

---

# **Logging library documentation**

**Nokia R&D Software Engineer Trainee position  
task**

**Subhan Hagverdiyev**

**Jul 01, 2020**

# 1. Problem Definition

Problem is about writing a logging library in preferred language. The program should contain log levels (DEBUG, INFO, WARNING, ERROR in this order. Log () , getErrors(), clear() functions should be provided. Log entries should have unique IDs. It also should contain some configurable and externally configurable settings. (described in task).

## 2.Solution Design

I have tried to implement everything in header only file. So the other person can easily include header file and can use logging library. Also if we provide a header only library then all the code gets compiled with the same compiler settings and against the same headers so a lot of problems go away (so we'll not end up with error reports on machines and with compilers you've never heard of). I have used singleton design pattern for header because we only need one object from this class. I have also used serialization for making service externally configurable.

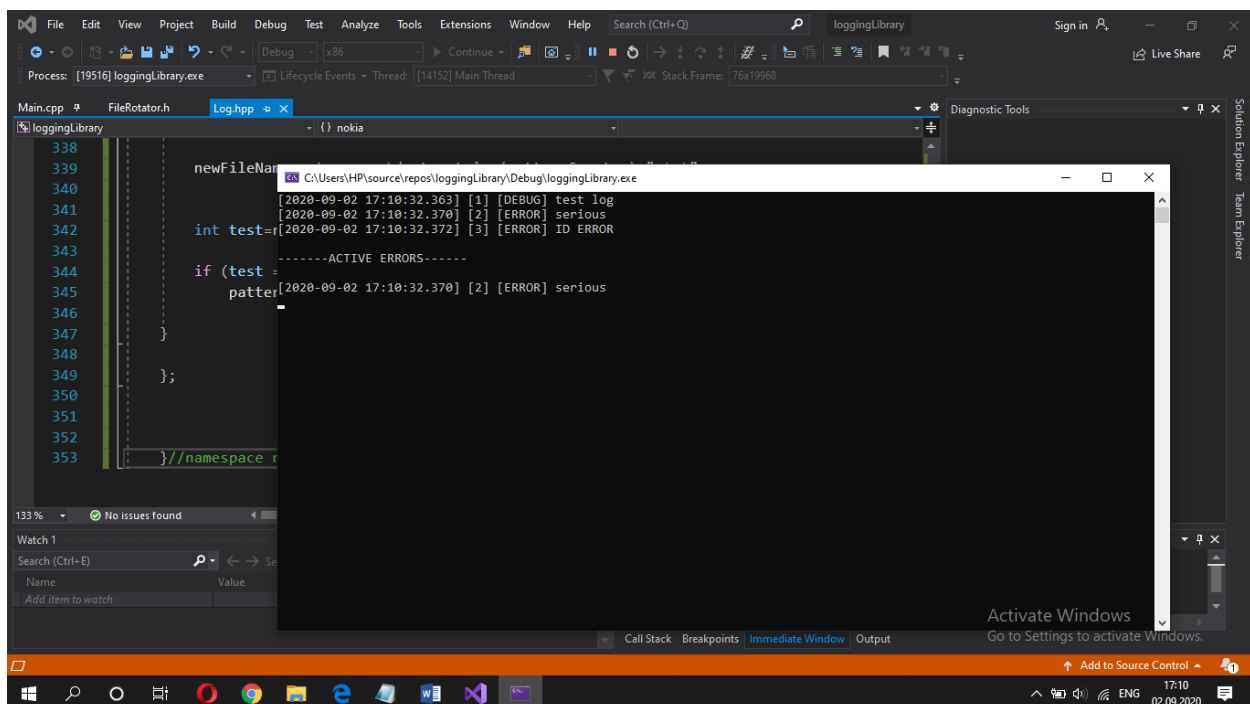
## 3.Detailed design(functions)

- Log.hpp- the main header that contains everything
- namespace nokia ()- for flexibility I have used namespace to make code more readable and maintainable also avoiding name collisions.
- class Log ()- class that contains everything about logging library
- enum class LogLevel()- is showing levels of log.
- enum class LogType()- with this parameter we can easily enable/disable logging to console or file.
- Inline stringify(LogLevel level)- takes the LogLevel enum class type and determine to return what type of log.
- std::map< int, std::string> errors- map to save errors. First int is the ID and the second parameter is the result string which is log entry. The reason that I have used map is for clear function. For deleting the element which has specific ID I used erase() function. If the specific ID could be found, then I am logging this error.
- Making constructor and destructor private I tried to implement singleton design pattern as we only need one instance of this log class.
- Timeformat()- is for setting the time that log entry executed and I have done some modifications to make it user-friendly(like adding brackets .etc)
- static int logid ()- generating unique IDs for log entries. I have thought that the main thing about an ID is the fact that it's unique. Incrementing from 0 is unique.SO I have used simple ID generating technique.
- static Log& get ()- this function helps us to have instance from Log class as I make constructor private.
- Void init () – function takes LogLevel and LogType parameter and use it later for making sure that we have logged correctly (like when if we set log level Warning then in the log function only Warnings and Errors will be printed). As text files will have pattern so taking it as parameter is redundant. I am starting from log.txt and then continue like log0.txt, log1.txt,log2.txt

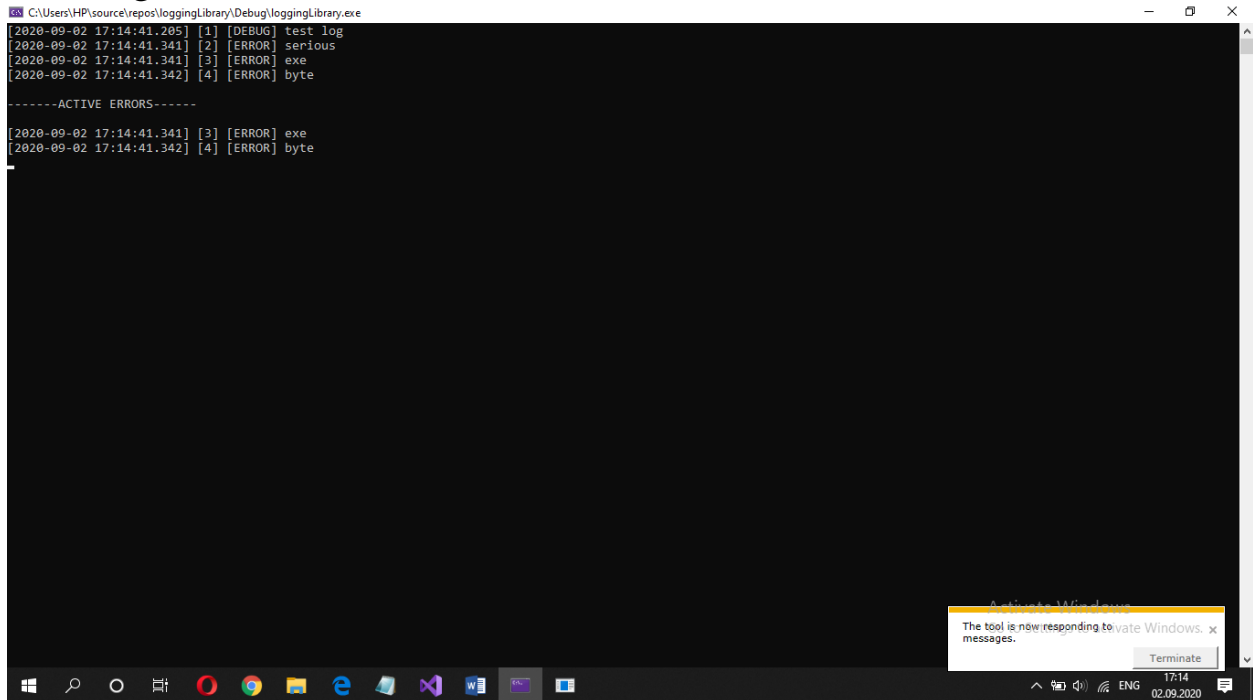
- `void open()`- opening the file. Here also I am checking if the current file full or not. If it is full I am renaming the file and creating another. If the file could not be opened then I am throwing the runtime error.
- `void log()`- this is the logging function. Here the string which contains log entry is save in errors map for later to use it in `getErrors()` function.
- `template<class T>` I am using this because log function can take either string, int and other variables.
- `void getErrors()` – List the active errors also taking into consideration that if the Logtype is file not printing the endlne(This can occur when we are logging to file not to the console and also we are calling `getErrors` function to see the errors on console. As there will be nothing before Active errors there is no need for endlne)
- `void clear(int id)`- takes the id and clear corresponding log from errors list. It is not listed in the `getErrors()` anymore
- `bool IsFull()`- It is for log rotation to check if the file is full or not. As I have set `maxFileSize` limit to 1024 bytes if the the file is “full” then I will create the new one. I am using `tellg()` function to get the size of current file.
- `void RenameFile()`- this functions helps to rename the file when it is full. Current file is `log.txt` and when it is full I am renaming it to `log0` and another `log.txt` is created. Then if it is full renaming it to `log1.txt` and so.

## 4.Testing

Checking the debug, error logs. Getting ID error when we want to clear the ID that does not exist.



Clearing the error which had id=2;



```
C:\Users\HP\source\repos\loggingLibrary\Debug\loggingLibrary.exe
[2020-09-02 17:14:41.285] [1] [DEBUG] test log
[2020-09-02 17:14:41.341] [2] [ERROR] serious
[2020-09-02 17:14:41.341] [3] [ERROR] exe
[2020-09-02 17:14:41.342] [4] [ERROR] byte

-----ACTIVE ERRORS-----
[2020-09-02 17:14:41.341] [3] [ERROR] exe
[2020-09-02 17:14:41.342] [4] [ERROR] byte
```

## 5.References:

<https://www.geeksforgeeks.org/templates-cpp/>

<https://github.com/jgaa/restc-cpp/issues/58>

<https://stackoverflow.com/questions/23831836/how-to-get-the-pointer-of-stdostream-write-for-serializing>