CS 3005: Programming in C++

Color Table Class

A color table is an array of colors. It is useful for translating a single number in a range into a color, reliably.

We will use a color table to translate Julia set escape values into colors to create images.

Assignment

Create a class named \texttt{Color} and one named \texttt{ColorTable}. These classes must have the methods listed below, with the specified behaviors.

Programming Requirements

Your \texttt{Color} class must store the following data.

- The integer representation of red, green and blue channels of a color.

Your \texttt{Color} class must have the following methods.

Constructors and Getters

- \texttt{Color( )}; Sets all color channels to value 0.
- \texttt{Color( const int\& red, const int\& green, const int\& blue )}; Sets the color channels to the values provided here. No range checking is applied.
- \texttt{int getRed( ) const}; Returns the value of the red channel.
- \texttt{int getGreen( ) const}; Returns the value of the green channel.
- \texttt{int getBlue( ) const}; Returns the value of the blue channel.
- \texttt{int getChannel( const int\& channel ) const}; Returns the value of the \texttt{channel}th channel. 0 == red, 1 == green, 2 == blue. Returns -1 if the channel is out of range.

Setters

- \texttt{void setRed( const int\& value )}; Changes the red channel to \texttt{value}. If \texttt{value} is less than 0, do not make any changes.
- \texttt{void setGreen( const int\& value )}; Changes the green channel to \texttt{value}. If \texttt{value} is less than 0, do not make any changes.
- \texttt{void setBlue( const int\& value )}; Changes the blue channel to \texttt{value}. If \texttt{value} is less than 0, do not make any changes.
- \texttt{void setChannel( const int\& channel, const int\& value )}; Changes the \texttt{channel}th channel to \texttt{value}. If \texttt{value} is less than 0, do not make any changes. 0 == red, 1 == green, 2 == blue. Does not make changes if \texttt{channel} is out of range.

Other Methods

- \texttt{void invert( const int\& max_color_value )}; Inverts the red, green and blue channels, using \texttt{max_color_value}. If \texttt{max_color_value} is less than the maximum color channel value, make no changes.
- \texttt{bool operator== ( const Color\& rhs ) const}; Returns \texttt{true} if \texttt{*this} and \texttt{rhs} have the same color values. Otherwise, returns \texttt{false}.

Your \texttt{ColorTable} class must store the following data.

- A linear collection of \texttt{Colors}.

Your \texttt{ColorTable} class must have the following methods.

Constructor and Getters

- \texttt{ColorTable( const size_t\& num_color )}; Sizes the \texttt{Color} collection to \texttt{num_color} items.
- \texttt{size_t getNumberOfColors( ) const}; Returns the number of \texttt{Color}s stored.

Setters

- \texttt{void setNumberOfColors( const size_t\& num_color )}; Resizes the collection to hold \texttt{num_color} items.
Previous `Color` contents may or may not be preserved.

### Operators

- `const Color& operator[]( const int& i ) const;` Returns the $i$th `Color` in the collection. If $i$ is out of range, returns a `static` memory `Color` object with all three channels set to $-1$.
- `Color& operator[]( const int& i );` Returns the $i$th `Color` in the collection. If $i$ is out of range, returns a `static` memory `Color` object with all three channels set to $-1$.

### Other Methods

- `void setRandomColor( const int& max_color_value, const size_t& position );` Assigns the $position$th color random values for all three channels. The random values are between 0 and `max_color_value`, inclusive. If `position` is out of range, no change is made. If `max_color_value` is less than 0, no change is made.
- `void insertGradient( const Color& color1, const Color& color2, const size_t& position1, const size_t& position2 );` Change the colors from `color1` to `color2` in the range from $position1$ to $position2$, inclusive, to be gradients from `color1` to `color2`. If `position1` is not less than `position2`, no change is made.

### Additional Documentation

- Consider using the `std::vector` class template.

### Show Off Your Work

To receive credit for this assignment, you must complete the unit tests available in CodeGrinder.