SEPM
Assignment 2

Submitted by:
Srividya Krishnakumar
CS6A
55

1. PRINCIPLES OF FUNDAMENTAL CODING

The principles that guide the coding task are closely aligned with:

- Programming style.
- Programming languages.
- Programming methods.

Fundamental coding principles are the following:

(i) Preparation principles.
(ii) Coding principles.
(iii) Validation principles.

Preparation Principles.

Before writing a line of code, we have to ensure the following:

- Understand the problem.
- Understand basic design principles & concepts.
- Pick a programming language.
- Select a programming environment.

Coding Principles.

As the coding proceeds, following things are ensured:

- Check your algorithms.
- Select data structures.
- Understand the software architecture and create interfaces that are consistent with it.
- Keep conditional logic as simple as possible.
- Create nested loops in a way that makes them easily testable.
- Select meaningful variable names & follow other local coding standards.
- Write code that is self-documenting.
- Create a visual layout (eg. Indentation & blank lines) that aids understanding.

Validation Principle

After completing the coding, following things are ensured:

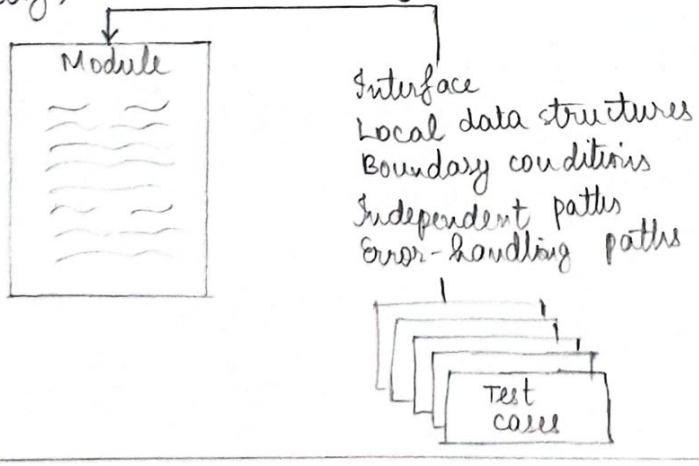
- Conduct a code walkthrough when appropriate.
- Perform unit tests & correct errors you've uncovered.
- Refactor the code.

2. UNIT TESTING.

- Unit testing focuses on verification of the smallest unit of software design, the software component or module.
- Important control paths are tested to uncover errors within the boundary of the module.
- It focuses on the internal processing logic & data structures within the boundaries of a component.
- This type of testing can be conducted in parallel for multiple components or modules.

Tests involved in Unit Testing.

- The module interface is tested to ensure that information properly flows into & out of the program under test.
- Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution.
- All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once.
- Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
- And finally, all error-handling paths are tested.



Common Errors Found in Unit Testing.

- Incorrect arithmetic precedence
- Mixed mode operations.
- Incorrect initialization.
- Inaccuracy.
- Incorrect symbolic representation of an expression.

Boundary Testing.

- Boundary Testing is one of the most important unit testing tasks.
- Software often fails at its boundaries.
- That is, errors often occur when the n^{th} element of an n-dimensional array is processed.

Anti Bugging.

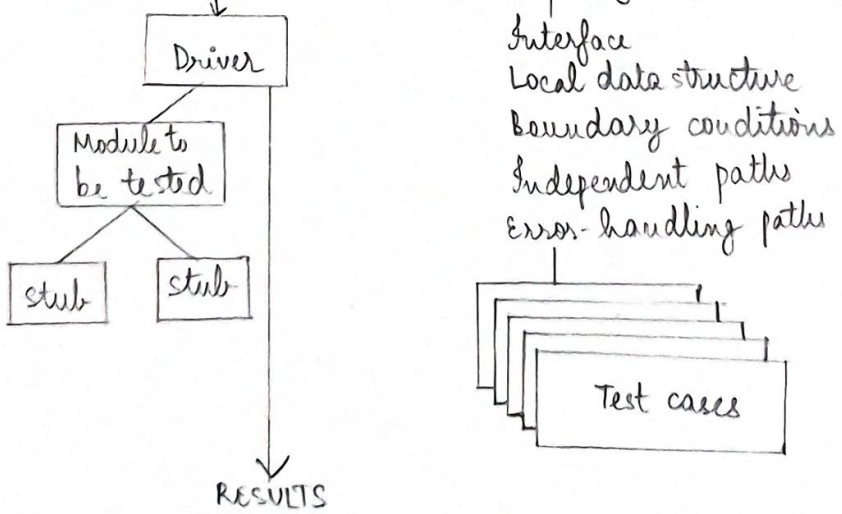
- A good design anticipates error conditions & establishes error-handling paths to reroute or cleanly terminate processing when an error does occur.

Unit Test Procedures.

- The design of unit tests can occur after source code has been generated.
- A review of design information provides guidance for establishing test cases.
- Each test case is coupled with a set of expected results.
- Driver & stub modules have to be developed for each unit test.

Driver & Stub Modules.

- Driver - a dummy "main program" that accepts test-case data & passes such data to the component to be tested, & prints relevant results.
- Stubs - a stub / "dummy subprogram" uses the subordinate module's interface, does minimal data manipulation, prints verification of entry, & returns control to the module undergoing testing.



3. BLACK BOX vs WHITE BOX TESTING.

Black box Testing	White box Testing.
• It is a way of software testing in which the internal structure or the program or the code is hidden & nothing is known about it.	• It is a way of testing software in which the tester has knowledge about the internal structure or the code or the program of the software.
• Mostly done by software testers.	• Mostly done by software developers.
• No knowledge of implementation is needed.	• Knowledge of implementation is needed.
• Outer/external software testing.	• Inner/internal software testing.
• Functional test of software.	• Structural test of software.
• Initiated on the basis of requirement specifications document.	• started after detailed design document.
• No knowledge of programming required.	• Mandatory to have knowledge of programming.
• Behavior testing of software	• Logic testing of software.
• Applicable to higher levels of testing of software.	• Generally applicable to lower levels of testing of software.
• It is also called closed testing.	• It is also called clear box testing.
• Least time consuming.	• Most time consuming.
• Not suitable/preferred for algorithm testing.	• Suitable for algorithm testing.
• Can be done by trial & error ways & methods.	• Data domains along with inner/internal boundaries can be tested.
eg: Search something on google by using keywords.	eg: By input to check & verify loops.