# Indentação

Indentação é um anglicismo com origem no termo *indentation*, que significa “um espaço no começo de uma linha escrita ou de um paragrafo” (tradução livre da definição do dicionário Merriam-Webster). Ao escrever um código-fonte é habitual adotar uma indentação que permita destacar cada uma das estruturas do programa. Para fazê-lo, a cada vez que se inicia um bloco de código aumenta-se a distância entre o texto e a margem esquerda. Quando se encerra este bloco, retoma-se a distância utilizada no bloco anterior (vide exemplos no fim do texto). Desta forma, a hierarquia entre os blocos fica evidenciada por suas distâncias à margem. Uma boa indentação é de extrema importância para permitir a futura leitura e compreensão de um código-fonte.

A indentação, ou seja, o afastamento entre o início do texto e margem, pode ser criada usando o caractere de tabulação horizontal ("tab") ou uma seqüência de espaços em branco. Durante muito tempo, fui defensor do uso da seqüência de espaços em branco, pois ela me permitia um maior controle no modo como meu código seria exibido, visto que muitos editores de texto adotam padrões diferentes para o tamanho da tabulação.

Com o passar dos anos, acabei mudando de opinião e adotando a tabulação que é mais simples e mais flexível. Na realidade, a maioria dos editores de texto permite que você configure quantos espaços serão usados para uma tabulação. Desta forma, é possível manter o controle sobre o modo como o código será exibido para mim. Além disso, caso o meu código deva ser lido por outro programador e ele deseje uma distância de indentação diferente da adotada por mim, basta que a quantidade de espaços usada na tabulação esteja configurada para essa distância. Adicionalmente, existe a vantagem de ser apenas um caractere para inserir ou retirar na hora de avançar ou recuar um nível de indentação.

Independe da sua preferência, o importante é escolher o seu padrão e mantê-lo sempre. Para isso, seguem algumas dicas para a indentação dos códigos no Vim:

O comando abaixo configura como **N** o número de espaços que será usado para cada tabulação (o padrão é 8):

```
:set tabstop=N
```

O comando abaixo faz com que os caracteres de tabulação inseridos sejam substituídos pelo número equivalente de espaços, conforme configurado no comando acima:

```
:set expandtab
```

O comando abaixo desativa a opção anterior, fazendo com que caracteres de tabulação sejam mantidos:

```
:set noexpandtab
```

O comando abaixo faz com que novas linhas sejam criadas com a mesma indentação da anterior.

```
:set autoindent
```

É possível aumentar ou diminuir em um nível a indentação de uma linha usando os comandos >> e <<, respectivamente. O comando que configura como **N** o número de espaços usados para cada nível de indentação é:

```
:set shiftwidth=N
```

Algumas considerações importantes sobre os comandos acima: se você salvar um arquivo com a propriedade **expandtab** ativa, todos os caracteres de tabulação que já existiam no arquivo serão convertidos no número de espaços configurados. O contrário não acontece, ou seja, ao salvar um arquivo com a propriedade desativada conjuntos de espaço nunca serão convertidos em caracteres de tabulação. Já no caso da indentação automática (**autoindent**), sempre que o Vim gera a indentação de uma nova linha,

um conjunto de espaços em branco com a quantidade configurada para a tabulação será convertido para a tabulação, exceto que a propriedade **expandtab** esteja ativa.

Todas essas configurações são relativas a sessão do Vim, ou seja, toda vez que você abrir o Vim elas voltarão para os seus valores padrão. Caso você deseje que um conjunto de configurações seja adotado sempre que você abrir o Vim, você pode criar, na pasta *home* do seu usuário, um arquivo com o nome **.vimrc** com as configurações desejdas.

Abaixo deixo exemplos de 3 modos diferentes de indentar um código em Pascal.

Conteúdo do bloco indentado, mas **begin** e **end**, não.

```
program indentation;

begin
    if ___ then
    begin
        for ___ do
        begin
            writeln('Estas linhas estão com 3 níveis de indentação.');
            writeln('Um do bloco principal, um do if, outro do for.');
            writeln('Desta forma, fica claro que estes comandos');
            writeln('estão dentro do for e poderão ser repetidos.');
        end;
        writeln('Mas este não.');
    end
    else
    begin
        for ___ do
            writeln('Bloco com apenas uma linha também é indentado.');
    end;
end.
```

**Begin** e **end** indentados junto do conteúdo dos blocos.

```
program indentation;

    begin
    if ___ then
        begin
        for ___ do
            begin
            writeln('Estas linhas estão com 3 níveis de indentação.');
            writeln('Um do bloco principal, um do if, outro do for.');
            writeln('Desta forma, fica claro que estes comandos');
            writeln('estão dentro do for e poderão ser repetidos.');
            end;
        writeln('Mas este não.');
        end;
    else
        begin
        for ___ do
            writeln('Bloco com apenas uma linha também é indentado.');
        end;
    end.
```

**Begin** e **end** indentados em um nível e conteúdo dos blocos indentado em outro.

```
program indentation;

    begin
        if ___ then
            begin
                for ___ do
                    begin
                        writeln('Diferente dos anteriores');
                        writeln('aqui são 6 níveis de indentação.');
                        writeln('Pois cada begin também acrescentou');
                        writeln('um nível.');
                    end;
                writeln('Aqui 2 níveis viraram 4.');
            end
        else
            begin
                for ___ do
                    writeln('Já aqui passou só de 3 para 5.');
                writeln('Pois não há begin neste for.');
            end;
    end.
```