Wrap up

Here's a summary of all the concepts we learned in this chapter!

Things to remember

Here are the things to remember about `std::variant` :

- It holds one of several alternatives in a type-safe way

- No extra memory allocation is needed. The variant needs the size of the max of the sizes of the alternatives, plus some little extra space for knowing the currently active value

- By default, it initialises with the default value of the first alternative

- You can access the value through `std::get` , `std::get_if` or through a form of a visitor

- To check the currently active type you can use `std::holds_alternative` or `std::variant::index`

- `std::visit` provides a way to perform an operation that is implemented for any possible type that might currently be the active one in the variant. Such a polymorphic operation is represented by a callable object that implements its call-operator for every possible type that this variant can hold

- Rarely `std::variant` might get into an invalid state, you can check this issue with the `valueless_by_exception()` method

Since all your concepts are refreshed now, head over to the quiz, to test your understanding!