Introduction

Intro provides a short overview of what refactoring with std::variant and std::optional means.

> **Did you know?**
>
> `std::variant` *and* `std::optional` *are called “vocabulary” types because you can leverage them to convey more design information.*

Your code can be much more compact and more expressive.

This chapter will show you one example of how `std::optional` and `std::variant` can help with the refactoring of one function. We’ll start with some legacy code, and through several steps, we’ll arrive with a much better solution. To give you a better understanding you’ll see the pros and cons of each step.