20 How to Write a C++ Class

1. Log class

开发和调试中很重要，输出日志等，是一个很好的例子

C++

```cpp
#include <iostream>

class Log
{
public:  // public method, variable, static variable可以分开放
    const int LogLevelError = 0;
    const int LogLevelWarning = 1;
    const int LogLevelInfo = 2;
private:
    int m_LogLevel = LogLevelInfo;  //stands for menber
public:
    void SetLevel(int level)
    {
        m_LogLevel = level; //可以区分哪些是成员变量，哪些是局部变量
    }

    void Error(const char* message)
    {
        if (m_LogLevel ≥ LogLevelError)
            std::cout << "[ERROR]:" << message << std::endl;
    }

    void Warn(const char* message)
    {
        if (m_LogLevel ≥ LogLevelWarning)
            std::cout << "[WARNING]:" << message << std::endl;
    }

    void Info(const char* message)
    {
        if (m_LogLevel ≥ LogLevelInfo)
        std::cout << "[INFO]:" << message << std::endl;
    }
};

int main()
{
    Log log;
    log.SetLevel(log.LogLevelWarning);  // m_LogLevel = 1
    log.Warn("Hello!");
    log.Error("Hello!");
    log.Info("Hello");    // Warn + Error
    std::cin.get();
}
```

这段代码很糟糕，但是符合一般思路。即先在main中写出可能用到的方法、变量，然后去拓展类。后面会学习如何改进。

C++

```cpp
class Log
{
public:
    enum Level
    {
        LevelError = 0,LevelWarning,LevelInfo
    };
private:
    Level m_LogLevel = LevelInfo;
public:
    void SetLevel(Level level)
    {
        m_LogLevel = level;
    }

    void Error(const char* message)
    {
        if (m_LogLevel ≥ LevelError)
            std::cout << "[ERROR]:" << message << std::endl;
    }

    void Warn(const char* message)
    {
        if (m_LogLevel ≥ LevelWarning)
            std::cout << "[WARNING]:" << message << std::endl;
    }

    void Info(const char* message)
    {
        if (m_LogLevel ≥ LevelInfo)
            std::cout << "[INFO]:" << message << std::endl;
    }
};

int main()
{
    Log log;
    log.SetLevel(log.LevelError);  // m_LogLevel = 0, Log::LevelError
    log.Warn("Hello!");
    log.Error("Hello!");
    log.Info("Hello");    // Warn + Error
    std::cin.get();
}
```