Espaçamento

Regras para Formatação Horizontal

Tabulação: Utilizar 2 caracteres (espaços) ao invés da tabulação.

Para manter a estrutura do código de modo que sua leitura seja fácil, o número máximo de caracteres por linha é **130**. Caso a expressão ultrapasse esse comprimento, a linha deve ser quebrada segundo essas regras:

1. A linha tem comparações? `('a = b', 'a <> b', 'a <= b', 'a >= b', 'a < b' ou 'a > b')`
    1. Envolver cada comparação com parênteses.
2. A linha contem negação? `('not')`
    1. Envolver cada negação com um parênteses.
3. A linha contém operadores lógicos? `('and' e 'or')`
    1. Envolver cada sequência de `'and'` com um parênteses;
    2. Envolver cada sequência de `'or'` com um parênteses.
4. A linha é uma atribuição? `(':=')`
    1. Adicionar uma quebra de linha após o sinal de atribuição `(':=')`, alinhar a nova linha com a linha original e indentar em um nível (2 espaços);
    2. Caso o código resultante não tenha ultrapassado a linha das 130 colunas, processo pode ser interrompido;
    3. Caso a atribuição seja uma String, ela pode ser quebrada em uma ou mais concatenações. Essas podem ser quebradas e alinhadas com a linha original, adicionando um nível de indentação;
    4. Caso a atribuição seja uma comparação (Não considerar comparações dentro de métodos aninhados), adicionar uma quebra de linha após o sinal de comparação `('=', '<>', '<=', '>=', '<' ou '>')`, e alinhar a nova linha com o primeiro operando.
5. A linha tem um 'if'?
    1. Caso a condição do if não tenha um parênteses envolvendo a condição inteira, envolvê-la com um;
    2. Adicionar uma quebra de linha após cada operador `'and'` ou `'or'`;
    3. Após cada quebra de linha, alinhar a nova linha com o parênteses onde ela está contida e adicionar um espaço;
    4. Caso o código resultante não tenha ultrapassado a linha das 1 colunas, processo pode ser interrompido;
    5. Caso alguma das condições do if tenha uma comparação (Não considerar comparações dentro de métodos aninhados), adicionar uma quebra de linha após o sinal de comparação `('=', '<>', '<=', '>=', '<'` ou `'>')`, e alinhar a nova linha com o primeiro operando.
    6. Caso o código resultante não tenha ultrapassado a linha das 130 colunas, processo pode ser interrompido.

6. A linha contém métodos aninhados?
    1. Aplicar essa regra do métodos mais externo para o mais interno;
    2. Adicionar uma quebra de linha exatamente antes do início do método aninhado;
    3. Alinhar a nova linha com o método onde ela está contida (Caso o método tenha uma negação (not), alinhar com o início dessa negação), adicionar ainda um nível de indentação (2 espaços);
    4. Caso o código resultante não tenha ultrapassado a linha das 130 colunas, processo pode ser interrompido.
7. A linha contém parâmetros?
    1. Ao realizar a quebra de linha de um parâmetro, ele deve estar indentado um nível (2 espaços) em relação ao nível do método proprietário do parâmetro (caso o método proprietário tenha uma negação (not), alinhar com o início dessa negação);
    2. Caso o código resultante não tenha ultrapassado a linha das 130 colunas, processo pode ser interrompido.
8. Após todas essas etapas a linha ainda ultrapassa a marca de 130 colunas?
    1. Caso ainda assim, a linha ultrapasse a marca de 130 colunas então deve-se verificar a existência de pontos na expressão (“.”), sendo que, ao realizar esse tipo de quebra de linha, o ponto deve ficar com a expressão na linha inferior, indentado em um nível em relação ao início da expressão.

Observação: Algumas vezes, quando uma linha ultrapassa as 130 colunas, é um sinal de que esse trecho pode ser refatorado. O programador pode então refatorar, se julgar que é necessário ou melhor.

Declaração de variáveis

Correto

```
var
  LTeste: Integer;
```

Incorreto

```
var
  LTeste : Integer;
  LTeste :Integer;
```

Atribuições

Correto

```
LTeste := 15;
```

Incorreto

```
LTeste:=15;
LTeste:= 15;
LTeste :=15;
```

Métodos

Correto

```
procedure Ex(AParametro: Integer);
procedure Ex(AParametro1, AParametro2: Integer);
procedure Ex(AParametro: Integer; AParametro2: String);
```

Incorreto

```
procedure Ex (AParametro1: Integer);
procedure Ex( AParametro: Integer);
procedure Ex(AParametro: Integer );
procedure Ex(AParametro1,AParametro2: Integer);
procedure Ex(AParametro1 , AParametro2: Integer);
procedure Ex(AParametro: Integer;AParametro2: String);
procedure Ex(AParametro: Integer ; AParametro2: String);
```

Matrizes

Correto

```
LTeste := LMatriz[0];
```

Incorreto

```
LTeste := LMatriz[ 0];
LTeste := LMatriz[0 ];
LTeste := LMatriz[ 0 ];
```

Operadores binários

Correto

```
LTeste := 1 + 1;
```

Incorreto

```
LTeste := 1+1;
LTeste := 1 +1;
LTeste := 1+ 1;
```

Operadores unários

Correto

```
LTeste := -1;
```

Incorreto

```
LTeste := - 1;
LTeste :=-1;
```

Subrotinas

Correto

```
function MeuMetodo: String;

    procedure SubMetodo;
    begin
      //Código SubMetodo
    end;

begin
  //Código MeuMetodo
end;
```

Incorreto

```
function MeuMetodo: String;

procedure SubMetodo;
begin
  //Código SubMetodo
end;
begin
  //Código MeuMetodo
end;
```

```
function MeuMetodo: String;

  procedure SubMetodo;
  begin
    //Código SubMetodo
  end;

begin
  //Código MeuMetodo
end;
```

Uses

> **Dica:** Para identificar se uma unit deve ser declarada na uses superior ou inferior, realize o seguinte teste: Copie a unit para a uses inferior e compile o programa, caso o processo de compilação não apresente erros, a unit deve permanecer na uses inferior. Caso contrário, mova a unit para a uses superior.

No primeiro “uses“ (logo abaixo de “interface”) declarar as units no .dfm e nas assinaturas dos métodos.

```
interface

uses
  Winapi.Windows,
  Winapi.Messages,
  System.SysUtils,
  System.Classes,
```

```
  Vcl.Graphics,
  Vcl.Controls,
  Vcl.Forms,
  Vcl.Dialogs;
```

No segundo “uses“ (logo abaixo de “implementation”) declarar as units usadas na implementação do código.

```
implementation

uses
  uExisteTabela,
  uProcuraRegistro,
  uGravaSistema;
```

A fim de evitar conflito em Merge devido a alterações em mesma linha, cada Unit deve ser declarada em linha diferente. Esse erro costuma acontecer principalmente quando há refatoração ou exclusão de algum componente visual.

> Correto

```
implementation

uses
  uExisteTabela,
  uProcuraRegistro,
  uGravaSistema;
```

> Incorreto

```
implementation

uses
  uExisteTabela, uProcuraRegistro, uGravaSistema;
```

Arrays

Ao indentar elementos de um array, eles devem seguir a lógica de ordenação de parênteses. Por não estarem em níveis diferentes de indentação, devem ser indentados da seguinte forma

>

Correto

```
TClasse.Metodo(
  Parametro1,
  Parametro2,
  [Item1, Item2, Item3
   Item4, Item5, Item6]);
```

Incorreto

```
TClasse.Metodo(
  Parametro1,
  Parametro2,
  [Item1, Item2, Item3
    Item4, Item5, Item6]);
```

```
TClasse.Metodo(
  Parametro1,
  Parametro2,
  [Item1, Item2, Item3
  Item4, Item5, Item6]);
```

IN em tratamento de condições

Correto

```
Result :=
  (Self in
    [cdfIncidenciadecisaojudicial,
     cdfIncidenciaDecisaoJudicial13Sal,
     cdfIncidenciaDecisaoJudicialAvisoPrevioIndenizado]);
```

Incorreto

```
Result :=
  Self in [cdfIncidenciadecisaojudicial,
           cdfIncidenciaDecisaoJudicial13Sal,
           cdfIncidenciaDecisaoJudicialAvisoPrevioIndenizado];
```

```
Result := (Self in [cdfIncidenciadecisaojudicial,
                    cdfIncidenciaDecisaoJudicial13Sal,
                    cdfIncidenciaDecisaoJudicialAvisoPrevioIndenizado]);
```

Revision #24
Created 29 December 2020 13:16:32 by Luã Martins
Updated 21 October 2021 19:35:31 by Gabriel Ferreira