

Instruções de Comando

Estas instruções são logicamente equivalentes a instruções de comando de outras linguagens de programação. Elas freqüentemente tem uma sintaxe similar, ou até idêntica.

Instruções Compostas

Por padrão, no IDL uma instrução de comando é uma única etapa da execução. Diversas etapas de execuções podem resultar em uma instrução de comando quando ela estiver agrupada por BEGIN-END. Isto é chamado de uma instrução composta. Por exemplo, esta instrução de comando IF-THEN.

```
If ( x lt 0 ) then print, x
```

Não precisa de BEGIN-END, enquanto esta instrução

```
If ( x lt 0 ) then begin
    print, x
    y = -x
endif
```

precisa.

Neste exemplo, a instrução END toma a forma de ENDIF. END é o suficiente para fechar qualquer BEGIN, mas usando o ENDIF deixa o código mais fácil de se entender, o que é desejável. Cada instrução de comando no IDL tem a sua versão de END. Note que o BEGIN e o END são palavras chaves reservadas, então você não pode usa-las como variáveis ou nome de programas.

Instruções Condicionais

Instruções condicionais são usadas para retornar diferentes resultados para diferentes entradas.

IF-THEN-ELSE

A instrução IF é usada para executar uma condição de uma instrução ou de um bloco de instruções.

Exemplo:

```
x = randomu (seed, 1) - 0.5
if ( x gt 0 ) then print, x
if ( x gt 0 ) then y = x else y = -x
```

ou desta forma

```
x = randomu (seed, 1) - 0.5
if ( x gt 0 ) then begin
    print, x
    y = x
endif else begin
    y = -x
```

```
endelse
```

Os parênteses em torno da condição não são necessários, mas eles deixam o código mais fácil de ser compreendido.

CASE

A instrução CASE é usada para selecionar uma instrução para que ela seja executada, dependendo o valor do seletor de expressão do CASE.

```
case pedacodetorta of
  0: begin
      torta = 'maca'
      topping = 'sorvete'
    end
  1: torta = 'abobora'
  else: torta = 'cherry'
endcase
```

A clausula ELSE é opcional, mas note que é preciso um sinal de dois pontos, diferente da clausula ELSE na instrução IF. Se a expressão resultar em um valor que não está especifico no CASE, e não existir o ELSE, resultara em um erro.

SWITCH

A instrução SWITCH é usada para selecionar uma ou mais instruções para execução, dependendo o valor do seletor de expressão do SWITCH. O SWITCH difere-se do CASE nisso, se uma combinação for encontrada os itens subsequentes da lista serão executados, até que um ENDSWITCH ou um BREAK seja encontrado.

CASE	SWITCH
<pre>x = 2 case x of 1: print, 'one' 2: print, 'two' 3: print, 'three' 4: print, 'four' endcase</pre>	<pre>x = 2 switch x of 1: print, 'one' 2: print, 'two' 3: print, 'three' 4: print, 'four' endswitch</pre>
<pre>IDL Mostra: two</pre>	<pre>IDL Mostra: two three four</pre>

```
x = randomn (seed)
switch 1 of
x lt 0: begin
    print, 'x é menor que zero'
    break
end
x gt 0: print, 'x é maior que zero'
x eq 0: print, 'x é igual a zero'
else:
endswitch
```

Note que se o x é maior que zero, as duas últimas instruções do SWITCH foram executadas.

Instrução de Repetição

Uma tarefa muito comum na programação é executar uma instrução, ou várias instruções, múltiplas vezes.

FOR-DO

A instrução de repetição FOR, executa uma instrução ou um bloco de instruções por um número específico de vezes.

```
for j = 0, 9 do print, j
```

É possível especificar a magnitude do contador da repetição, assim como a sua direção. Por exemplo, no código a seguir o contador da repetição está decrementando em dois com cada interação com a repetição.

```
sequencia = findgen(21)
for j=20, 0, -2 do begin
    print, j, sequencia[j], sequencia[j] mod 4
endfor
```

WHILE-DO

A instrução de repetição WHILE, executa uma instrução ou um bloco de instruções pelo tempo em que a condição teste for verdadeira. A condição é checada no ponto de entrada da repetição.

```
previous = 1 & current = 1
while (current lt 100) do begin
    next = current + previous
    previous = current
    current = next
endwhile
```

Este código gera a seqüência de Fibonacci: 1, 1, 2, 3, 5, 8, 13, ...

REPEAT-UNTIL

A instrução de repetição REPEAT é similar ao WHILE, exceto pela condição de teste da repetição ser no ponto de saída.

```
repeat begin
    n++
endrep until (n gt 20)
```

Instruções de Controle de Fluxo

A instrução de controle de fluxo pode ser usada para modificar o comportamento de instruções condicionais e interativas. A instrução de controle de fluxo permite que você saia de uma repetição, e possa retornar na próxima interação com a repetição.

BREAK

A instrução BREAK fornece uma forma imediata de sair de um FOR, WHILE, REPEAT, CASE, ou SWITH.

Exemplo:

```
FOR i = 0,n-1 DO BEGIN
    IF (array[i] EQ 5) THEN BREAK
ENDFOR
PRINT, i
```

CONTINUE

A instrução CONTINUE fornece uma forma imediata de começar a próxima repetição incluindo as instruções de repetição FOR, WHILE, ou REPEAT. Visto que a instrução BREAK sai do laço de repetição, a instrução CONTINUE sai apenas do atual laço de repetição, passando direto para a próxima repetição no mesmo laço.

Exemplo:

```
FOR I=1,10 DO BEGIN
    IF (I AND 1) THEN CONTINUE
    PRINT, I
ENDFOR
```

A Definição de Verdadeiro (true) e Falso (false)

O que é verdadeiro e o que não é, para os diferentes tipos de dados do IDL.

Variável ou tipo de expressão	Instruções Verdadeiras
Inteiros (<i>byte, integer, long</i>)	Valores ímpares são verdadeiros, valores pares são falsos.
Flutuantes (<i>float, double, complex</i>)	Valores diferentes de zero são verdadeiros, valores iguais a zero são falsos; a parte imaginária de um número complexo é ignorada.
<i>String</i>	Comprimento diferente de zero é verdadeiro, uma <i>string</i> nula é falsa.
Variáveis de Memória Heap (<i>pointers, objects</i>)	Variáveis de memória com dados válidos são verdadeiras, variáveis de memória com dados nulos são falsos.

Em uma unidade de programa, se o LOGICAL_PREDICATE for ativado pela instrução COMPILE_OPT, todos os valores iguais a zero ou nulos serão falsos, ou se não forem nulos ou diferentes de zero serão verdadeiros.

Para maiores informações sobre a instrução COMPILE_OPT, olhe no IDL Online Help.