Enumerated Variables

Just like the other types you've seen, you can create a variable of an enumerated type and initialize the variable with a scoped member of the type like this:

```
Coin c1 = Coin::penny;
// Coin c2 = 1;   // error
```

Note that you **can't** initialize the variable `c2` with its underlying `int` representation. The second line in the example above is an error. You may, however, initialize or assign an integral value, by **explicitly** using a `static_cast`.

```
Coin c3 = static_cast<Coin>(5); // OK, but why do this?
Coin c4 = static_cast<Coin>(3); // Just wrong. No 3-cent coin
```

As you can see, using `static_cast` in this way is **error prone**, and turns off the error checking that C++ provides. The variable `c4` in the example above is simply **undefined**. **Don't do this**.

However, if you want to get the "underlying" value of an enumerated type, you can use `static_cast<int>(c)` where `c` is a `Coin` variable like those shown above. Unlike casting from `int` to `Coin`, casting from `Coin` to `int` **is always safe**.