Estruturas de Repetição

FELIPE MACHADO 2 COMENTÁRIOS INICIANTE

As estruturas de repetição no **Delphi**, também conhecidas como ***loops***, são indispensáveis para o processo de desenvolvimento. Por isso é muito importante que você conheça bem a sua estrutura e domine a sua utilização.

Existem situações onde é necessário repetir um determinado trecho de um processo um certo número de vezes.

Imagine que você tenha que executar um determinado bloco de instruções 10 vezes. Sem as estruturas de repetição ficaria muito trabalhoso e em alguns casos praticamente impossível.

Assim, existem as estruturas de repetição no **Delphi**, que permitem que você execute estas tarefas de forma mais simples.

As estruturas de repetição também podem ser chamadas de **laços** ou *loops*, sendo que podem existir dois tipos de laços de repetição:

- Laços finitos: neste tipo de laço se conhece previamente o número de repetições que serão executadas; e
- Laços infinitos: neste tipo de laço não se conhece previamente o número de repetições que serão executadas. São também chamados **condicionais** pelo fato de encerrarem a sua execução mediante uma determinada condição.

Existem 3 estruturas de repetição no **Delphi**, são elas: *for, while…do e repeat…until*.

ESTRUTURAS DE REPETIÇÃO

for

O comando *for* executa repetitivamente um comando enquanto é atribuído um valor incremental a uma variável de controle, que chamamos de contador do *for*. A repetição, ou o *loop* só é interrompido quando o contador atinge o seu limite, que é definido também pela instrução *for*.

Sintaxe:

```
1. for variável := <início> to/downto <fim> do
2.   <instrução a ser repetida>;
```

ou (com mais de uma instrução)

```
1. for variável := <início> to/downto <fim> do
2. begin
3.     <instrução a ser repetida 1>;
4.     <instrução a ser repetida 2>;
5.     <instrução a ser repetida 3>;
6.     <…>
7. end;
```

Aonde:

variável = contador do *for*, qualquer valor inteiro.
<início> = valor a partir do qual o loop deve iniciar, qualquer valor inteiro.
<fim> = valor limite para o término do *loop*, qualquer valor inteiro (>= ao <início> quando usado o *to* e <= <início> quando usado o *downto*).

Exemplo: somar uma variável até 10

```
1. procedure Somar;
2. var
3.   Contador, Soma: Integer;
```

```
4. begin
5.    Soma := 0;
6.    for Contador := 1 to 10 do
7.       Soma := Soma + 1;
8. end;
```

No exemplo acima, temos duas variáveis: Contador e Soma. Contador será responsável por fazer o controle do *loop*. Ele será iniciado por 1 e irá interromper o laço quando chegar em 10.

A cada *loop* executado a variável Soma será incrementada em 1. Portanto, ao final do *loop*, a variável Soma terá o valor de 10.

Como só temos uma instrução no exemplo acima, não foi necessário utilizar a instrução *begin...end* para controle do bloco.

ESTRUTURAS DE REPETIÇÃO

while...do

Esta estrutura de repetição se caracteriza por efetuar um **teste lógico** no início do *loop*, verificando se é permitido executar o trecho de instruções abaixo dela.

A estrutura *while...do* tem o seu funcionamento controlado por condição. Desta forma, poderá executar um determinado conjunto de instruções enquanto a condição verificada permanecer verdadeira.

No momento em que a condição se torna falsa, o processamento da rotina é desviado para fora do *loop*. Se a condição for falsa logo no início do *loop*, as instruções contidas no *while...do* serão ignoradas.

Sintaxe:

```
1. while <condição> do
2.    <instrução a ser repetida para condição verdadeira>;
```

ou (com mais de uma instrução)

```
1. while <condição> do
2. begin
3.       <instrução a ser repetida para condição verdadeira 1>;
4.       <instrução a ser repetida para condição verdadeira 2>;
5.       <instrução a ser repetida para condição verdadeira 3>;
6.       <...>
7. end;
```

Veja como ficaria o mesmo exemplo usado no *for*, mas aplicando o *while...do*:

Exemplo: somar uma variável até 10

```
1. procedure Somar;
2. var
3.    Soma: Integer;
4. begin
```

```
5.    Soma := 0;
6.    while Soma <= 10 do
7.        Soma := Soma + 1;
8. end;
```

Repare no exemplo acima, que quando utilizamos o ***while...do***, não temos a necessidade de utilizar uma variável como Contador. Pois antes de iniciar o ***loop***, um teste lógico é efetuado, ou seja, o processo só será realizado enquanto Soma <= 10. Assim que a variável Soma atingir o valor 10, o ***loop*** será interrompido.

Como só temos uma instrução no exemplo acima, não foi necessário utilizar a instrução ***begin...end*** para controle do bloco.

ESTRUTURAS DE REPETIÇÃO

repeat...until

Esta estrutura é parecida com a estrutura ***while...do***, mas a principal diferença entre elas, é que no ***repeat... until***, o teste lógico é efetuado no final do ***loop***.

Seu funcionamento também é controlado por **decisão.** Esta instrução irá efetuar a execução de um conjunto de instruções pelo menos uma vez antes de verificar a validade da condição estabelecida.

Desta forma, ***repeat...until*** irá processar um conjunto de instruções, no mínimo uma vez, até que a condição se torne verdadeira. Para a estrutura ***repeat...until***, um conjunto de instruções é executado enquanto a condição se mantém falsa e até que se torne verdadeira.

Sintaxe:

```
1. repeat
2.    <instrução a ser repetida até que a condição seja verdadeira>;
3. until
4.    <condição>;
```

ou (com mais de uma instrução)

```
1. repeat
2.    <instrução a ser repetida até que a condição seja verdadeira 1>;
3.    <instrução a ser repetida até que a condição seja verdadeira 2>;
4.    <instrução a ser repetida até que a condição seja verdadeira 3>;
5.    <...>
6. until
7.    <condição>;
```

Veja como ficaria o mesmo exemplo usado no ***for*** e no ***while...do***, mas aplicando o ***repeat...until***:

Exemplo: somar uma variável até 10

```
01. procedure Somar;
02. var
03.    Soma: Integer;
04. begin
```

```
05.   Soma := 0;
06.   repeat
07.     Soma := Soma + 1;
08.   until
09.     Soma <= 10;
10. end;
```

No exemplo acima também não temos a necessidade de ter uma variável como Contador, o controle é feito pelo teste lógico ao final do ***loop***. Portanto, a variável Soma será incrementada em 1 até que ela atinga o valor 10.

Um outro fator importante a ser observado na instrução ***repeat...until***, é a ausência do ***begin...end*** para controlar o bloco quando se tem mais de uma instrução dentro do ***loop***. Portando, o ***begin...end*** nunca se faz necessário quando se utiliza o ***repeat...until***.

RESUMO DO QUE VOCÊ ACABOU DE APRENDER

As estruturas de repetição são indispensáveis para o processo de desenvolvimento. Por isso é importante dominar a sua utilização.

Existem 3 estruturas de repetição no **Delphi**:

for: necessita de um contador para controlar o ***loop***. O laço só é interrompido quando o contador atinge o seu limite.

while...do: efetua um teste lógico no início do ***loop***, verificando se é permitido executar o trecho de instruções abaixo dela.

repeat...until: efetua um teste lógico no final do ***loop***, portanto, as instruções serão processadas ao menos uma vez antes de verificar condição estabelecida.

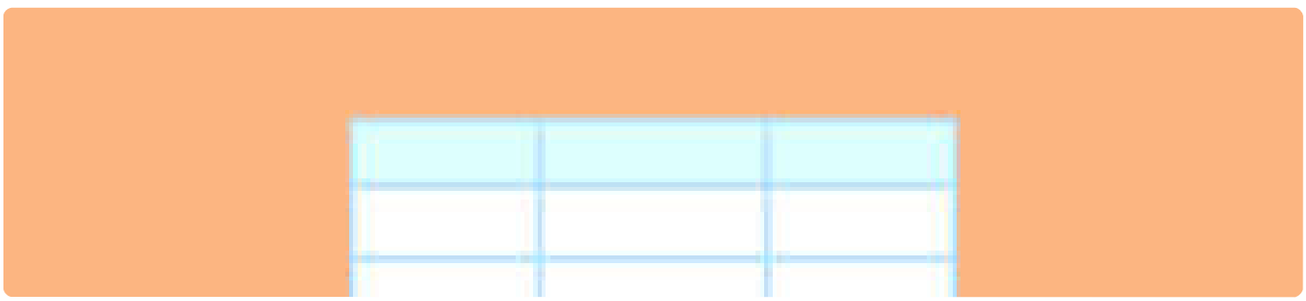
Espero que tenha gostado! ☺

Dúvidas? Deixe seu comentário abaixo.

Sobre Felipe Machado

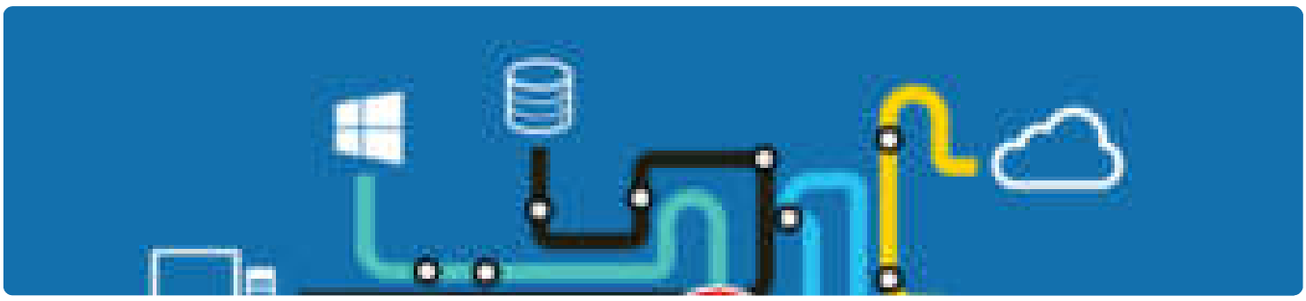
Formado em Ciência da Computação e Desenvolvedor Delphi há mais de 10 anos. Criei o blog para ajudar a todos que desejam **estudar** e realmente **aprender** a programar em Delphi. **Saiba mais aqui.**

Artigos relacionados



Mudar a Cor da Linha do DBGrid no Delphi

Diagnosticando e Recuperando Banco de Dados Firebird Corrompido



Afinal, o Delphi Morreu?

Seja Livre! Trabalhe de Qualquer Lugar

Dataware x Não Dataware – DBEdit x Edit

Como Ganhar Dinheiro Programando em Delphi

2 Comentários

Augusto

3 anos atrás

Link permanente

Excelente explicação, uma verdadeira aula! Parabéns!!

Felipe Machado

3 anos atrás

Link permanente

Olá Augusto, obrigado pela visita! Que bom que gostou, muito obrigado pelas palavras e seja sempre bem vindo!! Abraços.

Deixe uma resposta

O seu endereço de e-mail não será publicado. Campos obrigatórios são marcados com *

Comentário

Nome *

E-mail *

Site

Salvar meus dados neste navegador para a próxima vez que eu comentar.

Não sou um robô

reCAPTCHA
Privacidade - Termos

ENVIAR COMENTÁRIO

FELIPE MACHADO

Formado em Ciência da Computação e Desenvolvedor Delphi há mais de 10 anos. Criei o blog para ajudar a todos que desejam **estudar** e realmente **aprender** a programar em Delphi.

SAIBA MAIS

Mais Lidos

- Afinal, o Delphi Morreu?
- Seja Livre! Trabalhe de Qualquer Lugar
- Como Ganhar Dinheiro Programando em Delphi
- Diagnosticando e Recuperando Banco de Dados Firebird Corrompido

Páginas

- 10 Motivos para ser um Programador Delphi
- Delphi é uma Linguagem de Programação?
- História do Delphi
- Curso
- Contato

Curso

Carreira

Infográfico

MySQL query error