```
    // Note that the underlying type of Enum is implicit and will be
    // large enough to represent all of these values.
    static_assert(
        std::is_same<std::underlying_type<Enum>::type,unsigned>::value, "");
};

void setCurrentPortToCtrl()
{
    setPort(SysPort::e_CTRL);  // OK, SysPort::Enum promotes to long.
}
```

When the intended client will depend on the cardinal values of the enumerators during routine use, we can avoid tedious, error-prone, and repetitive casting by instead employing a classic, C-style **enum**, possibly nested within a **struct** to achieve explicit scoping of its enumerators. The sections that follow highlight specific cases in which classic, C-style, C++03 **enum**s are appropriate.

Misuse of enum class for collections of named constants

When constants are truly independent, we are often encouraged to avoid enumerations altogether, preferring instead individual constants; see Section 2.1."**constexpr** Variables" on page 302. On the other hand, when the constants all participate within a coherent theme, the expressiveness achieved using a *classic* **enum** to aggregate those values is compelling. Another advantage of an enumerator over an individual constant is that the enumerator is guaranteed to be a **compile-time constant** (see Section 2.1."**constexpr** Variables" on page 302) and a *prvalue* (see Section 2.1."*Rvalue* References" on page 710), which never needs static storage and cannot have its address taken.

For example, suppose we want to collect the coefficients for various numerical suffixes representing *thousands*, *millions*, and *billions* using an enumeration:

```
enum class S0 { e_K = 1000, e_M = e_K * e_K, e_G = e_M * e_K };  // (BAD IDEA)
```

A client trying to access one of these enumerated values would need to cast it explicitly:

```
void client0()
{
    int distance = 5 * static_cast<int>(S0::e_K);  // casting is error-prone
    // ...
}
```

By instead making the enumeration an explicitly scoped, *classic* **enum** nested within a **struct**, no casting is needed during typical use:

```
struct S1  // scoped
{
    enum Enum { e_K = 1000, e_M = e_K * e_K, e_G = e_M * e_K };
```