**Buying The Right Camera, How My Dear Aunt Sally Made Me a Better Programmer, and Structured Languages Increase Productivity**

David LaCivita

CTIM 281 Software Design and Development

Professor Chuck Bisson

September 29, 2020

**Buying the Right Camera**

Using a structured approach to buying a camera can help you save money and avoid regret once you have a little more photography experience. If you used an unstructured approach, you might buy a camera simply because it is on sale only to find later that it doesn’t suit your needs. Maybe it’s too big, too heavy, or comes with a lot of features you don’t understand or need. It’s likely the camera is missing some features you didn’t even know you needed until you’ve been using the camera for a while. You decide to sell your camera for a loss and buy a better camera that better fits your needs. Now you find yourself in a whole new system and need to sell your lenses at a loss to get new lenses that match your new camera body. You have lost time and money buying and using the wrong equipment.

If you had used a structured approach you could have gotten the right camera the first time. First, you would have decided what are important features you want and what you don’t want. You could read reviews and see what other users are saying about the cameras you are interested in. Once you’ve decided on the features and maybe model and manufacturer you can decide if a brand new camera is the right choice or save some money and get a used camera that still has everything you need. Maybe the camera you want costs more than you have so you wait and save a little money from each paycheck until you can afford it. Finally, you visit the local shops and online retailers for the best deal you can get. You may decide to wait until Black Friday for the absolute best price. By using the structured approach you feel much better about your choice, you have only bought one camera and have not lost any value for selling your equipment. Time to think about a tripod.

**How My Dear Aunt Sally Made Me a Better Programmer**

Operator precedence is of primary importance when determining the output from an expression. Learning the order of precedence and faithfully applying it will help reduce logic errors in your code. Since parenthesis are evaluated first, they should be used to clarify which operations should be evaluated first. Parentheses within parentheses are evaluated first and the temporary result can be operated on by the rest of the expression. Parentheses should be used to keep your intent obvious for debugging and for anyone doing maintenance later. For example:

withTip = total + total \* .20

is the same as

withTip = total + (total \* .20)

but the use of parentheses in the second version makes your intent much cleared to anyone trying to debug or maintain your code.

After parentheses, multiplication and division are of equal precedence and are evaluated from left to right. Finally, addition and subtraction are evaluated from left to right as well. The phrase, “Please excuse my dear Aunt Sally,” has been used for years to help remember the order of precedence. The assignment operator has the lowest precedence. Once all of the arithmetic operations on the right side of the assignment operator have been completed the value can be assigned to the variable on the left side.

**Modern, Structured Programming Languages Improve Productivity**

Modern programming languages that use a structured approach can greatly improve productivity. The three structures: sequence, selection and loops are all that is needed to write clear and professional code. These structures can easily be combined by stacking or nesting them together in an infinite number of ways to solve any programming problem. Additionally, the program can be modularized so multiple programmers can work independently and speed up the whole code writing process.

Unstructured languages are often called spaghetti code and use “goto” commands which make the logic difficult to follow. Programs written in these languages are usually not reusable and can not be modularized. Improving these programs is difficult or impossible and maintenance is too time consuming. It is often best to entirely rewrite these programs in a new, structured language rather than try to maintain them.