3 COMANDOS CONDICIONAIS E DE LAÇO

Nesta aula serão apresentados alguns comandos de condição, repetição e gráficos.

3.1 COMANDOS CONDICIONAIS

3.1.1 Comando IF

Como o próprio nome já indica, estes comandos executam determinadas ações desde que algumas condições sejam favoráveis (verdadeiras).

IF <<Condição>> THEN... END IF

Traduzindo: IF = Se, THEN = Então, END IF = Fim do Comando Se.

Quando colocamos uma estrutura condicional IF dentro do nosso programa estamos querendo que o computador avalie uma condição e conforme for o resultado siga determinado caminho. Veja um exemplo:

```
Dim Contador As Integer
Contador = 20
If Contador < 20 Then
        Print "Contador é menor que 20"
End If
Print "Fim do Programa"
```

O programa verifica: Se contador for menor que 20 então imprima no formulário "Contador é menor que 20". Quando terminar imprima "Fim do Programa".

Quando uma determinada condição que colocamos no comando IF é verdadeira ele começa a executar todas as linhas que existem entre IF e END IF. Se a condição não for verdadeira então o programa passa a execução para a linha após o END IF e ignora tudo que houver dentro da estrutura IF... END IF.

IF <<Condição>> THEN... ELSE... END IF

Traduzindo: IF = Se, THEN = Então, ELSE = Senão, END IF = Fim do Comando Se.

Acrescentamos na estrutura o comando ELSE, que é sempre executado quando a condição não é verdadeira.

```
Dim Contador As Integer
Contador = 20
If Contador < 20 Then
        Print "Contador é menor que 20"
Else
        Print "Contador é maior ou igual a 20"
End If
Print "Fim do Programa"
```

O programa verifica: Se o Contador for menor que 20 imprima a frase: "Contador é menor que 20", senão, imprima "Contador é maior ou igual a 20".

Perceba que agora o programa irá executar um comando da estrutura, seja um ou outro, mas pelo menor um será executado. Se a condição for verdadeira o comando executado é sempre o que esta após o comando IF, se for falsa ele ignora as instruções que existem após o comando IF e passa a execução para a linha após o ELSE e vai até o final.

Lembre-se: Sempre que usar o comando IF é necessário usar o comando THEN após a condição, e depois encerrar a estrutura condicional IF com END IF.

IF <<Condição>> THEN... ELSEIF... ENDIF

Traduzindo: IF = Se, THEN = Então, ELSEIF = Senão se, END IF = Fim do Comando Se.

Modificamos comando ELSE para ELSE IF, que é sempre executado quando a condição não é verdadeira, mas ainda dependendo de uma outra condição.

```
Dim Contador As Integer
Contador = 20
If Contador < 20 Then
        Print "Contador é menor que 20"
Elseif Contador = 20 Then
        Print "Contador é igual a 20"
Else
        Print "Contador é maior que 20"
End If
Print "Fim do Programa"
```

O programa verifica se o contador é menor que 20. Se não for verifica se é então igual a 20, senão irá imprimir que contador é maior que 20.

3.1.2 SELECT CASE

Significa "Seleção de Casos". Ou seja, colocamos várias possibilidades (vários casos) para o Visual Basic e ele escolhe um.

Veja a estrutura do comando:

```
op = 3
Select Case op
Case 1
        Print "Opção 1 acionada"
Case 2
        Print "Opção 2 acionada"
Case 3
        Print "Opção 3 acionada"
End Select
```

Veja que este comando verifica se a Opção é 1, depois verifica se é 2 e depois se é 3. Se a variável Opção for 3 ele executará as instruções contidas na próxima linha. Caso a variável Opção não seja 1,2 ou 3 então o comando SELECT CASE é encerrado.

```
op = 3
Select Case op
Case 1
        Print "Opção 1 acionada"
Case 2
        Print "Opção 2 acionada"
Case 3
        Print "Opção 3 acionada"
Case Else
        Print "Opção não é 1,2 ou 3"
End Select
```

Acrescentando na estrutura a cláusula CASE ELSE (caso contrário), o Visual Basic irá verificar se a variável é 1,2 ou 3, não sendo então será CASE ELSE, e a linha seguinte a esta cláusula será executada.

No comando Case podemos usar também intervalos, como por exemplo:

Case 2 to 4

Ou seja, caso a condição esteja entre 2 e 4.

Caso queira que na verificação da condição seja avaliado se a variável é maior ou menor que determinada expressão, usamos:

Case Is > 50

3.2 Comandos de Laços

Estes comandos criam círculos viciosos, ou seja, criam um estrutura condicional que se repetem até que a condição seja satisfeita.

3.2.1 Comando Do

DO WHILE <<Condição>> LOOP (Faça enquanto)

Executa todos os comandos que existem entre DO WHILE e o LOOP. Quando a execução do programa encontra o comando LOOP, o DO WHILE é reavaliado e a rotina continua até que a condição estabelecida esteja satisfeita.

```
Contador = 0
Do While Contador < 10
        Contador = Contador + 1
        Print Contador
Loop
Print "Fim da execução"
```

A variável de nome Contador foi inicializada com Zero, e pedimos para o programa: "Repita as instruções abaixo enquanto Contador for menor que 10". O comando LOOP faz com que o programa volte para a linha do DO WHILE e teste a condição de novo. Somente quando a condição for verdadeira, ou seja, quando Contador for maior ou igual a 10, que o programa executará a linha após o LOOP.

Perceba que dependendo do resultado da condição os comandos existentes dentro da estrutura podem não ser executado, passando direto para a linha após o comando LOOP.

DO ... LOOP WHILE <<Condição>> (Repita enquanto)

Neste caso o Comando Do abre a seqüência de repetição, mas não faz nenhum teste de condição. Este teste é feito no final da estrutura com o comando LOOP WHILE, ou seja, obrigatoriamente as instruções contidas após o comando DO serão executadas.

```
Contador = 0
Do
    Contador = Contador + 1
    Print Contador
Loop While contador < 10
Print "Fim da execução"
```

DO UNTIL <<Condição>> LOOP (Faça até que)

Executa todos os comandos que existem entre DO UNTIL e o LOOP. Quando a execução do programa encontra o comando LOOP, o DO UNTIL é reavaliado e a rotina continua até que a condição estabelecida esteja satisfeita.

```
Contador = 0
Do Until contador >= 10
        Contador = Contador + 1
        Print Contador
Loop
Print "Fim da execução"
```

Inicializamos uma variável de nome Contador com Zero, e pedimos para o programa: "Repita as instruções abaixo até que Contador seja maior ou igual que 10". O comando LOOP faz com que o programa volte para a linha do DO UNTIL e teste a condição de novo. Somente quando a condição for verdadeira, ou seja, quando Contador for realmente maior ou igual a 10, que o programa executará a linha após o LOOP.

Semelhante ao DO WHILE, dependendo do resultado da condição os comandos existentes dentro da estrutura podem não serem executados, passando direto para a linha após o comando LOOP.

DO ... LOOP UNTIL <<Condição>> (Repita até que)

O Comando DO abre a seqüência de repetição, mas não faz nenhum teste de condição. Este teste é feito no final da estrutura com o comando LOOP UNTIL, ou seja, obrigatoriamente as instruções contidas após o comando DO serão executadas.

```
Contador = 0
Do
    Contador = Contador + 1
    Print Contador
Loop Until Contador >= 10
Print "Fim da execução"
```

3.2.2 Comando For

FOR <<Intervalo>> ... NEXT (Conte de número inicial até numero final)

O comando FOR faz uma contagem de um determinado intervalo de números. Sempre que essa contagem encontra com o comando NEXT (próximo) a execução do programa retorna ao comando FOR até que o número final seja alcançado. Exemplo:

```
Contador = 0
For Contador  = 1 to 10
    Print "A variável Contador agora vale " &Contador
Next
Print "Fim da execução"
```

Neste programa o Visual Basic inicializa a variável Contador em zero, e o comando FOR avisa "vamos contar até 10, começando pelo 1". As instruções contidas na linha abaixo são executadas, e quando o comando NEXT é encontrado a execução volta para o FOR, e a variável Contador é incrementada em mais um e assim por diante. Quando Contador for igual a 10 a estrutura FOR/NEXT é desfeita.

STEP

Usamos o STEP em conjunto com o FOR para fazer com que a contagem seja incrementada. Exemplo: Se queremos que o comando conte de 2 em 2 colocamos FOR Variável = 1 TO 10 STEP 2

3.2.3 Comando Exit

EXIT <<Comando>>

- **Exit Sub**: Força a saída da Sub rotina. Quando a execução do programa encontra este comando, o Visual Basic transfere o controle do programa para a linha seguinte a aquela que chamou a rotina.

- **Exit Function**: Força a saída da função. Quando a execução do programa encontra este comando, o Visual Basic transfere o controle do programa para a linha seguinte a aquela que chamou a rotina.
- **Exit Do**: Força a saída de um LOOP, seja WHILE ou UNTIL, mesmo que a condição estabelecida não seja verdadeira:

```
Contador = 0
Do While Contador < 10
        Contador = Contador + 1
        Print Contador
        If Contador = 5 Then
          Exit Do
        End If
Loop
Print "Fim da execução"
```

- **Exit For**: Força a saída de um FOR..NEXT, mesmo que o número final não tenha sido alcançado.

```
Contador = 0
For Contador  = 1 To 10
    Print "A variável Contador agora vale " & Contador
    If Contador = 6 Then
        Exit For
    end if
Next
Print "Fim da execução"
```

4. Trabalhando com gráficos

Para se trabalhar com gráfico é aconselhável utilizar o componente "PictureBox" e não gerar o gráfico direto no formulário, porque assim você tem uma organização melhor no seu sistema e assim um maior controle.

Principais propriedades do componente PictureBox:

Name - definir o nome o padrão é "PCT_NOME"

AutoRedraw – define se o desenho será mantido no componente quando a tela perder o foco.

Os comandos básicos para montar um gráfico são:

Scale – definir a escala do objeto;

Line – para plotar uma linha ou um quadrado;

Circle – para plotar uma circunferência ou um círculo;

Print – para imprimir um texto no objeto;

4.1 Scale

O comando "Scale" define uma escala para o objeto, a sintaxe do comando é:

object.Scale (x1, y1) - (x2, y2)

(x1,y1) - pontos iniciais

(x2,y2) - pontos finais

Exemplo

Picture1.Scale (0, 0)-(100, 100)

Obs.: Este comando é importante para você definir o tamanho da escala do picturebox para você trabalhar com proporções (No momento que for desenhar um gráfico com escalas X e Y)

4.2 Line

O comando "line" desenha uma linha no objeto a sintaxe do comando é:

object.Line (x1, y1) - (x2, y2), [color], [B][F]

(x1,y1) - ponto inicial

(x2,y2) - ponto final

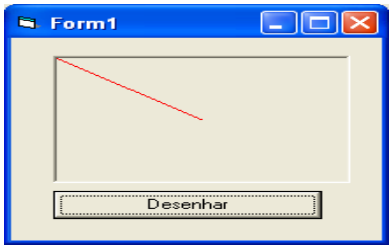
[color] - cor

[B] - desenha um quadrado com as coordenadas passadas

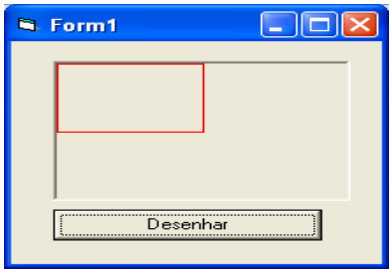
[F] - só pode ser utilizado quando o [B] foi utilizado e este define se o quadrado será preenchido ou vazio

Exemplos:

```
Private Sub Command1_Click()
Picture1.Scale (0, 0)-(100, 100)
Picture1.Line (0, 0)-(50, 50), 255
End Sub
```




```
Private Sub Command1_Click()
Picture1.Scale (0, 0)-(100, 100)
Picture1.Line (0, 0)-(50, 50), 255, B
End Sub
```




```
Private Sub Command1_Click()
Picture1.Scale (0, 0)-(100, 100)
Picture1.Line (0, 0)-(50, 50), 255, BF
End Sub
```

4.3 Circle

Através do comando "Circle" é possível desenhar círculos, através da seguinte sintaxe.

object.Circle (x, y), radius, [color, start, end, aspect]

(x,y) - são as coordenadas do ponto no objeto;

radius – é o raio do ponto.

Color – é a cor do ponto.

Start, End – define se o circulo será completo ou um semicírculo.

Aspect – se o ponto terá a aparência de um círculo ou pode ser oval, padrão círculo.

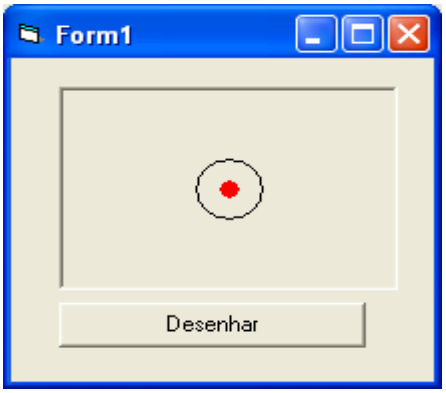
Observação: Para definir se o ponto vai ser preenchido ou vazio utiliza-se o comando "FillStyle". Para definir a cor do ponto é utilizado o comando "FillColor".

Exemplos

```
Objeto1.Fillstyle = 1 'o ponto aparecerá apenas as bordas
Objeto1.Fillstyle = 0 'o ponto aparecerá preenchido
Objeto1.FillColor = 0 'pinta o objeto de vermelho
```

Exemplo

```
Private Sub Command1_Click()
Picture1.Scale (0, 0)-(100, 100)
Picture1.FillStyle = 1
Picture1.Circle (50, 50), 10, 0
Picture1.FillStyle = 0
Picture1.FillColor = 255
Picture1.Circle (50, 50), 3, 255
End Sub
```



4.4 Print

Através do comando "Print" é capaz escrever textos no objeto, a sintaxe deste comando é:

Objeto1.Print "Texto"

Alguns comandos que estão relacionados com este comando alterar a visualização do texto são:

Objeto1.**FontName** = "Arial" 'Define a fonte do texto

Objeto1.**FontSize** = 12 'Define o tamanho do texto

Objeto1.**FontBold** = True 'Define se o texto será em negrito

Objeto1.**FontItalic** = True 'Define se o texto será em itálico

Objeto1.**FontUnderline** = False 'Define se o texto é sublinhado

Objeto1.**ForeColor** = 0 'Define a cor do texto

Objeto1.**CurrentX** = 10 'Posição em X que o texto será impresso

Objeto1.**CurrentY** = 10 'Posição em Y que o texto será impresso