9.x — Chapter 9 summary and quiz

ALEX JANUARY 5, 2024

Chapter Review

Scope creep occurs when a project's capabilities grow beyond what was originally intended at the start of the project or project phase.

Software verification is the process of testing whether or not the software works as expected in all cases. A **unit test** is a test designed to test a small portion of the code (typically a function or call) in isolation to ensure a particular behavior occurs as expected. **Unit test frameworks** can help you organize your unit tests. **Integration testing** tests the integration of a bunch of units together to ensure they work properly.

Code coverage refers to how much of the source code is executed while testing. **Statement coverage** refers to the percentage of statements in a program that have been exercised by testing routines. **Branch coverage** refers to the percentage of branches that have been executed by testing routines. **Loop coverage** (also called the **0, 1, 2 test**) means that if you have a loop, you should ensure it works properly when it iterates 0 times, 1 time, and 2 times.

The **happy path** is the path of execution that occurs when there are no errors encountered. A **sad path** is one where an error or failure state occurs. A **non-recoverable error** (also called a **fatal error**) is an error that is severe enough that the program can't continue running. A program that handles error cases well is **robust**.

A **buffer** is a piece of memory set aside for storing data temporarily while it is moved from one place to another.

The process of checking whether user input conforms to what the program is expecting is called **input validation**.

std::cerr is an output stream (like `std::cout`) designed to be used for error messages.

A **precondition** is any condition that must always be true prior to the execution of some segment of code. An **invariant** is a condition that must be true while some component is executing. A **postcondition** is any condition that must always be true after the execution of some code.

An **assertion** is an expression that will be true unless there is a bug in the program. In C++, runtime assertions are typically implemented using the **assert** preprocessor macro. Assertions are usually turned off in non-debug code. A **static_assert** is an assertion that is evaluated at compile-time.

Assertions should be used to document cases that should be logically impossible. Error handling should be used to handle cases that are possible.

Quiz time

Question #1

In a quiz for lesson 8.x -- Chapter 8 summary and quiz (https://www.learncpp.com/cpp-tutorial/chapter-8-summary-and-quiz/), we implemented a game of Hi-Lo.

Update your previous solution to handle invalid guesses (e.g. 'x'), out of bounds guesses (e.g. `0` or `101`), or valid guesses that have extraneous characters (e.g. `43x`). Also handle the user entering extra characters when the game asks them whether they want to play again.

Hint: Write a separate function to handle the user inputting their guess (along with the associated error handling).

Show Solution (javascript:void(0))

Next lesson
10.1 Implicit type conversion

Back to table of contents

Previous lesson
9.6 Assert and static_assert