

Problem 14.1

1. What did the designers of C get in return for not requiring subscript range checking?

The primary benefit of not requiring subscript range checking is a significant improvement in execution efficiency, which is critical in many system-level or embedded applications. The programmer can perform range checking at a higher level of abstraction that guarantees that subordinate subscripts are within the allowed range. However, this does place the onus for performing range checking onto the programmer.

Problem 14.7

7. In a language without exception-handling facilities, we could send an error-handling procedure as a parameter to each procedure that can detect errors that must be handled. What disadvantages are there to this method?

This approach adds to the function call overhead and can become very cumbersome when multiple exception handlers must be passed. In addition, it essentially requires that all exceptions be handled locally since any method of propagating exceptions to higher levels would be very complicated.

Problem 14.14

14. Summarize the arguments in favor of the termination and resumption models of continuation.

If the event that caused the exception is truly an error, then it is commonly the case that there is no meaningful way to truly recover from it and thus a graceful termination of the program makes the most sense. However, many errors are either recoverable (for instance, trying to write to a read-only file) or are merely normal, but unusual, occurrences (for instance, attempting to read past the end of a file) and ways to recover and resume execution well defined and desired.