**Enumerated types with enum class.**

But, in C++, it is possible to create real *enum* types that are neither implicitly convertible to *int* and that neither have enumerator values of type *int*, but of the *enum* type itself, thus preserving type safety. They are declared with *enum* *class* (or *enum* *struct*) instead of just *enum*:

|  |  |  |
| --- | --- | --- |
|  | enum class Colors {black, blue, green, cyan, red, purple, yellow, white}; |  |

Each of the enumerator values of an *enum* *class* type needs to be scoped into its type (this is actually also possible with *enum* types, but it is only optional). For example:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 | Colors mycolor;    mycolor = Colors::blue;  if (mycolor == Colors::green) mycolor = Colors::red; |  |

Enumerated types declared with *enum* *class* also have more control over their underlying type; it may be any integral data type, such as *char*, *short* or *unsigned* *int*, which essentially serves to determine the size of the type. This is specified by a colon and the underlying type following the enumerated type. For example:

|  |  |  |
| --- | --- | --- |
|  | enum class EyeColor : char {blue, green, brown}; |  |

Here, *Eyecolor* is a distinct type with the same size of a char (1 byte).