C static code analysis: Magic numbers should not be used

2 minutes

A magic number is a number that comes out of nowhere, and is directly used in a statement. Magic numbers are often used, for instance to limit the number of iterations of a loops, to test the value of a property, etc.

Using magic numbers may seem obvious and straightforward when you're writing a piece of code, but they are much less obvious and straightforward at debugging time.

That is why magic numbers must be demystified by first being assigned a name. This is classically done by using a constant (`constexpr` or `const` if your compiler does not support `constexpr` yet) or an enumeration.

-1, 0 and 1 are not considered magic numbers.

Note that since C++20, some well known mathematical constants, such as pi, are defined in the header `<numbers>`, and should be preferred over defining your own version (see {rule:cpp:S6164}).

Noncompliant Code Example

```
void doSomething(int var) {
  for(int i = 0; i < 42; i++) { // Noncompliant - 42 is a magic number
    // ...
  }

  if (var == 42) { // Noncompliant - magic number
    // ...
  }
}
```

Compliant Solution

```
enum Status {
STATUS_KO = 0,
STATUS_OK = 42,
};

void doSomething(Status var) {
  constexpr int maxIterations = 42; // Compliant - in a declaration
  for(int i = 0; i < maxIterations ; i++){ // Compliant: 0 is excluded, and maxIterations is a named constant
    // ...
  }

  if (STATUS_OK == var) { // Compliant - number comes from an enum
    // ...
```

```
    }
  }
```