```
    // ...
};
```

Note that the choice of using a nested classic **enum** was deliberate; see Section 2.1.“**enum class**” on page 332.

Replicating constant binary data

Especially in the context of **embedded development** or emulation, a programmer will commonly write code that needs to deal with specific constants, e.g., provided as part of the specification of a CPU or virtual machine, that must be incorporated in the program’s source code. Depending on the original format of such constants, a binary representation can be the most convenient or most easily understandable one.

As an example, consider a function decoding instructions of a virtual machine whose opcodes are specified in binary format:

```
~~#include <cstdint>  // std::uint8_t~~

void VirtualMachine::decodeInstruction(~~std::uint8_t~~ instruction)
{
    switch (instruction)
    {
        case 0b00000000u:  // no-op
            break;

        case 0b00000001u:  // add(register0, register1)
            d_register0 += d_register1;
            break;

        case 0b00000010u:  // jmp(register0)
            jumpTo(d_register0);
            break;

        // ...
    }
}
```

Replicating the same binary constant specified as part of the CPU’s or virtual machine’s manual or documentation directly in the source avoids the need to mentally convert such constant data to and from, say, a hexadecimal number.