**Breaking Points**

Existem algumas formas de escrever códigos em algoritmos. Na verdade, cada programador tem sua particularidade e lógica, por isso dos tópicos de boas práticas e práticas padrão para programação, para que “qualquer” dev consiga entender e fazer manutenção no seu código.

Mas imagine que por algum motivo o compilador não leia um **for**, ou um **if, do while**, enfim, qualquer estrutura do seu algoritmo, por qualquer motivo desconhecido que seja. Se você estiver escrevendo o seu código em uma plataforma que aponte os erros, ela não identificará, pois não é um erro de sintaxe, ou estrutural, em si, pode acontecer. Mas mesmo que não seja erro do compilador. Dependendo da estrutura do código fica difícil identificar o que de fato está dando errado, se você apenas inverteu alguma lógica no algoritmo por exemplo. Daí vem os breaking points! Eles são nada menos que a sinalização de que o seu compilador precisa interromper a leitura e execução do código naquele ponto e apresentar o que foi encontrado até então, como variáveis e seus valores, mensagens de alertas e erros. Ou seja, o breaking point auxilia no teste de funcionalidade do código. O breaking point irá ajudar a encontrar com maior facilidade e agilidade o problema que faz sua variável aparecer com valor diferente do esperado ou qualquer coisa do gênero, e assim você poderá corrigir o projeto e dar sequência ao trabalho.

Fica como na figura abaixo a representação do breaking point no DEV C++ (editor de código C++ com console de execução), existem algumas variações de interface para interface de IDE, mas no geral, uma sinalização em cima do número da linha e um destaque na linha inteira.

