Document number: D

Date: 2021-06-14

Project: <iomanip> Extension for Color

Reply-to: David Aaron Braun da.braun1@hotmail.com

Table of Contents:

Introduction:

C++ is a low-level language which often finds itself used for servers and other passive software, thus there are a lot of C++ programs running in the console/terminal.

That being said there is a very noticeable lack of a certain feature, one found in most other languages today: Standardized Colored Text Output.

I propose to add this feature to the <iomanip> header or another header if there is risk of an ABI break.

Motivation:

The need for at least basic color output is mainly aesthetic, but it is very useful for making text easier to read and for drawing attention to certain information such as errors.

There is no solution in the current standard (C++20) and you're stuck with operation-system specific code.

However, recently Microsoft Windows 10 added support for ANSI Escape Sequences. Their first OS since MS-DOS to support this feature (although it's not enabled by default and some sequences are not supported), thus I believe this is the perfect time to add support for colored text output to C++.

Windows Implementation (Working)

```
#pragma once
#include <iostream>

#ifdef _WIN32
#include <windows.h>
#endif

#ifdef _WIN32
HANDLE  hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

/// <summary>
/// Change the console output foreground color
/// </summary>
const enum class color {
        normal = 0x8,
        black = 0x0,
        white = 0x7,
        bright_white = 0xF,
        red = 0x4,
        bright_red = 0xC,
        yellow = 0x6,
        bright_yellow = 0xE,
        green = 0x2,
        bright_green = 0xA,
        blue = 0x1,
        bright_blue = 0x9,
        cyan = 0x3,
        bright_cyan = 0xB,
        magenta = 0x5,
        bright_magenta = 0xD
}; //End of enum class color

/// <summary>
/// Apply new foreground color
/// </summary>
/// <param name="os"></param>
/// <param name="c"></param>
/// <returns></returns>
std::ostream& operator<<(std::ostream& os, const color c)
{
     if (os.rdbuf() == std::cout.rdbuf())
     {
        SetConsoleTextAttribute(hConsole,
static_cast<WORD>(c));
          return os;
     } //End if
     else
     {
          return os;
     } //End else
} //End of ostream&
```

```
std::cout << color::red << "Hello World!\n";
```

ANSI Implementation (Untested!)

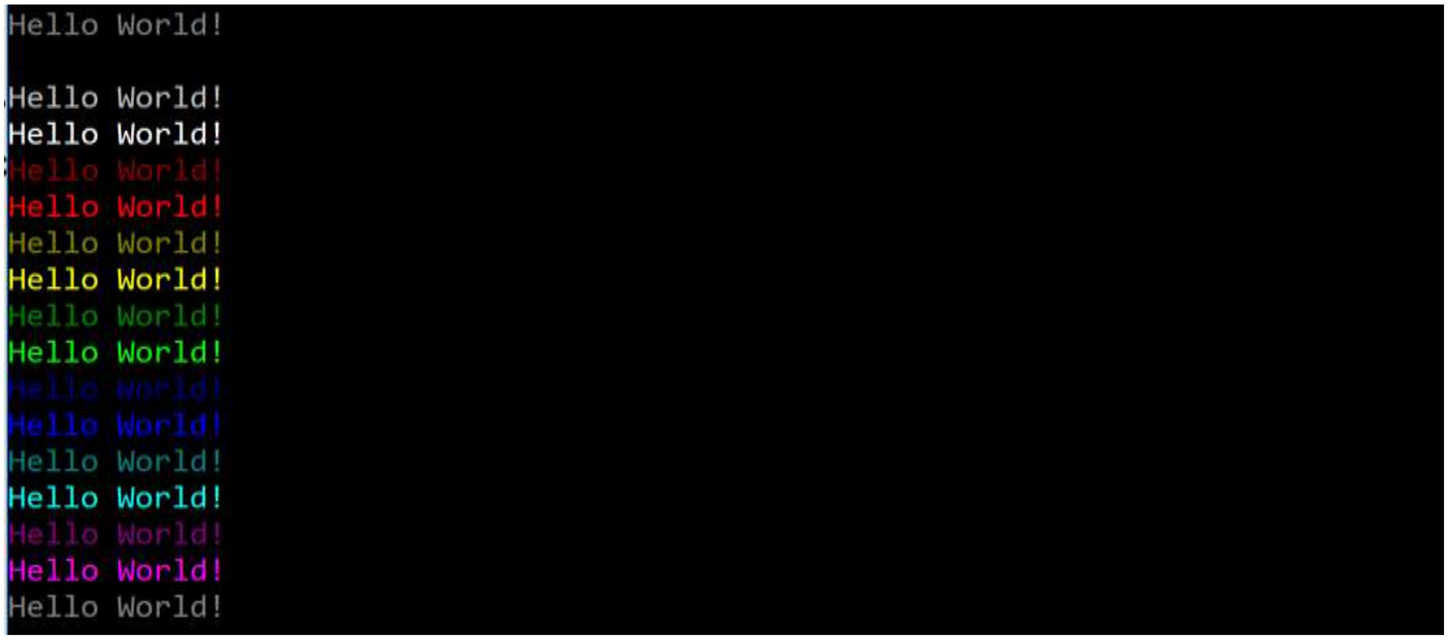
```
#pragma once
#include <iostream>

/// <summary>
/// Change the console output foreground color
/// </summary>
const enum class color {
        normal = 39,
        black = 30,
        white = 37,
        bright_white = 97,
        red = 31,
        bright_red = 91,
        yellow = 33,
        bright_yellow = 93,
        green = 32,
        bright_green = 92,
        blue = 34,
        bright_blue = 95,
        cyan = 36,
        bright_cyan = 96,
        magenta = 35,
        bright_magenta = 95
};//End of enum class color

/// <summary>
/// Apply new foreground color
/// </summary>
/// <param name="os"></param>
/// <param name="c"></param>
/// <returns></returns>
std::ostream& operator<<(std::ostream& os, const color c)
{
     if (os.rdbuf() == std::cout.rdbuf())
     {
        return os << "\033[" << static_cast<int8_t>(c) <<
"m";
     } //End if
     else
     {
          return os;
     } //End else
} //End of ostream&
```

```
std::cout << color::red << "Hello World!\n";
```

The usage will be identical to that of the other functions available in <iomanip> and will produce output similar to this for each color defined in the above enum class:



```
Hello World!

Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

One might argue that this design is so simple anyone could create it with a little research; so why add it to the standard?

I will counter argue that because it is so simple, there is no good reason it should not be added.

The addition of colored text output to the standard will vastly simplify many projects and allow many projects to become cross-platform with no additional effort.

I will also argue that due to the sheer amount of C++ console applications in existence, this should have been a standard feature long ago.

It’s also notable that many other programing languages have had cross-platform colored text output in the console for years (if not decades) now. C++ should have been a leader here and not fallen behind.

Impact On the Standard:

As far as I know my proposal will not present any negative impacts to the current C++ standard.

Design Decisions:

I chose the traditional Win32 Implementation instead of ANSI escape sequences for the Windows implementation because it is the default, and works on all versions of windows going back to Windows 2000.

The ANSI implementation for Mac/Unix/Linux is untested as of the writing of this document. I am looking for suggestions for improvements.

Technical Specifications:

One goal for this addition to the standard is to be simple and light-weight. Thus only 16 colors and only on the foreground. Many more colors are possible even on Windows. However, I wanted a system that worked in a similar way to the rest <iomanip> and one designed for text output as opposed to ASCII graphics

This addition to the <iomanip> library requires:

- An enum class named “color” with each color implemented as in the above table
- An overload of the shift left operator (<<) calling the appropriate native OS functions
- Access to the ostream class from <iostream>
- A safety check making sure that the passed ostream class is std::cout, ensuring text coloring won’t be accidently applied to other steams such as file output streams.

There are no rules regarding accurate color reproduction across different operating systems. (this is likely impossible to achieve with software alone)

There is no support for colors not listed above.

Acknowledgements:

Thank You to the creators of the current C++ standard.

References: none.

goto Top;