C++ static code analysis: Functions should not have more than one argument of type "bool"

2-3 minutes

`bool` is often used to implement an `enum` with two values. But when used carelessly, this leads to error-prone code. While it may seem reasonable when defining a boolean function parameter, the meaning might not be so clear at the function call point.

If the function only has a single boolean argument, the potential ambiguity is mitigated: the function name can indicate its purpose, and what `true` and `false` mean. But when a function has more than one boolean argument, it becomes increasingly difficult to come up with descriptive names. Moreover, it becomes very easy for callers to inadvertently swap the argument order. In such cases, it is much clearer to use an explicit `enum` or, if the boolean has a logical relation to another argument, to package them together in a `struct`, where the data member name can give meaning to the boolean. Another option is to split dealing with the multiple boolean arguments into multiple functions because sometimes multiple boolean parameters indicate a function that's trying to do too much.

Noncompliant Code Example

```
class Image;

Image* loadImage(const char *path, bool bw, bool lowRes, bool
createEmptyIfNotExist); // Noncompliant
bool saveImage(const Image *image, const char *path, bool color,
bool highRes);  // Noncompliant

void f(const char* path) {
  Image *image = loadImage(path, false, true, false);
  // do some processing on image
  saveImage(image, path, true, false);
}
```

Compliant Solution

```
class Image;
enum class ColorMode {RGB, BlackAndWhite};
enum class Resolution {Full, Low};
enum class NotExistFallback {CreateEmpty, ReturnNullPtr};

Image* loadImage(const char *path, ColorMode mode, Resolution
res, NotExistFallback fallback);
bool saveImage(const Image *image, const char *path, ColorMode
mode, Resolution res);

void f(const char* path) {
  Image *image = loadImage(path, ColorMode::RGB,
Resolution::Low, NotExistFallback::ReturnNullPtr);
  // do some processing on image
  saveImage(image, path, ColorMode::RGB, Resolution::Low);
}
```