The Problem

If you're lucky, the code will have some extra checking to catch this, and report an error. If you are unlucky and the code actually sends too much radiation to the patient, then they would die, just as in the original Therac-25 incident.

In other words, because the user of the `Time` structure set a single field to a nonsensical value, it's possible that your program could cause real injury. **This is clearly unacceptable, and you will need to do something about it.**

There are two problems with implementing `Time` as a `struct`.

- **Structures do not enforce invariants**. Structures use "naked" variables to represent data, so **any part of the program** can modify those variables without any validation. `Time` **expects** certain relationships between its data members, **but cannot enforce those relationships**.
- `Time` is **represented in a particular way**, as two `int` members. We say that code which uses the `Time` data type is **tightly coupled to that implementation**.

Both of these are real problems, and this is what C++ programming is like with raw structures. Code is **brittle**, bugs are more likely, and **changes are more difficult**. So, in the next lesson, let's change gears and represent `Time` in a slightly different way.