Notes part 4

Clean C++20. Ch. 2. Build a Safety Net

• Unit-Testing of low-level code is key to good development
• Focus on test of low-level code and components
• Avoid Ice Cream Cone and Cup Cake Anti-Patterns in testing, that mainly focus at Integration, System and GUI testing
• High coverage with Unit-Testing prevents time-consuming and frustrating debugging sessions
• Testing combined with functional programming eliminates most basic errors
• *Design for testability* is good *design for usability*
• Test-Driven Development is faster on bigger systems due to less debugging time
• Software Testing isn't a sole concern of the quality assurance department
– and with small teams the programmer have the sole responsibility of quality assurance
• The quality assurance department is the second safety net, while the programmer is the first
• Test code should have same high quality as production code (p. 22-33)

Clean C++20. Ch. 3. Be Principled

• KISS – Keep it as simple as possible, not simpler
• YAGNI – if You Ain't Gonna Need It – don't design or code it
• DRY – don't repeat it
– copy and paste is a design error
– every piece of knowledge (code, comments etc.) must have a single, unambiguous, authoritative representation in a system
– creating an adequate common abstraction from duplicated code can be tricky as well as deteriorate readability and comprehensibility of the code
• Information Hiding/modularity
– one piece of code should not know the internals of another piece of code
– changing internals does not affect the calling code
• Strong cohesion (p. 53-60)
• Be careful with optimization – wait till later in the development process
• PLA/POLS – Principle of Least Astonishment/Surprise
– important in API design
– calling a function shouldn't result not have unexpected behaviour or mysterious side effects
• The Boy Scout Rule – always leave the code cleaner than you found it