Enumeration Scoping in C++

Thursday August 5, 2021

The Problem

I spent way too much time trying to figure out what was wrong with this code:

```
#include <iostream>

using namespace std;

enum enum_1 { ok, warning, error };

enum enum_2 { ok, warning, error };

int main() {
  enum_1 my_value1 = enum_1::ok;
  enum_2 my_value2 = enum_2::error;

  if (my_value1 == enum_1::ok) {
    cout << "my_value1 is OK!" << endl;
  } else {
    cout << "my_value1 is not OK!" << endl;
  }

  if (my_value2 == enum_2::ok) {
    cout << "my_value2 is OK!" << endl;
  } else {
    cout << "my_value2 is not OK!" << endl;
  }

  return (0);
}
```

Trying to build this code produces the following errors:

```
main.cpp:9:15: error: redefinition of enumerator 'ok'
enum enum_2 { ok, warning, error };
              ^
main.cpp:6:15: note: previous definition is here
enum enum_1 { ok, warning, error };
              ^
main.cpp:9:19: error: redefinition of enumerator 'warning'
enum enum_2 { ok, warning, error };
                  ^
main.cpp:6:19: note: previous definition is here
```

```
enum enum_1 { ok, warning, error };
                  ^
main.cpp:9:28: error: redefinition of enumerator 'error'
enum enum_2 { ok, warning, error };
                           ^
main.cpp:6:28: note: previous definition is here
enum enum_1 { ok, warning, error };
                           ^
main.cpp:13:30: error: no member named 'error' in 'enum_2'
enum_2 my_value2 = enum_2::error;
                    ~~~~~~~~~^
main.cpp:21:20: error: no member named 'ok' in 'enum_2'; did you mean simply
'ok'?
if (my_value2 == enum_2::ok) {
                 ^~~~~~~~~~
                 ok
main.cpp:6:15: note: 'ok' declared here
enum enum_1 { ok, warning, error };
              ^
```

A quick search of StackOverflow, and I learned that “old style” enumerations in C++ are unscoped. Since the individual members in enums are global, the member names have to be unique.

Solution 1 (C++11)

If your compiler supports the C++11 standard, the fix is easy. Just add “class” to your enum declarations:

```
#include <iostream>

using namespace std;

enum class enum_1 { ok, warning, error };

enum class enum_2 { ok, warning, error };

int main() {
  enum_1 my_value1 = enum_1::ok;
  enum_2 my_value2 = enum_2::error;

  if (my_value1 == enum_1::ok) {
    cout << "my_value1 is OK!" << endl;
  } else {
    cout << "my_value1 is not OK!" << endl;
  }

  if (my_value2 == enum_2::ok) {
```

```
    cout << "my_value2 is OK!" << endl;
  } else {
    cout << "my_value2 is not OK!" << endl;
  }

  return (0);
}
```

Solution 2

Alternatively, you can wrap your enums in namespaces:

```
namespace scope1 {
  enum enum_1 { ok, warning, error };
}

namespace scope2 {
  enum enum_2 { ok, warning, error };
}
```

Then, you can access the members as scope1::enum_1::ok, scope2::enum_2::warning, etc.