Another way of looking at C++ reverse iterators

devblogs.microsoft.com/oldnewthing/20211112-00

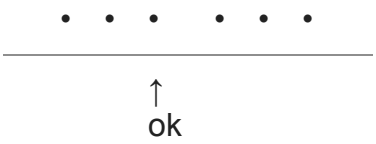
November 12, 2021

Raymond Chen

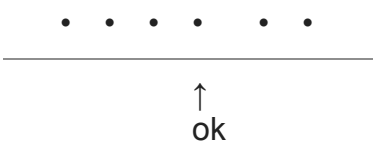
C++ has this thing called a `reverse_iterator` that takes a regular iterator and goes in the opposite direction. You might think that an iterator and its reverse point to the same thing, and it's just the increment and decrement operators that are reversed, but that's not true. An iterator and its reverse point to *different things*.

This reason for this comes down to a quirk of the C++ language: You are allowed to have a pointer "one past the end" of a collection, but you are not allowed to have a pointer "one before the beginning" of a collection.

When moving forward through a collection, you can use the "one past the end" pointer as the sentinel that means "you're done."



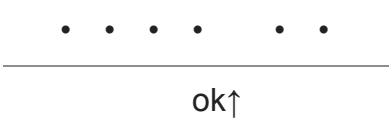
But you don't have such a luxury when going backward:



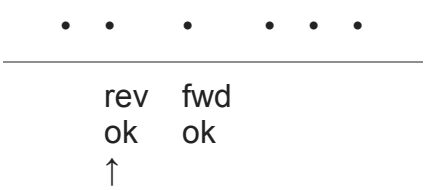
You want to create an *end* sentinel value immediately before the first element, but you can't because the C++ language forbids it.

The standard library finesses the problem by making the reverse iterator "off by one": The element referenced by the iterator is the one *before* the one it nominally points to.

This off-by-one behavior is a bit tricky to wrap your brain around, but here's a way of looking at it that's a bit less wacky: View the pointer as pointing, not at the center of an element, but at its start. When iterating forward, you look to the element in *front* of the pointer, and when iterating backward, you look to the element in *back* of the pointer. In both cases, you look in the direction of motion.



Now the off-by-one behavior is easier to see. A pointer when interpreted as a forward iterator looks forward one element, but when interpreted as a reverse iterator looks backward one element.



Raymond Chen

Follow