**97 things every programmer should know**

**▼ Title Of Book and Chapter**

# Chapter 9 Check Your Code First before Looking to Blame Others

**▼ What are Three Things I learned today**

1. Before – I used to think that something compiler or anything else has the problem not my code.

After - But it turns out usually that it’s my mistake after all.

2. Before – Whenever I tell that my compiler causes and error. I might simply lying to myself 99% of cases. After - But after investigating the problem, I usually find out the following reasons: an array overrun; an uninitialized variable; a misprint; a synchronization error in a parallel program; a non volatile variable used; code leading to undefined behavior; etc.

3. Before - Many went through fixing such errors. Many read about them. But it doesn't prevent them from blaming the compiler for all sins again and again. Each time it seems that it's exactly it which is guilty.

The compiler, of course, might also contain errors. But this probability is very small unless you use some exotic compiler for a microcontroller. During many years of working with Visual C++ I saw only once that it had generated an incorrect assembler code.

After - Before starting to blame the compiler and write about it in the code or on a forum, carry out a thorough investigation. First, I will eliminate an error in my code sooner. Second, I won't look silly in other programmers' eyes who will point out my slip-up.

**▼ Title Of Book and Chapter**

# Chapter 10 Choose Your Tools with Care

**▼ What are Three Things I learned today**

1. Before – I thought new technologies, applications and any software are built from scratch but I was wrong.

After – After reading this chapter I’ve learned that new technologies, applications and any software. They are assembled using existing tools — components, libraries, and frameworks.

2. Before – I thought they build software production.

After – But Software production and maintenance is human-intensive work, so buying may be cheaper than building.

3. Before – I thought that new techonologies/new tools would be better in the development industry but I was wrong.

After - Different tools have different lifecycles, and upgrading one of them may become an extremely difficult and time-consuming task since the new functionality, design changes, or even bug fixes may cause incompatibilities with the other tools. The greater the number tools the worse the problem can become.

**▼ Title Of Book and Chapter**

# Chapter 11. Code in the Language of the Domain

**▼ What are Three Things I learned today**

1. Before – I don’t mind my code as long I understand it and others don’t.

After – I’ve learned that I should make a clean code for the next programmer to see my code or my teammates will have an easy time understanding it too.

2. Before – I used to criticize others code.

After – I realized that as a boy scout rule we should clean other’s mess and shouldn’t be blaming someone.

3. Before – I didn’t know that bad domain is not good for the industry

After – Domains should be improved over time for the next programmer who comes along a few months later to work on the code will thank me. The programmer who comes along a few months later might be me.