Wrap Up

You can play with the code in:

`Chapter Refactoring With Optional And Variant/refactoring_optional_variant.cpp.`

In this chapter, you've seen how to refactor lots of ugly-looking output parameters to a nicer `std::optional` version. The optional wrapper clearly expresses that the computed value might be absent. Also, you've seen how to wrap several function parameters into a separate struct. Having one separate type lets you easily extend the code while keeping the logical structure untouched.

And finally, if you need the full information about errors inside a function, then you might also consider an alternative with `std::variant`. This type gives you a chance to return a full error code.

Head over to the next chapter to learn about enforcing code contracts.