

Go ahead. Stop programming. This ensures you from any mistakes.

# Go ahead. Stop programming. This ensures you from any mistakes.

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-06/msg01307.html>

---

- *From:* "valentin tihomirov" <[V\\_tihomirov@xxxxxxxx](mailto:V_tihomirov@xxxxxxxx)>
  - *Date:* Fri, 8 Jun 2007 15:16:27 +0300
- 

In other words, the "outer" variable `i` has scope which includes the part of the block before it was declared. There's then an extra rule:

It is a fact that statements are executed sequentially, one after another. It is truth that a block of instructions is executed as one statement. It is truth that the variables in the imperative languages, which allow deferred declarations, are visible only past declaration. Acquiring these basic matters, everybody understands that the parent block variables declared past the block are not visible inside the block and that the block internal variables are not visible past the block. The other considerations are irracionale.

<quote>In certain situations but not in the example above, this could lead to a compile-time error if the statements of the block were later to be rearranged.  
</quote>

Rearranging code always can yield errors. Does this argument give me power to infer arbitrary rules?

They tell us they pursue language simplicity. The rule "do not define a variable more than once in the same context" is natural, and simplest therefore. All normal languages obey it therefore. Overcomplicating a grammar by injecting more barriers is a path right away from simplicity.

Go ahead. Stop programming. This ensures you from any mistakes.

Another obstacle at the plain place introduced in C# for no reason is blocking "fall throughs" in switch — the feature which can be very useful sometimes.

Very useful sometimes when it's deliberate, but also very painful when you don't want it and accidentally have it. Note that you *can* use multiple cases for a single block, you just can't have case/code/case/code without a break.

I suspect that when working with C, I ran into more times when I forgot to include the break than times when I deliberately wanted to fall through.

I suppose that the first thing the programmers should know is the sequence of instruction execution. Normally, the statements are evaluated sequentially. Should we put an explicit branch after each and every statement to avoid the natural "fall through"? The fallthrough ban does not save you from the infinite kinds of errors you still can make, including wrong branching. It just infers a fair amount of code where it is unnecessary. You should consider avoid programming be safe. I doubt that the requirement to pile up the loads of syntactic salt improves the quality of code the unconscious people produce.