um nome *default* à sua *user-view*, para sobrescrever esse nome, basta digitar o nome de sua preferência sobre o campo.



### Lembre-se!

- Campos do banco de dados não podem ser movidos ou copiados;
- Os formatos e tamanhos dos campos do banco de dados não podem ser mudados;
- Os campos do banco dados não podem ser renomeados;
- Os campos do banco de dados podem ser redefinidos usando a instrução *REDEFINE*;
- Os valores de inicialização não podem ser definidos para campos do banco de dados.



## Opção FILLER na Área de Dados Externa

A opção *FILLER* também está disponível na área de dados externa, no entanto, não é preciso mencionar a *keyword FILLER* quando usar essa opção, nem o formato e tamanho.

I	T	L	Name	F	Leng	Index/Init/EM/Name/Comment
All	-		-----	-	-----	-----
--						
V	1		VEHICLES-VIEW			VEHICLES
	2		PERSONNEL-ID	A	8	/* INTERN.
	2		MAKE	A	20	/* LOCAL/GENERIC
	2		MODEL	A	20	/* LOCAL/GENERIC
	2		COLOR	A	10	/* LOCAL/GENERIC
	2		YEAR	N	4.0	/* INTERN.
	2		DATE-ACQ	N	8.0	/* INTERN.
R	2		DATE-ACQ			
	3		4X			
	3		#MONTH	A	2	
	3		2X			
	1		#START-MAKE	A	20	
	1		#END-MAKE	A	20	

Uma das decisões que você precisará tomar ao definir os seus dados será a escolha da área de dados. Além disso, pode haver a necessidade de mover as definições dos dados de uma área externa para uma área de dados interna (ou vice-versa).

Há dois comandos para isso:

- **Generate** - Usado para mover as definições de uma área de dados externa para uma área de dados interna.
- **I(xxxxxxxx)**- Usado para mover as definições de uma área de dados interna para uma área de dados externa (válido somente para o editor Natural no mainframe).

E esse comando permite que você mova as definições contidas no editor da área de dados para o editor do programa. Quando o comando GEN é emitido (dentro do editor da área de dados externa), um *copycode* é criado. As definições são colocadas na LDA do *copycode*.

Para aproveitar essas definições num programa, basta digitar o comando *SET TYPE* "P" para mudar o tipo de objeto de *copycode* para programa.

## Comando **.I** (xxxxxxxx) (somente para Nat. Mainframe)

Esse comando permite que você mova as definições contidas no editor do programa para uma área de dados externa. Antes de inserir essas definições, devemos nos certificar que o objeto que contém os dados a serem “puxados” foi previamente catalogado.

Emitir o comando **.I**(<nome do objeto>) na última linha editada da área de dados, ou então na primeira linha do editor, caso não haja nenhum campo editado.

Selecione os campos que deseja incluir na área de dados através de uma das seguintes opções:

- ***All local variables and parameters***
- ***All local variables***
- ***Only internally defined local variables***
- ***All parameters***
- ***Only internally defined parameters***

# Exemplo

```
I T L Name                                F Leng Index/Init/EM/Name/Comment
All - -----
V 1 CARS                                  VEHICLES
  2 REG-NUM                               A   15 /* LOCAL FORMAT
  2 PERSONNEL-ID                          A    8 /* INTERN.
  2 MAKE                                   A   20 /* LOCAL/GENERIC
  2 MODEL                                  A   20 /* LOCAL/GENERIC
  2 COLOR                                  A   10 /* LOCAL/GENERIC
  2 COLOUR                                 A   10 /* LOCAL/GENERIC
  2 YEAR                                   N   4.0 /* INTERN.
*                                           /* INSERTED FROM SANPGM01
  1 #OPTION                                A    1
*                                           /* INSERTED FROM SANMAP01
V 1 EMPL                                  EMPLOYEES
  2 PERSONNEL-ID                          A    8
*                                           /* END OF SANMAP01
*                                           /* END OF SANPGM01
```

## Unidade A - Processamento Condicional

- Introdução
- Tomando uma decisão
- Controle de loop
- Variável lógica
- Resumo

## Condição Lógica

Para aplicar funções que envolvem o processamento condicional a seus programas, você pode usar qualquer uma das declarações abaixo.

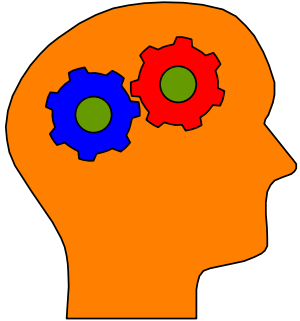
Condição Lógica	Descrição
<b>IF/THEN/ELSE</b>	Testa a condição lógica e divide-se em uma ou duas operações.
<b>DECIDE</b>	Estrutura condicional de múltiplas opções.
<b>REPEAT</b>	Usado para criar um processamento de loop. O loop continua até que uma certa condição seja satisfeita, ou até que um determinado número de iterações tenha sido completado.
<b>FOR</b>	Usado para criar um processamento de loop. O loop continua até que um certo número de iterações tenha sido completado.

### Instrução IF

Essa instrução define uma condição lógica que é executada em função da condição associada a ela. A declaração IF contém 3 componentes:

- **IF** - condição lógica que deve ser encontrada;
- **THEN** - instrução a ser executada caso a condição seja verdadeira;
- **ELSE** - (opcional) instrução a ser executada caso a condição seja falsa.



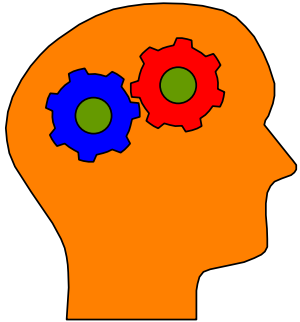


## Lembre-se!

- Você pode usar expressões aritméticas com IF.
- A cláusula ELSE não é essencial.
- Valores em campos alfanuméricos podem ser checados para um formato e tamanho específicos, para que você saiba se pode ou não converter o valor para usá-lo em outro formato. Os formatos válidos para conversão são: N, F, D, T, P, I. Use a função VAL para mover o valor para um campo de outro formato.

## Exemplo

```
IF #ALPHA IS (N5,3) THEN  
    #NUM := VAL(#ALPHA)  
END-IF
```

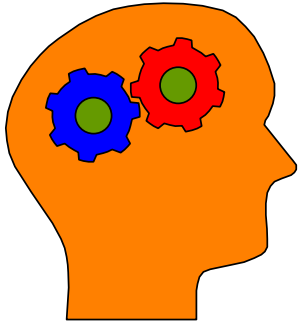


## Lembre-se!

- Você pode combinar vários operadores booleanos dentro de uma única instrução IF. A ordem em que os operadores são avaliados é a seguinte: (), NOT, AND e OR.

## Exemplo

```
IF YEAR = 80 THRU 89 AND MAKE='AUDI'  
    AND (COLOR = 'GREEN' OR ='BLACK')  
THEN  
    INPUT USING MAP 'CARMAP'  
END-IF
```

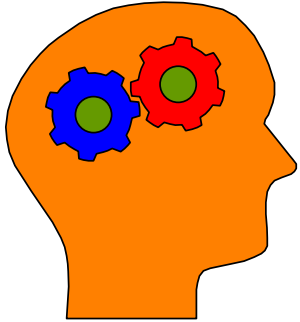


## Lembre-se!

- Você pode usar a opção SUBSTRING para comparar parte de um campo alfanumérico.

## Exemplo

```
IF SUBSTRING (#DATE,1,2) GT '12'  
  REINPUT  
    'MÊS INCORRETO PARA A MÁSCARA (MMDDYYYY)'  
END-IF
```



## Lembre-se!

- Você pode consultar posições selecionadas um campo de acordo com um conteúdo específico usando a opção MASK.

## Exemplo

```
IF #DATE NE MASK (DD/MM/YYYY)
```

```
  REINPUT
```

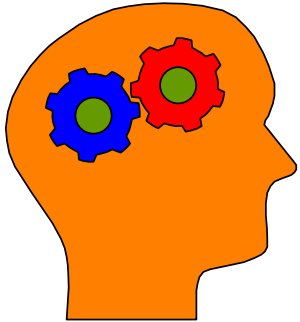
```
    'POR FAVOR, ENTRE COM O FORMATO "DD/MM/YYYY" '
```

```
END-IF
```

```
IF #SSN NE MASK (NNNNNNNNNN)
```

```
  REINPUT 'SSN MUST BE 9 DIGITS'
```

```
END-IF
```



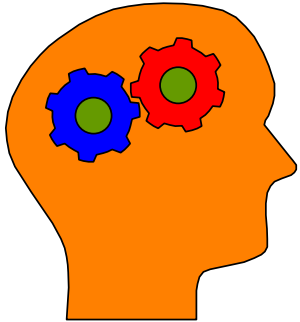
## Lembre-se!

- Você pode verificar se um valor termina com um caracter específico. No exemplo abaixo, a condição será verdadeira tanto se houver um 'E' na última posição do campo, ou o último 'E' no campo for seguido de brancos.

## Exemplo

```
IF #FIELD NE MASK ( * 'E' / )  
  REINPUT 'POR FAVOR, ENTRE COM UM NOME QUE TERMINE COM "E" '  
END-IF
```

# Tomando uma decisão

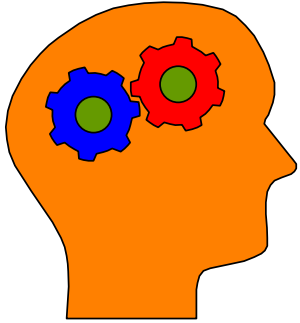


## Lembre-se!

- Campos numéricos também podem ser verificados dessa forma.

## Exemplo

```
IF #YEAR NE MASK ( *'9'/)  
  REINPUT 'POR FAVOR, ENTRE COM UM ANO QUE TERMINE COM "9" '  
END-IF
```



## Lembre-se!

- Instruções que são embutidas dentro de outras instruções são chamadas *nested* (aninhadas). Instruções IF podem ser aninhadas desde que uma condição leve a outra e assim por diante.
- Instruções com as cláusulas THEN ou ELSE, podem por sua vez, conter outras declarações IF/THEN/ELSE. Essa alternativa de aninhar cria vários caminhos que possibilitam a execução do programa.

# Tomando uma decisão

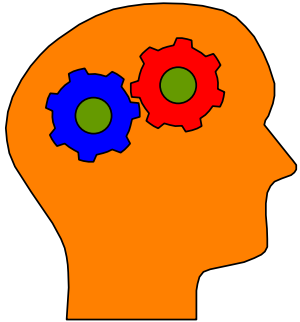
```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO IF
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 EMPL VIEW OF EMPLOYEES
0070   2 NAME
0080   2 JOB-TITLE
0090   2 PERSONNEL-ID
0100   2 CITY
0110   2 COUNTRY
0120 1 #PID (A8)
0130 1 #CV1 (C)
0140 1 #CV2 (C)
0150 1 #MESSAGE (A60)
0160 END-DEFINE
0170 *
0180 REPEAT
0190   INPUT USING MAP 'EMPLMAP'
...

```



## Tomando uma decisão

```
0200     IF #PID=' '
0210         ESCAPE BOTTOM
0220     END-IF
0230     FIND EMPL WITH PERSONNEL-ID = #PID
0240         MOVE (AD=P) TO #CV1 #CV2
0250         INPUT USING MAP 'EMPLMAP'
0260         IF #CV2 MODIFIED
0270             UPDATE
0280             END TRANSACTION
0290             MOVE (AD=P) TO #CV1 #CV2
0300             #MESSAGE := 'UPDATE DONE'
0310             INPUT USING MAP 'EMPLMAP'
0320         END-IF
0330     END-FIND
0340         RESET EMPL #CV1 #CV2 #MESSAGE #PID
0350 END-REPEAT
0360 *
0370     END
```



### Lembre-se!

- A opção *MODIFIED* é usada para testar o conteúdo de um campo que recebeu atributos dinamicamente foi modificado durante a execução de uma instrução *INPUT*.
- No primeiro *input* do mapa, as variáveis de controle são sempre referenciadas com o *status* “*NOT MODIFIED*” na declaração *INPUT*. Sempre que o conteúdo de um campo (que está ligado a uma variável de controle) é modificado, a variável de controle recebe o *status* “*MODIFIED*”.

### Instrução *DECIDE*

A instrução *DECIDE* permite a execução de múltiplas opções num processamento condicional. Assim como o *IF*, você avalia as condições lógicas ou valores de um campo. O tipo de avaliação irá determinar o tipo de declaração *DECIDE* que você usará. Há duas formas básicas:

- ***DECIDE FOR*** - refere-se a um ou mais campos.
- ***DECIDE ON*** - baseia-se nos valores de uma única variável.

## Exemplo - *DECIDE FOR*

```
DECIDE FOR FIRST CONDITION
  WHEN #FUNCTION ='A' AND #PARM ='X'
    PERFORM ROUTINE-A
  WHEN #FUNCTION ='B' AND #PARM ='X'
    PERFORM ROUTINE-B
  WHEN #FUNCTION ='B' AND #PARM ='Z'
    PERFORM ROUTINE-BZ
  WHEN NONE
    REINPUT 'INVALID FUNCTION ENTERED' MARK *#FUNCTION
END-DECIDE
```

## Exemplo - *DECIDE ON*

```
DECIDE ON FIRST VALUE OF *PF-KEY
  VALUE 'PF1'
    PERFORM ROUTINE-UPD
  VALUE 'PF2'
    PERFORM ROUTINE-ADD
  ANY VALUE
    END TRANSACTION
  WRITE 'RECORD HAS BEEN MODIFIED'
  NONE VALUE
    IGNORE
END-DECIDE
```

### Declaração *DECIDE* - OPÇÕES

Opções	Descrição
<b>FIRST</b>	Para a checagem até a primeira condição verdadeira ser encontrada
<b>EVERY</b>	Checa todas as condições. Processa todas as condições verdadeiras.
<b>ANY</b>	Processa a declaração se qualquer uma das condições especificadas for verdadeira.
<b>ALL</b>	Processa as declarações se todas as condições especificadas forem verdadeiras.
<b>NONE</b>	Opção obrigatória em todas declarações. É processada se nenhuma condição for verdadeira. A palavra IGNORE indica que nenhuma ação adicional é exigida se nenhuma condição for verdadeira.

### ***IF vs. DECIDE FOR***

Como regra geral, o número máximo de *IF* aninhados num programa é três, pois a depuração de *ifs* aninhados é difícil, Portanto, é aconselhável para mais de três níveis o uso da instrução *DECIDE*.

Vejam como fica a leitura de um mesmo programa, com o emprego de duas possibilidades diferentes: uma com o uso do *IF* e o outro com o usos do *DECIDE*.

## Exemplo com uso da instrução *IF*:

```
0010 *****
0020 * ILUSTRACAO O USO DA DECLARACAO IF
0030 *****
0040 DEFINE DATA LOCAL
0050 1 PERSON VIEW OF EMPLOYEES
0060 2 PERSONNEL-ID
0070 2 NAME
0080 2 MAR-STAT
0090 2 SEX
0100 1 #STATUS (A25)
0110 END-DEFINE
0120 *
0130 READ PERSON BY NAME
0140 IF MAR-STAT = 'M' THEN
0150     IF SEX = 'M' THEN
0160         #STATUS := 'THIS MAN IS MARRIED'
0170     ELSE
0180         #STATUS := 'THIS WOMAN IS MARRIED'
0190     END-IF
0200 ELSE
```



```
0210     IF MAR-STAT = 'S' THEN
0220         IF SEX = 'M' THEN
0230             #STATUS := 'THIS MAN IS SINGLE'
0240         ELSE
0250             #STATUS:= 'THIS WOMAN IS SINGLE'
0260         END-IF
0270     ELSE
0280         #STATUS := 'STATUS IS UNKNOWN'
0290     END-IF
0300 END-IF
0310 DISPLAY NOTITLE
0320 10T PERSONNEL-ID NAME 'MARITAL STATUS' #STATUS
0330 END-READ
0340 END
```

## Exemplo com uso da instrução *DECIDE*:

```
0010 *****
0020 * ILUSTRACAO O USO DA DECLARACAO DECIDE FOR
0030 *****
0040 DEFINE DATA LOCAL
0050 1 PERSON VIEW OF EMPLOYEES
0060 2 PERSONNEL-ID
0070 2 NAME
0080 2 MAR-STAT
0090 2 SEX
0100 1 #STATUS (A25)
0110 END-DEFINE
0120 *
0130 READ PERSON BY NAME
0140  DECIDE FOR FIRST CONDITION
0150     WHEN MAR-STAT='M' AND SEX='M'
0160         #STATUS := 'THIS MAN IS MARRIED'
0170     WHEN MAR-STAT='M' AND SEX='F'
0180         #STATUS := 'THIS WOMAN IS MARRIED'
0190     WHEN MAR-STAT='S' AND SEX='M'
0200         #STATUS := 'THIS MAN IS SINGLE'
```

```
0210     WHEN NONE
0220         #STATUS := 'STATUS IS UNKNOWN'
0230     END-DECIDE
0240 DISPLAY NOTITLE
0250 10T PERSONNEL-ID NAME 'MARITAL STATUS' #STATUS
0260 END-READ
0270 END
```

# Tomando uma decisão

PERSONNEL	NAME	MARITAL STATUS
ID		
1234567		STATUS IS UNKNOWN
99990000		STATUS IS UNKNOWN
55555551		STATUS IS UNKNOWN
44444444		STATUS IS UNKNOWN
95555551		STATUS IS UNKNOWN
60008339	ABELLAN	THIS MAN IS SINGLE
30000231	ACHIESON	THIS MAN IS SINGLE
50005800	ADAM	THIS WOMAN IS MARRIED
20009800	ADKINSON	STATUS IS UNKNOWN
20012700	ADKINSON	STATUS IS UNKNOWN
20013800	ADKINSON	THIS MAN IS MARRIED
20019600	ADKINSON	THIS MAN IS MARRIED
20008600	ADKINSON	STATUS IS UNKNOWN
20005700	ADKINSON	THIS MAN IS SINGLE
20011000	ADKINSON	THIS MAN IS MARRIED
11300313	AECKERLE	THIS WOMAN IS MARRIED
20013600	AFANASSIEV	THIS MAN IS SINGLE
20023500	AFANASSIEV	STATUS IS UNKNOWN

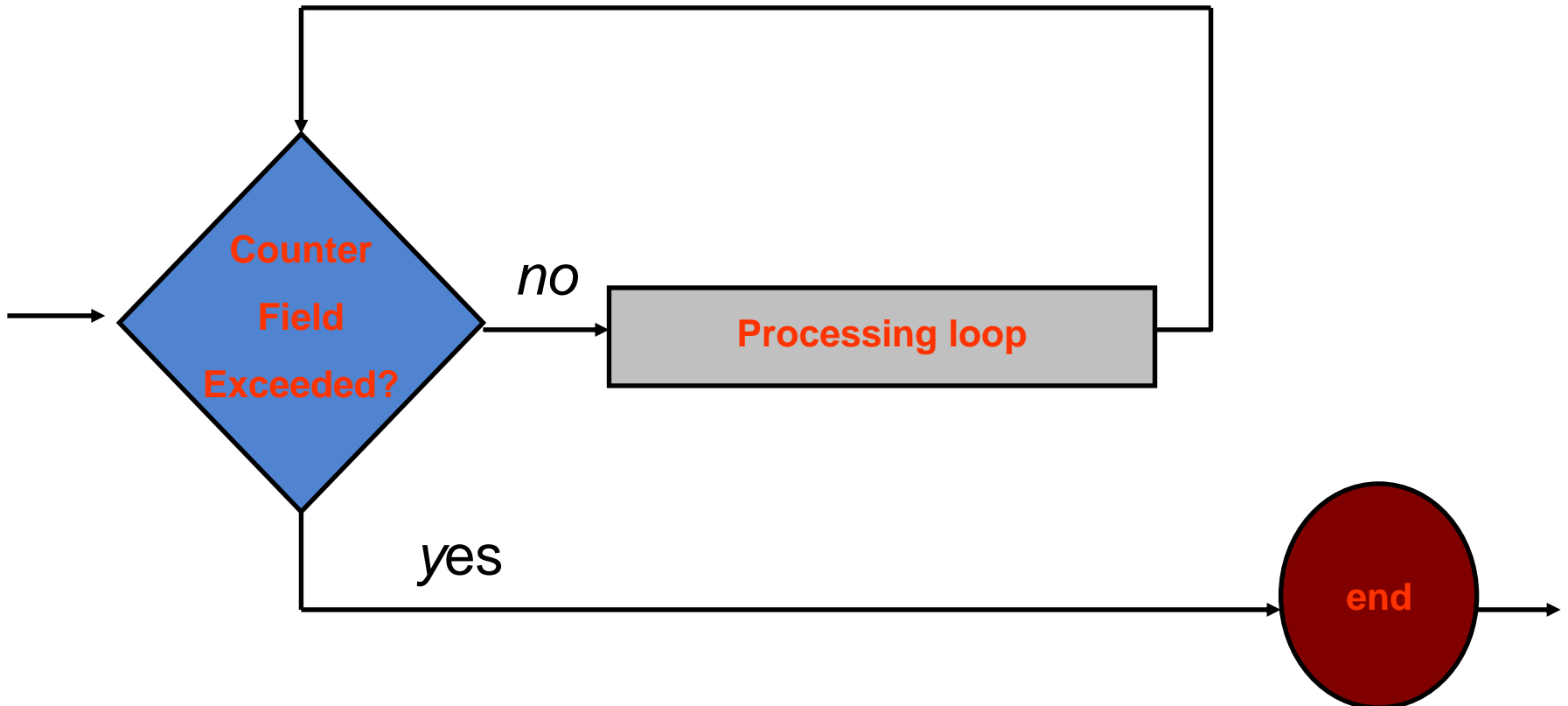
### Processos de Repetição

O Natural fornece duas instruções para ajudar a controlar o processamento de *loop*: *FOR* e *REPEAT*.

- ***FOR*** - Inicia um *loop* que é executado um número exato de vezes. Um campo de controle é usado para contar o número de iterações. O valor desse campo é incrementado num certo valor (chamado *STEP*) cada vez que o *loop* é processado. Esse campo deve ser referenciado dentro do *loop*.
- ***REPEAT*** - Você especifica uma ou mais instruções, que devem ser executadas repetidamente. Você pode ainda, definir uma condição lógica para que as instruções sejam executadas somente se a condição for encontrada. Nesse caso, você deverá usar as cláusulas *UNTIL* ou *WHILE*. Elas podem ser definidas no início ou no fim do *loop*.

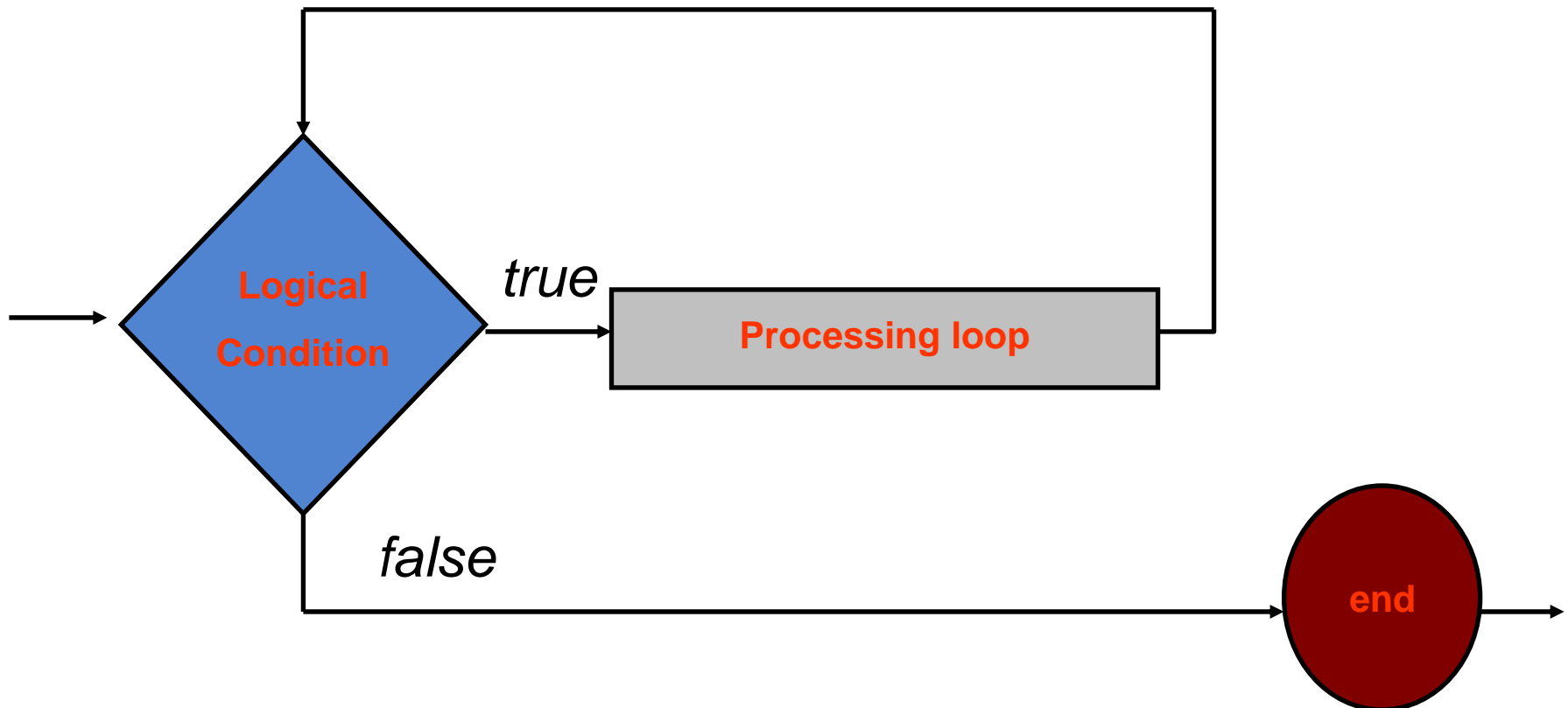
## **FOR**

O número exato de repetições é conhecido de antemão.



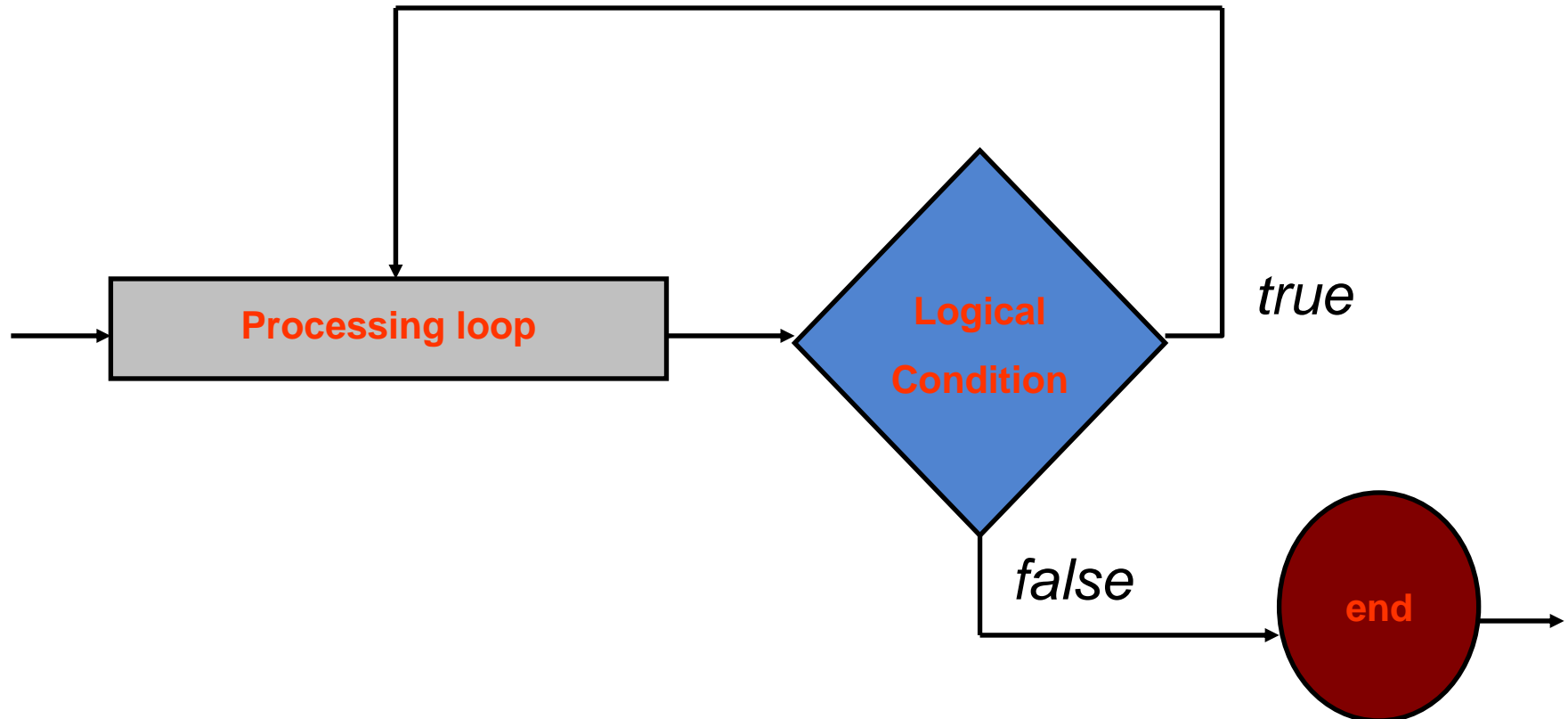
### ***REPEAT[UNTIL]***

Condição lógica no início do *loop*.



### ***REPEAT[WHILE]***

Condição lógica no fim do *loop*.





### Processos de Repetição

Se você não definir condição lógica alguma, a saída do *loop*, com a instrução *REPEAT*, deverá ser feita usando uma das seguintes instruções: *ESCAPE*, *STOP* ou *TERMINATE*.

### Exemplos

Nesse primeiro exemplo, a instrução *FOR* é usada para incrementar o valor de *#INDEX* de 1 até 5 a fim de criar um relatório das raízes quadradas dos números 1 ao 5.

```
0010 *****
0020 * ILUTSRA O USO DA DECLARACAO FOR NUM LOOP DE PROCESSAMENTO
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 #INDEX (I1)
0070 1 #ROOT (N2.7)
0080 END-DEFINE
0090 *
0100 FOR #INDEX 1 TO 5
0110 COMPUTE #ROOT = SQRT(#INDEX)
0120 WRITE NOTITLE 'THE SQUARE ROOT OF' #INDEX 'IS ' #ROOT
0130 END-FOR
0140 END
```

## Saída:

```
THE SQUARE ROOT OF 1 IS 1.0000000
THE SQUARE ROOT OF 2 IS 1.4142135
THE SQUARE ROOT OF 3 IS 1.7320508
THE SQUARE ROOT OF 4 IS 2.0000000
THE SQUARE ROOT OF 5 IS 2.2360679
```

### Exemplos

Nesse segundo exemplo, o *REPEAT* é usado para repetidamente permitir que os usuários entrem com um *PERSONNEL-ID* por vez até não ser mais necessário entrar com nenhum *ID* (ou seja, quando o usuário deixa o em branco).

```
0010 *****
0020 * ILUTSRA O USO DA DECLARACAO REPEAT
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 EMPLOY-VIEW VIEW OF EMPLOYEES
0070 2 PERSONNEL-ID
0080 2 NAME
0090 1 #PERS-NR (A8)
0100 END-DEFINE
0110 *
```

## Controle de *loop*

```
0120 REPEAT
0130   INPUT 'ENTER A PERSONNEL ID' #PERS-NR
0140     IF #PERS-NR = ' '
0150       ESCAPE BOTTOM
0160   END-IF
0170     FIND EMPLOY-VIEW WITH PERSONNEL-ID = #PERS-NR
0180     IF NO RECORDS FOUND
0190       REINPUT 'NO RECORDS FOUND'
0200     END-NOREC
0210     DISPLAY NOTITLE NAME
0220   END-FIND
0230 END-REPEAT
0240 END
```

## Saída:

```
NO RECORDS FOUND
```

```
ENTER A PERSONNEL ID 23
```

## Exemplos

Nesse terceiro exemplo, a opção *UNTIL* do *REPEAT* é usada para sair do *loop* quando *#X* tiver o valor 6. Note que a a opção *UNTIL* pode ser colocada no início ou no final do *loop*.

```
0010 *****
0020 * ILUSTRACAO DA DECLARACAO REPEAT
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 #X (I1) INIT <0>
0070 1 #Y (I1) INIT <0>
0080 END-DEFINE
0090 *
0100 REPEAT
0110  #X := #X + 1
0120  WRITE NOTITLE '=' #X
0130  UNTIL #X=6

0140 END-REPEAT

0150 END
```

## Saída:


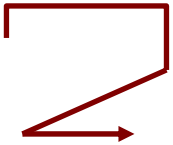
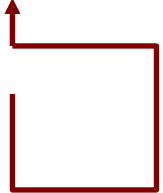
```
#X: 1  
#X: 2  
#X: 3  
#X: 4  
#X: 5  
#X: 6
```

### Finalizando os processamentos de loop

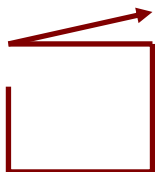
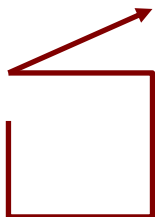
- **ESCAPE** - Finaliza a execução de um processamento de loop baseado na condição lógica. Há três opções para essa declaração: *ESCAPE TOP*, *ESCAPE BOTTOM* e *ESCAPE ROUTINE*.
- **STOP** - É usada para finalizar a execução de uma aplicação inteira Natural.
- **TERMINATE** - É semelhante à instrução *STOP* no que se refere a parar a aplicação inteira. Além disso, essa instrução sai do ambiente Natural.



# Finalizando os processamentos de *loop*

Declaração	Ilustração	Descrição
ESCAPE TOP		Retorna ao topo do loop e inicia o processamento da próxima iteração.
ESCAPE BOTTOM[(r)] [IMMEDIATE]*		Termina e sai do loop para o qual o label se refere. O processamento continua a partir da 1.ª declaração após o loop.
ESCAPE ROUTINE [IMMEDIATE]		A rotina atual Natural abandona o controle. Para subrotinas, o processamento continua a partir da 1.ª declaração após aquela usada para chamar a subrotina.

### Finalizando os processamentos de *loop*

Declaração	Ilustração	Descrição
STOP		Finaliza o programa/aplicação e retorna o controle ao Natural.
TERMINATE		Finaliza o programa/aplicação e sai do Natural.

#### Observação:

Se a opção IMMEDIATE for utilizada, ambos, a última declaração AT BREAK e AT END OF DATA são executadas antes da saída do loop.

### *O que são variáveis lógicas?*

Quando o valor de um campo pode ser definido como verdadeiro/falso, você pode lhe designar o formato L para o campo. O uso da variável lógica em seu programa permite que você o referencie da seguinte maneira:

```
0010 DEFINE DATA
0020 LOCAL
0030 1 #ROUTINE-DONE (L) INIT <TRUE>
0040 END-DEFINE
0050 .
0060 .
0070 .
0080 IF #ROUTINE-DONE = TRUE
0090 THEN ...
0100 END-IF
0110 END
```

# Variáveis Lógicas

## Máscaras de edição

Para tornar o uso das variáveis lógicas mais significativo, a edição de máscara *EM=FALSE/TRUE* pode ser definida. Essa edição pode ser trocada por *OFF/ON*, *NO/YES*, *REJECT/ACCEPT*.

```
0010 DEFINE DATA
0020 LOCAL
0030 1 #SWITCH (L) INIT <TRUE>
0031 1 #INDEX (I1)
0040 END-DEFINE
0060 FOR #INDEX 1 5
0070     WRITE #SWITCH (EM=FALSE/TRUE) 5X 'INDEX = ' #INDEX
0070     WRITE #SWITCH (EM=FALSE/TRUE) 5X 'INDEX = ' #INDEX
0071 SKIP 1
0080 IF #SWITCH = TRUE
0090     MOVE FALSE TO #SWITCH
0100 ELSE
0110     MOVE TRUE TO #SWITCH
0120 END-IF
0130 END-FOR
0140 END
```

## Saída:

Page	1		
TRUE	INDEX =	1	
TRUE	INDEX =	1	
FALSE	INDEX =	2	
FALSE	INDEX =	2	
TRUE	INDEX =	3	
TRUE	INDEX =	3	
FALSE	INDEX =	4	
FALSE	INDEX =	4	
TRUE	INDEX =	5	
TRUE	INDEX =	5	

## Unidade B - Usando os objetos Natural efetivamente

- Programas
- Subrotinas
- Subprogramas
- Copycode

Os programas são fundamentais para qualquer aplicação. Desde que tenham uma área de dados interna, não precisam de qualquer outro objeto para ser executado. Nas grandes aplicações, no entanto, eles servem como navegador e também podem ser usados para chamar outros objetos.

### Instrução ***FECTCH***

Um programa pode chamar outro através da instrução *FETCH* ou *FETCH RETURN*. A diferença entre eles é que a declaração *FETCH RETURN* retorna ao programa chamador e o outro não.

### A pilha do Natural (*stack*)

A pilha do Natural é uma porção da área de trabalho usada para guardar informações para uso futuro. Os dados são acessados por um objeto através da instrução *INPUT*. O uso da pilha para compartilhar dados é mais lento que o uso da GDA. No entanto a pilha permite a passagem de dados através de outras aplicações, o que não acontece com a GDA.



## Controlando pilha

- Usando os objetos programáveis você pode carregar comandos e/ou dados na pilha;
- Você pode carregar esses itens no topo ou na base da pilha;
- Você pode colocar os dados na pilha no modo *Form* ou no modo *Delimiter*;
- Você pode ver o que está no topo da pilha através da variável de sistema \*DATA. Essa variável pode assumir os seguintes valores:

Valor	Descrição
0	A pilha está vazia
-1	O topo da pilha contém um comando
n	O topo da pilha contém n elementos

## Passando dados

Quando você estiver passando dados de um programa para outro, lembre-se dos seguintes pontos em relação à GDA e a pilha:

### Pilha (*Stack*)

- A pilha passa dados via lista de argumentos;
- Os dados da pilha são recebidos no programa chamador dia declaração *INPUT*;
- Informações limitadas podem ser passadas em uma única entrada da.

## **GDA** (*Global Data Area*)

- Inclua o mesmo nome de GDA em ambos objetos- o chamado e o chamador;
- Os dados da GDA no programa chamador está disponível para o programa chamado;
- A GDA é mais eficiente e flexível que a pilha.

## Exemplo

```
DEFINE DATA
  GLOBAL USING GDA1
LOCAL
1 #PARM1 (A10)
1 #PARM2 (A10)
...
END-DEFINE
...
FETCH RETURN 'PGM2' #PARM1 #PARM2
...
END
```

GDA

NATURAL  
*Stack*

```
DEFINE DATA
  GLOBAL USING GDA1
LOCAL
1 #ARG1 (A10)
1 #ARG2 (A10)
...
END-DEFINE
...
INPUT #PARM1 #PARM2
...
END
```

## Carregando a pilha

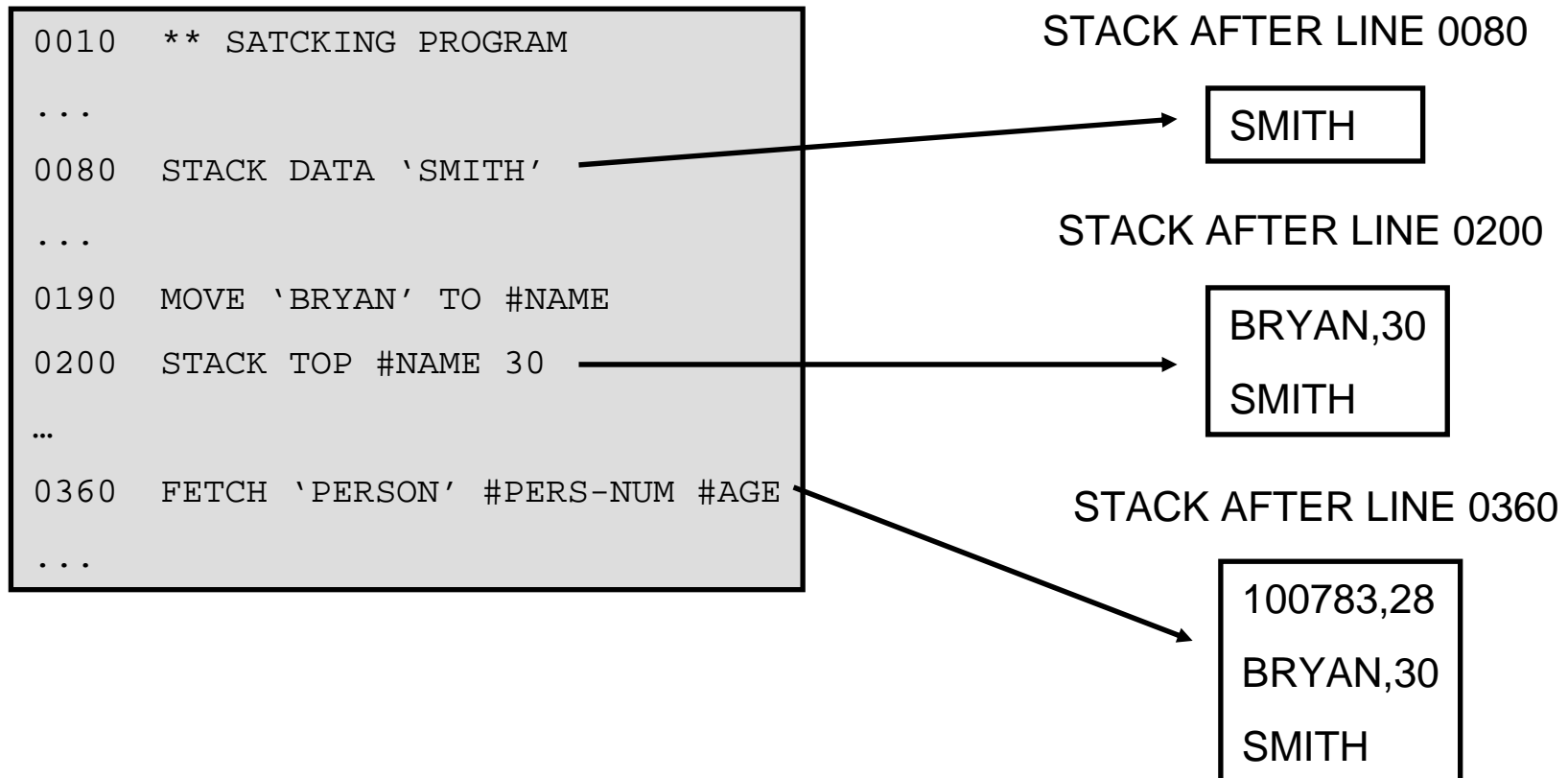
- Você pode carregar a pilha usando o comando *EXECUTE*;
- Com as declarações *FETH* e *FETCH RETURN*;
- Com a declaração *STACK*
- Essa declaração, por “*default*” coloca os dados na base da pilha no modo *delimiter*.

## Limpendo a Pilha

- Você pode limpar a pilha através da declaração *RELEASE STACK*;
- Com o comando de terminal %%;
- Pressionando a tecla *CLEAR*.

O comando de terminal “%P” apaga a primeira entrada e o comando “%S” lê a primeira entrada sem apagá-la.

## Como a pilha recebe os dados



## Como a pilha recebe os dados

```
0010  ** INPUT PROGRAM
0020  ** 'PERSON'
...
0120  INPUT #PERS-NUM AGE
0130  INPUT #PERS-NAME #NUM
...
0200  INPUT #LASTNAME
...
```

STACK AFTER LINE 0360

100783,28  
BRYAN,30  
SMITH

STACK AFTER LINE 0130

SMITH

As subrotinas são tipicamente usadas para carregar funções específicas em seu sistema. Você tem dois tipos de subrotinas disponíveis no Natural:

### Subrotinas Internas

- São codificadas dentro do objeto;
- Está disponível somente para esse objeto;
- Têm acesso à GDA e a LDA do objeto nos qual foi codificada, assim como aos dados passados ao usar-se a PDA.



## Subrotinas Externas

- São codificadas como um objeto separado;
- Podem ser acessadas por múltiplos objetos.

O que constitui uma subrotina

```
PERFORM subroutine-name  
...  
DEFINE SUBROUTINE subroutine-name  
    (processing statements)  
END-SUBROUTINE  
...
```

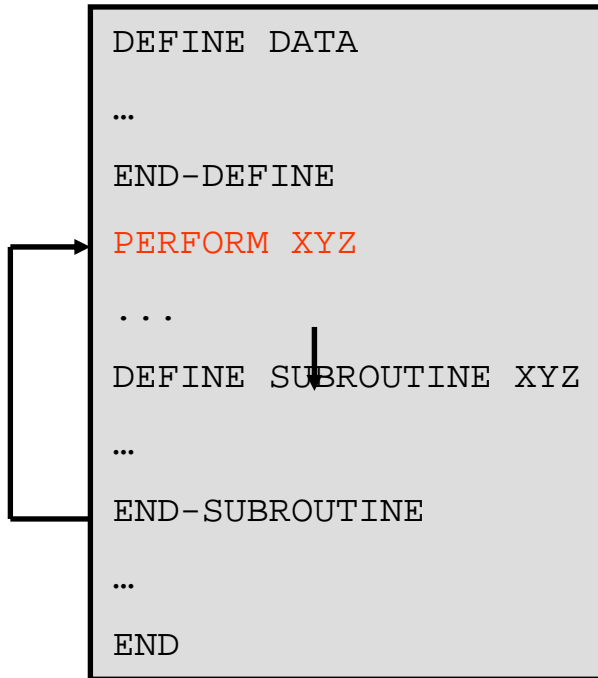
### Subrotinas Externas

- Tudo que você codifica entre as declarações *DEFINE SUBROUTINE* e *END-SUBROUTINE* é considerado parte da subrotina. Qualquer processamento de *loop* iniciado dentro de uma subrotina deve ser fechado antes da declaração *END-DEFINE*.
- A declaração *PERFORM* é usada para chamar tanto a subrotina interna quanto a externa. Ela busca o nome da subrotina definido na declaração *DEFINE SUBROUTINE* e não o nome do objeto.

# Subrotinas

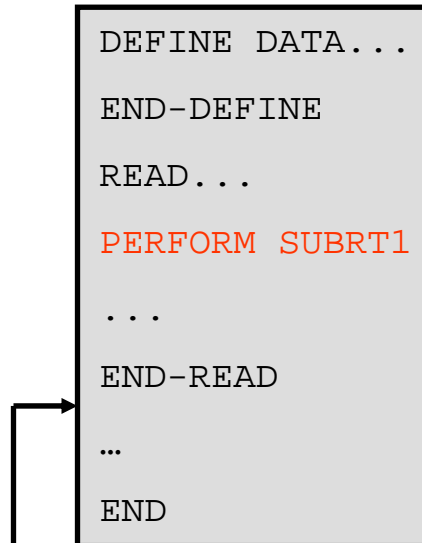
## Subrotina Interna

### PGMA

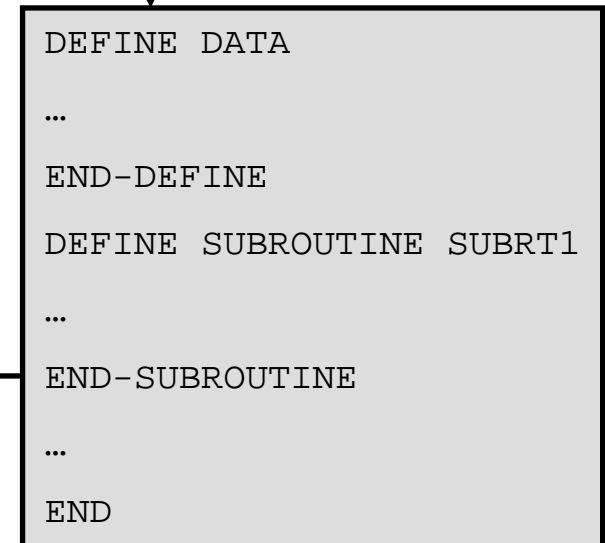


## Subrotina Externa

### PGM1



### SUBRT1



## Prós e contras das subrotinas internas

### Prós

- Auxilia a modularização interna do objeto;
- O teste e a depuração fica confinado a um objeto apenas;
- Melhor performance de todos os módulos.

### Contras

- O objeto no qual está inserido irá possuir um número maior de linhas de código;
- Não é reutilizável e não pode ser compartilhado por outros objetos.

## Prós e contras das subrotinas externas

### Prós

- Auxilia a padronização do sistema;
- Mantém os módulos com tamanhos manipuláveis;
- Acesso à GDA e a PDA;
- Facilidade nas funções de segurança;
- Compartilhável entre os outros objetos da aplicação;
- Permite o uso do XREF.

## Prós e contras das subrotinas externas

### Contras

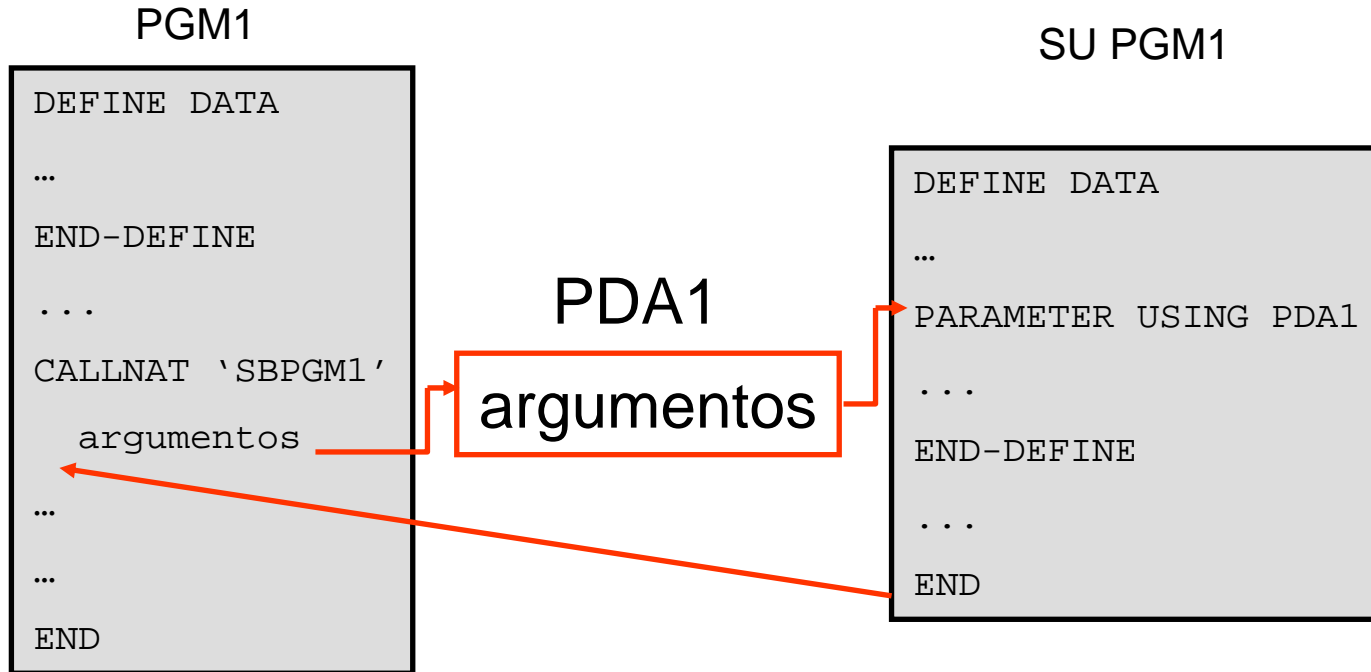
- O teste e a depuração envolve muitos objetos;
- Aumenta o uso do *buffer pool*;
- Os dados compartilhados através da GDA ou da pilha limita a reutilização e o controle da interface.

### Subprogramas vs. Subrotinas

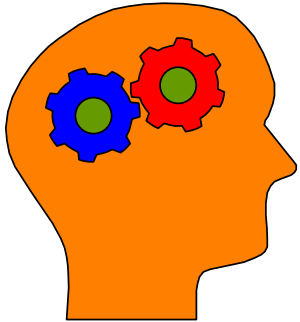
Em vez de chamar uma subrotina, você pode chamar um subprograma. Basta emitir a declaração *CALLNAT*.

Os subprogramas diferem das subrotinas na forma como compartilham os dados no objeto chamador. Os subprogramas só podem acessar os dados através de um conjunto de parâmetros definidos na PDA; as subrotinas podem acessar os dados através da GDA e da PDA.

# Subprogramas

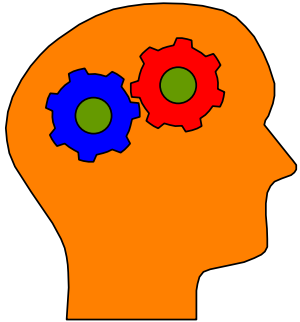






## Lembre-se!

- Os subprogramas fornecem um meio mais eficiente de compartilhar dados que a pilha, uma vez que os dados são meramente referenciados e não passados;
- Os subprogramas passam apenas referências para os dados definidos em suas GDA ou PDA;
- Por “*default*”, um parâmetro é passado por referência para um subprograma/subrotina, ou seja, é transferido pelo seu endereço. Um campo definido como parâmetro numa declaração *CALLNAT/PERFORM* deve ter o mesmo formato/tamanho que o campo correspondente;

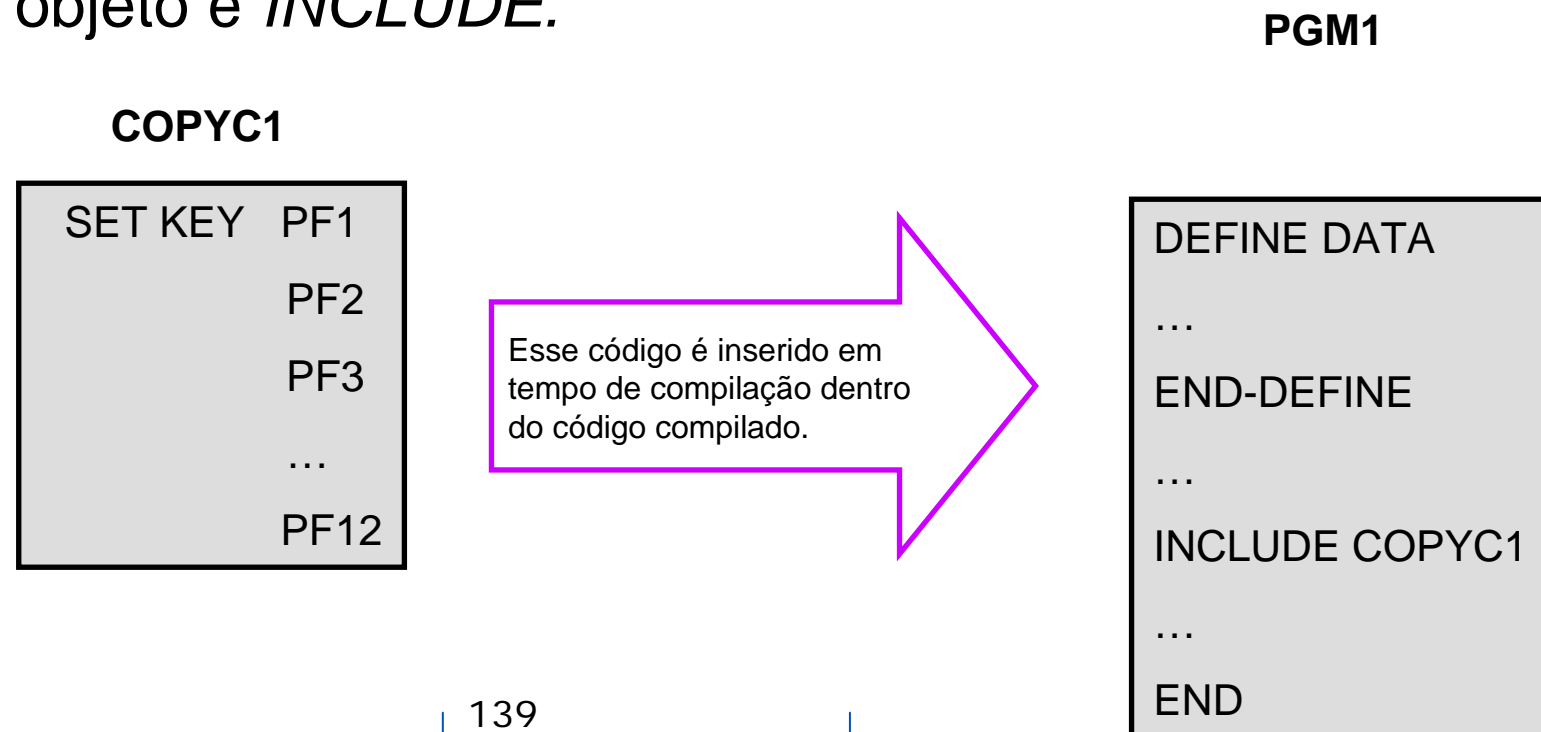


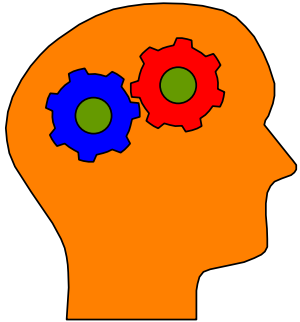
## Lembre-se!

- Se parâmetros são passados por valor (*by value*), é o valor atual do parâmetro que é passado e não o endereço, logo o campo do subprograma não precisa ter o mesmo formato e tamanho que o parâmetro da declaração *CALLNAT/PERFORM*;
- Se os valores dos parâmetros que foram modificados no subprograma devem ser retornados ao programa chamador você tem que definir esses campos como *BY VALUE RESULT*.

Com o *copycode* você pode inserir rotinas especiais dentro do seu objeto em tempo de compilação em vez de ter que codificar essas linhas repetidamente.

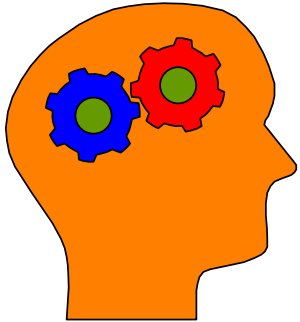
A declaração usada para incorporar o *copycode* ao seu objeto é *INCLUDE*.





## Lembre-se!

- Você não pode ter a declaração END em seu *copycode*;
- *Copycode* é salvo, nunca catalogado;
- O *copycode* não pode ser checado;
- O *copycode* não pode conter um trecho de outra declaração (só pode conter uma ou mais declarações completas);
- O teste e a depuração pode ser difícil;
- Se você modifica um *copycode*, você deve recompilar todos os objetos que o utiliza.



## Lembre-se!

- O *copycode* aumenta o tamanho do objeto compilado que o utiliza. O tamanho do código fonte não sofre aumento;
- Valores podem ser inseridos dinamicamente dentro do *copycode* usando-se a notação &n&.

## Unidade C - Manipulação de Dados

- Calculando valores
- Atribuindo valores
- Combinando e separando valores
- Trabalhando com *Strings*
- Trabalhando com Data e Time
- Processamento de *arrays*

## Operadores aritméticos

Cada operador deve ser precedido por um caracter branco para não ser confundido com o nome da variável. Os seguintes símbolos são usados como operadores:

Operador	Símbolo
Add	+
Subtract	-
Multiply	*
Divide	/
Parênteses	()
Exponencial	**

## Ordem do processamento

Os cálculos aritméticos possuem vários operadores para serem processados dentro de uma única declaração. Para assegurar a exatidão desses cálculos, a seguinte ordem deve ser seguida:

1. Parênteses;
2. Exponencial;
3. Multiplicação/Divisão (da esquerda para direita);
- 4 Adição/Subtração (da esquerda para direita).



Operador	Descrição
/	<p><i>DIVIDE[ROUNDED] oper1 INTO oper2[GIVING oper3] [REMAINDER oper4]</i></p> <p>O resultado é colocado:</p> <ul style="list-style-type: none"><li>-operador2 <code>DIVIDE #A INTO #B</code></li><li>-operador3 <code>DIVIDE #A INTO #B GIVING #RESULT</code></li></ul>
*  /  +  -	<p><code>COMPUTE[ROUNDED] {oper1}...expressão aritmética</code></p> <p>O resultado é colocado:</p> <ul style="list-style-type: none"><li>-operador1 <code>COMPUTE #A = (#A + 1) + (#B / (#C * 8))</code></li></ul> <p>Notas:</p> <ul style="list-style-type: none"><li>- Use os parênteses para agrupar operações complexas;</li><li>- A palavra chave “COMPUTE” é opcional em report mode;</li><li>- A palavra chave “COMPUTE” não é usada com a notação “:=“;</li><li>- A função DIVIDE é sempre ROUNDED.</li></ul>

Operador	Descrição
+	<p><i>ADD[ROUNDED] oper1...TO oper2[GIVING oper3]</i></p> <p>O resultado é colocado:</p> <p>-operador2   ADD 7 TO #FIELD1</p> <p>-operador3   ADD 3 TO #FIELD1 GIVING #RESULT</p>
-	<p><i>SUBTRACT[ROUNDED] oper1...FROM oper2[GIVING oper3]</i></p> <p>O resultado é colocado:</p> <p>-operador2   SUBTRACT #A FROM SALARY</p> <p>-operador3   SUBTRACT #A FROM SALARY GIVING #NETPAY</p>
*	<p><i>MULTIPLY[ROUNDED] oper1...BY oper2[GIVING oper3]</i></p> <p>O resultado é colocado:</p> <p>-operador1   MULTIPLY #A BY #B</p> <p>-operador3   MULTIPLY #B BY 6 GIVING #YTD</p>

### ***RESET***

Essa instrução devolve à variável o valor nulo ou seu valor original. Os seguintes elementos podem ser atualizados com o *RESET*:

- Campos elementares de qualquer formato Natural válido;
- *Arrays*;
- Campos de grupo;
- Variáveis do sistema;
- Toda *user view* ou campos do banco dentro dessas *views*.

## Exemplo

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO RESET INITIAL
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 #A (P4.2)INIT <119.2>
0070 1 #B (P4.2)INIT <17>
0080 1 #C (P4.2)
0090 1 #D (P4.2)
0100 END-DEFINE
0110 DIVIDE #B INTO #A GIVING #C REMAINDER #D
0120 WRITE // 'DEPOIS DE TER DIVIDIDO #B DE #A: ' 40T #A #B #C #D
0130 RESET INITIAL #A #B #C #D
0140 WRITE // 'RETORNANDO AOS VALORES INICIAIS: ' 40T #A #B #C #D
0150 RESET #A #B #C #D
0160 WRITE // 'LIMPANDO OS VALORES DAS VARIÁVEIS: ' 40T #A #B #C #D
0170 END
```

## Saída:

Page 1 03-01-16 15:50:38

DEPOIS DE TER DIVIDIDO #B DE #A: 119.20 17.00 7.01 0.03

RETORNANDO AOS VALORES INICIAIS: 119.20 17.00 0.00 0.00

LIMPANDO OS VALORES DAS VARIAVEIS: 0.00 0.00 0.00 0.00

### ***RESET*** com *arrays*

Ao fazer uso da instrução *RESET* com *arrays*, índices devem ser fornecidos. Quando o “asterisco” é informado [ex.: #ARRAY(\*)], todo *array* é atualizado com nulo ou com seus valores iniciais. Mas quando o índice é informado [ex.: #ARRAY(2:5)], somente as ocorrências apontadas são atualizadas.

## Exemplo

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO RESET
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 #A (A30) INIT <'BUILDING NATURAL APPLICATIONS'>
0070 1 #B (P3.2)
0080 1 #C (I4)
0090 1 #ARRAY (A4/12)
0100 1 #GRP
0110 2 #GRP-F-1 (A10)
0120 2 #GRP-F-2 (N5)
0130 1 PERSON VIEW OF EMPLOYEES
0140 2 NAME
0150 2 BIRTH
0160 2 CITY
0170 END-DEFINE
0180 RESET #A #B #C #ARRAY(*) #GRP PERSON
0190 END
```

### ***MOVE vs. ASSIGN***

Essa instrução copia um valor para dentro de uma variável. A palavra *ASSIGN* pode ser omitida se usarmos a notação “:=”. Ambas as formas são válidas:

```
ASSIGN #MAKE = 'FORD'
```

```
#MAKE := 'FORD'
```

A instrução *MOVE* também copia um valor para dentro de uma variável. Se os valores que estão sendo movidos têm um formato diferente da variável que está recebendo, o Natural primeiro converte o dado antes de copiá-lo.

```
MOVE 'FORD' TO #MAKE
```



## Exemplo:

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO ASSIGN
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 MODEL
0080 2 COLOR
0090 2 YEAR
0100 1 #MAKE(A20)
0110 END-DEFINE
0120 *
0130 ASSIGN #MAKE = 'FORD' /* MOVE 'FORD' TO #MAKE seria a outra alternativa
0140 FIND CARS WITH MAKE=#MAKE
0150 DISPLAY NOTITLE YEAR MAKE MODEL
0160 END-FIND
0170 END
```

## Saída:

YEAR	MAKE	MODEL
1982	FORD	SIERRA
1980	FORD	CAPRI
1980	FORD	ESCORT
1980	FORD	CAPRI
1980	FORD	CAPRI
1980	FORD	FIESTA
1980	FORD	ESCORT
1985	FORD	ESCORT LASER
1983	FORD	TRANSIT
1978	FORD	GRANADA
1984	FORD	SIERRA 2.0
1986	FORD	SCORPIO 2.0 I GL
1984	FORD	FIESTA XR2
1985	FORD	SIERRA L TURNIER
1986	FORD	ESCORT XR3I
...		

### Opções da instrução MOVE

Opção	Descrição
<b>MOVE ROUNDED</b>	Pode ser usado quando o valor a ser recebido é uma variável numérica. O valor é arredondado antes de ser movido.
<b>MOVE EDITED</b>	Copia o valor de acordo com o formato da máscara de edição (EM=).
<b>MOVE BY NAME</b>	Copia individualmente campos de uma estrutura de dados para outra baseada no nome do campo sem levar em consideração sua posição.

### Opções da instrução *MOVE*

Opção	Descrição
<b>MOVE ALL</b>	Copia o valor para dentro de uma variável até que ela esteja totalmente preenchida. A opção <i>UNTIL</i> limita o número de posições.
<b>MOVE BY POSITION</b>	Copia o conteúdo de um campo de uma estrutura para outra baseada na posição que ela ocupa. O mesmo número de campos devem existir em cada estrutura.
<b>MOVE SUBSTRING</b>	Copia <i>substrings</i> de um campo para uma variável.
<b>MOVE LEFT/RIGHT JUSTIFIED</b>	Faz com que o dado seja posicionado à esquerda/direita ou justificado quando é movido para a variável que recebe.

## Exemplo da opção *MOVE EDITED*

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO ASSIGN
0030 *****
0040 DEFINE DATA LOCAL
0050 1 #DATEA-RESULT      (D)
0060 1 #NUMERIC-RESULT   (N6)
0070 1 #ALPHA-DATEA     (A8) INIT <'96/01/20'>
0110 END-DEFINE
0120 *
0130 MOVE EDITED #ALPHA-DATE TO #NUMERIC-RESULT (EM=99/99/99)
0140 WRITE '=' #NUMERIC-RESULT '=' #ALPHA-DATE
0150 **
0160 MOVE EDITED #ALPHA-DATE TO #DATE-RESULT      (EM=YY/MM/DD)
0170 WRITE '=' #DATE-RESULT '=' #ALPHA-DATE
0180 **
0190 END
```

## Saída:

Page 1

03-01-16 16:06:34

#NUMERIC-RESULT: 960120 #ALPHA-DATE: 96/01/20

#DATE-RESULT: 96-01-20 #ALPHA-DATE: 96/01/20

### ***COMPRESS***

A instrução *COMPRESS* combina os valores de dois ou mais campos em um único campo alfanumérico.

Se o campo que irá receber o conteúdo dos valores comprimidos for maior que o conteúdo combinado o restante é preenchido com brancos.

Se o campo que irá receber o conteúdo for menor, o valor será truncado. Zeros à esquerda em campos numéricos e brancos entre campos alfanuméricos são suprimidos quando combinados. Se os zeros à esquerda forem necessários, o campo numérico deve ser redefinido como alfanumérico.

### Opções da declaração *COMPRESS*

Opção	Descrição
<b>ALL</b>	Usado com a opção <i>WITH DELIMITER</i> . O delimitador é colocado campo alvo para cada branco que não é transferido.
<b>SUBSTRING</b>	Permite transferir parte de um campo.
<b>LEAVING NO SPACE</b>	Não haverá brancos ou qualquer outro delimitador entre os campos.
<b>WITH DELIMITER</b>	Haverá um delimitador especial entre os campos.



## Combinando e separando valores

Opção	Descrição
<b>FULL</b>	Os valores dos campos fontes são combinados para incluir todos os brancos e zeros..
<b>NUMERIC</b>	Pontos e sinais no campo fonte serão transferidos para o campo alvo.

## Exemplo:

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO COMPRESS
0030 *****
0040 DEFINE DATA LOCAL
0050 1 VIEWEMP VIEW OF EMPLOYEES
0060 2 FIRST-NAME
0070 2 MIDDLE-I
0080 2 NAME
0090 1 #FULL-NAME (A20)
0100 END-DEFINE
0110 **
0120 READ (5) VIEWEMP BY NAME STARTING FROM 'JONES'
0130 COMPRESS FIRST-NAME MIDDLE-I NAME INTO #FULL-NAME
0140 DISPLAY NOTITLE FIRST-NAME MIDDLE-I NAME #FULL-NAME
0150 SKIP 1
0160 END-READ
0170 END
```

## Saída:

FIRST-NAME	MIDDLE-I	NAME	#FULL-NAME
VIRGINIA	J	JONES	VIRGINIA J JONES
MARSHA		JONES	MARSHA JONES
ROBERT	B	JONES	ROBERT B JONES
LILLY	P	JONES	LILLY P JONES
EDWARD	C	JONES	EDWARD C JONES

### ***SEPARATE***

A instrução *SEPARATE* divide o conteúdo de um campo alfanumérico em dois ou mais campos ou em múltiplas ocorrências de um *array*.

Se você não especificar um delimitador, qualquer caracter diferente de letra ou número será tratado como tal. A separação ocorre sempre que um delimitador é encontrado.

### Opções da declaração *SEPARATE*

Opção	Descrição
WITH INPUT DELIMITER	Usa o caracter <i>default</i> de entrada como delimitador.
WITH DELIMITER	Define o delimitador que será usado para separar. Brancos são ignorados.
LEFT JUSTIFIED	Brancos entre o delimitador e o próximo caracter não branco são removidos no campo alvo.
GIVING NUMBER	Retorna o número de campos alvo que receberam dados durante a separação.

## Combinando e separando valores

Opção	Descrição
<b>IGNORE REMAINDER</b>	Evita mensagens de erro quando você define um número de campos alvo menor que o esperado.
<b>RETAINED DELIMITERS</b>	Posiciona os delimitadores indicados dentro dos campos alvos
<b>SUBSTRING</b>	Define a porção do campo a ser processada

## Exemplo:

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO SEPARATE
0030 *****
0040 DEFINE DATA LOCAL
0050 1 #PRODUCT-NAME      (A40) INIT <'ENTIRE APPC SERVER'>
0060 1 #PRODUCT-CATEGORY (A10) /* EX.: NATURAL, ADABAS, ENTIRE
0070 END-DEFINE
0080 *
0090 SEPARATE #PRODUCT-NAME INTO #PRODUCT-CATEGORY IGNORE
0100 WRITE NOTITLE
0110    // 4X '=' #PRODUCT-NAME / '=' #PRODUCT-CATEGORY
0120 END
```

## Saída:

```
#PRODUCT-NAME: ENTIRE APPC SERVER  
#PRODUCT-CATEGORY: ENTIRE
```



### ***EXAMINE***

A instrução *EXAMINE* varre o conteúdo de um campo alfanumérico ou de um *array* a procura de um *string*. Há duas formas para essa instrução:

*ABSOLUTE* - Qualquer ocorrência do *string* é procurada.

*WITH DELIMITERS* - As ocorrências devem ser delimitadas por brancos ou por um caracter especial.

### **Opção *DELETE/REPLACE***

Use essas opções para excluir ou substituir cada valor do operando1 que é idêntico ao operando2.

## Opções da instrução *EXAMINE*

Clausula	Descrição
<b>EXAMINE PATTERN</b>	Examina o campo através de um caracter padrão.
<b>EXAMINE SUBSTRING</b>	Define uma porção do campo a ser examinada.
<b>EXAMINE TRANSLATE</b>	Transforma os dados de maiúscula para minúscula através de uma tabela de conversão.

## Clausula GIVING:

Clausula	Descrição
GIVING NUMBER	Número de ocorrências do <i>string</i> .
GIVING POSITION	Byte inicial onde o primeiro <i>string</i> foi encontrado.
GIVING LEFT	Tamanho final do valor do campo examinado após todas as exclusões e substituições terem sido feitas..
GIVING INDEX	Índice do <i>array</i> onde a primeira ocorrência do <i>string</i> foi encontrada.

## Exemplo:

```
EXAMINE #TELE FOR PATTERN '(...)' GIVING NUMBER #NUMBER  
EXAMINE #NAME FOR ' ' GIVING NUMBER #NUM GIVING POSITION #POS  
EXAMINE FULL #NAME FOR ' ' GIVING NUMBER #NUM GIVING POSITION #POS  
EXAMINE #ARRAY(*) FOR 'X' GIVING INDEX #INDEX
```

## Trabalhando com Data e Hora

No natural os campos Data e Hora têm seus próprios formatos: D para data e T para hora. Quando os definimos, basta indicar seus formatos. Internamente eles são definidos como campos numéricos compactados (P6 e P12).

### Combinando campos

É possível fazer cálculos com esses campos. A tabela abaixo mostra os formatos recomendados para os campos de destino.

Combinação de formatos	Resultados
Numérico(I,n,P) com D ou T	Colocado no campo Data ou hora
Data com Hora	Colocado no campo Hora
Data com Data	Colocado num campo P6
Hora com Hora	Colocado num campo P12

## Exemplo:

```
0010 *****
0020 * ILUSTR A O USO DA DE CAMPOS DATA
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMPL VIEW OF EMPLOYEES
0060 2 NAME
0070 2 PERSONNEL-ID
0080 2 LEAVE-START (1)
0090 2 REDEFINE LEAVE-START
0100 3 #LEAVE-START-A (A6)
0110 2 LEAVE-END (1)
0120 2 REDEFINE LEAVE-END
0130 3 #LEAVE-END-A(A6)
0140 1 #LEAVE-DUE (P6)
0150 1 #START-DATE (D)
0160 1 #END-DATE (D)
0170 END-DEFINE
0180 *
```

## Exemplo:

```
0190 READ (10) EMPL BY NAME FROM 'A'  
0200     MOVE EDITED #LEAVE-START-A TO #START-DATE (EM=YYMMDD)  
0210     MOVE EDITED #LEAVE-END-A TO #END-DATE (EM=YYMMDD)  
0220     COMPUTE #LEAVE-DUE = #END-DATE - #START-DATE +1  
0230     DISPLAY NAME PERSONNEL-ID LEAVE-START(1) LEAVE-END(1)  
0240           'LEAVE/DUE' #LEAVE-DUE  
0250 END-READ  
0260 END
```

## Saída:

NAME	PERSONNEL ID	LEAVE START	LEAVE END	LEAVE DUE
ABELLAN	60008339	1998/11/04	1998/11/09	1
ACHIESON	30000231	1998/12/27	1998/12/31	1
ADAM	50005800	1999/08/01	1999/08/31	1
ADKINSON	20009800	1998/02/22	1998/02/23	1
ADKINSON	20012700	1998/01/02	1998/01/03	1
ADKINSON	20013800	1998/01/12	1998/01/12	1
ADKINSON	20019600	1998/01/02	1998/01/03	1
ADKINSON	20008600	1998/01/12	1998/01/12	1
ADKINSON	20005700	1998/01/12	1998/01/12	1
ADKINSON	20011000	1998/01/02	1998/01/03	1



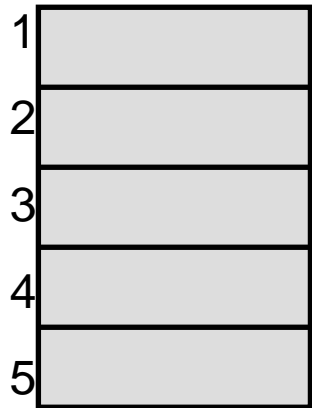
*Arrays* são tabelas multi-dimensionais com dois ou mais dados logicamente relacionados e identificados através de um único nome. *Arrays* podem ser usadas para:

- Armazenar dados para serem processados mais tarde;
- Processar dados numa ordem seqüencial;
- Guardar múltiplos valores em um único campo;
- Auxiliar em cálculos matemáticos.

# Processamento de *Array*

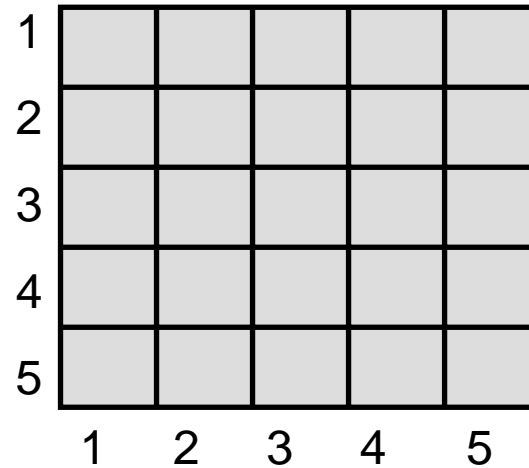
Array de uma-dimensão

#FLD (A15/5)



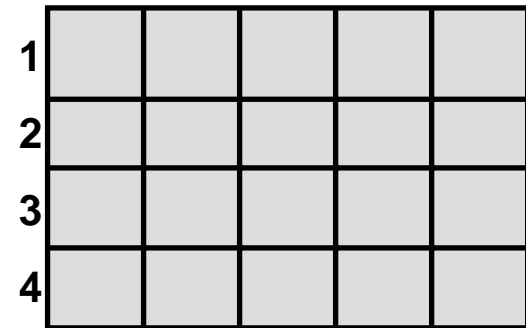
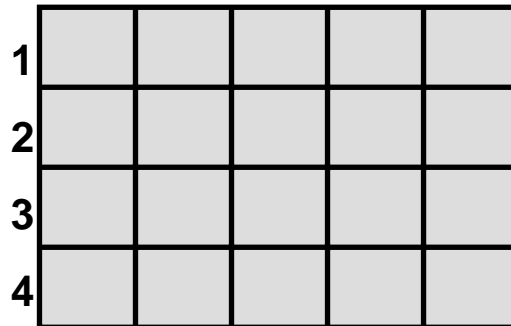
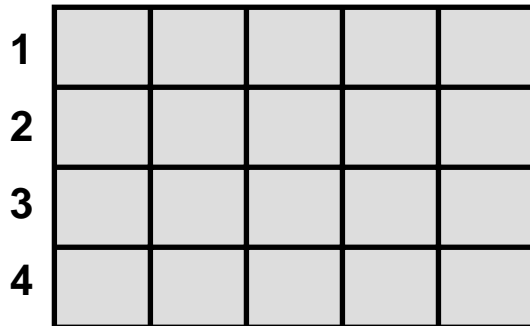
Array de duas-dimensões

#FLD(A15/5,4)



Array de três-dimensões

#FLD(A15/4,5,3)

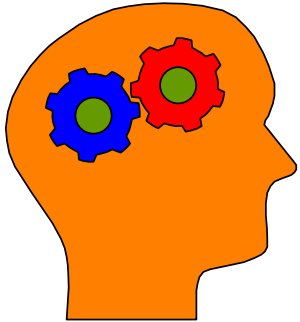


## Índices de *array*:

Índice	Descrição
#ARRAY(1)	1. <sup>a</sup> ocorrência de um <i>array</i> unidimensional..
#ARRAY(7:12)	Byte inicial onde o primeiro <i>string</i> foi encontrado.
#ARRAY(#IND+5)	Ocorrência localizada na posição #IND mais cinco, em um <i>array</i> unidimensional.
#ARRAY(5,3:7)	A 5. <sup>a</sup> ocorrência da 1. <sup>a</sup> dimensão e o intervalo da 3. <sup>a</sup> à 7. <sup>a</sup> da 2. <sup>a</sup> dimensão .

## Índices de *array*:

Índice	Descrição
#ARRAY(5,3:7,1:4)	A 5. <sup>a</sup> ocorrência da 1. <sup>a</sup> dimensão, o intervalo da 3. <sup>a</sup> à 7. <sup>a</sup> da 2. <sup>a</sup> dimensão e o intervalo da 1. <sup>a</sup> à 4. <sup>a</sup> ocorrências da 3. <sup>a</sup> dimensão.
#ARRAY(*)	Todas as ocorrências do <i>array</i> .



### Lembre-se!

- Quando expressões aritméticas são usadas como índices, os operadores “+” e “-” devem ser precedidos e seguidos por um caracter em branco;
- Se um valor constante possui 4 dígitos, coloque-o entre parênteses, pois o Natural interpreta esse número como um número de linha. Portanto um *array* que possui ocorrências com 4 dígitos devem ser precedidos por uma barra “/”. Exemplo:

#ARRAY(/1000) vs. #ARRAY(1000)

Para exibir *arrays* em mapas externos é preciso definir a dimensão, o tamanho e como eles deverão aparecer no mapa.

### 1. Acessar o editor de mapa

- Chamar a função de edição de *array*.

### 2. Determinar o tamanho do *array*

- quantas dimensões ele tem;
- qual o tamanho de cada dimensão

Nota: Se o campo for puxado para dentro do mapa este tamanho já está previamente definido.

### 3. Verificar como o *array* deverá aparecer no mapa

- quantas ocorrências de cada dimensão deverá ser exibida?
- Cada dimensão será exibida horizontalmente ou verticalmente?
- Quantas linhas em branco deverá aparecer entre cada linha?
- Quantos espaços em branco deverá aparecer entre cada campo?

## Definindo *Arrays* em mapas

Quando você quiser exibir apenas uma porção do *array*, use os parâmetros *STARTING FROM* e *NUMBER OF OCURRENCES*.

O objeto que chama o mapa deve fornecer a funcionalidade de exibir o mapa tantas vezes quantos forem necessárias até que todas as ocorrências tenham sido exibidas. Para isso, você precisa definir um processamento de *loop* que mostra o mapa múltiplas vezes. Exemplo:

```
FOR #ITERATION 1 N STEP N  
    INPUT USING MAP '_____  
END-FOR
```



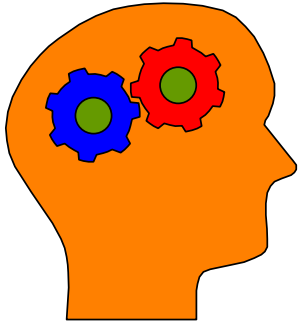
Múltiplos e Periódicos são tratados como *arrays* no Natural. As ocorrências podem ser definidas tanto na área de dados interna como na externa usando o editor de programa ou o editor da área de dados.

O contador de ocorrência binária (BOC) está disponível para uso. Ele contém o número de ocorrências que existem no banco para os campos de valores múltiplos e periódicos. Para ter acesso ao BOC, defina um campo com o mesmo nome do campo múltiplo ou periódico precedido por “C\*”.

Exemplo:

**C\*ADDRESS-LINE** (Esse campo irá conter o número de ocorrências do múltiplo ADDRESS-LINE que existe no banco)

Variável	Função
$C*fieldname$	Retorna o número de valores de um múltiplo e o número de ocorrências do periódico.
$C*fieldname$	Para uso em campo periódico apenas. Usando a notação “n” o Natural irá retornar dentro da nth ocorrência do periódico



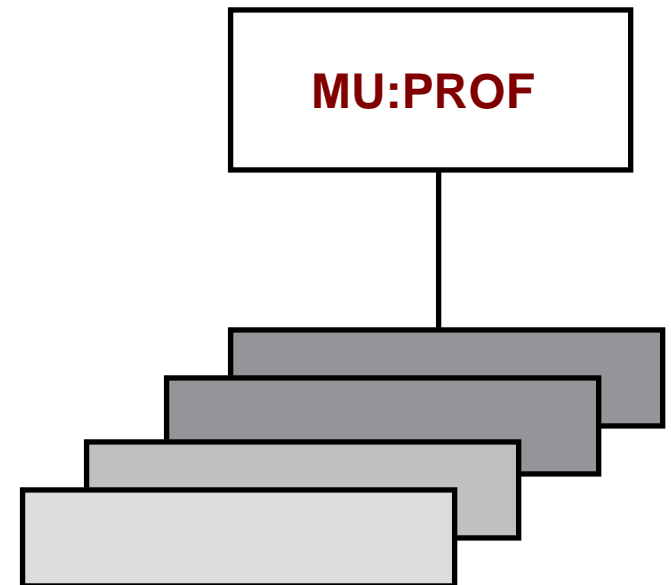
## Lembre-se!

- A variável C\* contém o número e ocorrências que existem no banco, não o número de ocorrências que você escolhe para retornar ao programa.
- Para definir a variável C\* no editor da área de dados, verifique o comando de edição apropriado. Exemplo  
“ \*”  
▪ .

## 1.º Exemplo

As ocorrências do Campo de valor múltiplo PROF. Podem ser referenciadas da seguinte forma:

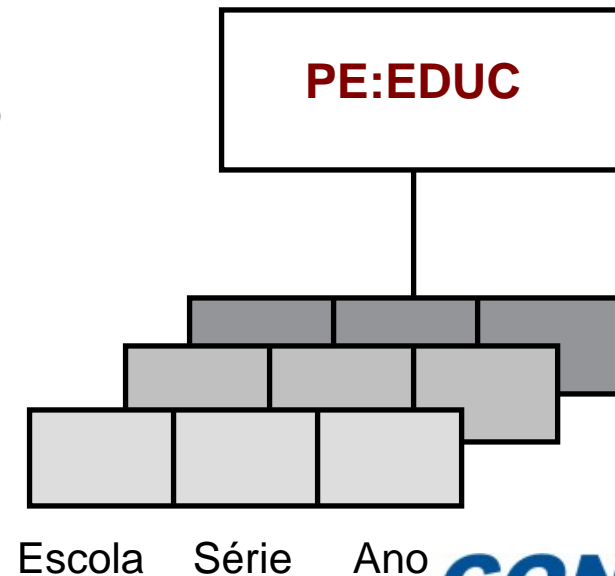
```
DISPLAY NAME PROF (2:4)  
MOVE PROF(#I) TO #HLD  
DISPLAY PROF (#I:#J)
```



## 2.º Exemplo

As ocorrências do Periódico EDUC composta por um grupo de três campos, podem ser referenciadas da seguinte forma:

```
DISPLAY EDUC (1:3)  
DISPLAY SERIE (1:#J) GR-ANO (1:#J)  
ASSIGN #HLD = SERIE (#K)
```



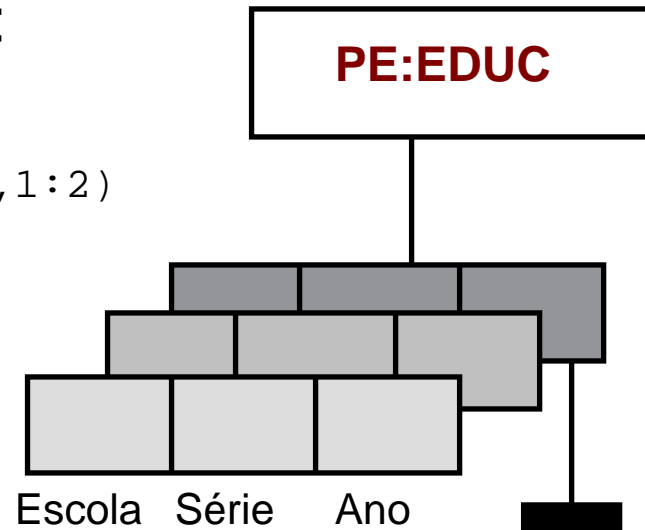
## 3.º Exemplo

Esse exemplo ilustra um campo de valor múltiplo chamado CURSOS contido em um periódico. As ocorrências de um campo múltiplo definido dentro de um periódico podem ser referenciadas da seguinte forma:

```
DISPLAY CURSOS (1,2:4) CURSOS (2,1:2)
```

```
DISPLAY CURSOS (1:3,1:4)
```

```
DISPLAY CURSOS (#I:,* )
```



Cursos

## Exemplo:

```
0010 *****
0020 * ILUSTRA O USO DE CAMPOS MÚLTIPLOS
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMP VIEW OF EMPLOYEES
0060 2 C*INCOME
0070 2 SALARY (5)
0080 2 BONUS (5,5)
0090 2 PERSONNEL-ID
0100 2 NAME
0110 1 #TOTAL-INCOME (P12)
0120 1 #NO-OF-SALARIES (N2)
0130 END-DEFINE
0140 READ (15) EMP BY NAME STARTING FROM 'G'
0150 #NO-OF-SALARIES := C*INCOME
0160 ADD SALARY(*) TO #TOTAL-INCOME
0170 NEWPAGE LESS THAN #NO-OF-SALARIES
0180 DISPLAY NOTITLE (SF=3)
0190 8T 'EMPLOYEE/ID' PERSONNEL-ID 'EMPLOYEE/NAME' NAME
```

## Exemplo:

```
0200    'SALARY/HISTORY' SALARY(1:#NO-OF-SALARIES) (EM=ZZZ,ZZZ,ZZ9 HC=R)
0210    'TOTAL/INCOME' #TOTAL-INCOME (EM=ZZZ,ZZZ,ZZ9 HC=R)
0220    SKIP 1
0230    RESET #TOTAL-INCOME
0240    END-READ
0250    END
```



## Unidade A - Definição e Implementação de Mapas

- Construindo uma interface para o usuário
- Conceitos sobre mapas
- Layouts
- Formulários

### Por que criar uma *interface* padrão para o usuário?

Cada vez mais as organizações estão se voltando pra a criação de uma interface consistente como padrão para todas as suas aplicações, pois, ela resolve vários problemas, tanto para o programador como para o usuário. E ssa padronização é conhecida como *commom user access* (CUA).

### ***Common User Access* - Benefícios**

Um *layout* uniforme, com um menu que direciona as opções dos usuários, oferece uma maneira consistente de arranjar e apresentar as informações.

Navegação uniforme para guiar o usuário através do sistema. Desde que o usuário possa contar com os mesmos comandos em qualquer tela, essa uniformização torna o sistema fácil de ser aprendido e usado.

Uniformização dos serviços (como o Help) também facilita o uso do sistema, pois há uma pequena possibilidade de aprendizado envolvendo o uso de novas funções.

## Linhas Gerais para o desenvolvimento de *Interfaces*

Se você estiver criando uma *interface* para uma aplicação baseada em caracter no *mainframe* ou para uma aplicação gráfica numa estação de trabalho, os usuários, geralmente querem as mesmas coisas em suas *interfaces*. Considere os seguintes objetivos ao criar uma *interface* para o usuário:

Objetivo	Propósito
Controle do usuário	O usuário deveria sempre estar seguro do controle da interface, e não o contrário. Isso significa que você deveria apresentar várias opções de input e fornecer caminhos com rotas de saída.

# Construindo uma *interface* padrão para o usuário

Objetivo	Propósito
Controle do usuário	O usuário deveria sempre estar seguro do controle da interface, e não o contrário. Isso significa que você deveria apresentar várias opções de <i>input</i> e fornecer caminhos com rotas de saída.
Consistência	Uma vez que os usuários aprendem uma interface, eles são capazes de aplicar as técnicas de navegação e a terminologia que já conhecem para cada aplicação nova. Adote padrões que sejam aderentes dentro de toda aplicação.
Atratividade	Dedique uma atenção especial à estética assim como uma boa tela e desenvolvimento gráfico.

## Construindo uma *interface* padrão para o usuário

Objetivo	Propósito
Feedback	Os usuários deveriam sempre receber imediatamente mensagens de retorno de suas ações. Eles não deveriam e questionar sobre o que fizeram de errado.
Recall	Evite rechamadas. Os seres humanos podem lembrar-se de um número grande de informações, mas uma interface de fácil uso não deveria solicitá-las. Controle as informações em trechos manipuláveis. Use elementos de reconhecimento do <i>prompt</i> .
Esquecimentos	Usuários cometem erros. Forneça opções que lhes permitam facilmente reverter suas ações.

## Programação Interativa

Os mapas Natural permitem aos usuários comunicar-se com os programas. Um programa iterativo controla o mapa para que ele possa enviar e obter informações do usuário. A maneira do Natural fornecer essa iteração se dá pelo uso da declaração *INPUT*.

## Especificações do Mapa

- O tamanho máximo da página (n.º de linhas) é 250, e o tamanho da linha (n.º de colunas) deve compreender 5-249;
- Todo campo do mapa deve ter nome e esses nomes devem corresponder aos nomes de campos usados no programa chamador;
- Os campos exibem atributos que podem ser sobrescritos usando as variáveis de controle;
- O nível do campo e o nível do mapa podem ser incorporados dentro do mapa;
- As regras de processamento podem ser definidas em seu mapa, embora a maioria prefiram definir nos programas.



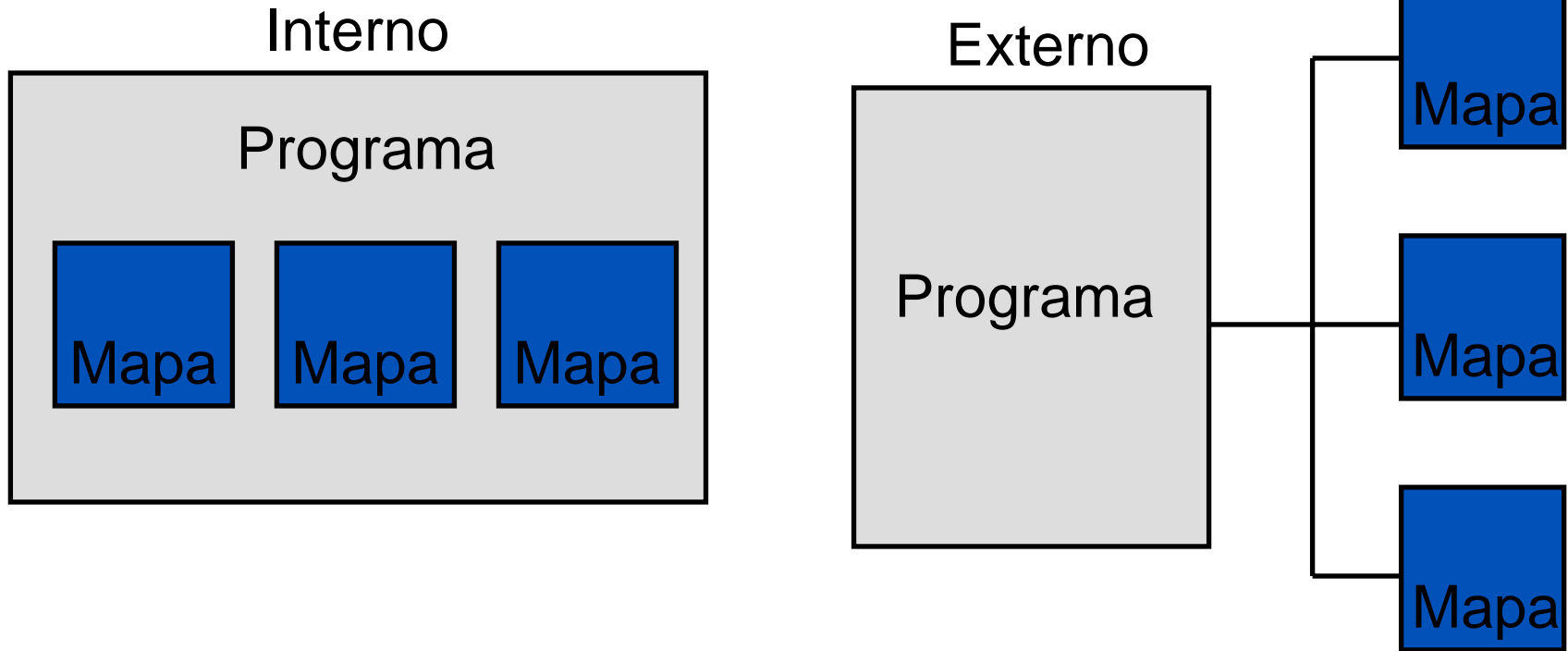
## Tipos de Mapa

- Interno

Os mapas internos são definidos dentro do programa usando-se o editor de programa e são utilizados unicamente por esse programa em particular.

- Externo

Como a área de dados externo, esses programas podem ser usados por diferentes programas. Esses objetos são definidos fora do programa usando-se o editor de mapa.



# Exemplo de Mapa Externo

DATA: (XXXXXXXXXX)

HORA: (XXXXXXXXXX)

BIBLIOTECA: (XXXXXXXXXX)

MAPA: (XXXXXXXXXX)

SISTEMA DE WORKSHOP  
MENU PRINCIPAL

---

- A - INCLUIR UM NOVO REGISTRO
- B - ATUALIZAR UM REGISTRO
- C - EXCLUIR UM REGISTRO
- D - EXIBIR LISTA DE ENDERECO
- E - TERMINAR

OPCAO: )X

ID DO FUNCIONARIO: )XXXXXXXX (OBRIGATORIO PARA AS OPCOES A, B E C)

(XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX)

---

PF3 - ENCERRA

- **Mapa Interno**

Para criar um mapa interno você vai utilizar a instrução `INPUT`. Todos os campos definidos na declaração `INPUT` são são campos de entrada, por *default*.

Se você quiser mudar o tipo do campo para *output* ou *modifiable*, use as definições de atributos (AD) para aquele campo. Essas definições permitem que você defina a função e aparência dos campos no mapa.

Atributo	Variações Disponíveis
Representação do Campo	B = Blinking (flash on and off) I = Intensified (Negrito) N = No display D = Default
Alinhamento do Campo	L = left-justified (default para campos alfanuméricos) R = right-justified (default para campos alfanuméricos) Z = Zero print (imprime zeros de campos numéricos )
Tipo de Campo	A = Campo de input (usuários podem digitar informações) M = Campo Modifiable (usuários podem alterar as informações que o Natural irá apresentar) O = Campo Output (usuários não podem alterar as informações apresentadas pelo Natural).

Atributo	Variações Disponíveis
Tamanho das letras no Campo	T = Transforma tudo no campo para maiúscula W = Transforma tudo no campo para minúscula
Caracteres de Preenchimento	'c' = Qualquer caracter que você escolher para preencher o campo (exemplo: '_'). O default é o espaço em branco.

### Posicionamento do Cursor no Mapa

Quando um mapa é gerado o cursor é automaticamente colocado no primeiro campo de *input* ou modificável.

Entretanto, ocorrerá certas situações em que o mapa poderá solicitar o posicionamento do cursor, não no primeiro mas em qualquer outro campo do mapa.

A clausula *MARK* das declarações *INPUT*(e *REINPUT*) permitem que você posicione o cursor. Isto pode ser feito usando:

- Nome de campo (precedido de um '\*');
- Variável numérica;
- Constante numérica.

### Opção *MARK POSITION*

Você não apenas pode determinar o lugar em que seu cursor irá aparecer como pode também definir que o cursor irá ser colocado em uma posição particular para o valor daquele campo.

Essa característica pode ser usada quando você gostaria que o usuário modificasse, por exemplo, o valor do item dia do campo data 11/25/1996.

```
MARK POSITION 4 IN *#DATE (AD=MM/DD/YYYY)
```



# Conceitos sobre Mapas

```
0010 *****
0020 * ILUSTRA O POSICIONAMENTO DO CURSOR NO MAP
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 #MARCA-INICIAL (A20)
0070 1 #MARCA-FINAL (A20)
0080 1 CARS VIEW OF VEHICLES
0090 2 MAKE
0100 2 MODEL
0110 2 YEAR
0120 END-DEFINE
0130 *
0140 INPUT MARK *#MARCA-FINAL /////  
0150 7T 'ENTRE COM A MARCA INICIAL E FINAL PARA A LEITURA DO ARQUIVO '  
0160 // 17T 'MARCA INICIAL:' #MARCA-INICIAL (AD=AIT'_' )  
0170 // 19T 'MARCA FINAL :' #MARCA-FINAL (AD=AIT'_' )  
0180 *  
0190 READ CARS BY MAKE STARTING FROM #MARCA-INICIAL ENDING AT #MARCA-FINAL  
0200 DISPLAY MAKE MODEL YEAR  
0210 END-READ  
0220 *  
0230 WRITE / 10T 'O RELATORIO ESTA COMPLETO '  
0240 *  
0250 END
```

ENTRE COM A MARCA INICIAL E FINAL PARA A LEITURA DO ARQUIVO

MARCA INICIAL: \_\_\_\_\_

MARCA FINAL: datsun\_\_\_\_\_



Cursor

# Conceitos sobre Mapas

MAKE	MODEL	YEAR
-----	-----	-----
ALFA ROMEO	GIULIETTA 2.0	1985
ALFA ROMEO	SPRINT GRAND PRIX	1984
ALFA ROMEO	QUADRIFOGLIO	1984
AMERICAN MOTOR	HORNET	1977
AMERICAN MOTOR	AMBASSADOR	1983
AMERICAN MOTOR	HORNET	1977
AUDI	QUATRO	1983
AUDI	QUATRO TURBO	1983
AUDI	QUATRO TURBO	1986
AUDI	100 CD	1982
AUDI	80 S	1980
...	...	...
DATSUN	300ZX	1986
DATSUN	CHERRY	1983
DATSUN	SUNNY	1981

O RELATORIO ESTA COMPLETO

- **Mapas Externos**

Os mapas externos são criados no editor de mapa e são invocados no programa através da declaração *INPUT USING MAP*.

Iremos mostrar 2 tipos de mapas. Um é simples e o outro usa várias opções em sua declaração.

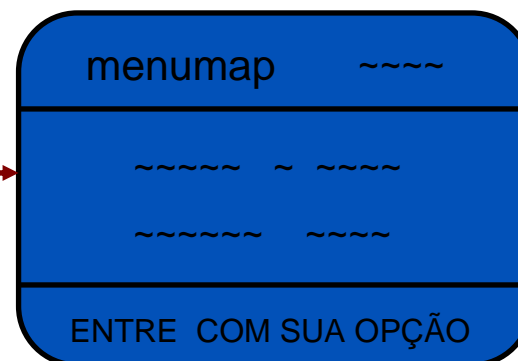
# Chamando um mapa externo

## Programa

```
...  
/*Exemplo 1  
INPUT USING MAP 'MENU MAP'  
...
```

```
...  
/*Exemplo 2  
INPUT WITH TEXT  
. 'ENTRE COM SUA OPÇÃO'  
. 'MARK *#CHOICE ALARM  
. USING MAP 'MENU MAP'  
.   
END
```

## Mapa Externo



### Definições do Perfil do Mapa

Para cada mapa externo que você cria, você pode mudar várias definições que alteram sua aparência e seu comportamento. Por exemplo, você pode controlar o tamanho do mapa, se as teclas de função irão aparecer ou não, etc.

Quando você inicia um novo mapa, estas definições estão com seus valores padrões, mas, você pode sobrescrevê-los.

### Que tipos de definições existem

#### *Format*

As definições de formato permitem que você determine como seu mapa será formatado. Qual o tamanho de seu mapa? Que PF-keys serão usadas? etc.

#### *Context*

Permitem dizer como o mapa está sendo usado, se é permitido usar características especiais para a tela como o pisca-pisca ou vídeo reverso, se o *help* está definido a nível de mapa, etc.

### Que tipos de definições existem

#### Caracter de Preenchimento

Um caracter de preenchimento padrão pode ser definido. Essas definições podem ser sobrescritas por outro caracter utilizando o editor de campo estendido.

#### Delimitadores (em alguns ambientes apenas)

Em alguns ambientes os delimitadores são usados para atribuir inicialmente certos atributos dos campos ou textos como cor, por exemplo.



### Que tipos de definições existem

Delimitadores (em alguns ambientes apenas)

Qualquer caracter especial pode ser definido como delimitador, exceto o caracter de controle e de notação de ponto decimal. Eles aparecem sempre na primeira posição do campo no mapa externo.

Obs.: Nos ambientes nos quais os delimitadores não são usados, costuma-se usar o parâmetro AD para definir os atributos de um campo.

### Definindo campos no mapa externo

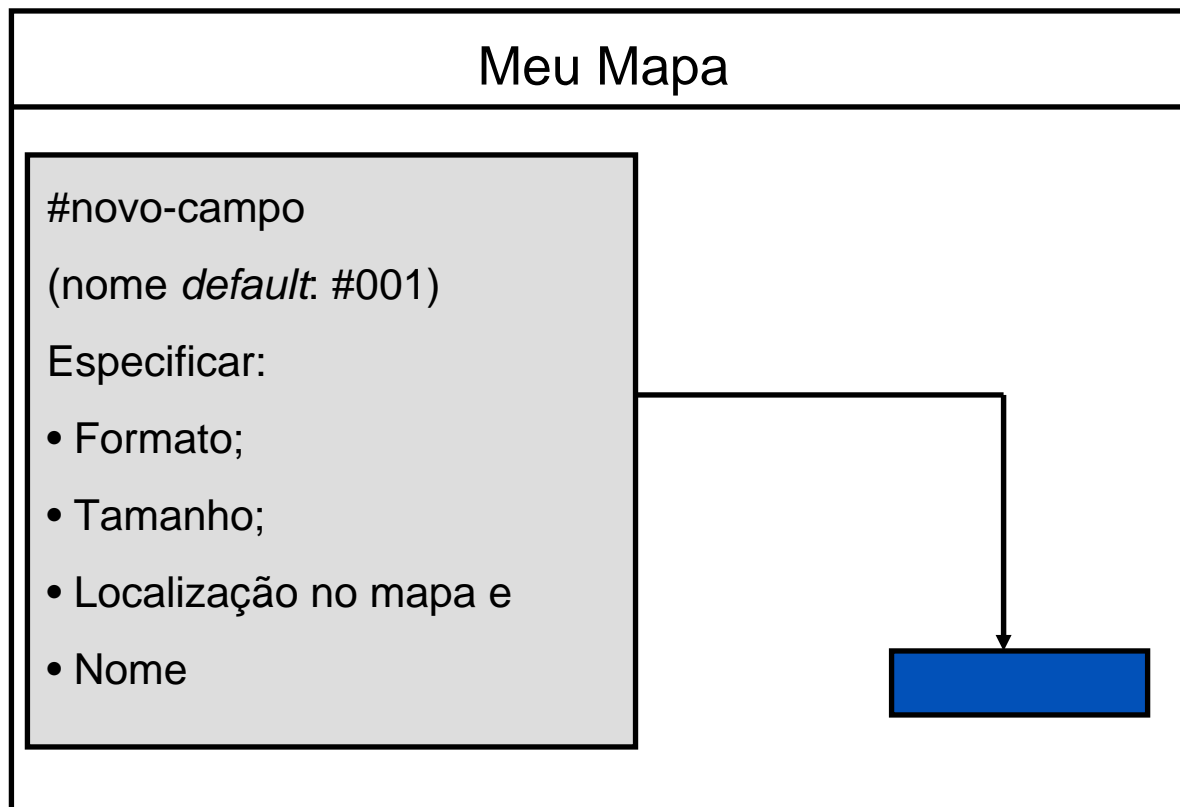
- Método I

É possível definir um novo campo no mapa diretamente indicando o delimitador, o formato e o tamanho e posteriormente o nome.

- Método II

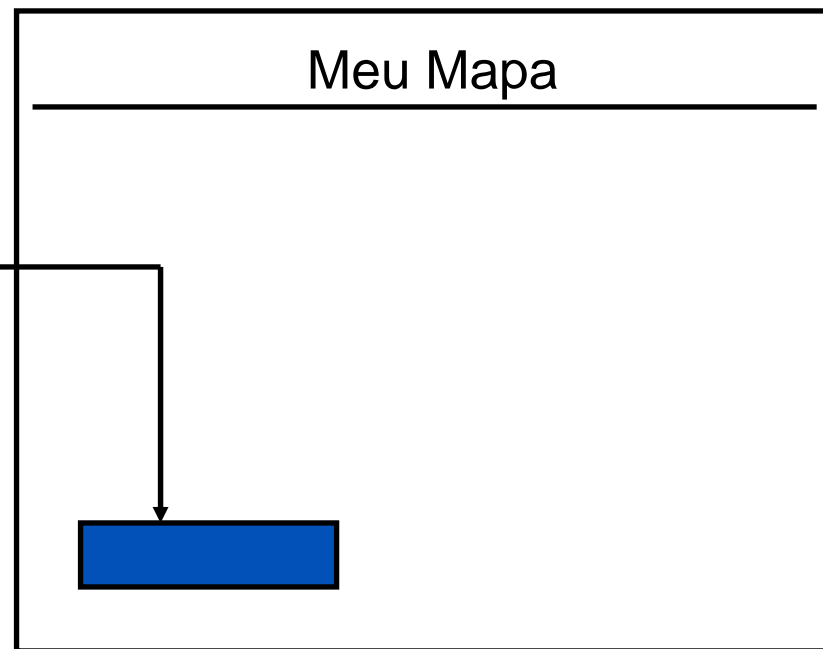
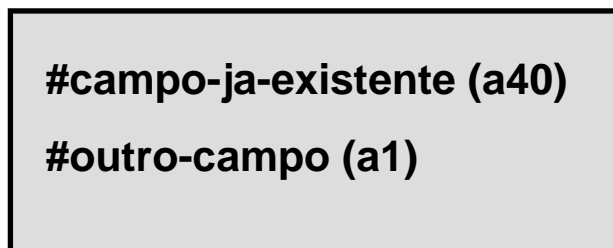
Uma outra forma de definir um campo no mapa é puxá-lo de uma área de dados ou da DDM. Nesse caso, o nome, formato e o tamanho são definidos automaticamente.

## Método I



## Método II

Definição de Dados existente



### Atribuindo nome aos campos

Antes de catalogar o mapa, todos os campos no seu mapa devem ser nomeados. Se os campos não foram puxados de nenhuma área de dados, elas recebem nomes temporários (ex.: #001, #002, #003, etc). Esses nomes devem ser alterados antes do mapa ser catalogado.

Todos os campos definidos no seu mapa externo devem corresponder àqueles definidos no programa chamador, bem como os respectivos formatos e tamanhos.

Para assegurar que todos os seus mapas tenham a mesma aparência, você pode optar por usar uma tela de *layout* padronizada. Os dados localizados no mesmo lugar nas diversas telas melhora a eficiência de seu processamento.

### Tipos de *Layout*

Estático: Serve apenas como um ponto inicial para a criação de um novo mapa. Você usa o mapa de *layout* como um “modelo” para a definição de novos mapas. As mudanças realizadas no *layout* não têm efeito sobre o novo mapa.

Dinâmico: Ele permite que você tenha um modelo consistente e modifique facilmente o *layout* de tal forma que as alterações efetuadas no *layout* sejam repassadas para o outros mapas da sua aplicação.

## Procedimento para criar um mapa de *Layout*

- Crie o mapa de *layout*;
- Crie um novo mapa;
- Especifique o nome o mapa de *layout* na tela de *profile* do novo mapa;
- Adicione os novos campos, textos e os campos definidos pelo usuário do novo mapa;
- catalogue o mapa.



### O que são formulários (*Forms*)

São mapas que não têm campos de *input* ou campos modificáveis. São criados no editor do mapa como uma alternativa para compor declarações complexas com o *WRITE*.

Para identificar o mapa como um formulário, você deve chamar o formulário através da declaração *WRITE USING FORM*.

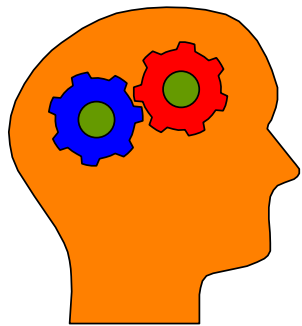
Dessa forma:

- As declarações *WRITE* são criadas nos bastidores;
- Se estiver trabalhando numa plataforma que usa delimitadores, eles devem estar definidos para aceitar apenas campos do tipo output.

## Chamando um formulários

Uma vez criado, para chamar um formulário a partir de outro objeto, basta usar a declaração *WRITE USING FORM* seguido do nome do formulário.

Múltiplos formulários serão escritos para a mesma página ou tela.



**Você pode usar regras de processamento no formulário mas, Lembre-se!**

- A regra não pode usar a declaração *REINPUT* ;
- As regras de PF nunca são processadas;
- Todas as regras são processadas depois que o formulário é escrito e antes do controle ser retornado ao objeto chamador.

## Programa com a instrução *WRITE USING FORM*

```
DEFINE DATA
...
END-DEFINE
...
WRITE USING FORM 'HELLO1'
...
END
```



```
Hello Jeff Smith
 Bem vindo ao Building Natural Applications!
 Esperamos que ache esse curso uma experiência educacional gratificante.
```

## Unidade B - Editando campos do Mapa

- Editando campos do Mapa
- Variáveis de Controle

### Características do campo

Ao criar um mapa, você pode definir todas as características que você deseja para um campo. Muitas características estão associada a definição de algum parâmetro. Segue na tabela abaixo alguns desses parâmetros.

<b>Característica</b>	<b>Parâmetro</b>	<b>Propósito</b>
Nome do campo	s/p	Nome do campo. Qualificador de alto nível incluído para campos do banco.
Tamanho e Formato	s/p	Formato e tamanho. Só podem ser alterados para campos criados originalmente no mapa.

<b>Característica</b>	<b>Parâmetro</b>	<b>Propósito</b>
Definição de Atributos	AD	Exibe a função de seu campo( ex.: input/output/modifiable; required/optional) e como o campo será exibido (ex.: intensificado, com caracter de preenchimento).
<b>Help em nível de campo e variáveis de controle</b>		
Variável de Controle	CV	Você pode ligar uma variável de controle em nível de campo usando esse parâmetro
Help	HE	Você pode ligar um mapa de help ou uma helproutine em nível de campo usando esse parâmetro.

<b>Característica</b>	<b>Parâmetro</b>	<b>Propósito</b>
Help	HE	Entre com “HE=+” abre uma janela para que você defina até 20 parâmetros para serem passados para a helproutine.
<b>Parâmetros Especiais</b>		
Atributos de String Dinâmico	DY	Usado para definir certas características contidas num texto para controlar a definição de atributos.
<b>Campos informativos não-modificáveis</b>		
Número de regras de processamento	s/p	Número atual de regras de processamento definidas para o campo

Característica	Parâmetro	Propósito
Modo de Definição	s/p	Indica como o campo foi definido:  DATA = selecionados a partir da DEFINE DATA;  SYS = variável de sistema;  UNDEF = criada no mapa e assinalada com um nome dummy;  USER = criada no mapa e definida através do editor do campo;  VIEW = selecionada a partir da DDM.



## Controlando a exibição do campo

Tamanho do campo de exibição	AL NL FL	Altera o tamanho de exibição dos campos mostrando somente as primeiras posições definidas pelo parâmetro.
Máscara de edição	EM	Edita especificações de máscaras dependendo do formato do campo.
Definição de cor	CD	Campos podem ser exibidos com qualquer cor suportada pelo ambiente.
Impressão de zeros	ZP	Determina se os zeros serão impressos para campos numéricos com dados nulos

## Controlando a exibição do campo

Posição do Sinal	SG	Determina se é alocada ou não uma posição extra no início do campo para incluir o seu sinal.
Modo de Impressão	PM	Permite alternar definições de caracter e a direção de impressão utilizadas.

### Definições de atributo para campo dinâmico

Os mapas do Natural têm a intenção de interagir com o usuário da forma mais eficiente possível. Para que o usuário possa entender melhor a utilização de cada mapa você pode ligar variáveis de controle e usá-las junto com as instruções *DISPLAY*, *INPUT*, *PRINT* e *WRITE*.

Um campo com formato C pode ser usado para definir delimitadores dinamicamente. Dessa forma você poderá modificar as características desses campos.

### Possíveis usos das variáveis de controle

- Enfatizar mensagens de erro ou outras mensagens importantes;
- Permitir que um único mapa seja usado para múltiplos propósitos;
- Podem ser usadas com *arrays* em campos de seleção para tornar invisíveis as ocorrências dos campos que não têm dados a serem selecionados;
- Verificar se o conteúdo de um campo que recebeu atributos dinamicamente foi modificado durante a execução de uma instrução *INPUT*.

## Categoria de delimitadores

### Representação do campo (Parâmetro AD)

B	<i>Blinking</i>
C	<i>Cursive/Italic</i>
D	<i>Default intensity</i>
I	<i>Intensified</i>
N	<i>Non-display</i>
U	<i>Underlined</i>
V	<i>Reverse video</i>
Y	<i>Dynamic attributes</i>

### Características de Input/Output do campo (Parâmetro AD)

P	<i>Temporarily protected</i>
---	------------------------------

## Categoria de delimitadores

### Exibição de cor (Parâmetro CD)

BL	<i>Blue</i>
GR	<i>Green</i>
NE	<i>Neutral</i>
PI	<i>Pink</i>
RE	<i>Red</i>
TU	<i>Turquoise</i>
YE	<i>Yellow</i>

### Usando variáveis de controle em nível de campo e mapa

A variável de controle tem formato C e não tem tamanho. Elas devem ser definidas e receber atributos antes da chamada do mapa.

Passo	Ação
1	No objeto que chama o mapa, defina a variável de controle, por exemplo: #CNTL (C) Antes de chamar o mapa, mova ou defina atributos à variável. MOVE (AD=B CD=YE) TO #CNTL

Passo	Ação
2	<p>Use o parâmetro <i>control variable</i> para ligar a variável de controle em nível de campo ou mapa.</p> <p><b>Nível de Campo:</b> esse parâmetro encontra-se na tela de edição do campo. Ela sobrepõe a definição em nível de mapa, se essa opção foi usada.</p> <p><b>Nível de mapa:</b> esse parâmetro está na tela de <i>profile</i> do mapa (PF2).</p>
3	<p>Verifique que o atributo Y (AD=Y) foi definido para cada campo a ser controlado pela variável de controle. Em algumas plataformas um delimitador pode ser usado para esse fim.</p>



# Variáveis de Controle

```
0010 *****
0020 * ILUSTRACAO DO USO DA VARIAVEL DE CONTROLE
0030 *****
0040 DEFINE DATA
0050 GLOBAL USING DIAGDA
0060 LOCAL
0070 1 #LANME      (A20)
0080 1 #OPTION     (A1)
0090 1 #CTLVAR1   (C)          /* NIVEL DE MAPA
0100 1 #CTLVAR2   (C)          /* NIVEL DE CAMPO
0110 1 #MESSAGE  (A60)
0120 END-DEFINE
0130 REPEAT
0140     INPUT
0150         ////'POR FAVOR ENTRE COM O ULTIMO NOME: ==> ' #LNAME (AD=AILT'_' )
0160         /'OU ENTRE COM A PALAVRA "QUIT" PARA SAIR' (CD=RE)
0170     IF #LNAME = ' '
0180         REINPUT 'OPCAO INVALIDA! ENTRE COM O NOME OU "QUIT"' MARK *#LNAME
0190     END-IF
0200     IF #LNAME='QUIT'
```

# Variáveis de Controle

```
0210      'WRITE NOTITLE 10/6 'VOCE SOLICITOU O FIM DA SESSAO...' *USER
0220      '....' 6T 'TENHA UM BOM DIA!'
0230 STOP
0240 END-IF
0250 F1. FIND(1) EMPL-VIEW WITH NAME=#LNAME
0260 INPUT USING MAP 'CNTLMAP1'
0270      DECIDE ON FIRST VALUE OF #OPTION
0280      VALUE 'Q'
0290          ESCAPE BOTTOM
0300      VALUE 'U'
0310          UPDATE (F1.)
0320      END OF TRANSACTION
0330      MOVE 'UPDATE DONE' TO #MESSAGE
0340          MOVE (CD=RE AD=P) TO #CTLVAR2
0350      VALUE 'D'
0360          DELETE (F1.)
0370      END OF TRANSACTION
0380      MOVE 'DELETE DONE' TO #MESSAGE
0390          MOVE (CD=NE AD=P) TO #CTLVAR2
0400 NONE
```

# Variáveis de Controle

```
0410 REINPUT 'OS VALORES CORRETOS SAO D(DELETE), U(UPDATE), Q(QUIT)'  
0420     MARK *#OPTION  
0430     END-DECIDE  
0440     MOVE (AD=P) TO #CTLVAR1  
0450     INPUT USING MAP 'CNTLMAP1'  
0460     RESET EMPL-VIEW #CTLVAR2 #OPTION #MESSAGE  
0470     END-FIND  
0480     END-REPEAT  
0490     END
```

## Unidade C - Validando as entradas do mapa

- Instrução REINPUT
- Regras de processamento
- Regras de processamento internas
- Reutilização das regras internas
- Resumo

### EXIBINDO MENSAGENS DE ERRO

Se seu programa testa a entrada de dados em um campo do mapa e um erro é detectado, este erro precisa ser comunicado ao usuário. Você pode fazer isso utilizando a instrução *REINPUT*.

### A CLAUSULA *MARK*

Você pode usar a clausula *MARK* com a instrução *REINPUT* para posicionar o cursor no campo em que o erro ocorreu.

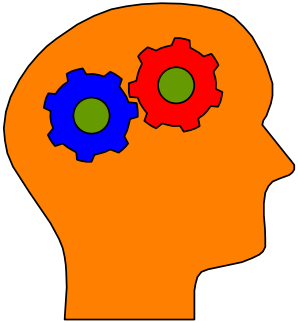
Além disso, você tem a opção de exibir as mensagens de erro com cores diferentes ou em negrito.

## EXIBINDO MENSAGENS DE ERRO

```
0010 *****
0020 * EXEMPLO DA DECLARAÇÃO REINPUT
0030 *****
0040 DEFINE DATA LOCAL
0050 1 #LNAME (A20)
0060 END-DEFINE
0170 INPUT //// 'PLEASE ENTER A LAST NAME' #LNAME (AD=AIT'_' )
0180          / 'OR ENTER THE WORD' '"QUIT"' (CD=RE)
0190 IF #LANME = ' '
0200     REINPUT 'PLEASE ENTER A LAST NAME OR "QUIT" ' MARK *#LNAME
0210 END-IF
0240 END
```

### Saída:

```
PLEASE ENTER A LAST NAME OR 'QUIT'
PLEASE ENTER A LAST NAME_____
OR ENTER THE WORD 'QUIT'
```



## Lembre-se!

- Nenhuma instrução *WRITE* ou *DISPLAY* pode ser executada entre as declarações *INPUT* e *REINPUT*.
- A instrução *REINPUT* pode ser usada em mapas internos e externos, porém seu uso é mais eficiente quando codificada em mapas externos.
- Em virtude da instrução *REINPUT* retornar ao último *INPUT*, todas as regras de processamento no mapa são re-executadas.

### ***O que são regras de processamento?***

Trata-se de uma edição ligada ao campo no mapa externo. Elas são usadas para avaliar o conteúdo do campo e para tratar os valores da variável de sistema *\*PF-KEY*. Quando o mapa é catalogado, elas passam a fazer parte do código do mapa.



## Exemplo

```
0010 IF #END = FALSE
0020     IF & = 'D' OR = 'U' OR = 'Q'
0030         IGNORE
0040     ELSE
0050         REINPUT 'CORRECT VALUE ARE "D" = DELETE, "U" = UPDATE, "Q" = QUIT'
0060         (AD=I) MARK *&
0070     END-IF
0080 END-IF
```

## Vantagens

- É possível testar o mapa e toda lógica associada a ele antes da criação do objeto chamador;
- Ajuda a executar sua aplicação de forma mais eficiente;
- As regras de processamento podem ser compartilhada entre outros programas e aplicações. Isso pode ser feito com o auxílio do *Predict*, que fornece 3 tipos de regras:
  - *Inline rules* (regras internas)
  - *Free rules* (regras livres)
  - *Automatic rules* (regras automáticas)

- **Regras *Inline***

Elas são definidas diretamente no mapa para serem usadas somente por ele. Esse tipo de regra não precisa estar documentada no *Predict* e outros mapas na aplicação não precisam acessá-la.

- **Regras *Free***

São criadas para serem usadas em vários mapas. Como forma de centralizar as informações são documentadas no *Predict*. Para ser usada por mais de um mapa deve ser criada de forma geral e em virtude desse compartilhamento não deve ser modificada antes que seus efeitos sejam avaliados nos outros mapas.

- **Regras Automáticas**

São definidas para campos do banco dentro da DDM pelo DBA ou pelo administrador do banco. Elas são automaticamente usadas sempre que campos do banco são usados no mapa. Elas são criadas no *Predict* e não podem ser alteradas em nível de mapa.

## Níveis de Processamento

Um único campo pode ter até 100 regras. A ordem do processamento é feita atribuindo níveis para cada regra. As regras são processadas na ordem ascendente do nível 0 ao 99 e de acordo com a posição do campo no mapa - da esquerda para a direita e de cima para baixo. As regras associadas com as \**PF-KEY* são processadas primeiro. Elas agem como se fossem o primeiro campo do mapa.

## Definindo os níveis da regra

Se suas regras não são executadas na ordem desejada, você pode mudar a ordem de execução alterando o seu nível.

Rank	Regra de processamento
0	Regra de Finalização
1-4	Regras Automáticas
5-24	Checagem de formato
25-44	Checagem de valor para campos individuais
45-64	Checagem de valores entre campos

# Regras de Processamento

Rank	Regra de processamento
65-84	Acesso ao banco
85-99	Propósitos especiais

### Criando Regras Internas

O método usado para criar regras variam de acordo com a plataforma operacional. No *mainframe* basta entrar com .p sobre o delimitador do campo ao qual deseja-se associar a regra.

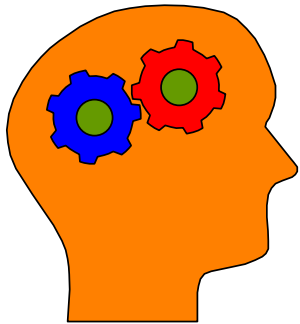
### Notação &

Ao editar a regra você pode utilizar o caracter “&” para substituir o nome do campo. Essa opção possibilita a padronização de regras, permitindo que a mesma regra seja usada em diferentes campos com o mesmo propósito.



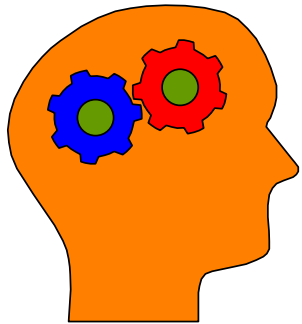
## Exemplo

```
0010 IF #END = FALSE
0020     IF & = 'D' OR = 'U' OR = 'Q'
0030         IGNORE
0040     ELSE
0050         REINPUT
0060         'CORRECT VALUES ARE "D" = DELETE, "U" = UPDATE, "Q" = QUIT'
0070         (AD=I) MARK *&
0080     END-IF
0090 END-IF
```



## Lembre-se!

- A declaração *END* não é permitida dentro das regras de processamento;
- Referências a números de linha não são permitidas dentro das regras de processamento;
- Você pode checar, testar ou salvar uma regra de processamento.



### Lembre-se!

- Evite a execução de objetos externos a partir de regras de processamento (ex.: uso do *PERFORM* ou *CALLNAT*) pois há uma enorme possibilidade do controle ser passado para fora do mapa antes da validação dos campos de entrada;
- Regras de processamento podem conter a declaração *DEFINE DATA*, pois você pode precisar usar uma variável que não está definida no mapa ou uma *view* e seus campos.

## Unidade D - Definindo *Function Keys*

- Interface da *Function Key*
- Instrução *SET KEY*

Você pode tornar seus programas mais amigáveis atribuindo-lhes processos a serem executados quando uma determinada função chave é pressionada. Entre essas funções, podemos incluir:

- *Program attention keys 1-3* (PA 1-3)
- *Program function keys 1-24* (PF 1-24)
- *ENTER key* (ENTR)
- *CLEAR key* (CLR)
- *Ligth pen* (PEN)

Há 4 passos que envolvem a adição de funções em sua aplicação:

1. No editor do mapa, diga que você quer exibir as funções chaves;
2. Adicione uma regra de processamento à variável de sistema \*PF-KEY para executar qualquer ação necessária;
3. Ative as chaves programaticamente. Use a declaração SET KEY no objeto. Você pode ainda definir um nome para essa função;
4. Codifique seu programa usando as variáveis \*PF-KEY ou PF-NAME.

### Variáveis de sistema

\*PF-KEY - Contém a identificação da última função que foi pressionada. Seu formato e tamanho é A4 e o conteúdo não pode ser modificado.

\*PF-NAME - Contém a identificação da última função que foi pressionada cuja denominação é dada através da cláusula *NAMED*. Seu formato e tamanho é A10.

## Instrução SET KEY

Essa instrução assinala funções para as PF, PA, *ENTER* e *CLEAR*. As funções que podem ser definidas para essas chaves, incluem:

- Interrogação pelo programa ativo;
- Nome do programa e/ou do comando;
- Comandos de terminal.

### Sintaxe:

```
SET KEY
```

```
PFn = KEYWORD
```

```
PFn NAMED 'name'
```



## Instrução SET KEY

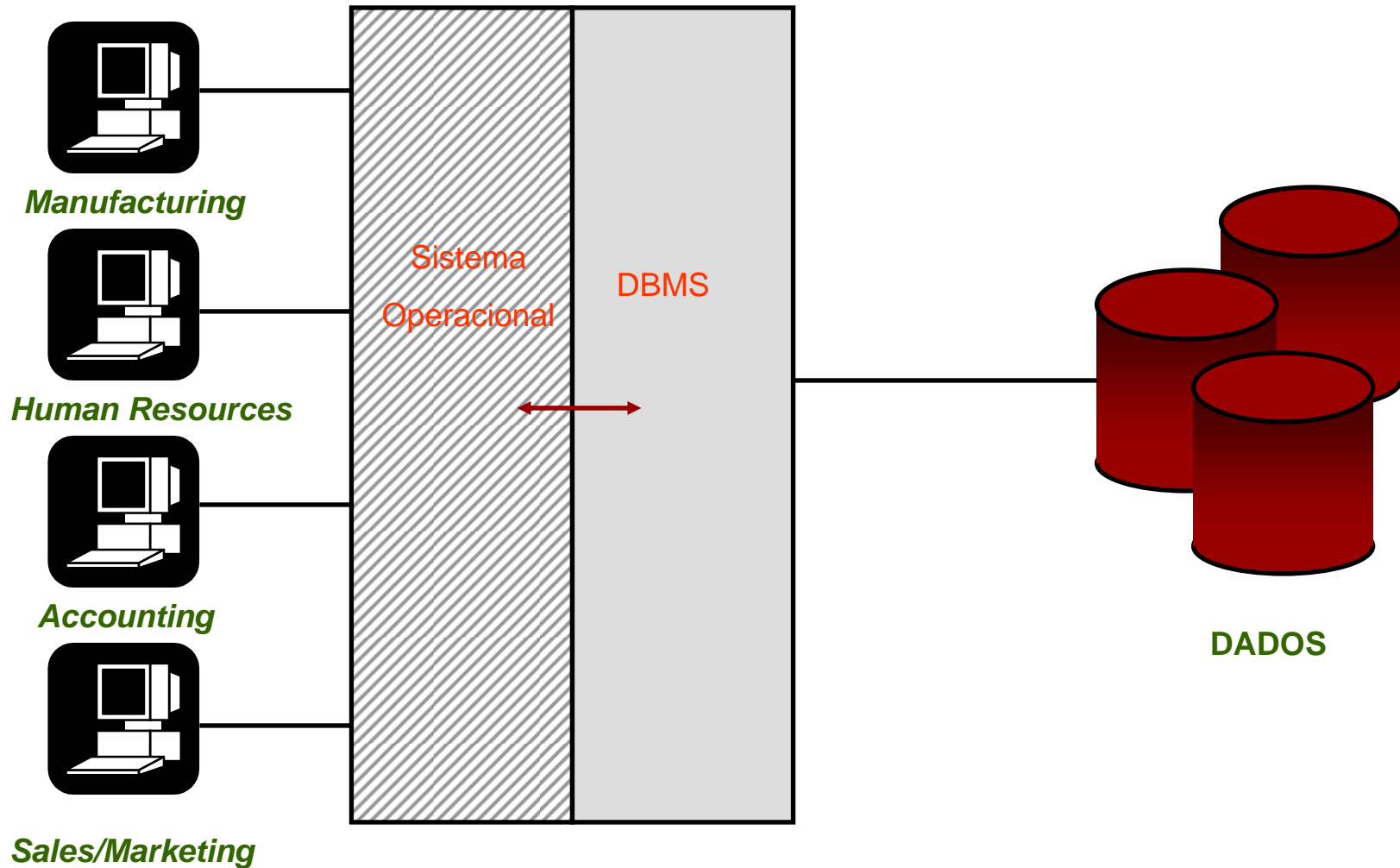
SET KEY Keyword	Descrição
OFF	Desativa todas as funções e retorna o controle ao monitor de TP do sistema.
ON	Reativa o nome do programa ou do comando definido para a função.
ALL	Faz com que todas as chaves tornem-se sensitivas, ou desativa qualquer definição anterior.
PFn=HELP	Quando PFn é pressionado, o help é acionado. Também pode acionar help rotinas e mapas de help.
PFn=OFF	Desativa a PFn.

# Instrução SET KEY

SET KEY Keyword	Descrição
PFn	Torna a PFn sensitiva ou a reativa.
PFn='command'	O comando apontado é atribuído à PFn (ex.: QUIT, SAVE,etc).
PFn=#X	O valor da variável #X é atribuída a PFn.
PFn='program' NAMED 'XXXX'	Um programa é atribuído á PFn. O rótulo definido na clausula <i>NAMED</i> aparece na tela para essa função.
PFn NAMED 'XXXX'	Altera o nome mas não a função da PFn.

- Sistema de Gerenciamento de Banco de Dados
- Visão Geral sobre acesso ao banco
- Processamento seqüencial
- Métodos para limitar o processamento seqüencial
- Processamento randomico
- Métodos para limitar o processamento randomico
- Métodos especiais de acesso
- Uso de condições lógicas
- Acesso a múltiplos arquivos

# Sistema de Gerenciamento de Banco de Dados (DBMS)



### Instruções de acesso ao banco

Para que seu programa possa manipular dados, ele terá primeiro que acessá-los e recuperá-los. O Natural executa essa tarefa através de instruções definidas dentro dos programas. A partir delas, é possível também especificar critérios de recuperação para esses mesmos dados.

As instruções possuem as seguintes características em comum:

- Elas podem acessar um ou mais arquivos do banco (geralmente um de cada vez);
- A maioria podem iniciar processamento de *loops*;
- Elas podem retornar dados para o objeto questionando-o. (Se sua instrução de acesso inclui critérios de pesquisa e nenhum dado atendeu àquele critério, então nenhum dado é retornado ao seu programa);
- Elas podem colocar os dados em “*hold*” para garantir que outros usuários não acessem os dados até que eles tenham sido completamente processados.

### Métodos de Acesso

O Natural suporta dois métodos: Seqüencial e Randômico. A declaração que você escolher para fazer o acesso irá determinar o tipo do método.

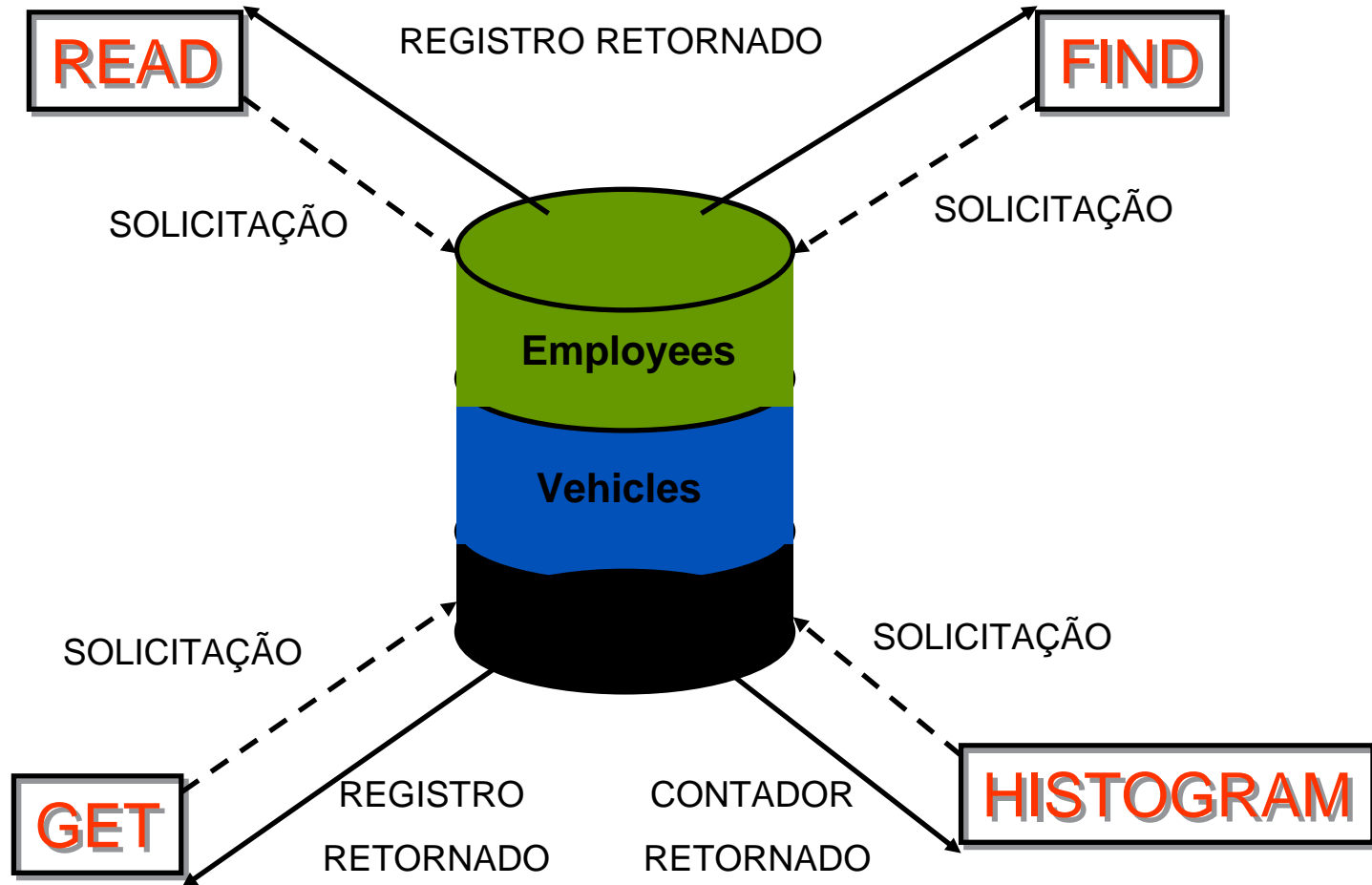
- **Seqüencial**

Seu uso é adequado para acessar um número grande de registros.

- **Randômico**

Seu uso é adequado para acessar um número restrito de registros ou para registros que satisfazem algum critério de pesquisa.

# Visão Geral sobre Acesso ao Banco





### Instrução *READ*

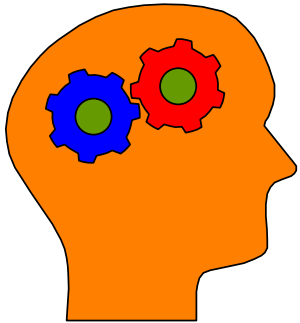
É o método de acesso mais eficiente para processar todo o arquivo. Os arquivos podem ser recuperados do banco:

- Na ordem em que foram fisicamente armazenados (*READ PHYSICAL*);
- Através do número seqüencial interno (*READ BY ISN*);
- Na ordem dos valores de um campo chave (*READ LOGICAL*).

### ***READ PHYSICAL***

Essa instrução gera um *loop* de processamento para acessar os dados no qual os registros são retornados na ordem em que forma armazenados fisicamente. A leitura começa com o início do dados e continua até que o último registro seja lido. Podemos interromper esse processamento estabelecendo um limite, por exemplo, definindo um número de registros a serem lidos.

O *READ PHYSICAL* é útil quando a ordem dos registros retornados não é importante ou quando o critério de seleção não é necessário.



## Lembre-se!

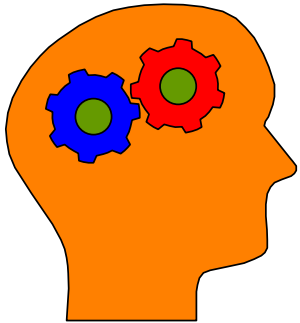
- Use o *READ PHYSICAL* para ler um número grande de registros por ordem física de armazenamento;
- Cuidado ao usar o *READ PHYSICAL* para obter e atualizar todos os registros num arquivo. Em alguns DBMSs você poderia atualizar inadvertidamente um registro duas vezes se o registro atualizado não for colocado de volta na mesma localização física após a modificação.

```
0010 *****
0020 * ILUSTRA O READ PHYSICAL
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 MODEL
0080 2 COLOR
0090 2 YEAR
0100 1 #CAR-TYPE (A30)
0110 END-DEFINE
0120 *
0130 FORMAT SF=3 PS=21
0140 *
0150 READ (50) CARS PHYSICAL
0160 COMPRESS YEAR MAKE MODEL INTO #CAR-TYPE
0170 DISPLAY NOTITLE 5T' ' *COUNTER (UC= NL=4)
0180          'TIPO/DE/CARRO' #CAR-TYPE
0190          'COR' COLOR
0200 END-READ
```

	TIPO DE CARRO	COR
	-----	-----
1	1980 RENAULT R9	ROUGE
2	1980 RENAULT R5	BLANCHE
3	1980 PEUGEOT 305	GRISE
4	1985 PEUGEOT 305	BLANCHE
5	1984 FIAT PANDA	ROUGE
6	1982 RENAULT R4	BLEUE
7	1982 RENAULT R18	GRISE
8	1982 PEUGEOT 205	BLANCHE
9	1982 FIAT UNO	BLANCHE
10	1980 FIAT UNO	CREME
11	1983 BMW 30	GRISE
12	1986 RENAULT R25	GRISE
13	1984 CITROEN CX	BLEUE
14	1984 CITROEN BX 19	BLANCHE
15	1982 RENAULT R5	BLANCHE
16	1982 RENAULT R5	BLEUE

### ***READ BY ISN***

Essa instrução gera um *loop* de processamento para acessar os dados no qual os registros são retornados na ordem do número seqüencial interno. A busca começa com o ISN 1 e continua até o último ISN ser lido. Para DBMSs que suportam *ISNs* esse é o método mais rápido de acesso e assegura que o registro será atualizado apenas uma vez.



## Lembre-se!

- *ISNs* não são disponíveis em todos os DBMSs. Tente identificar o tipo de seu DBMSs para usar essa característica.

# Processamento Seqüencial

```
0010 *****
0020 * ILUSTRA O READ BY ISN
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 MODEL
0080 2 COLOR
0090 2 YEAR
0100 1 #CAR-TYPE (A30)
0110 END-DEFINE
0120 *
0130 FORMAT SF=3 PS=21
0140 *
0150 READ (60) CARS BY ISN
0160 COMPRESS YEAR MAKE MODEL INTO #CAR-TYPE
0170 DISPLAY NOTITLE 5T ' ' *COUNTER (UC= NL=4)
0180          'TIPO/DE/CARRO' #CAR-TYPE
0190          'COR' COLOR
0200          'IDENTIFICACAO/DO/REGISTRO' *ISN (NL=6)
0210 END-READ
0220 *
0230 END
```



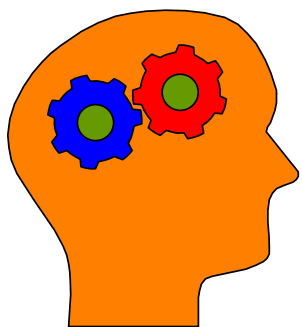
# Processamento Seqüencial

	TIPO DE CARRO	COR	IDENTIFICACAO DO REGISTRO
	-----	-----	-----
1	1980 RENAULT R9	ROUGE	1
2	1980 RENAULT R5	BLANCHE	2
3	1980 PEUGEOT 305	GRISE	3
4	1985 PEUGEOT 305	BLANCHE	4
5	1984 FIAT PANDA	ROUGE	5
6	1982 RENAULT R4	BLEUE	6
7	1982 RENAULT R18	GRISE	7
8	1982 PEUGEOT 205	BLANCHE	8
9	1982 FIAT UNO	BLANCHE	9
10	1980 FIAT UNO	CREME	10
11	1983 BMW 30	GRISE	11
12	1986 RENAULT R25	GRISE	12
13	1984 CITROEN CX	BLEUE	13
14	1984 CITROEN BX 19	BLANCHE	14
15	1982 RENAULT R5	BLANCHE	15
16	1982 RENAULT R5	BLEUE	16
...			

### ***READ LOGICAL***

Essa instrução gera um *loop* de processamento para acessar os dados no qual os registros são retornados na ordem ascendente de um campo chave. A busca começa com o primeiro registro que satisfaz o valor definido no campo chave e continua até o fim do arquivo. Podemos interromper esse processamento estabelecendo um limite ou escapando do *loop*.

O *READ LOGICAL* é indicado para a leitura de um número grande de registro em que a ordem é importante. Esse tipo de instrução é bem eficiente quando os registros do banco são armazenados por ordem de descritor.



## Lembre-se!

- Use o *READ LOGICAL* para ler um grande número de registros pela ordem do campo chave.
- A menos que você defina explicitamente o número de registros a ser lido, ou defina um valor final, a instrução *READ LOGICAL* retorna os registros definidos a partir do valor especificado de início e continua até o fim do arquivo.

# Processamento Sequencial

```
0010 *****
0020 * ILUSTRA O INSTRUCAO READ LOGICAL
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 MODEL
0080 2 COLOR
0090 2 YEAR
0100 1 #CAR-TYPE (A30)
0110 END-DEFINE
0120 *
0130 FORMAT SF=3 PS=21
0150 READ (50) CARS BY MAKE
0160 COMPRESS YEAR MAKE MODEL INTO #CAR-TYPE
0170 DISPLAY NOTITLE 5T ' ' *COUNTER (UC= NL=4)
0180          'TIPO/DE/CARRO' #CAR-TYPE
0190          'COR' COLOR
0200          'IDENTIFICACAO/DO/REGISTRO' *ISN (NL=4)
0210 END-READ
0220 *
0230 END
```

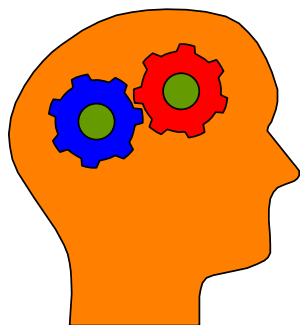
# Processamento Seqüencial

	TIPO DE CARRO	COR	IDENTIFICACAO DO REGISTRO
	-----	-----	-----
1	1985 ALFA ROMEO GIULIETTA 2.0	VERMELHO	205
2	1984 ALFA ROMEO SPRINT GRAND P	ROT	206
3	1984 ALFA ROMEO QUADRIFOGLIO	BLAU-MET.	207
4	1977 AMERICAN MOTOR HORNET	YELLOW	299
5	1983 AMERICAN MOTOR AMBASSADOR	BLACK	302
6	1977 AMERICAN MOTOR HORNET	YELLOW	328
7	1983 AUDI QUATRO	ROUGE	102
8	1983 AUDI QUATRO TURBO	BLANCHE	103
9	1986 AUDI QUATRO TURBO	GRISE	104
10	1982 AUDI 100 CD	WEISS	108
11	1980 AUDI 80 S	WEISS	109
12	1984 AUDI 100 CC	GOLD-MET.	113
13	1985 AUDI 80 LS	ROT	114
14	1984 AUDI 100 CD	BLAU-MET.	118
15	1985 AUDI 100 CC	GOLD-MET.	123
16	1984 AUDI 100 CD 5E	DUNKELGRAU	125

...

## Limitando o número de registros a serem lidos

Basta acrescentar, entre parênteses, o número de registros a serem lidos após a palavra chave *READ*.

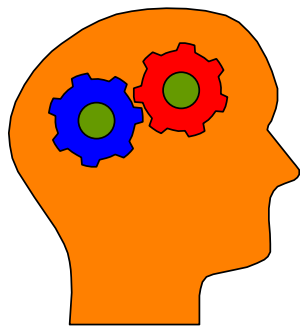


### Lembre-se!

- Para ler registros na casa do milhar, defina um zero à esquerda do número (por exemplo (0nnnn), pois o Natural entende qualquer número da casa do milhar entre parênteses como uma linha de referência de alguma declaração.

## As cláusulas **STARTING / ENDING**

Com as opções *EQUAL/STARTING FROM* nas cláusulas *BY* ou *WITH*, você pode definir o valor a partir do qual a leitura deveria começar. Adicionando as opções *THRU/ENDING AT*, você pode também definir um ponto final para a operação de leitura. Sem essa opção a leitura dos registros continuaria até o fim do arquivo.



### Lembre-se!

- Os valores lidos incluem o valor definido após o *THRU/ENDING AT*. Por exemplo, se você estiver fazendo uma leitura *READ EMPLOYEES BY JOB-TITLE='A' THRU 'C'*, todos os registros que começam com 'A', 'B' são lidos; se houver algum *job-title 'C'*, esse também seria lido, mas o próximo não.

# Métodos para Limitar o Processamento Seqüencial

```
0010 *****
0020 * ILUSTRA O USO DO READ COM OPCAO DE STARTING FROM / ENDING AT
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMP VIEW OF EMPLOYEES
0060 2 PERSONNEL-ID
0070 2 FIRST-NAME
0080 2 NAME
0090 2 DEPT
0100 2 JOB-TITLE
0110 1 #START (A20)
0120 1 #END (A20)
0130 1 #MAX (P2)
0140 END-DEFINE
0150 *
0160 FORMAT SF=3 PS=21
0170 *
0180 INPUT   /// '          ENTRE COM O NOME INICIAL:' #START (AD=AIT'_' )
0190         / '          E O NOME FINAL:' #END (AD=AIT'_' )
0200        / 'NUMERO DE REGISTROS A SEREM LIDOS:' #MAX (AD=AIT'_' )
```



## Continuação

```
0210 *  
0220 READ (#MAX) EMP BY NAME STARTING FROM #START THRU #END  
0230 DISPLAY NOTITLE FIRST-NAME / 2X NAME DEPT PERSONNEL-ID JOB-TITLE  
0240 END-READ  
0250 *  
0260 END
```

## Tela de *Input*:

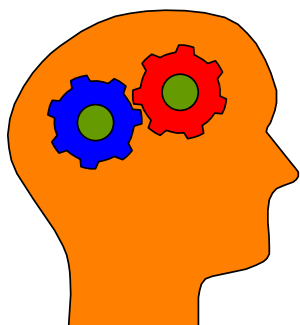
```
ENTRE COM O NOME INICIAL: _____  
E O NOME FINAL: _____  
NUMERO DE REGISTROS A SEREM LIDOS: _____
```

## Continuação

FIRST-NAME NAME	DEPARTMENT CODE	PERSONNEL ID	CURRENT POSITION
VIRGINIA JONES	SALE30	20007500	MANAGER
MARSHA JONES	MGMT10	20008400	DIRECTOR
ROBERT JONES	TECH10	20021100	PROGRAMMER
LILLY JONES	MGMT10	20000800	SECRETARY
EDWARD JONES	TECH10	20001100	DBA
MARTHA JONES	SALE00	20002000	TRAINEE
LAUREL JONES	SALE20	20003400	SALES PERSON
KEVIN	COMP12	30034045	V D U OPERATOR
...			

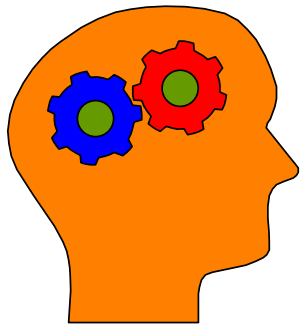
## A clausula **WHERE**

O *READ LOGICAL* com a clausula *WHERE* fornece outro método para definir um critério de pesquisa. Ela permite que você verifique os valores de dados que não foram definidos como chave.



### Lembre-se!

- A clausula *WHERE* é diferente das clausulas *WITH/BY* em dois aspectos:
  1. O campo definido na clausula *WHERE* não precisa ser descritor;
  2. As opções que seguem a clausula *WHERE* são condições lógicas (igual, menor que, maior que, etc.).



## Lembre-se!

- Todo critério com a cláusula *WHERE* é avaliado após a leitura dos registros pelo DBMS e antes que qualquer processamento seja executado no registro.
- Se um processamento com limites é definido numa declaração *READ* contendo a cláusula *WHERE*, os registros rejeitados como resultado não são testados contra o limite.
- Evite usar a cláusula *WHERE* em *loops* de processamento que contenham as declarações *UPDATE* ou *DELETE*. Em alguns DBMS a cláusula *WHERE* pode rejeitar registros suficientes para exceder o número permitido de registros em *hold*.

# Métodos para Limitar o Processamento Seqüencial

```
0010 *****
0020 * ILUSTRA A DECLARACAO READ COM A CLAUSULA WHERE
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMP VIEW OF EMPLOYEES
0060 2 PERSONNEL-ID
0070 2 FIRST-NAME
0080 2 NAME
0090 2 DEPT
0100 2 JOB-TITLE
0110 2 SALARY (1)
0120 END-DEFINE
0130 *
0140 FORMAT SF=3 PS=20
0150 *
0160 READ EMP BY PERSONNEL-ID WHERE SALARY (1) = 60000
0170 DISPLAY NOTITLE PERSONNEL-ID SALARY(1) JOB-TITLE FIRST-NAME / 2X NAME
0180 END-READ
0190 *
0200 END
```

# Métodos para Limitar o Processamento Seqüencial

PERSONNEL ID	ANNUAL SALARY	CURRENT POSITION	FIRST-NAME NAME
11700312	60000	SYSTEMBERATER	HEINZ GRAF
20000900	60000	DIRECTOR	HAZEL WYLLIS
20021900	60000	DIRECTOR	RICHARD GEE

### Declaração *FIND*

Gera um *loop* de processamento onde os registros não são retornados numa ordem específica ao definir-se a chave. No ADABAS, os registros são retornados por ordem de ISN.

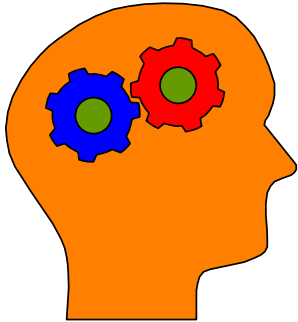
### Uso de variáveis de sistema

#### \**NUMBER*

Em alguns DBMSs essa variável contém o número de registros que satisfizeram o critério da clausula *WITH*.

#### \**COUNTER*

É uma variável automaticamente incrementada a cada interação do *loop* de leitura. Se for referenciada fora do *loop* de processamento ao qual ela se aplica, a referência deve ser fornecida.



## Lembre-se!

- A menos que a opção *SORT* seja usada, a declaração *FIND* retorna os registros numa ordem qualquer.
- Quando estiver fazendo atualizações com a declaração *FIND*, é importante lembrar que todos os registros são colocados em *hold*.
- Os campos que você definiu após a declaração *WITH* devem ser campos do banco definidos na *view*. Esse campo deve ser um descritor.



```
0010 *****
0020 * EXEMPLO DO USA DA DECLARACAO FIND
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 MODEL
0080 2 COLOR
0090 2 YEAR
0100 1 #CAR-TYPE (A30)
0110 1 #MARCA (A20)
0120 END-DEFINE
0130 *
0140 FORMAT SF=3 PS=21
0150 *
0160 INPUT /// 10T 'POR FAVOE ENTRE COM A MARCA DESEJADA:' #MARCA (AD=AIT'_' )
0170 *
0180 F1.
0190 FIND CARS WITH MAKE = #MARCA
0200 COMPRESS YEAR MAKE MODEL INTO #CAR-TYPE
0210 DISPLAY NOTITLE 5T ' '*COUNTER (UC= NL=3 AD=I) 'TIPO/DE/CARRO' #CAR-TYPE
0220 'COR' COLOR 'IDENTIFICACAO/DO/REGISTRO'*ISN (NL=3)
0230 END-FIND
0240 *
0250 END
```

# Processamento Randômico

POR FAVOE ENTRE COM A MARCA DESEJADA: \_\_\_\_\_

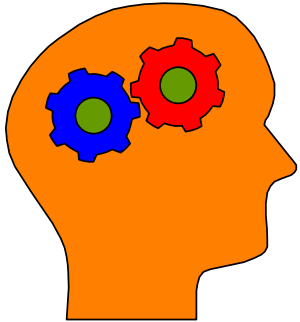
TIPO	COR DE CARRO	IDENTIFICACAO	DO REGISTRO
-----		-----	-----
1	1984 FIAT PANDA	ROUGE	5
2	1982 FIAT UNO	BLANCHE	9
3	1980 FIAT UNO	CREME	10
4	1980 FIAT PANDA	ROUGE	17
5	1985 FIAT UNO	GRISE	105
6	1985 FIAT PANDA 45	GELB	176
7	1984 FIAT RITMO 75 CL	GRUEN-MET.	177
8	1982 FIAT 127 S	ROTBRAUN	178
9	1984 FIAT DINO	BLUE	312
10	1984 FIAT DINO	BLUE	341
...			

### Instrução *FIND...SORTED BY*

Lembre-se, quando usa-se a declaração *FIND* os registros são retornados numa ordem qualquer. Você pode controlar a ordem em que os registros são retornados ao programa usando a clausula *SORTED BY*. Com essa chave você pode definir até 3 campos chaves.

Exemplo:

```
FIND CARS WITH  
  MAKE = 'BMW' THRU 'FORD'  
  SORTED BY COLOR  
END-FIND
```



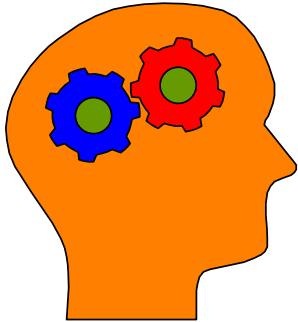
## Lembre-se!

- Enquanto o *FIND...SORTED BY* permite que você classifique os dados numa ordem específica, ele pode, por outro lado deixar o processamento de sua máquina mais lento quando houver muitos arquivos para classificar.
- Outras opções para retornar os dados classificados incluem o uso das declarações *READ LOGICAL* ou instruções de *SORT*.

## Clausula *WHERE*

Com a clausula *WHERE* da declaração *FIND* você pode especificar um critério de seleção adicional que é avaliado após os registros terem sido lidos e antes que o processamento seja executado no registro.

```
FIND EMPL WITH NAME = 'SMITH'  
  WHERE SALARY(1) > 10000  
END-FIND
```



### Lembre-se!

- Todo critério que usa a cláusula *WHERE* são avaliados após os registros terem sido lidos pelo DBMS e antes que qualquer processamento seja executado.
- Se um limitador é definido junto com a declaração *FIND* que contém a cláusula *WHERE*, os registros rejeitados como resultado da cláusula *WHERE* não são testados contra o limitador.
- Evite usar a cláusula *WHERE* em *loops* de processamento contendo a declaração *UPDATE* ou *DELETE*.

# Métodos para Limitar o Processamento Randômico

```
0010 *****
0020 * EXEMPLO DO USO DO FIND COM CLAUSULA WHERE
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 MODEL
0080 2 COLOR
0090 2 YEAR
0100 1 #CAR-TYPE (A30)
0110 1 #MAKE      (A20)
0120 END-DEFINE
0130 *
0140 FORMAT SF=3 PS=20
0150 *
0160 INPUT /// 10T 'POR FAVOR ENTRE COM A MARCA DESEJADA:' #MAKE (AD=AIT'_' )
0170 *
0180 F1.
0190 FIND CARS WITH MAKE = #MAKE
0200 WHERE YEAR = 1983
0210 COMPRESS YEAR MAKE MODEL INTO #CAR-TYPE
0220 DISPLAY NOTITLE 5T ' ' *COUNTER (UC= NL=3 AD=I)
0230           'TIPO/DE/CARRO' #CAR-TYPE
0240           'COR' COLOR
0250 END-FIND
0260 *
0270 END
```

# Métodos para Limitar o Processamento Randômico

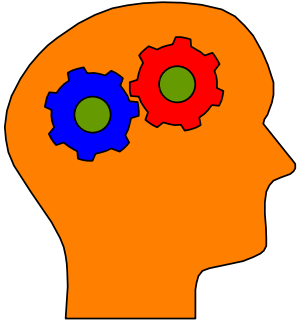
POR FAVOR ENTRE COM A MARCA DESEJADA: \_\_\_\_\_

	TIPO DE CARRO	COR
	-----	-----
1	1983 FORD TRANSIT	BLAU
2	1983 FORD FIESTA	METAL BL\$
3	1983 FORD ESCORT 1.3	METAL GR\$
4	1983 FORD ESCORT	WHITE
5	1983 FORD ESCORT	WHITE
6	1983 FORD ESCORT	WHITE
7	1983 FORD ESCORT	WHITE
8	1983 FORD ESCORT	YELLOW
9	1983 FORD ESCORT	YELLOW
10	1983 FORD ESCORT 1.3	BLUE



### Instrução *FIND NUMBER*

Apresenta o número de registros que passaram pelo critério de pesquisa sem a necessidade do fornecimento de dados relativos aos campos. O resultado é colocado na variável de sistema \**NUMBER*.



## Lembre-se!

- *FIND NUMBER* é uma declaração de acesso randômico.
- Nenhum registro do banco é retornado com o seu uso.
- Essa declaração não inicia *loop* de processamento, portanto não exige o *END-FIND*.
- As cláusulas *WHERE*, *SORTED BY* e *IF NO RECORDS FOUND* não estão disponíveis para ela.
- Ela é bem eficiente quando usada com um banco do tipo *relational-like*.

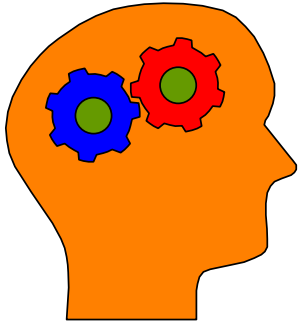
# Métodos Especiais de Acesso

```
0010 *****
0020 * EXEMPLO DO USO DO FIND NUMBER
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 MODEL
0080 2 COLOR
0120 END-DEFINE
0130 *
0160 INPUT /// 10T 'POR FAVOR ENTRE COM A MARCA DESEJADA:' #MAKE
(AD=AIT'_' )
0170 *
0180 F1.
0190 FIND NUMBER CARS WITH MAKE = 'FORD'
0200     AND COLOR = 'BLUE'
0220 WRITE NOTITLE // 5T 'O numero de carros 'FORD' azuis sao:' *number
0270 END
```

O numero de carros FORD azuis são: 38

### Instrução *HISTOGRAM*

Essa declaração pode ser usada tanto para ler apenas os valores de um campo do banco, como para determinar o número de registros que passaram pelo critério de pesquisa definido.



## Lembre-se!

- Deve-se incluir uma *user view* na declaração *DEFINE DATA* que define somente o campo descritor através do qual o *HISTOGRAM* se guiará.
- Somente um descritor pode ser definido por declaração *HISTOGRAM*.
- Se a cláusula *WHERE* estiver sendo usada, ela só poderá conter um critério que utilize o mesmo descritor de *HISTOGRAM*.
- As cláusulas *STARTING FROM* e *ENDING AT* estão disponíveis; assim como as variáveis de sistema \**COUNTER* e \**NUMBER*.

```
0010 *****
0020 * EXEMPLO DO USO DA DECLARACAO HISTOGRAM
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 END-DEFINE
0080 *
0090 H1. HISTOGRAM CARS FOR MAKE
0100 DISPLAY NOTITLE 25T 'NBR' *NUMBER (NL=3) 2X MAKE
0110 END-HISTOGRAM
0120 *
0130 WRITE // 20T 'NUMERO DE MARCAS DIFERENTES:' *COUNTER (H1.)
0140 *
0150 END
```

## Saída:

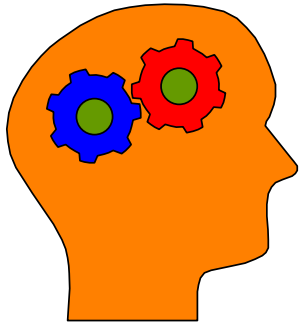
NBR	MAKE
---	-----
3	ALFA ROMEO
3	AMERICAN MOTOR
29	AUDI
25	AUSTIN
37	BMW
57	CHRYSLER
29	CITROEN
1	DAIHATSU
10	DATSUN
1	FERRARI
16	FIAT
...	
15	VOLVO
16	VW
NUMERO DE MARCAS DIFERENTES: 45	

### Instrução *GET*

É usada para operações que envolvem um único registro. Segue algumas características:

- Retorna um registro do banco através de um *ISN* conhecido. A variável *\*ISN* pode ser usada para fornecer o valor do *ISN*, caso o registro tenha sido previamente acessado.
- O registro retornada através do *GET* pode ser colocado em *hold* para um processamento posterior. Somente o atual usuário pode atualizar o registro posto em *hold*.
- É uma forma eficiente de acessar um único registro.





### Lembre-se!

- Assim como a variável de sistema *\*ISN*, a declaração *GET* não está disponível para todos os DBMS.

# Métodos Especiais de Acesso

```
0010 *****
0020 * EXEMPLO DO USO DA DECLARACAO GET
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMP VIEW OF EMPLOYEES
0060 2 PERSONNEL-ID
0070 2 NAME
0080 2 FIRST-NAME
0090 2 JOB-TITLE
0100 2 SALARY(1)
0120 END-DEFINE
0130 *
0140 FORMAT PS=21
0150 *
0160 READ-ISN.
0170 READ (5) EMP BY ISN
0180 DISPLAY NOTITLE (SF=7) 8T'IDENTIFICACAO/DO/REGISTRO(ISN)' *ISN
0190 PERSONNEL-ID 'NOME' NAME(AL=9)
0200 'SALARIO' SALARY(1)
0210 END-READ
```

## Continuação

```
0230 *****
0240 * A VARIAVEL *ISN IRA CONTER O VALOR DO ISN DO ULTIMO REGISTRO ACESSADO
0250 *****
0260 *
0270 GET EMP *ISN (READ-ISN.)
0280 *
0290 WRITE  NOTITLE
0300 / 'O ULTIMO REGISTROLIDO FOI:' //
0310 3X 'ISN:          ' *ISN(AL=3 AD=L)/
0320 3X 'PERSONNEL ID:' PERSONNEL-ID /
0330 3X 'NOME:         ' NAME /
0340 *
0350 END
```

## Saída:

IDENTIFICACAO DO REGISTRO ( ISN )	PERSONNEL ID	NOME	SALARIO
-----	-----	-----	-----
1	50005800	ADAM	159980
2	50005600	MORENO	165810
3	50005500	BLOND	172000
4	50005300	MAIZIERE	166900
5	50004900	CAOUDAL	167350

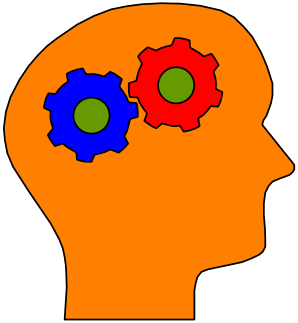
O ULTIMO REGISTRO LIDO FOI:

ISN: 5  
PERSONNEL ID: 50004900  
NOME: CAOUDAL

### Expressões Lógicas

Ao definir os critérios de seleção nas declarações *READ* ou *FIND*, você pode fazer uso de condições lógicas. Ou seja, você pode escolher um intervalo de valores para seus dados tais como: 'salários menores que \$30.000' ou 'nome igual a *SMITH*'. Essas condições são introduzidas no programa através das expressões lógicas, que classificam-se em dois tipos:

- Condições Simples,
- Condições Múltiplas



## Lembre-se!

- Se você estiver usando mais que uma expressão lógica simples e/ou lógica *booleana*, você deveria usar um dos seguintes conectores:
  - `()`
  - *NOT*
  - *AND*
  - *OR*
  - *THRU*

# Usando Condições Lógicas

```
0010 *****
0020 * EXEMPLO DO FIND COM OPERADORES
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 MODEL
0080 2 COLOR
0090 2 YEAR
0100 END-DEFINE
0110 *
0120 FORMAT SF=3 PS=21
0130 FIND CARS WITH (MAKE='FORD' OR='HONDA')
0140         AND (COLOR ='BLUE' OR ='RED')
0150         SORTED BY COLOR
0160         DISPLAY NOTITLE MAKE MODEL COLOR YEAR
0170 END-FIND
0180 *
0190 END
```

# Usando Condições Lógicas

MAKE	MODEL	COLOR	YEAR
FORD	MERCURY	BLUE	1982
FORD	MERCURY	BLUE	1986
FORD	MERCURY	BLUE	1982
FORD	MERCURY	BLUE	1986
FORD	MUSTANG	BLUE	1984
FORD	GRANADA	BLUE	1977
FORD	MUSTANG	BLUE	1977
FORD	LTD	BLUE	1982
FORD	LTD	BLUE	1982
FORD	ORION 1.6 GHIA	BLUE	1985
FORD	ORION 1.6 GHIA	BLUE	1985
FORD	ORION 1.6 GHIA	BLUE	1985
FORD	ORION 1.6 GHIA	BLUE	1985
FORD	ORION 1.6 GHIA	BLUE	1985
FORD	ORION 1.6 GHIA	BLUE	1985
FORD	ORION 1.6 GHIA	BLUE	1985
FORD	ORION 1.6 GHIA	BLUE	1986
FORD	ORION 1.6 GHIA	BLUE	1986

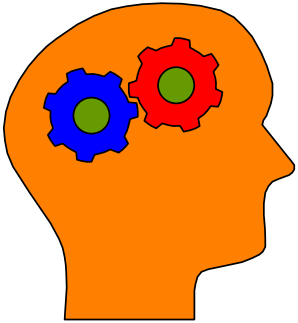


## Instrução *ACCEPT / REJECT*

As instruções *ACCEPT/REJECT* são usadas para avaliar os registros baseados no critério lógico. Se os registros satisfazem ao critério *ACCEPT*, eles serão processados; se satisfizerem ao critério *REJECT*, não serão processados.

```
FIND EMPLOYEES  
  WITH JOB-TITLE = 'DBA' OR='ADMINISTRATOR'  
  ACCEPT IF SEX = 'M'  
END-FIND
```

```
READ EMPLOYEES  
  BY JOB-TITLE WHERE JOB-TITLE='DBA' OR='ADMINISTRATOR'  
  REJECT IF SEX = 'F'  
END-READ
```



### Lembre-se!

- Essas declarações podem ser colocadas em qualquer lugar no seu *loop* de processamento.
- O campo usado para base de critério pode ser tanto descritor como não-descritor.
- Se uma declaração *LIMIT* ou outra notação de limite for definido para o *loop* de processamento junto com *ACCEPT* e *REJECT*, cada registro processado é avaliado contra o limite para ser aceito ou rejeitado.

### Clausula *IF NO RECORDS FOUND*

Se dentro da instrução *FIND* nenhum registro satisfizer ao critério de pesquisa, as declarações dentro do *loop* de processamento não serão executadas. Se esse for o caso, você pode utilizar a clausula *IF NO RECORDS FOUND* para definir que processamento será executado caso nenhum registro seja encontrado.

# Usando Condições Lógicas

```
0010 *****
0020 * EXEMPLO DA CLAUSULA IF NO RECORDS FOUND
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMPLOY-VIEW VIEW OF EMPLOYEES
0060 2 PERSONNEL-ID
0070 2 NAME
0080 1 #PERS-NR (A8)
0090 END-DEFINE
0100 *
0110 REPEAT
0120     INPUT 'ENTRE COM A IDENTIFICACAO DO FUNCIONARIO:' #PERS-NR
0130     IF #PERS-NR = ' '
0140         ESCAPE BOTTOM
0150     END-IF
0160     FIND EMPLOY-VIEW WITH PERSONNEL-ID = #PERS-NR
0170     IF NO RECORDS FOUND
0180         REINPUT 'NENHUM REGISTRO ENCONTRADO'
0190     END-NOREC
0200     DISPLAY NOTITLE NAME
0210     END-FIND
0220 END-REPEAT
0230 *
0240 END
```

### Acessando mais que um arquivo

Até agora, temos recuperado dados armazenados em apenas um arquivo. Entretanto, a maioria das aplicações acessam mais que um arquivo por banco. Esse processo é conhecido como *coupling* (acoplamento).

O acoplamento permite a extração de dados de um arquivo, baseado nos dados encontrados em outro arquivo. Há três tipos de acoplamento:

- Lógico;
- *Soft*;
- Físico.

### Linhas Gerais

- Para executar o acoplamento, deve existir um campo chave comum em cada arquivo.
- O acoplamento lógico é uma técnica de codificação em Natural.
- O acoplamento *Soft* é executado pelo seu DBMS.

### Acoplamento Lógico

É um processo que permite que você leve vantagem do relacionamento lógico entre dois ou mais arquivos, se estiverem ou não acoplados fisicamente no DBMS. Com essa característica você pode acessar dois ou mais arquivos usando um campo descritor comum. Duas declarações *FIND* ou *READ* são usadas para acoplar arquivos logicamente onde um *loop* interno é montado para cada registro selecionado no *loop* externo.

# Acesso Múltiplo-Arquivo

```
0010 *****
0020 * EXEMPLO DE ACOPLAMENTO LOGICO
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 PERSONNEL-ID
0080 2 MODEL
0090 2 COLOR
0100 *
0110 1 EMPL VIEW OF EMPLOYEES
0120 2 PERSONNEL-ID
0130 2 NAME
0140 2 FIRST-NAME
0150 END-DEFINE
0160 *
0170 FIND EMPL WITH NAME = 'JONES'
0180     FIND CARS WITH PERSONNEL-ID = EMPL.PERSONNEL-ID AND MAKE = 'FORD'
0190     DISPLAY PERSONNEL-ID MAKE (AL=10) MODEL (AL=10) COLOR *NUMBER
0200             FIRST-NAME (AL=10) NAME (AL=10)
0210     END-FIND
0220 END-FIND
0230 *
0240 END
```



# Acesso Múltiplo-Arquivo

## Saída:

PERSONNEL-ID	MAKE	MODEL	COLOR	NMBR	FIRST-NAME	NAME
20000800	FORD	ESCORT	BLACK	1	LILLY	JONES
30034233	FORD	ESCORT 1.3	GREEN	1	GREGORY	JONES

### Acoplamento *Soft*

O acoplamento *soft* está disponível com declaração *FIND* e pode ser emitida para criar *loops* aninhados onde o *loop* interno é criado para cada registro selecionado do *loop* externo. Esta característica permite que você acesse um arquivo baseado nos descritores a partir de dois arquivos que possuem dados em comum.

# Acesso Múltiplo-Arquivo

```
0010 *****
0020 * EXEMPLO DE ACOPLAMENTO LOGICO
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 MAKE
0070 2 PERSONNEL-ID
0080 2 MODEL
0090 2 COLOR
0100 *
0110 1 EMPL VIEW OF EMPLOYEES
0120 2 PERSONNEL-ID
0130 2 NAME
0140 2 FIRST-NAME
0150 END-DEFINE
0160 *
0170 FIND CARS WITH MAKE = 'FORD' AND COUPLED TO EMPL
0180     VIA PERSONNEL-ID = PERSONNEL-ID WITH NAME= 'JONES'
0190     DISPLAY PERSONNEL-ID MAKE MODEL) COLOR *NUMBER
0200
0210 END-FIND
0230 *
0240 END
```

# Acesso Múltiplo-Arquivo

PERSONNEL-ID	MAKE	MODEL	COLOR	NMBR
20000800	FORD	ESCORT	BLACK	2
30034233	FORD	ESCORT 1.3	GREEN	2

### Unidade A - Criação de Relatórios

- Instruções *Display* e *WRITE*
- Exibição de atributos
- Parâmetros de sessão
- Controle de saída para relatórios
- *WRITE TITLE* e *WRITE TRAILLER*
- *AT TOP OF PAGE* e *AT END OF PAGE*
- Declaração *PRINT*

## ***Display***

Produz uma saída em formato de coluna onde cada coluna é representada pelo valor do campo. A ordem de exibição segue a seqüência na qual os campos são definidos. Além disso, possui um cabeçalho (*title*), que consiste do número da página, hora e data.

## **Sintaxe**

***DISPLAY* [(*rep*)] [*NOTITLE*] [*NOHDR*] [parâmetros] {elemento de saída [/...]}...**

**elemento de saída = [{ nX, nT}] [{ '=' , 'text' , 'c'(n)}] ['=']  
operando 1 [(parâmetros)]**

### Recursos Adicionais - Display

- Controle de relatório escrito;
- Controle de títulos e cabeçalhos automáticos;
- Sobreposição de cabeçalhos de campos d DDM;
- Inserção de caracteres em frente ao campo.
- A barra provoca o avanço de linha dentro da coluna especificada.
- Somente a primeira instrução *DISPLAY* tem controle do título e cabeçalho.

### ***Write***

Produz uma saída em formato livre. Quando o tamanho do texto não couber na linha definida há um avanço automático para a linha seguinte. Não possui cabeçalhos automáticos.

### **Sintaxe**

*WRITE* [(*rep*)] [*NOTITLE*] [{*nX*, *nT*, *x/y*}] [{{“*text*”, ‘*c*’(*n*)}, ‘=’, /...}] operando 1...

Os títulos da página são gerados, exceto quando suprimidos.



<b>Características</b>	<b>DISPLAY</b>	<b>WRITE</b>
Orientação da saída	Coluna	Linha
Imprime títulos das páginas automaticamente	Sim	Sim
Imprime cabeçalhos automaticamente	Sim	Não
Estouro de linha permitido	Não	Sim
Espaço permitido para os campos no relatório	Determinado pelo campo maior - cabeçalho ou o próprio campo.	Determinado pelo tamanho do campo.

Características	DISPLAY	WRITE
Espaço entre os campos	Controlado pelo parâmetro <i>Spacing Factor</i> , mas pode ser sobreposto pelas notações nX e nT. O padrão é um espaço.	Para obter mais que um espaço deve-se usar a notação nX, nT, ou x/y.
Uso da barra “ / ”	Pula uma linha dentro da própria coluna	Inicia uma nova linha a partir da margem esquerda
Impressão de Arrays	Lista verticalmente na coluna	Lista horizontalmente através da linha.

### Parâmetro AD

Controla a forma na qual os campos são exibidos.

- Representação (normal, intensificado, etc);
- Alinhamento (esquerda, direita, zeros)
- Caracteres (maiúsculos, minúsculos, misturados).

### Parâmetro CD

Define a cor na qual os campos são exibidos

. Exemplo:

```
DISPLAY NAME (AD=I CD=YE) 'HOME TOWN' (AD=I CD=BL) CITY (AD=DR)
```

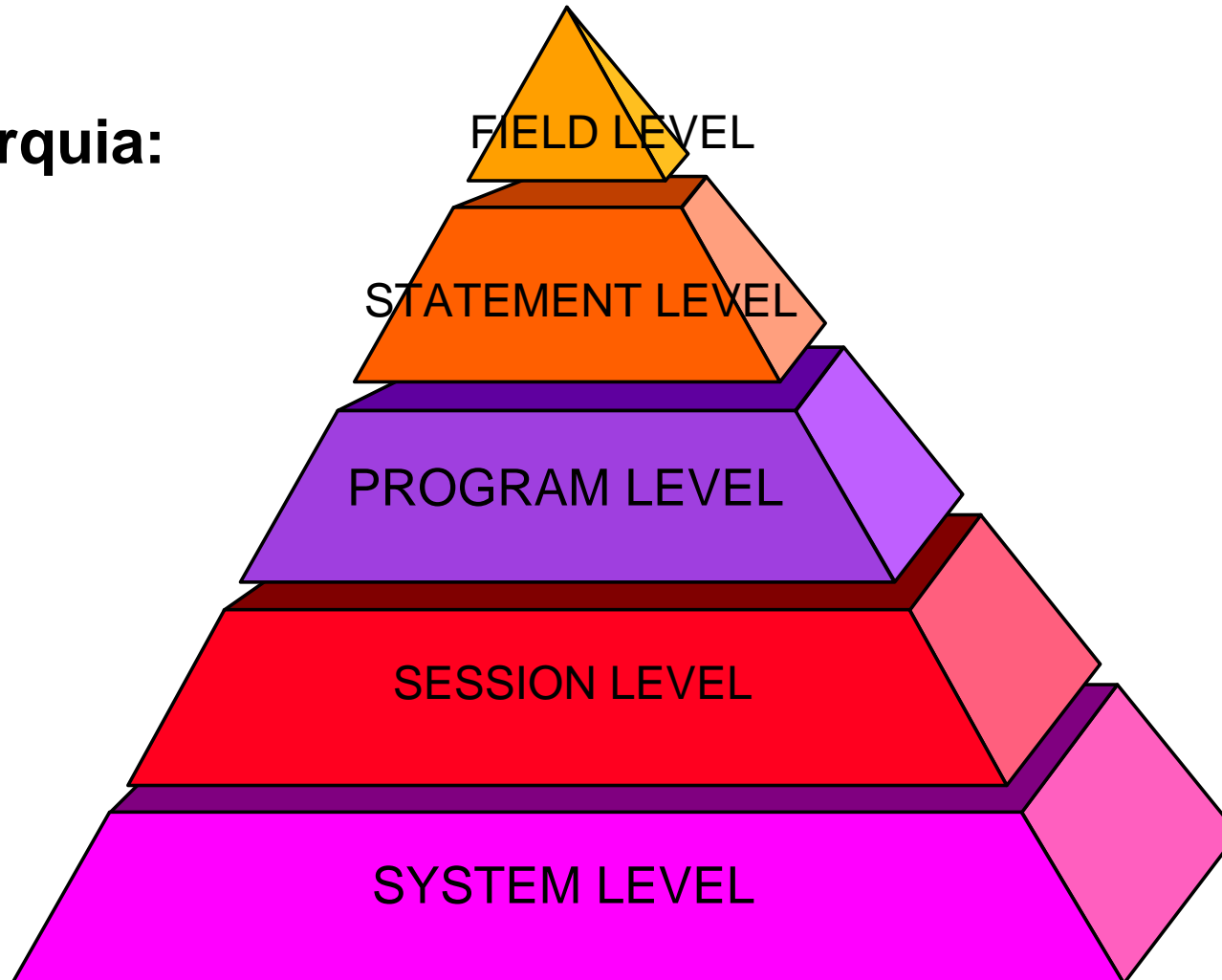
```
WRITE 'THE NUMBER OF CARS:' (AD=I) #CNT (CD=RE AD=IL)
```

Códigos	Parâmetros	Valores	Disponível p/ declaração(ões)
LS	Line Size Default=tamanho do dispositivo	10-250	DISPLAY e WRITE
PS	Page Size Default=tamanho do dispositivo	10-250	DISPLAY e WRITE
SF	Spacing Factor	1-30 default=1	DISPLAY
AL	Alphanumeric Length	1-tamanho da linha	DISPLAY e WRITE
NL	Numeric Length	1-tamanho da linha	DISPLAY e WRITE

Códigos	Parâmetros	Valores	Disponível p/ declaração(ões)
HC	Header Centering	Center, Left, e Rigth default=center	DISPLAY
FC	Filler Character	Qualquer caracter default=branco	DISPLAY
GC	Filler character for group headers	Qualquer caracter default=branco	DISPLAY
UC	Underlining character	Qualquer caracter default=hífen	DISPLAY
DF	Date Format	S - ano com 2 díg. l - ano com 4 díg.	DISPLAY e WRITE

Códigos	Parâmetros	Valores	Disponível p/ declaração(ões)
IS	Identical Supress	ON/OFF default=ON	DISPLAY, WRITE, FORMAT
ES	Supressão de linhas em branco	ON/OFF default=OFF	DISPLAY, WRITE, FORMAT

**Hierarquia:**



### \****LINE-COUNT***

- Indica a linha atual contida na página do relatório;
- Um contador separado é mantido para cada relatório gerado pelo programa;
- O valor desse contador é atualizado durante a execução das declarações *WRITE*, *SKIP*, *DISPLAY*, *PRINT* ou *INPUT*;
- As instruções *NEWPAGE* e *EJECT* limpam esse valor.
- Seu formato/tamanho é P5 e seu conteúdo não pode ser modificado.



### \***PAGE-NUMBER**

- Indica o número da página atual do relatório;
- Um contador separado é mantido para cada relatório gerado pelo programa;
- Um programa Natural pode modificar essa variável;
- O seu valor é atualizado durante a execução das declarações *WRITE*, *SKIP*, *DISPLAY* ou *NEWPAGE*;
- A declaração *EJECT* não provoca a atualização do contador;
- Seu formato/tamanho é P5.

# Controle de Saída para Relatórios

```
0010 *****
0020 ** ILUSTRA O USO DAS VARIAVEIS DE SISTEMA *PAGE-NUMER E *LINE-COUNT
0030 *****
0040 DEFINE DATA
0050     LOCAL USING LOC1
0060 EM-DEFINE
0070 *
0080 WRITE TITLE 'PAGE:'*PAGE-NUMBER (AD=L) 70T *DAT4E //
0090 '***** RELATORIO DE CARROS *****' (AD=I)
0100 SKIP 1
0110 READ CARS BY MAKE
0120 IF *LINE-COUNT GT 21 THEN
0130     NEWPAGE          /* NEWPAGE RESETS *LINE-COUNT
0140 END-IF
0150 DISPLAY (HC=L) 15X 'MARCA E MODELO' (AD=I) MAKE / ' ' MODEL
0160     10X 'COR DO CARRO E' (AD=I) COLOR / 'MODELO DO ANO' (AD=I) YEAR
0170 SKIP 1
0180 END-READ
0190 END
```

## Saída:

PAGE: 1

05/02/2003

\*\*\*\*\* RELATORIO DE CARROS \*\*\*\*\*

MARCA E MODELO -----	COR DO CARRO E MODELO DO ANO -----
ALFA ROMEO GIULIETTA 2.0	VERMELHO 1985
ALFA ROMEO SPRINT GRAND PRIX	ROT 1984
ALFA ROMEO QUADRIFOGLIO	BLAU-MET. 1984
AMERICAN MOTOR HORNET	YELLOW 1977

### ***EJECT***

Provoca um avanço de página sem a impressão dos títulos e cabeçalhos. É freqüentemente usada com uma condição que avalia o número de linhas a vencer na página.

### ***EJECT (rep.)***

Se você está produzindo vários relatórios, o repositório de relatório (rep) pode ser usado para identificar a que relatório pertence a declaração *EJECT*.

### ***NEWPAGE***

Provoca um avanço de página com a impressão dos títulos e cabeçalhos. Se *EJECT* ou *NEWPAGE* não forem usadas, o avanço de página é controlado automaticamente pelo parâmetro de sessão *Natural Page Size (PS)*.

### ***SKIP***

Gera uma ou mais linhas em branco. Você pode definir o número de linhas em branco que serão introduzidas. Essa declaração não é executada dentro da condição *AT-TOP-OF-PAGE*.

## Exemplo: *EJECT*

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO EJECT
0030 *****
0040 DEFINE DATA LOCAL
0050 1 VIEWEMP VIEW OF EMPLOYEES
0060   2 NAME
0070   2 CITY
0080   2 DEPARTMENT
0090 END-DEFINE
0100 *
0110 FORMAT PS=10
0120 READ VIEWEMP BY CITY STARTING FROM 'ARLINGTON'
0130   DISPLAY NOTITLE 'LOCALIZACAO' CITY (IS=ON) NAME 'DEPT' DEPARTMENT
0140   SKIP 1
0150   AT BREAK OF CITY
0160     EJECT
0170   END-BREAK
0180 END-READ
0190 END
```

# Controle de Saída para Relatórios

## Saída:

Page 1

03-02-0514:50:23

LOCALIZACAO	NAME	DEPT
ARLINGTON	LORIE	TECH
	ELLIOT	SALE
	BRANDIN	MGMT

ASHBOURNE	DALE	PROD
	BURTON	TECH

## Exemplo: *NEWPAGE*

Page	1	03-02-05	14:45:12
LOCALIZACAO	NAME	DEPT	
-----	-----	-----	
ARLINGTON	LORIE	TECH	
	ELLIOT	SALE	
	BRANDIN	MGMT	

Page	2	03-02-05	14:47:43
LOCALIZACAO	NAME	DEPT	
-----	-----	-----	
ASHBOURNE	DALE	PROD	
	BURTON	TECH	



## Instrução *WRITE TITLE*

- Essa declaração pode ser usada somente uma vez em cada relatório;
- A opção *SKIP* faz com que as linhas sejam saltadas logo após o título;
- Como padrão o título é centralizado e não-sublinhado.

Exemplo:

```
WRITE TITLE 'SAME TITLE'  
  
          / 'PAGE:' *PAGE-NUMBER (AD=L)  
  
          SKIP 2  
  
WRITE (5) TITLE 'REPORT NUMBER 5'  
  
          / *DATU  
  
          / 'PAGE:' *PAGE-NUMBER (5)
```

## Instrução *WRITE TRAILER*

- Essa declaração pode ser usada somente uma vez em cada relatório;
- A instrução é executada sempre que o Natural detecta o fim da página devido a uma instrução *DISPLAY*, *WRITE*, *SKIP* ou *NEWPAGE*;
- O tamanho lógico da página (PS) deve ser menor que o tamanho físico para que as informações de rodapé apareçam. Exemplo:

```
WRITE (3) TRAILER LEFT JUSTIFIED
```

```
50T `RUNNING TOTAL:` #TOTAL
```

```
SKIP 2
```

```
WRITE TRAILER *PAGE-NUMBER
```

# Exemplo

```
0010 *****
0020 * ILUSTRA O SUO DA DECLARACAO WRITE TITLE E WRITE TRAILER
0030 *****
0040 DEFINE DATA
0050 LOCAL
0060 1 #CNT          (N2)
0070 1 #REG-NUM     (A20)
0080 1 CARS VIEW OF VEHICLES
0090  2 REG-NUM
0100  2 MAKE
0110  2 MODEL
0120  2 COLOR
0130  2 YEAR
0140 END-DEFINE
0150 *
0160 FORMAT PS=20
0170 WRITE TITLE UNDERLINED 1T *DAT4E 70T *TIMX
0180   / 'SISTEMA DE WORKSHOP - RELATORIO'
0190   SKIP 1
0200 WRITE TRAILER / '-' (79)
0210 'PAGE:' *PAGE-NUMBER (AD=L)
```

# Exemplo

```
0220 INPUT MARK *#CNT
0230     //// 7T 'ENTRE COM O VALOR INICIAL'
0240     // 7T 'E O NUMERO DE REGISTROS A SEREM LIDOS NO ARQUIVO DE VEICULOS'
0250     /// 9T 'NUMERO DO REGISTRO INICIAL:' #REG-NUM (AD=AIT'_' )
0260     /// 9T 'NUMERO DE REGISTROS:          '#CNT (AD=AIT'_' )
0270 R1. READ (#CNT) CARS BY REG-NUM STARTING FROM #REG-NUM
0280     DISPLAY 7T 'REGISTRO' REG-NUM MAKE MODEL 'ANO' YEAR
0290 END-READ
0300 WRITE
0310     // 'O NUMERO TOTAL DE REGISTROS PROCESSADOS:' *COUNTER (R1.)
0320 END
```

## Saída:

ENTRE COM O VALOR INICIAL

E O NUMERO DE REGISTROS A SEREM LIDOS NO ARQUIVO DE VEICULOS

NUMERO DO REGISTRO INICIAL: \_\_\_\_\_

NUMERO DE REGISTROS:            22\_\_\_\_

05/02/2003

18:01:28

SISTEMA DE WORKSHOP - RELATORIO

---

REGISTRO	MAKE	MODEL	ANO
A-C 712	CITROEN	BX LEADER	1985
A-KS 731	ALFA ROMEO	SPRINT GRAND PRIX	1984
ABC 345X	AUSTIN	MINI	1980
ABY 449Z	TALBOT	SOLARA	1983
ACD 897X	AUSTIN	MINI	1980
ADF 367Y	LADA	RIVA	1982
ADF 567Y	CITROEN	CV2	1982
AFG 456Y	VAUXHALL	ASTRA 1.3	1982
AFG 563Y	VAUXHALL	CHEVETTE	1982
AIC-H 334	AUDI	90 QUATTRO	1984
ARS 668X	AUSTIN	MINI	1981
ASD 536X	AUSTIN	MINI	1981
A112 DDD	VAUXHALL	ASTRA 1.3	1984

---

PAGE :

1

05/02/2003

18:05:13

SISTEMA DE WORKSHOP - RELATORIO

---

REGISTRO	MAKE	MODEL	ANO
A567 EGH	VAUXHALL	NOVA	1983
A567 VVF	VAUXHALL	NOVA	1983
A612 CGH	FORD	ESCORT 1.3	1984
A632 DGH	FORD	FIESTA	1984
A654 BUN	CITROEN	PALLAS	1983
A654 YHR	VAUXHALL	CAVALIER	1983
A655 FGH	FORD	ESCORT 1.3	1983
A657 ERB	VAUXHALL	ASTRA 1.3	1983
A665 TAT	FORD	ORION 1.6	1983

O NUMERO TOTAL DE REGISTROS PROCESSADOS: 22

---

PAGE:

2

## ***AT TOP OF PAGE***

Executa um processamento quando uma nova página é iniciada. Essa declaração é executada quando o tamanho lógico da página é encontrado ou devido a uma instrução *NEWPAGE*. A saída é impressa uma linha abaixo do título.

## ***AT END OF PAGE***

Executa um processamento quando o fim da página é detectado ou devido a instrução *SKIP*, *NEWPAGE*. A checagem do fim da página é feita após o processamento das instruções *DISPLAY* ou *WRITE*. A saída é impressa após qualquer informação de rodapé.



## EXEMPLO:

```
0010 *****
0020 * ILUSTRA O USO DA INSTRUCAO AT TOP OF PAGE E AT END OF PAGE
0030 *****
0040 DEFINE DATA LOCAL
0050 1 VIEWEMP VIEW OF EMPLOYEES
0060 2 NAME
0070 2 FIRST-NAME
0080 2 CITY
0090 END-DEFINE
0100 *
0110 WRITE TITLE '*****TITULO*****'
0120 SKIP 2
0130 WRITE TRAILER '*****RODAPE*****'
0140 *
0150 AT TOP OF PAGE
0160     WRITE ' ESTE EH O LUGAR ONDE EH IMPRESSO O TOPO DA PAGINA'
0170 END-TOPPAGE
0180 AT END OF PAGE
0190     WRITE ' ESTE EH O LUGAR ONDE EH IMPRESSO O FIM DA PAGINA'
0200 END-ENDPAGE
```

## AT TOP OF PAGE / AT END OF PAGE

```
0210 *
0220 READ (10) VIEWEMP BY NAME
0230   AT START OF DATA
0240     WRITE ' INICIO DOS DADOS...'
0250   END-START
0260   DISPLAY NAME FIRST-NAME CITY
0270   AT END OF DATA
0280     WRITE ' FIM DOS DADOS '
0290   END-ENDDATA
0300 END-READ
0310 END
```

# AT TOP OF PAGE / AT END OF PAGE

\*\*\*\*\*TITULO\*\*\*\*\*

ESTE EH O LUGAR ONDE EH IMPRESSO O TOPO DA PAGINA

NAME

FIRST-NAME

CITY

-----  
INICIO DOS DADOS...

ABELLAN	KEPA	MADRID
ACHIESON	ROBERT	DERBY
ADAM	SIMONE	JOIGNY
ADKINSON	PHYLLIS	BEVERLEY HILLS
ADKINSON	HAZEL	NEW YORK
FIM DOS DADOS		

\*\*\*\*\*RODAPE\*\*\*\*\*

ESTE EH O LUGAR ONDE EH IMPRESSO O FIM DA PAGINA

## Instrução *PRINT*

Produz a saída de relatórios em formato livre. Assemelha-se à instrução *WRITE*, mas é diferente em função de 3 aspectos:

- Zeros a esquerda e brancos desnecessários são suprimidos;
- A impressão de cada campo é feita de acordo com o tamanho do valor campo;
- Se o tamanho da linha é excedido, a impressão continuará na próxima linha.

# Instrução *PRINT*

## Exemplo:

```
0010 *****
0020 ** ILUSTRA O USO DA DECLARACAO PRINT
0030 *****
0040 DEFINE DATA LOCAL
0050 1 VIEWEMP VIEW OF EMPLOYEES
0060 2 NAME
0070 2 FIRST-NAME
0080 2 JOB-TITLE
0090 END-DEFINE
0100 *
0110 READ (4) VIEWEMP BY NAME STARTING FROM 'A'
0120 WRITE NOTITLE /
0130 'EXEMPLO DO WRITE =' NAME 'TRABALHA COMO UM ' JOB-TITLE
0140 PRINT
0150 'EXEMPLO DO PRINT =' NAME 'TRABALHA COMO UM ' JOB-TITLE
0160 END-READ
0170 END
```

# Instrução *PRINT*

## Saída:

```
EXEMPLO DO WRITE = ABELLAN          TRABALHA COMO UM  
MAQUINISTA
```

```
EXEMPLO DO PRINT = ABELLAN TRABALHA COMO UM MAQUINISTA
```

```
EXEMPLO DO WRITE = ACHIESON        TRABALHA COMO UM  
DATA BASE ADMINISTRATOR
```

```
EXEMPLO DO PRINT = ACHIESON TRABALHA COMO UM DATA BASE ADMINISTRATOR
```

```
EXEMPLO DO WRITE = ADAM            TRABALHA COMO UM  
CHEF DE SERVICE
```

```
EXEMPLO DO PRINT = ADAM TRABALHA COMO UM CHEF DE SERVICE
```

```
EXEMPLO DO WRITE = ADKINSON        TRABALHA COMO UM  
DBA
```

```
EXEMPLO DO PRINT = ADKINSON TRABALHA COMO UM DBA
```

## Unidade B - Agrupando os dados de saída

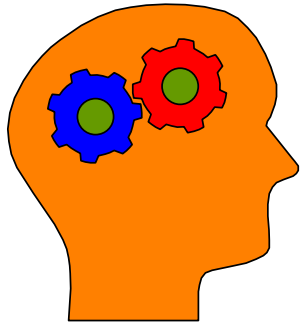
- Instrução ***SORT***
- Processamento de quebra automático
- Instruções de quebra
- Funções do Sistema

## Instrução *SORT*

Executa uma operação de classificação nos registros. O uso do programa de classificação online do Natural possui uma limitação quanto ao tamanho do *buffer sort* da *user work area*. Essa declaração possui as seguinte cláusulas:

Clausula	Descrição
<b>BY</b>	Especifica até 10 campos de classificação.
<b>USING</b>	Indica quais campos escrever para um armazenamento intermediário além dos campos classificados. Se nenhum campo adicional for solicitado, a cláusula USING KEYS deve ser definida.
<b>GIVE</b>	Define qualquer função do sistema Natural a ser avaliada no SORT.





## Lembre-se!

- Antes de usar a declaração *SORT*, você deve fechar todos os processamentos de *loop* com a cláusula *END-ALL*;
- Você deve definir até 10 campos de classificação. Cada campo pode ser classificado em ordem ascendente ou descendente;
- Campos de classificação podem ser variáveis definidas pelo usuário ou campos do banco, sendo escritores ou não;
- A declaração *SORT* inicia seu próprio processamento de *loop* e termina com a declaração *END-SORT*;
- Para processar um número grande de registros use o processamento *batch*.

# Instrução ***SORT***

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO SORT
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMPVIEW VIEW OF EMPLOYEES
0060 2 NAME
0070 2 JOB-TITLE
0080 2 CITY
0090 2 DEPT
0100 END-DEFINE
0110 *
0120 FIND EMPVIEW WITH DEPT = 'TECH10'
0130 END-ALL
0140 SORT BY CITY NAME USING JOB-TITLE
0150 DISPLAY NOTITLE CITY (IS=ON) NAME JOB-TITLE
0160 END-SORT
0170 END
```

# Instrução ***SORT***

CITY	NAME	CURRENT POSITION
ALBUQUERQUE	LINCOLN	ANALYST
ANN ARBER	MARON	ANALYST
ARGONNE	ROLLING	DBA
ARLINGTON	LORIE	PROGRAMMER
AUSTIN	HALL	ANALYST
BALTIMORE	ALEXANDER	PROGRAMMER
	ZINN	PROGRAMMER
BEDFORD	BONNER	ANALYST
BERKELEY	SMITH	ANALYST
BEVERLY HILLS	ADKINSON	DBA
BIRMINGHAM	GUTENBERG	ANALYST
BOSTON	PERREAULT	PROGRAMMER
	STANWOOD	PROGRAMMER
BOULDER	DAY	PROGRAMMER
BUFFALO	CHU	ANALYST
CAMDEN	FORRESTER	PROGRAMMER
	JONES	DBA

## ***AT START OF DATA***

Define o processamento que irá ser executado logo após a leitura do primeiro registro. Essa declaração deve estar dentro de um *loop* de processamento. Exemplo:

```
READ VIEW-NAME  
    AT START OF DATA  
        (PROCESSING STATEMENTS)  
    END-START  
END-READ
```

## ***AT END OF DATA***

Define o processamento que irá ser executado logo após todos os registros do *loop* de processamento terem sido processados. Essa declaração deve estar dentro de um *loop* de processamento. Exemplo:

```
READ VIEW-NAME
  AT END OF DATA
    (PROCESSING STATEMENTS)
  END-ENDDATA
END-READ
```

## ***AT BREAK***

Define um processamento a ser executado sempre que um controle de quebra ocorrer; ou seja, sempre que o valor de um campo definido para essa declaração muda. Geralmente esses campos são campos do banco. Exemplo:

```
READ VIEW-NAME
  AT BREAK OF DEPT
    (PROCESSING STATEMENTS)
  END-BREAK
END-READ
```

# Processamento de quebra automático

```
0010 *****
0020 ** ILUSTRA O USO DA DECLARACAO AT BREAK
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMP VIEW OF EMPLOYEES
0060   2 FIRST-NAME
0070   2 NAME
0080   2 DEPT
0090   2 REDEFINE DEPT
0100     3 #DEPT (A4)
0110 END-DEFINE
0120 FORMAT PS=21
0130 READ (20) EMP
0140 END-ALL
0150 SORT BY DEPT USING FIRST-NAME NAME
0160   NEWPAGE LESS THAN 5
0170   DISPLAY (AL=15) DEPT FIRST-NAME NAME
0180   AT BREAK DEPT /4/
0190     NEWPAGE LESS THAN 4
0200     PRINT (AD=L) // '-' (79)
```

# Processamento de quebra automático

```
0210      / 'NO DEPT: ' OLD(#DEPT) 'HA ' COUNT(DEPT) 'PESSOAS '  
0220      SKIP 3  
0230      END-BREAK  
0240      AT END OF DATA  
0250      PRINT (AD=L) / '-' (79)  
0260      / 'ESTATISTICAS FINAIS: HA ' COUNT (NAME) 'PESSOAS '  
0270      END-ENDDATA  
0280      END-SORT  
0290      END
```



## Saída:

DEPARTMENT CODE	FIRST-NAME	NAME
COMP01	MICHEL	HEURTEBISE
COMP02	LOUIS	D'AGOSTINO
COMP70	PATRICK	BAILLET
COMP70	MARC	LEROUGE
COMP73	ANDRE	GRUMBACH

-----

NO DEPT: COMP HA 5 PESSOAS

## Saída:

DEPARTMENT CODE	FIRST-NAME	NAME
FINA01	DANIEL	JOUSSELIN
NO DEPT: FINA HA 1 PESSOAS		
MARK06	JEAN-MARIE	MARX
NO DEPT: MARK HA 1 PESSOAS		

# Processamento de quebra automático

DEPARTMENT CODE	FIRST-NAME	NAME
VENT01	JEAN	MONTASSIER
VENT02	ROBERT	CHAPUIS
VENT03	BERNARD	VAUZELLE
VENT04	MICHELE	GUERIN
VENT05	BERNARD	VERDIE
VENT06	ALBERT	CAOUDAL
VENT07	HUMBERTO	MORENO
VENT27	PAUL	GUELIN
VENT29	JACQUELINE	REIGNARD
VENT30	JEAN-CLAUDE	REISKEIM
VENT54	ELISABETH	MAIZIERE
DEPARTMENT CODE	FIRST-NAME	NAME
VENT56	ALEXANDRE	BLOND
VENT59	SIMONE	ADAM
NO DEPT: VENT HA 13 PESSOAS		
ESTATISTICAS FINAIS: HA 20 PESSOAS		

## ***BEFORE BREAK***

Define as instruções a serem executadas antes do controle de quebra, ou seja, antes do valor do campo ser avaliado independente da posição que ocupa no loop. Exemplo:

```
READ VIEW-NAME
    BEFORE BREAK PROCESSING
        (PROCESSING STATEMENTS)
    END-BEFORE
END-READ
```

# Processamento de quebra automático

```
0010 *****
0020 * ILUSTRA O USO DA DECLARACAO BEFORE BREAK PROCESSING
0030 *****
0040 DEFINE DATA LOCAL
0050 1 EMP VIEW OF EMPLOYEES
0060 2 CITY
0070 2 COUNTRY
0080 2 JOB-TITLE
0090 2 SALARY (1)
0100 1 #LOCATION (A20)
0110 END-DEFINE
0120 FORMAT SF=3
0130 READ (10) EMP      BY CITY WHERE COUNTRY = 'USA'
0140  BEFORE BREAK PROCESSING
0150  COMPRESS CITY 'USA' INTO #LOCATION
0160  END-BEFORE
0170  DISPLAY (HC=L) 'LOCATION' #LOCATION JOB-TITLE SALARY(1) (AD=L)
0180  AT BREAK OF #LOCATION
0190  SKIP 1
0200 END-BREAK
0210 END-READ
0220 END
```

## Saída:

LOCATION	CURRENT POSITION	ANNUAL SALARY
-----	-----	-----
ALBUQUERQUE USA	SECRETARY	22000
ALBUQUERQUE USA	MANAGER	34000
ALBUQUERQUE USA	MANAGER	34000
ALBUQUERQUE USA	ANALYST	41000
ANN ARBER USA	MANAGER	36000
ANN ARBER USA	SALES PERSON	30000
ANN ARBER USA	ANALYST	43000
ARGONNE USA	SECRETARY	16000
ARGONNE USA	DBA	41500
ARLINGTON USA	PROGRAMMER	35000

### Informações padronizadas de quebra de nível

O sistema de funções Natural são um conjunto de funções estatísticas e matemáticas que podem ser aplicadas aos dados após o processamento dos registros e antes que a quebra seja avaliada.

Essas funções só podem ser definidas nas declarações *WRITE*, *DISPLAY*, *PRINT*, *COMPUTE* ou *MOVE* que estejam codificadas dentro de blocos de processamento de quebra de nível.

## Processamento de quebra automático

Função do sistema	Retorna essa informação
<b>AVER</b>	Média de todos os valores para um campo.
<b>NAVER</b>	Média de todos os valores para um campo, com exceção dos nulos.
<b>MAX</b>	Valor máximo de um campo.
<b>MIN</b>	Valor mínimo de um campo.
<b>NMIN</b>	Valor mínimo de campo, com exceção dos nulos.
<b>OLD</b>	Valor mais antigo do campo.
<b>SUM</b>	Soma de todos os valores do campo.
<b>TOTAL</b>	Valor total de todos os campos.



# Processamento de quebra automático

Função do sistema	Retorna essa informação
COUNT	Número de passagens pelo processamento de <i>loop</i> .
NCOUNT	Número de passagens pelo processamento de <i>loop</i> , com exceção dos nulos.

## Mapas de Help e Helprotinas

- Criando Mapas de Help
- Criando Helprotinas
- Janelas Pop-Up
- Como as Helprotinas passam os dados

O mapa de *help* é um mapa Natural cuja função é dar ao usuário uma explicação sobre a tela em uso. Geralmente contém textos explicativos, embora seja possível incluir campos de entrada e/ou saída.

- Você pode ligar um *help* ao mapa inteiro (*map level help*);
- Ou, ligar o *help* a um campo particular (*field level help*).

Quando um *help* é solicitado para um campo e esse não tem *help* definido, o Natural exibe o *help* definido em nível de mapa.

### Parâmetro HE

Em nível de mapa, basta fornecer o nome do objeto na tela de “profile” do mapa. Em nível de campo, basta fazer o mesmo na tela de edição do campo. O nome do objeto deve estar entre apóstrofes. Exemplo: HE='XXXHH1'.

Em vez de definir o nome do objeto , você pode usar uma variável alfanumérica que contenha o nome do objeto (ex.: #HELP-OBJ). Nesse caso não é necessário o uso de apóstrofes.

## Criando Mapas de *Help*

Os objetos do tipo *help* que você criou podem ser chamados tanto pelo usuário como pelo programa.

### Controle do usuário

O usuário pode solicitar um *help* digitando uma marca em determinada posição do campo, no mapa; ou pressionando uma tecla chave que ative o *HELP* (ex.: PF1).

### Controle do Programa

O programador pode definir a instrução *REINPUT USING HELP* para solicitar o *help*. Exemplo:

```
IF &= ' ' THEN
    REINPUT USING HELP MARK *&
END-IF
```

# Criando Mapas de Help

DATA: 04/02/2003

HORA: 12:49

MAPA: DIAMIAE

PROGRAMA: DIAPINC

INCLUSAO

CODIGO: 1111

SOBRENOME: Santos\_\_\_\_\_ NOME: Paula\_\_\_\_\_

CIDADE: ?

CARGO: \_

ESCOLHA A CIDADE

- 1 - ARARAQUARA
- 2 - ITU
- 3 - POA
- 4 - PINDAMONHANGABA
- 5 - SANTANA DO PARNAIBA

OPCAO: \_

SALARIO

BONUS

SALARIO

BONUS

PF3 - ENCERRA

PF5 - MENU

# Criando Mapas de *Help*

DATA: 04/02/2003

HORA: 12:53

MAPA: DIAMIAE

PROGRAMA: DIAPINC

INCLUSAO

CODIGO: 1111  
SOBRENOME: SANTOS\_\_\_\_\_ NOME: PAULA\_\_\_\_\_  
CIDADE: ARARAQUARA\_\_\_\_\_  
CARGO: ?

SELECCIONE O CARGO COM 'X'

- ACCOUNTANT
- ACCOUNTING ASSISTANT
- ACCOUNTING CLERK
- ACCOUNTING MANAGER
- ADMIN.BASE DE DATOS

SALARIO

BONUS

SALARIO

BONUS

PF3 - ENCERRA

PF5 - MENU

As helprotinas são *helps* iterativos pois auxiliam os usuários a decidir o que colocar na tela. Elas só podem ser chamadas a partir dos mapas. Elas oferecem ao usuário múltiplas telas de *help*, com a possibilidade de escolha de dados e com a opção de dispor as informações em janelas.

As helprotinas podem ser definidas no programa, em nível de declaração e campo; e no mapa, em nível de mapa e campo.



# Criando Helprotinas

```
*****
0020 * DESCRICAO: ROTINA DE AUXILIO PARA A ESCOLHA DO CAMPO CIDADE
0030 * PROGRAMA : DIAHH1
0040 * MAPAS      : DIAMIAE, DIAMH1
0050 * DATA      : 04/06/2001          AUTOR: DIANA PATRICIA
0060
*****
0070 DEFINE DATA
0080 PARAMETER
0090 1 #CIDADE-ESCOLHIDA (A20)
0100 LOCAL
0110 1 #OPT              (N1)
0120 1 #I                (I1)
0130 1 #CIDADE (A20/5) INIT (1) <'ARARAQUARA'>
0140                      (2) <'ITU'>
0150                      (3) <'POA'>
0160                      (4) <'PINDAMONHANGABA'>
0170                      (5) <'SANTANA DO PARNAIBA'>
0180 END-DEFINE
0190 DEFINE WINDOW HELP
0200 SIZE AUTO
```

# Criando Helprotinas

```
0210 BASE CURSOR
0220 TITLE 'ESCOLHA A CIDADE'
0230 CONTROL WINDOW
0240 FRAMED ON
0250 INPUT WINDOW ='HELP' USING MAP 'DIAMH1'
0260 IF NOT (#OPT=0 OR #OPT GT 5)
0270     MOVE #CIDADE(#OPT) TO #CIDADE-ESCOLHIDA
0280 END-IF
0290 END
```

Uma janela é um segmento de uma página lógica, construída por um programa, exibida na tela do usuário. O tamanho da janela pode ser definido pelo cláusula *SIZE* da declaração *DEFINE WINDOW*.

Essa declaração é usada para definir o tamanho, a posição, e os atributos da janela. Exemplo:

```
DEFINE WINDOW NICE-WIN  
    SIZE AUTO  
    BASE CURSOR  
    TITLE 'A NICE WINDOW'  
    CONTROL WINDOW  
    FRAMED ON
```

A instrução *DEFINE WINDOW* não ativa a janela, apenas define. Para ativar a janela usa-se a clausula *SET WINDOW* ou a clausula *WINDOW* da declaração *INPUT*. Exemplo:

```
INPUT WINDOW = 'WINDOW-NAME'
```

```
INPUT WINDOW = 'WINDOW-NAME' USING MAP 'MAP-NAME'
```

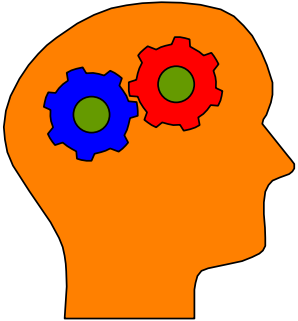
As *help* rotinas só podem acessar a GDA corrente. Mas, os dados podem ser passados via parâmetros. Uma *help* rotina pode ter até 20 parâmetros explícitos e um implícito.

### Parâmetro explícito

```
HE= 'MYHELP' , #HELPME
```

### Parâmetro implícito

```
INPUT #A (A5) (HE= 'YOURHELP' , #HELPME)
```



## Lembre-se!

- Quando um *help* é chamado, a *help* rotina é chamada antes dos dados serem passados da tela para área de dados. Isto significa que as *help* rotinas não podem acessar os dados de entrada dentro da mesma transação de tela.
- Uma vez completo o processamento de *help*, os dados da tela são atualizados. Qualquer campo modificado por uma rotina de *help* é atualizado (com exceção dos campos modificados pelos usuários antes da chamada da *help* rotina).

## Atualizando a Base de Dados

- Store, Update e Delete
- Proteção dos Dados
- Transações Lógicas
- Exemplos

As instruções para modificações no Banco são *STORE*, *UPDATE* e *DELETE*. Elas operam em nível de registro individualmente.

Declaração	Descrição
<b>STORE</b>	Adiciona um novo registro e valores ao campo do registro.
<b>UPDATE</b>	Atualiza valores de campos em um registro existente. Para que o registro seja atualizado, ele deve ser acessado e colocado em “hold” antes que a atualização ocorra.
<b>DELETE</b>	Exclui todo o registro. Para que o registro seja excluído, ele deve ser acessado e colocado em “hold” antes que a exclusão ocorra.



### Declarações de *Hold*

Qualquer registro a ser atualizado ou excluído deve, em primeiro lugar, ser lido e posto em “hold”. Usando o registro é colocado em “hold” para um usuário, ele se torna indisponível para atualização/exclusão para outro usuário.

No Natural o registro é colocado em “hold” se estiver dentro de um *loop* de processamento das declarações READ e FIND. Caso a instrução GET seja usada para acessar o registro, deve-se usar um *label* para referenciar o número da linha onde o GET esta sendo emitido.

### Ordem do Processamento

1. Uma solicitação é feita para colocar o registro em “hold” a fim de proceder a atualização;
2. Uma exigência é feita para verificar se alguém já prendeu o registro;
3. Se ninguém estiver “prendendo” o registro, ele é colocado em “hold”, e a atualização/exclusão pode seguir;
4. Se alguém estiver “prendendo” o registro, seu DBMS pode tentar colocar o registro em “hold” novamente ou enviar uma mensagem de erro informando que o registro está “preso”.

Uma transação lógica consiste de um ou mais comandos que juntos provocam a atualização do banco. Isso envolve a modificação de um ou mais registros. Uma transação lógica começa com o 1.º comando que coloca o registro em “hold” e termina quando a declaração *ET (END TRANSACTION)* ou *BT (BACKOUT TRANSACTION)* é emitida.

O tempo limite para efetuar uma transação é definido pelo DBA. Se a transação ultrapassar esse tempo, o usuário será “lançado” ao último ET.

## Transações Lógicas

Declaração	Descrição
<b>END TRANSACTION</b> (ET)	Encerra as modificações do banco. A localização da declaração ET depende do DBMS.
<b>BACKOUT TRANSACTION</b> (BT)	Desconsidera todas as modificações a partir do último ET.

## Exemplo:

```
0010 *****
0020 * ILUSTRA O USO DA DECLARAÇÃO STORE
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 PERSONNEL-ID
0070 2 MAKE
0080 2 MODEL
0090 2 COLOR
0100 2 YEAR
0110 1 #PID (A8)
0120 1 #ADD (A4)
0130 END-DEFINE
0140 INPUT (AD='_') 'ENTER THE OWNER ID:' #PID
0150 IF #PID=SCAN 'QUIT'
0160 STOP
0170 END-IF
0180 *
```

## Exemplo:

```
0190 FIND. FIND CARS WITH PERSONNEL-ID = #PID
0200     IF NO RECORDS FOUND
0210         MOVE #PID TO CARS.PERSONNEL-ID
0220         INPUT (AD='_' ) 10X 'ADD RECORD'
0230         // 'ID NUMBER:' CARS.PERSONNEL-ID (AD=0)
0240         / 'MAKE         :' CARS.MAKE
0250         / 'MODEL        :' CARS.YEAR
0260         / 'YEAR         :' CARS.YEAR
0270         / 'COLOR        :' CARS.COLOR
0280         /// 'DO YOU WANT TO ADD THIS RECORD:' #ADD
0290 DECIDE ON FIRST VALUE OF #ADD
0300     VALUE 'YES'
0310         STORE CARS
0320         END TRANSACTION
0330     VALUE 'NO'
0340     IGNORE
0350     VALUE 'QUIT'
0360     STOP
```

## Exemplo:

```
0370  NONE VALUE
0380  REINPUT 'PLEASE ENTER "YES", "NO", OR "QUIT" '
0390  END-DECIDE
0400  END-NOREC
0410  END-FIND
0420  IF *NUMBER (FIND.) > 0
0430  REINPUT 'PLEASE ENTER NEW ID NUMBER, RECORD ALREADY EXISTS'
0440  END-IF
0450  END
```

## Exemplo:

```
0010 *****
0020 * ILUSTR A O USO DA DECLARAÇÃO UPDATE E DELETE
0030 *****
0040 DEFINE DATA
0050 GLOBAL USING EMPLGDA
0060 LOCAL
0070 1 #LNAME (A20)
0080 1 #OPTION (A01)
0090 1 #CTLVAR1 (C)
0100 1 #CTLVAR2 (C) INIT <(AD=I CD=GR)>
0110 1 #MESSAGE (A60)
0120 END-DEFINE
0130 REPEAT
0140 INPUT
0150 ///'PLEASE ENTER A LAST NAME: ==> ` #LNAME (AD=AILT'_' )
0160 / `OR ENTER THE WORD' ` "QUIT" ' (CD=RE)
0170 IF #LNAME=' ` THEN
0180 REINPUT `PLEASE ENTER A LAST NAME OR "QUIT" .' MARK #LNAME
0190 END-IF
```



## Exemplo:

```
0200     IF #LNAME = 'QUIT'
0210         WRITE NOTITLE 10/6 'YOU HAVE REQUESTED TO END YOUR SESSION' *USER
0220         '...' / 6T 'HAVE A NICE DAY!'
0230         STOP
0240     END-IF
0250 F1. FIND (1) EMPL-VIEW WITH NAME = #LNAME
0260     IF NO RECORDS FOUND
0270         REINPUT 'EMPLOYEES:1:NOT FOUND. RE-ENTER NAME OR QUIT.' , #LNAME
0280     END-NOREC
0290     INPUT USING MAP 'CNTLMAP1'
0300     DECIDE ON FIRST VALUE OF #OPTION
0310         VALUE 'Q'
0320         ESCAPE BOTTOM
0330         VALUE 'U'
0340         UPDATE (F1.)
0350         END OF TRANSACTION
0360         MOVE 'UPDATE DONE' TO #MESSAGE
0370         MOVE (CD=RE AD=P) TO #CTLVAR2
```

## Exemplo:

```
0380     VALUE 'D'
0390         DELETE(F1.)
0400         END OF TRANSACTION
0410     MOVE 'DELETE DONE' TO #MESSAGE
0420     MOVE (CD=NE AD=P) TO #CTLVAR2
0430     NONE
0440     REINPUT 'CORRECT VALUES ARE D (DELETE), U (UPDATE), Q (QUIT).'
```

No exemplo a seguir, se não usarmos a declaração GET, todos os registros lidos seriam colocados em “hold”. Usando o GET para fazer uma re-leitura somente daqueles registros que satisfazem ao critério estabelecido, somente esses seriam considerados.

Essa técnica necessita de uma lógica adicional para assegurar a integridade dos dados entre as leituras.

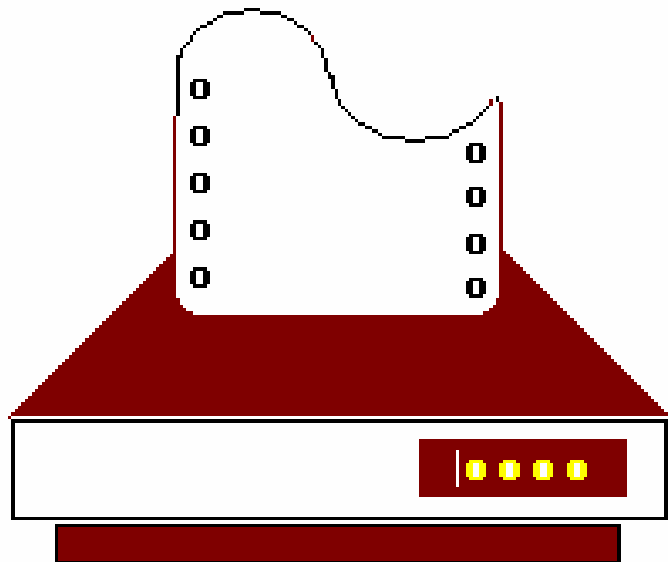
A instrução GET não está disponível para todos os DBMS.

# Reduzindo o número de registros em *hold*

```
0010 *****
0020 * ILUSTRA O USO DA DECLARAÇÃO GET PARA REDUZIR O NÚMERO DE REGISTROS
0030 *****
0040 DEFINE DATA LOCAL
0050 1 CARS VIEW OF VEHICLES
0060 2 PERSONNEL-ID
0070 2 MAKE
0080 2 MODEL
0090 2 CLASS
0100 2 REG-NUM
0110 2 MAINT-COST (5)
0120 END-DEFINE
0130 READ CARS BY ISN
0140 IF CLASS = 'C' /* COMPANY CAR
0150     GET. GET CARS *ISN /* RE-READ LAST RECORD READ
0160     MAINT-COST(*) := 0 /* NO MAINTENANCE IF COMPANY CAR
0170     UPDATE RECORD (GET.) /* REFER BACK TO THE RECORD OBTAINED WITH GET
0180     END TRANSACTION
0190 END-IF
0200 END-READ
0210 END
```

## Processamento Batch

- Visão Geral do Processamento Batch (mainframe)
- Instrução Define Work File
- Instrução Write Work File
- Instrução Read Work File



**PROCESSAMENTO BATCH**



**PROCESSAMENTO ONLINE**

Algumas funções na aplicação não exigem a interação do usuário para que o controle seja passado ao Natural. A maioria das funções não-iterativas requerem tempo de processamento alto e deveriam ser executadas em modo batch, pois executar um programa que lê e grava centenas de milhões de registros ou que produz relatórios enormes torna o tempo de resposta da máquina muito mais lento.

## Funções não-iterativas

Algumas razões para usar o processamento batch acontecem quando:

- ◆ um programa executa uma classificação numerosa (sort);
- ◆ seu programa é executado à noite sem a interferência do usuário;
- ◆ conjuntos de dados seqüenciais são criados;
- ◆ longos relatórios devem ser gerados.



### Como usar o Processamento Batch

Quando um programa Natural é executado em modo batch, um núcleo Natural separado é inicializado para disponibilizar os recursos que permitirão o uso das funções batch. Isso acontece quando um job é submetido. Assim o procedimento geral que você deve seguir, envolve:

- ◆ a submissão de um job batch;
- ◆ o chamado do Natural batch e
- ◆ a execução de um programa Natural em modo batch.

## O que deve-se definir no job?

- ◆ Entrada e saída dos datasets;
- ◆ As impressoras de destino;
- ◆ Os Work files;
- ◆ O arquivo Adabas que armazena os programas do sistema Natural;
- ◆ A biblioteca de logon;
- ◆ Os programas Natural a serem executados.

### **O que são nomes de link?**

São palavras reservadas que definem uma função particular e variam conforme a aplicação.

### **Parâmetros Natural Batch**

Funcionam da mesma forma que os parâmetros online. Exemplo: PS (Page Size), serve para controlar o tamanho da página que será exibida.

## Nomes de link

<b>Descrição</b>	<b>Nomes de Link-OS</b>	<b>Nomes de Link-OS</b>
<b>Primary Input</b>	CMSYNIN	SYSRDR
<b>Object Input</b>	CMOBJIN	SYSIPT
<b>Primary Report</b>	CMPRINT	SYSLST
<b>Additional Reports</b>	CMPRTnn	SYS041
<b>Work Files</b>	CMWKFnn	SYS001...SYS032

## Parâmetros Natural Batch

<b>NATPARM</b>	<b>Função</b>	<b>Valor</b>
<b>INTENS</b>	Intensidade p/ saída de relatórios	1-10
<b>IM</b>	Modo para entrada de terminal	F/D
<b>PS</b>	Tamanho da página	5-250
<b>LS</b>	Tamanho da linha	35-250
<b>MT</b>	Limite do tempo de CPU	No limit
<b>MAXCL</b>	Máximo de objetos eternos chamados entre telas gravadas	No limit
<b>MADIO*</b>	Máximo de chamadas ao Adabas entre gravações.	No limit

\* Só faz sentido se você estiver usando arquivos Adabas.

## Sintaxe

**DEFINE WORK FILE** *n operand1* [**TYPE** *operand2*]

## Definição

É usada para atribuir um nome a um arquivo de trabalho Natural. Isso permite que você construa ou altere as definições de um arquivo dinamicamente dentro da sessão. Quando essa instrução for executada e o arquivo já estiver aberto, ela provocará, implicitamente, o fechamento do arquivo. Ele possui um número que o identifica. Esse número varia de 1 até 32.

## Sintaxe

**WRITE WORK [FILE]** *work-file-number* **[VARIABLE]** *operand1...*

## Definição

É usada para gravar registros na seqüência física num arquivo. Em ambientes mainframe só pode ser usada em modo batch sob COM-PLETE, TSO, CMS e TIAM. É possível gravar registros com diferentes campos no mesmo work file usando diferentes instruções WRITE WORK FILE. Nesse caso, a variável de entrada deve ser definida em todas as instruções WRITE WORK FILE.

## Sintaxe

**READ WORK [FILE] *work-file-number* [ONCE]**

**RECORD *operand1***  
**[AND] [SELECT]    *OFFSET n***  
**{ *FILLER nx ... operand2 ...* }**  
**{ *[GIVING LENGTH operand3]* }**

**AT [END] [OF] [FILE]**

*statement...*  
**END-ENDFILE**  
*statement...*

**END-WORK**



## Definição

É usada para ler dados de um arquivo seqüencial não-Adabas. Os dados são lidos na ordem física independente da forma como foram gravados. Em ambientes mainframe só pode ser usada sob COM-PLETE, TSO, CMS e TIAM ou em modo batch. Ao ser lido, o JCL deve ser fornecido.

Essa instrução provoca um loop de processamento, assim, os processos que quebra automática devem ser executados dentro do loop de leitura. Quando a condição de fim de arquivo ocorre, o Natural automaticamente fecha o work file.

## WRITE WORK FILE, READ WORK FILE, DEFINE WORK FILE

local

1 #registro (a50)

1 redefine #registro

2 #reg

3 cidade (a8)

3 filler 2x

3 nome (a20)

2 codigo\_emp (n3)

1 emp view of employees

2 codigo\_emp

2 nome

2 cidade

```
define work file 1 'c:\w-trab\arq1.txt'
read emp by city
    move by name emp to #reg
    write work file 1 #registro
end-read
.....
read work file 1 [once] #registro
    move by name #reg to emp
    store emp... end-transaction
end-work
```

## Editores do *Mainframe*

- Editor de Programa
- Editor da Área de Dados
- Editor de Mapa

## Editor de Programa

Para entrar no editor de programa do *mainframe* basta selecionar a opção *Create -> Object* no menu principal e depois informar o tipo e o nome do objeto. Uma outra alternativa é entrar com o comando *e (editing) p (program)* na linha de comando.

# EDITOR DO MAINFRAME

19:55:30 \*\*\*\*\* NATURAL \*\*\*\*\* 2003-02-26

User CTDPS - Development Functions - Library SYSTEM  
Mode Reporting  
Work area empty

Code Function

C Create Object  
E Edit Object  
R Rename Object  
D Delete Object  
X Execute Program  
L List Object(s)  
S List Subroutines Used  
? Help  
. Exit

Code .. **c** Type .. **p**  
Name .. **progl**\_\_\_\_\_

Command ==>

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---  
Help Menu Exit Canc

## Comandos de linha

- Marcar linhas **.X, .Y;**
- Copiar linhas **.C(nn), .CX(nn), .CY(nn), .CX-Y(nn);**
- Mover linhas **.MX, .MY, .MX-Y;**
- Inserir linhas **.I, .I(nn), .I(pgm-name[,sss,nn]);**
- Excluir linhas **.D, .D(nn);**
- Dividir uma linha **.S;**
- Unir uma linha **.J;**
- Cancelar todas as mudanças de uma linha **L.**

### Comandos Adicionais

- ADD - Adiciona linhas em branco no fim ou no início do programa;
- CLEAR - Limpa a área de trabalho-fonte;
- DELETE - Permite ao usuário escolher entre excluir o código-fonte, o código-objeto ou ambos;
- PURGE - Exclui o código-fonte do programa;
- SCRATCH - Exclui o código objeto do programa;
- LAST - Mostra os últimos comandos emitidos com a possibilidade de reexecução.

## Comandos Adicionais

- **RENAME** - Usado para dar um novo nome ao objeto;
- **LIST VIEW** - Emite uma lista com os diversos tipos de objetos. Quando apenas um objeto é especificado, o comando listará o conteúdo desse objeto;
- **SET TYPE** - Troca o tipo de objeto Natural;
- **RENUMBER** - Usado para renumerar as linhas do programa-fonte;
- **STRUCT** - Gera o fonte tabulado na área de trabalho;



## Subcomandos do Editor

- Excluir um conjunto de linhas:

DX, DY, DX-Y;

EX, EY, EX-Y.

- Obter mais linhas para o programa:

*ADD.*

- Cancelar as marcas X e Y:

*RESET.*

## Subcomandos do Editor

- Renumerar o programa:

N.

- Exibir/Repetir último comando:

\* ou \*=.

- Subcomandos de posição:

T, --; B, ++; +P, -P, +, -; +h, -H; +nnnn, -nnnn; nnnn;

X, Y.

### Subcomandos **SCAN** e **CHANGE**

O **SCAN** é usado para procurar dados contidos na área do fonte. Sintaxe: **SCAN**['*scan-value*'].

O **CHANGE** é usado para pesquisar um valor digitado e substituir cada ocorrência localizada. Sintaxe: **CHANGE**'*scandata*'*replacedata*'.

### Editor da Área de Dados

Para entrar no editor da área de dados do *mainframe* basta selecionar a opção *Create -> Object* no menu principal e depois informar o tipo e o nome do objeto. Uma outra alternativa é entrar com o comando *e (editing) l (local), g (global) ou p (parameter)* na linha de comando.

# EDITOR DO MAINFRAME

```
20:23:58          ***** NATURAL *****          2003-02-26
User CTDPS        - Development Functions -          Library SYSTEM
                                                         Mode Reporting
                                                         Work area empty

Code  Function

C    Create Object
E    Edit Object
R    Rename Object
D    Delete Object
X    Execute Program
L    List Object(s)
S    List Subroutines Used
?    Help
.    Exit

Code .. c      Type .. l
Name .. local1_____

Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help  Menu  Exit                                     Canc
```

## Descrição das colunas do editor da Área de Dados

Local	LOCAL1	Library	SYSTEM	DBID	23	FNR	231
Command							> +
<b>I T L</b>	<b>Name</b>			<b>F</b>	<b>Leng</b>	<b>Index/Init/EM/Name/Comment</b>	
All	-	-	-	-	-	-	-

Coluna	Informação
I	<p>Campo de informação (não modificável pelo usuário):</p> <p>E=erro detectado</p> <p>I=valores de inicialização definido</p> <p>M=máscara de edição definida</p> <p>S=máscara de edição e valores de inicialização definidos</p>

Coluna	Informação
T	Tipo de campo (type): branco= campo elementar G= campo grupo P= campo grupo periódico M= campo múltiplo *= linha de comentário V= visão dos dados C= constante B= bloco de dados R= campo redefinido

Coluna	Informação
L	Nível do campo na estrutura de dados (level):1 a 9
Name	Nome da variável, do bloco, da view ou opção de FILLER (Nx) 0<n<25. Mínimo 3 caractere e máximo de 32 caracteres.
F	Formato do campo.
Length	Tamanho do campo. Não preencher para formatos C, D, T e L.
Index/Init/EM Name/ Comment	Contém comentários (/*) ou definições tais como: inicializações (init <3>), máscaras de edições (EM=999.99) e definições de tabelas (2,2).





# EDITOR DO MAINFRAME

Local	LLL	Library	DIANA	DBID	23	FNR	231
View EMPLOYEES							
I	T	L	Name	F	Leng	Index/Init/EM/Name/Comment	
X	2		PERSONNEL-ID	A	8		
	G 2		FULL-NAME				
X	3		FIRST-NAME	A	20		
	3		MIDDLE-I	A	1		
X	3		NAME	A	20		
	2		MIDDLE-NAME	A	20		
X	2		MAR-STAT	A	1		
	2		SEX	A	1		
	2		BIRTH	A	6		
	G 2		FULL-ADDRESS				
	M 3		ADDRESS-LINE	A	20	(1:191) /* MU-FIELD	
X	3		CITY	A	20		
	3		ZIP	A	10		
	3		POST-CODE	A	10		
X	3		COUNTRY	A	3		
	G 2		TELEPHONE				
	3		AREA-CODE	A	6		



## Redefinição dos campos

Local	DIALDA1	Library	DIANA	DBID	23	FNR	231
Command							> +
I T L	Name			F	Leng	Index/Init/EM/Name/Comment	
All	-	-----	-	-----	-----	-----	-----
V	1	EMPL				EMPLOYEES	
	2	PERSONNEL-ID		A	8		
	2	FIRST-NAME		A	20		
.	r	NAME		A	20		
	2	CITY		A	20		
	2	JOB-TITLE		A	25		
	2	SALARY		P	9.0	(1:3)	
M	2	BONUS		P	9.0	(1:3,1:2)	

# EDITOR DO MAINFRAME

```
Local      DIALDA1  Library DIANA          DBID    23 FNR    231
Command                                         > +
I T L Name                                     F Leng Index/Init/EM/Name/Comment
All - ----- - -----
  V 1 EMPL                                     EMPLOYEES
    2 PERSONNEL-ID                            A    8
    2 FIRST-NAME                              A   20
    2 NAME                                     A   20
  R 2 NAME                                     /* REDEF. BEGIN : NAME
    3 #LETRA-INICIAL                          A    1 /* LETRA INICIAL DO SOBRENOME
    2 CITY                                     A   20
    2 JOB-TITLE                              A   25
    2 SALARY                                  P  9.0 (1:3)
  M 2 BONUS                                  P  9.0 (1:3,1:2)
```

```
0010 DEFINE DATA
0020 LOCAL USING DIALDA1
0030 END-DEFINE
0040 READ (10) EMPL
0050   DISPLAY PERSONNEL-ID NAME CITY #LETRA-INICIAL
0060   END-READ
0070 END
```

### Editor de Mapa

Para entrar no editor de mapa do *mainframe* basta selecionar a opção *Create -> Object* no menu principal e depois informar o tipo de objeto “m” e o nome do objeto. Uma outra alternativa é entrar com o comando *e (editing) m (map)* na linha de comando. Você será direcionado para o editor de mapa.

# EDITOR DO MAINFRAME

```
20:23:58          ***** NATURAL *****          2003-02-26
User CTDPS        - Development Functions -          Library SYSTEM
                                                         Mode Reporting
                                                         Work area empty

Code  Function

C    Create Object
E    Edit Object
R    Rename Object
D    Delete Object
X    Execute Program
L    List Object(s)
S    List Subroutines Used
?    Help
.    Exit

Code .. c      Type .. m
Name .. xxxmap1_____

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Menu  Exit                                     Canc
```

# EDITOR DO MAINFRAME

```
17:25:27          ***** NATURAL MAP EDITOR *****          2003-03-05
User CTDPS          - Edit Map -          Library SYSTEM

      Code      Function
      ----      -
      D      Field and Variable Definitions
      E      Edit Map
      I      Initialize new Map
      H      Initialize a new Help Map
      M      Maintenance of Profiles & Devices
      S      Save Map
      T      Test Map
      W      Stow Map
      ?      Help
      .      Exit

      Code .. I      Name .. XXXMAP1_      Profile .. SYSPROF_

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
              Help          Exit  Test  Edit
```



Ao iniciar um novo mapa, você cairá numa tela que contém as definições do mapa, tais como: tamanho da tela, caracter de preenchimento, os tipos de delimitadores, se o mapa usa algum *layout*, se usa variável de controle, etc. Você pode definir as características do se mapa inicialmente ou depois com o auxílio da tecla PF2.

# EDITOR DO MAINFRAME

```

17:28:20                Define Map Settings for MAP                2003-03-05

Delimiters                Format                Context
-----
Cls Att CD  Del          Page Size ..... 23          Device Check .... 4880
  T  D      BLANK       Line Size ..... 79          WRITE Statement  _
  T  I      ?           Column Shift ... 0 (0/1)      INPUT Statement  X
  A  D      _           Layout ..... _____  Help _____
  A  I      )           dynamic ..... N (Y/N)      as field default N (Y/N)
  A  N      ^           Zero Print ..... N (Y/N)
  M  D      &          Case Default ... UC (UC/LC)
  M  I      :          Manual Skip .... N (Y/N)      Automatic Rule Rank 1
  O  D      +          Decimal Char ... .          Profile Name .... SYSPROF
  O  I      (          Standard Keys .. N (Y/N)
                          Justification .. L (L/R)
                          Print Mode ..... _
                          Control Var .... _____

Filler Characters
-----
Optional, Partial ....
Required, Partial ....
Optional, Complete ...
Required, Complete ...

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help                Exit                                Let
  
```

## Tela do editor de mapa

```
Ob _                               Ob D CLS ATT  DEL      CLS ATT  DEL
.                                  .      T  D   Blnk    T  I   ?
.                                  .      A  D   _       A  I   )
.                                  .      A  N   ^       M  D   &
.                                  .      M  I   :       O  D   +
.                                  .      O  I   (
.
001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10---PF11---PF12---
      Help  Mset  Exit  Test  Edit  --   -   +   Full  <   >   Let
```

### Criação de um mapa básico

O texto é colocado no mapa ao digitarmos o delimitador de texto + palavra;

Campos alfanuméricos são colocados no mapa ao digitarmos o delimitador de campo + X(n), onde 'n' é o tamanho do campo;

Campos numéricos são colocados no mapa ao digitarmos o delimitador de campo + 9(n), delimitador de campo + 9999...n ou delimitador de campo + 0000...n; onde 9 justifica o campo à esquerda e 0 à direita;

campos recebam suas definições.

## Definindo os campos

```
Ob _          Ob D CLS ATT  DEL      CLS ATT  DEL
.             .      T  D   Blnk     T  I   ?
.             .      A  D   _         A  I   )
.             .      A  N   ^         M  D   &
.             .      M  I   :         O  D   +
.             .      O  I   (
.
001  --010---+-----+-----030---+-----+-----050---+-----+-----070---+-----

                novo menu principal

                matricula: )x(20)
                   nome:  )x(20)
                   idade:  )9(2)
                   n.o inscr: )0(5)

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --   -   +   Full  <   >   Let
```

## Após o ENTER

```
Ob _                               Ob D CLS ATT  DEL      CLS ATT  DEL
.                                  .      T  D   Blnk    T  I    ?
.                                  .      A  D   _      A  I    )
.                                  .      A  N   ^      M  D    &
.                                  .      M  I   :      O  D    +
.                                  .      O  I   (
.
001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

                                novo menu principal

                                matricula: )XXXXXXXXXXXXXXXXXXXXX
                                nome: )XXXXXXXXXXXXXXXXXXXXX
                                idade: )99
                                n.o inscr: )00000

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Mset  Exit  Test  Edit  --   -   +   Full  <   >   Let
```

## Ao tentar dar PF3 sem nomear os campos

```
17:56:46                Field and Variable Definitions - Summary                2003-03-05

Cmd Field Name (Truncated)                Mod Format Ar Ru Lin Col
___ #001_____                A20                5  41
___ #002_____                A20                6  41
___ #003_____                N2                 7  41
___ #004_____                N5                 8  41
```

## Definindo os respectivos nomes

```
17:56:46                Field and Variable Definitions - Summary                2003-03-05

Cmd Field Name (Truncated)                Mod Format Ar Ru Lin Col
___ #MATRICULA_____                A20                5  41
___ #NAME_____                A20                6  41
___ #BIRTH_____                N2                 7  41
___ #INSCRICAO_____                N5                 8  41
```

### Salvando e catalogando o mapa

Para salvar o mapa basta sair da tela de edição com PF3 e escolher a opção “S” (*Save Map*) no menu da tela “Natural Map Editor”. Para catalogar, escolha a opção “W” (*Stow Map*).



# EDITOR DO MAINFRAME

```
18:06:22          ***** NATURAL MAP EDITOR *****          2003-03-05
User CTDPS          - Edit Map -          Library SYSTEM

          Code      Function
          ----      -
          D      Field and Variable Definitions
          E      Edit Map
          I      Initialize new Map
          H      Initialize a new Help Map
          M      Maintenance of Profiles & Devices
          S      Save Map
          T      Test Map
          W      Stow Map
          ?      Help
          .      Exit

          Code .. W      Name .. XXXMAP1_      Profile .. SYSPROF_

Command ==>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help      Exit      Test      Edit
STOW command executed successfully.
```

## Testando o mapa

Você poderá testar o mapa diretamente na tela do editor de mapa com a opção PF4, saindo da tela de edição com PF3 e escolhendo a opção “T” (*Test Map*) no menu da tela “*Natural Map Editor*” ou chamando o mapa a partir de um programa através da instrução *INPUT USING MAP* ‘<nome-do-mapa>’.

```
novο menu principal
```

```
matricula: _____  
nome: _____  
idade: ____  
n.o inscr: _____
```

### “Capturando” os campos de outras áreas de dados

Em vez de definir os campos, você pode ainda “puxá-los” de outras áreas com suas respectivas definições. Os tipos válidos são: (P) Programas, (V) *View* (DDM), (G) Área de Dados Global, (L) Área de Dados Local e (M) Mapa. As (S) Subrotinas, (N) Subprogramas e (H) *Helprotinas* só poderão exibir os dados se contiverem a instrução *DEFINE DATA* com uma Área de Dados Local ou *Parameter*.

Basta levar o cursor à primeira posição do campo, fornecer o delimitador adequado e o número do campo que aparece na parte superior tela.

# EDITOR DO MAINFRAME

```
Ob P XXXPG1                               Ob D CLS ATT DEL      CLS ATT DEL
. EMP          *V1      .      T  D   Blnk      T  I   ?
1 NAME        A20      .      A  D   _          A  I   )
2 CITY        A20      .      A  N   ^          M  D   &
3 SALARY      P9.0    .      M  I   :          O  D   +
4 PERSONNEL-ID A8      .      O  I   (
.
001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----
```

MENU PRINCIPAL

IDENTIFICACAO: ) 4

NOME: ) 1

SALARIO: ) 3

## Após o ENTER

```
Ob P XXXPG1                               Ob D CLS ATT DEL      CLS ATT DEL
. EMP                *V1                .   T  D   Blnk      T  I   ?
1 NAME              A20                .   A  D   _       A  I   )
2 CITY              A20                .   A  N   ^       M  D   &
3 SALARY            P9.0                .   M  I   :       O  D   +
4 PERSONNEL-ID     A8                  .   O  I   (
.
.
001  --010-----+-----+-----030-----+-----+-----050-----+-----+-----070-----+-----
                                     MENU PRINCIPAL

IDENTIFICACAO: )XXXXXXXXX
                NOME: )XXXXXXXXXXXXXXXXXXXXXXXXX
                SALARIO: )999999999
```

## Comandos do editor de mapas

Comandos de campo:

- .M - move o campo para a posição do cursor;
- .R - repete o campo na posição do cursor;
- .C - centralizar o campo entre campos adjacentes;
- .D - exclui o campo;
- .T - trunca a linha a partir do campo marcado até o final da linha.

## Comandos de linha:

- ..M, ..Mn, ..M\* - move linha(s) para a posição do cursor;
- ..R, ..Rn - repete linha(s) na posição do cursor;
- ..C, ..Cn, ..C\* - centraliza linha(s);
- ..D, ..Dn, ..D\* - exclui linha(s);
- ..I, ..In, ..I\* - insere linha(s) em branco;
- ..Fc - preenche espaços vazios na linha com o caracter 'c';
- ..S, ..Sn - divide a(s) linha(s) na posição do cursor;
- ..J, ..Jn, ..J\* - une linhas.

### Uso das variáveis do sistema

As variáveis de sistema devem ser usadas com delimitador do tipo O I (output intensified). Eis algumas delas:

- \*USER - A identificação do usuário;
- \*LIBRARY-ID - Nome da atual biblioteca;
- \*PROGRAM - Nome do programa/mapa em execução;
- \*DATU - Data no formato americano;
- \*DAT4E - Data com os quatro dígitos para ano;
- \*TIME - Hora no formato HH:MM:SS.T.



### Regras de Processamento

Trata-se de um grupo de instruções anexadas a uma variável em um mapa externo. É utilizada para realizar a validação do campo ao qual está associada. Uma regra não pode conter a instrução *END*.

### Como colocar regras em campos do mapa

Para associar uma regra a um campo, basta entrar com **.P** sobre o delimitador do campo. Ao digitar *enter*, você será direcionado para o editor de regras.

Para associar uma regra a uma PF, basta entrar com **..P** no início de qualquer linha em branco do editor de mapa.

Para fechar o editor de regras basta digitar **."** Na linha de comando.

O símbolo **"&"** é utilizado na regra para representar o campo para o qual a regra está sendo definida.



# EDITOR DO MAINFRAME

```
Variables used in current map                                     Mod
#IDENT(A8)                                                       U
#NOME(A20)                                                        U
#SALARIO(N5)                                                     U

Rule _____ Field #IDENT
> . > + Rank 0          S 3      L 1      Report Mode
ALL  ....+....10...+....+....+....30...+....+....+....50...+....+....+....70.
    0010 IF '&=' '
    0020 REINPUT 'campo deve ser preenchido!' MARK *&
    0030 END-IF
    0040
    0050
```

## Testando o campo no mapa

MENU PRINCIPAL

IDENTIFICACAO: \_\_\_\_\_

NOME: \_\_\_\_\_

SALARIO: \_\_\_\_\_

campo deve ser preenchido!

### Utilização de MU e PEs em mapas

O usuário pode definir uma tabela e colocar os campos MUs e PEs dentro do mapa através de um programa que contenha na Área de Dados Local todos os campos que pretende utilizar, certificando-se do número de ocorrências para os campos repetidos.

Outra opção é definir os campos no mapa e depois definir as ocorrências e dimensões através da tela de edição que pode ser acessada digitando-se **.A** sobre o delimitador do campo que pretende definir como uma tabela.

# EDITOR DO MAINFRAME

```
Name #SALARIO                               Upper Bnds 3____ 1____ 1____
-----
Dimensions                                Occurrences  Starting from  Spacing
0 . Index vertical                        1____         _____    0   Lines
1 . Index horizontal                      3____         _____    3   Columns
0 . Index (h/v) V                         1____         _____    0   Cls/Ls

001  --010---+-----+-----+---030---+-----+-----+---050---+-----+-----+---070---+-----

                                MENU PRINCIPAL

                                IDENTIFICACAO: )XXXXXXXXX
                                NOME: )XXXXXXXXXXXXXXXXXXXXX

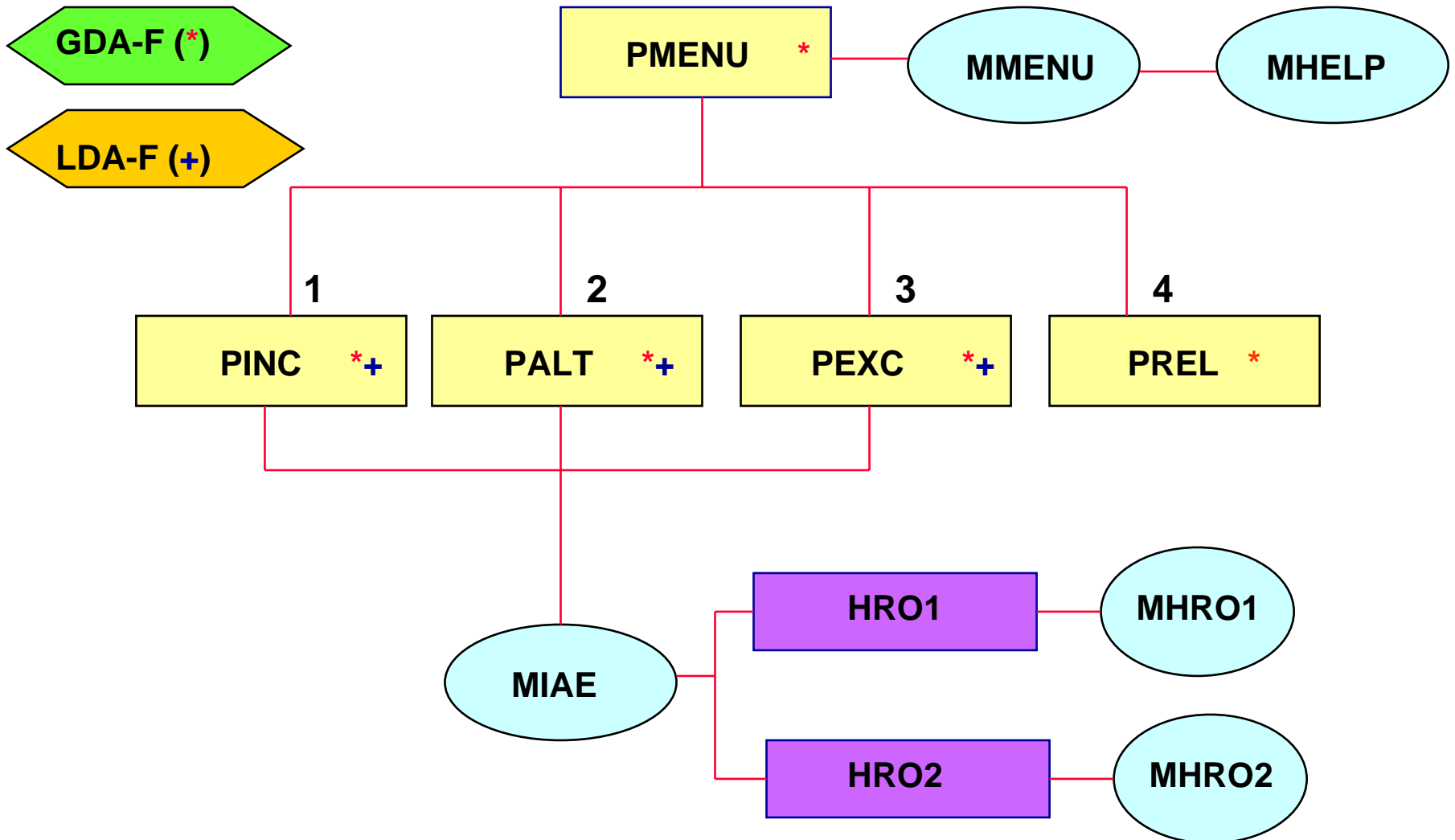
                                SALARIO: .A9999  )99999  )99999
```



# Exercícios



# SISTEMA DE FUNCIONÁRIOS



1) Criar uma Área de Dados Global chamada 'GDA-F' que tenha a variável #MENSAGEM (A30).

2) Criar uma local externa de nome 'LDA-F' com os seguintes campos da DDM EMPLOYEES:

PERSONNEL-ID (codigo),

NAME (sobrenome), FIRST-NAME (nome),

SEX , CITY, JOB-TITLE (cargo),

SALARY (1:3) e

BONUS (1:3,1:2)

O nome da view deve ser alterada para FUNC

### 3) Criar o programa 'PREL'.

- a) Incluir em sua área de dados a local LDA-F utilizando o comando *.I (nome-da-local)*
- b) Alterar o número de ocorrências do salário para uma só ocorrência e retirar o campo bonus
- c) Ler a view utilizando o campo CITY como chave

## d) Layout do relatório PREL

Data: 28/03/2003	Sexta-Feira, 28 de Março de 2003	Pag: 0001			
Hora: 16:39	FUNCIONARIOS CADASTRADOS POR CIDADE	Prog: PREL			
-----					
CIDADE	COD	NOME COMPLETO	S	CARGO	SALARIO
-----					
BELEM	14	MIRANDA MARLUI	F	CANTORA	R\$ 23.121,00
BELO HORIZONTE	12	TISO WAGNER	M	MUSICO	R\$ 45.755,00
	26	BOSCO JOAO	M	COMPOSITOR	R\$ 55.000,00
	41	DIAS WILMA	F	COZINHEIRA	R\$ 32.299,00
BRASILIA	31	PINHEIRO JULIO	M	MINISTRO	R\$ 999.999,99
CAMPO GRANDE	48	MURILO MARCIO	M	ATLETA	R\$ 17.000,00
-----					
PF3 - SAIDA					

```
e) Esquema de programação :  
** programa de relatório PREL  
define data  
    local using LDA-F  
end-define  
set key pf3='%%'  
format ps=21  
write title left ...  
write trailer ...  
read func by city  
    display ...  
end-read  
end
```



## 5) Criar o mapa MMENU usando como layout o mapa MLAY.

**MENU DE EMPREGADOS**

1 - Inclusão

2 - Alteração

3 - Exclusão

4 - Relatório

? - Auxílio

Opção:

Código:  (opções 1, 2 ou 3)

Cidade ini:  (somente opção 4)

Cidade fin:  (somente opção 4)

Definir as seguintes consistências no mapa MMENU:

■ Variável #OPCAO (N1):

if not (#opcao = 1 thru 4)

reinput '...' (cd=re ad=l) mark \*#opcao alarm

end-if

■ Variável #CODIGO (A8):

if #opcao = 1 thru 3

if #codigo = ' '

reinput '...' mark \*#codigo alarm

end-if

fff. find number func with personnel-id = #codigo

end-find



- Variável #CODIGO (continuação) :

```
if *number (fff.) = 0 /* não encontrou o código
  if #opcao = 2 thru 3 /* alteração ou exclusão
    reinput '...' mark *#codigo alarm
  end-if
else /* encontrou o código
  if #opcao = 1 /* inclusão
    reinput '...' mark *#codigo alarm
  end-if
end-if
```

- Variável #CIDADE-INI (A20):

```
if #opcao = 4 /* relatório
  if #cidade-ini = ' '
    reinput '...' mark *#cidade-ini alarm
  end-if
end-if
```

- Variável #CIDADE-FIN(a20):

```
if #opcao = 4 /* relatório
  <<idem #cidade-ini>>
  if #cidade-fin < #cidade-ini
    reinput '...' mark *#cidade-fin alarm
  end-if
  << verifique se existem cidades no intervalo fornecido>>
end-if << usar a instrução histogram>>
```

### 6) Criar o programa PMENU (menu principal) com o seguinte esquema:

```
** programa menu principal PMENU  
define data  
  global using gda-emp  
  local  
    1 ... variáveis do mapa MMENU  
end-define  
set key pf3  
input with text #mensagem using map 'mmenu'  
reset #mensagem  
...
```

## (continuação do programa PMENU)

```
...  
decide on first value of #opcao  
  value 1 fetch 'pinc' #codigo  
  value 2 fetch 'palt' #codigo  
  value 3 fetch 'pexc' #codigo  
  value 4 fetch 'prel' #cidade-ini  #cidade-fin  
  none ignore  
end-decide  
end
```

### 7) Alterar o programa PREL (relatório) :

- **set key pf3='%%' → set key pf3='pmenu'**
- **inserir antes do read → input #cidade-ini #cidade-fin**
- **acrescentar a cláusula *starting from ... thru* na instrução read  
selecionar os registros que estão no intervalo das cidades**
- **inserir no término do relatório → fetch 'pmenu'**

## 8) Criar o mapa MIAE usando como layout o mapa MLAY.

XXXXXXXXXX

**CODIGO** : XXXXXXXXX

**SOBRENOME** :  **NOME** :

**SEXO** :  **CIDADE** :

**CARGO** :

SALARIO	BONUS	SALARIO	BONUS	SALARIO	BONUS
<input type="text"/> (1)	<input type="text"/> (1,1) <input type="text"/> (1,2)	<input type="text"/> (2)	<input type="text"/> (2,1) <input type="text"/> (2,2)	<input type="text"/> (3)	<input type="text"/> (3,1) <input type="text"/> (3,2)

Obs: O primeiro campo é a variável #TITULO (A10), os outros campos deverão ser puxados da local LDA-F.

## Outras observações para o mapa MIAE :

- A variável #TITULO e o campo FUNC.PERSONNEL-ID deverão ser protegidos (tipo “output protected”), os demais campos deverão ser do tipo “output modifiable”
- No rodapé do mapa (onde está situada a **PF3** ) definir também a **PF5 – MENU** e a sua consistência deverá ser :

```
if *pf-key = 'PF5'  
    fetch 'pmenu'  
end-if
```

- Fazer a consistência do campo FUNC.SEX → (M ou F)
- Consistência do campo FUNC.SALARY: Se o usuário preencher qualquer BONUS, obrigar o preenchimento do campo SALARIO correspondente àquele BONUS
- Para os demais campos do tipo “output modifiable” , obrigar o preenchimento, utilizando o “&” no lugar do nome do campo
- Definir a variável de controle #CONTROL para todos os campos do tipo “output modifiable”

### 9) Fazer o programa de inclusão PINC com o seguinte esquema:

```
** programa de inclusão PINC  
define data  
  global using gda-f  
  local using lda-f  
  local /* (outras variáveis do mapa MIAE e seus "init") end-define  
set key pf3 pf5  
input func.personnel-id /* para receber o código vindo do PMENU  
input using map 'MIAE'  
store func  
end transaction  
move 'Funcionário incluído com sucesso' to #mensagem  
fetch 'PMENU'  
end
```



10) Fazer o programa de alteração PALT com o seguinte esquema:

**\*\* programa de alteração PALT**

```
define data /* igual ao programa PINC, só que mudando o título
set key pf3 pf5
input func.personnel-id /* para receber o código vindo do PMENU
find func with personnel-id = func.personnel-id
    input using map 'MIAE'
    if #control modified
        update
        end transaction
        move 'Funcionário alterado com sucesso' to #mensagem
    else
        backout transaction reset #mensagem end-if
end-find
fetch 'PMENU'
end
```

11) Fazer o programa de exclusão PEXC, copiando do programa PALT e fazer os ajustes necessários para a exclusão.

Observações:

- Proteger a variável de controle: move (ad=p) to #control
- Depois da instrução INPUT USING MAP ... e antes de efetuar a exclusão, incluir a rotina :

```
input no erase 20/15 'Confirme a exclusão (S/N) ?:' (cd=re ad=i)
  #conf (ad=mt) /* definí-la na local interna com INIT <'N'>
if #conf = 'S' /* emitir o comando DELETE ,
                /* END TRANSACTION
                /* mover 'Funcionário excluído com sucesso'
                /* para #MENSAGEM, caso contrário, mover
                /* 'Exclusão não efetuada' para #MENSAGEM
```

- 12) Criar o mapa MHELP com o tamanho de 5 linhas por 15 colunas. Este mapa deverá ser do tipo “help text”. Layout :

```
selecione  
a opção  
desejada
```

- 13) Editar o mapa MMENU, entrar no campo #OPCAO e colocar o mapa ‘MHELP’ na opção de “Help Routine” do campo
- 14) Testar o mapa MMENU colocando o caracter ? no campo #OPCAO

## 15) Criar o mapa MHRO1 conforme o layout :

```
1 - XXXXXXXXXXXXXXXXXXXXXXXXXXXX
2 - XXXXXXXXXXXXXXXXXXXXXXXXXXXX
3 - XXXXXXXXXXXXXXXXXXXXXXXXXXXX
4 - XXXXXXXXXXXXXXXXXXXXXXXXXXXX
5 - XXXXXXXXXXXXXXXXXXXXXXXXXXXX
Opcao : 0
```

- Este mapa deverá ter 6 linhas por 27 colunas ;
- Deve conter 2 variáveis: o array #A-CIDADE (A20/1:5) e #OPCAO (N1)
- O array #A-CIDADE deverá ser tipo “output protected”
- Fazer a consistência para a variável #OPCAO (valores válidos de 1 a 5)

16) Criar a helprotina HRO1 conforme o esquema :

```
** helprotina HRO1  
define data  
  parameter  
    1 #cidade (a20)  
  local  
    1 #a-cidade (a20/1:5) init (1) <'SÃO PAULO'> (2) ...  
    1 #opcao (n1)  
end-define  
define window janela  
  size 9 * 30  
  base 8 / 20  
  title 'Escolha a Cidade'  
  control window      framed on
```

### Continuação da helprotina HRO1

```
set window 'janela'  
input using map 'mhro1'  
move #a-cidade (#opcao) to #cidade  
set window off  
end
```

- 17) Editar o mapa 'MIAE', entrar no campo FUNC.CITY e colocar a helprotina 'HRO1' na opção de 'Help Routine' do campo.
- 18) Testar o mapa MIAE colocando o caracter ? no campo FUNC.CITY

## 19) Criar o mapa MHRO2 conforme o layout :

```
X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

- Este mapa deverá ter 5 linhas por 30 colunas ;
- Deve conter 2 array's : #A-OPCAO (A1/1:5) e #A-CARGO (A25/1:5)
- O array #A-CARGO deverá ser tipo “output protected”
- Fazer a consistência para o array #A-OPCAO :  
examine #a-opcao (\*) for 'X' giving number #i  
if #i = 0 or #i > 1 /\* → fazer a consistência devida

20) Criar a helprotina HRO2 conforme o esquema :

```
** helprotina HRO2  
define data  
  parameter  
    1 #cargo (a25)  
  local  
    1 #a-opcao (a1/1:5)  
    1 #a-cargo (a25/1:5)  
    1 #i (i2)  
    1 func view of employees  
      2 job-title  
end-define
```



## Continuação da helprotina HRO2

```
** leitura dos 5 primeiros cargos  
histogram (5) func for personnel-id starting from 'a'  
  move *counter to #i  
  move job-title to #a-cargo (#i)  
end-histogram  
define window janela /* mesma definição da HRO2, ajustando o  
  /* tamanho e usando o título  
  /* 'Marque o Cargo com X'
```

### Continuação da helprotina HRO2

\*

```
set window 'janela'
```

```
input using map 'mhro2'
```

```
examine #a-opcao (*) for 'X' giving index #i
```

```
move #a-cargo (#i) to #cargo
```

```
set window off
```

```
end
```

21) Editar o mapa 'MIAE', entrar no campo FUNC.JOB-TITLE e colocar a helprotina 'HRO2' na opção de 'Help Routine' do campo

22) Testar o mapa MIAE colocando o caracter ? no campo  
FUNC.JOB-TITLE



# Término do Curso

**NATURAL**

**The Power Tool for the Enterprise**