在传统 C++中，枚举类型并非类型安全，枚举类型会被视作整数，则会让两种完全不同的枚举类型可以进行直接的比较（虽然编译器给出了检查，但并非所有），甚至枚举类型的枚举值名字不能相同，这不是我们希望看到的结果。
C++11 引入了枚举类（enumaration class），并使用 `enum class` 的语法进行声明：

```
enum class new_enum : unsigned int {
 value1,
 value2,
 value3 = 100,
 value4 = 100
};
```

这样定义的枚举实现了类型安全，首先他不能够被隐式的转换为整数，同时也不能够将其与整数数字进行比较，更不可能对不同的枚举类型的枚举值进行比较。
但相同枚举值之间如果指定的值相同，那么可以进行比较：

```
if (new_enum::value3 == new_enum::value4) {
 // 会输出
 std::cout << "new_enum::value3 == new_enum::value4" << std::endl;
}
```

在这个语法中，枚举类型后面使用了冒号及类型关键字来指定枚举中枚举值的类型，这使得我们能够为枚举赋值（未指定时将默认使用 int）。
而我们希望获得枚举值的值时，将必须显式的进行类型转换，不过我们可以通过重载 `<<` 这个算符来进行输出，可以收藏下面这个代码段：

```
#include <iostream>
template<typename T>
std::ostream& operator<<(typename std::enable_if<std::is_enum<T>::value, std::ostream>::type& stream, const T& e)
{
 return stream << static_cast<typename std::underlying_type<T>::type>(e);
}
```

这时，下面的代码将能够被编译：

```
std::cout << new_enum::value3 << std::endl
```