```
    e_OCT = (10 << 4) + 0x1,
    e_NOV = (11 << 4) + 0x0,
    e_DEC = (12 << 4) + 0x1
};

bool hasThirtyOneDays(MonthOfYear month)
{
    return static_cast<std::underlying_type<MonthOfYear>::type>(month) & 0x1;
}
```

In the example above, we are using a new cross-cutting feature of all enumerated types that allows the client defining the type to specify its underlying type precisely. In this case, we have chosen an **unsigned char** to maximize the number of flag bits while keeping the overall size to a single byte. Three bits remain available. Had we needed more flag bits, we could have just as easily used a larger underlying type, such as **unsigned short**; see Section 2.1. "Underlying Type '11" on page 829.

In case **enum**s are used for encoding purposes, the public clients are not intended to make use of the cardinal values; hence, clients are well advised to treat them as implementation details, potentially subject to change without notice. Representing this enumeration using the modern **enum class**, instead of an explicitly scoped classic **enum**, deters clients from making any use (apart from same-type comparisons) of the cardinal **values** assigned to the enumerators. Notice that implementors of the hasThirtyOneDays function will require a verbose but efficient **static_cast** to resolve the cardinal value of the enumerator and thus make the requested determination as efficiently as possible.

Potential Pitfalls

Strong typing of an enum class can be counterproductive

The additive value in using a scoped enumeration is governed *solely* by whether the stronger typing of its enumerators, *not* the implicit scoping, would be beneficial in typical anticipated usage. If the expectation is that the client will never need to know the specific values of the enumerators, then use of the modern **enum class** is often just what's needed. But if the cardinal values themselves are ever needed during typical use, extracting them will require the client to perform an explicit cast. Beyond mere inconvenience, encouraging clients to use casts invites defects.

Suppose, for example, we have a function, setPort, from an external library that takes an integer port number:

```
int setPort(int portNumber);
    // Set the current port; return 0 on success and a nonzero value otherwise.
```

Suppose further that we have used the modern **enum class** feature to implement an enumeration, SysPort, that identifies well-known ports on our system: