Why not using std::cout?

- Depending on when you run your application you may wish to see messages on the console or store them into a file or neither of those behaviours. To make logging behaviour controllable by users of the application we provide the utility log feature. One must include the header file util_log.h
    - util::ConfigFile settings;
    - util::Log logging;
    - std::string const newline = util::Log::newline();
    - ...
    - settings.load("my_config_file.cfg");
    - util::LogInfo::on() = util::to_value<bool>(settings.get_value("logging"));
    - util::LogInfo::console() = util::to_value<bool>(settings.get_value("console"));
    - util::LogInfo::filename() = settings.get_value("log_file");
    - logging << "hello" << newline;
- Here we use a configuration file to obtain end-users decided logging behaviour. The behaviour is controlled by setting “on”, “console” and “filename” properties on the LogInfo class. Here after one can use a util::Log instance in much the same way as one would use std::cout or std::cerr streams.

Sprinkles on top

- The utility library contains other features that are nice in combination with logging. For instance util_timestamp.h contains a convenience tool to mark ones log with the time of the day
  - logging << util::timestamp() << newline;
- The util_string_helpers.h is quite useful too. For instance the util::to_value and util::to_string functions are convenient.