Pseudocode vs flowcharts:

Learned flowcharting years ago as a way to plan out code before starting work.

Doesn’t work as well when requirements are undefined or changing.

People I work with can’t read flow charts, and it’s even worse if they try making a flow chart.

Not willing to plan things out before they start coding and build on the fly.

Have not used it for years, but as I recall it went quicker for the project it was used for. Three people were involved and handing off the pseudocode seemed to have fewer issues than a flowchart.

Benefits of GitHub:

By using GitHub I have been able to access my code and working documents from multiple locations, using at least four different machines. Two of those machines have no ports for removable media, and the other two the ability to write to removable media has been disabled.

So whether I’m at home, travelling, or have some spare time at work – I can access the files without having to carry a laptop everywhere I go.

Separation of Concerns:

New and really plays well with my desire to organize code to make it more manageable.

Code reuse:

Not something I learned in this class, but something I’m glad to see being discussed. A surprising number of coders I’ve worked with seem to avoid use of subroutines, functions, or other forms of reusable code. Instead, they hard code full operations each time the need it a program – making it harder to diagnose errors and update code as necessary. Engineers (non-software) seem to be more likely to do this.

The worst case I’ve seen was a VBA macro in an Excel workbook that was 35 pages (8.5 x 11) printed out, most of which was repeated code. A single outdated variable was preventing it from running, but as it declared in at least nine places along with repeating the code that was using it – the user spent three years using a manual work around. After recoding to use subroutines it was reduced to about seven pages.