```
    // ...
};
```

Making default values obvious for documentation purposes

Configuration objects that bundle a large number of different properties are a popular artifact in large systems. Though the values might be loaded from, e.g., an appropriate configuration file, they often have meaningful default values. In C++03, the default values for these properties would typically be documented in the header file (`.h`) but actually effected in the implementation file (`.cpp`), which opens the opportunity for the documentation to become out of sync with the implementation:

```
// my_config.h:

#include <string>  // std::string

struct Config
{
   std::string d_organization;  // default value "ACME"
   long long   d_maxTries;      // default value 1
   double      d_costRatio;     // default value 13.2

   Config();
};

// my_config.cpp:

#include <my_config.h>

Config::Config()
    : d_organization("Acme")    // Bug, doesn't match documentation
    , d_maxTries(3)             // Bug, went out of sync in maintenance
{ }                             // Bug, d_costRatio not initialized
```

Default initializers can be used in such cases to work as active documentation:

```
#include <string>  // std::string

struct Config
{
   std::string d_organization = "Acme";
   long long   d_maxTries     = 3;
   double      d_costRatio    = 13.2;

   Config() = default; // no user-provided definition needed any longer
};
```