CS 3005: Programming in C++

Overloaded Operators

This assignment requires extending the text-based application for working with PPM images. The user will now be able to add two images, take their difference, and multiple or divide them by a number.

The result will be the ability to blend two images, and to change the overall brightness of an image.

Assignment

In this assignment, you will update the `ppm_menu` program from the previous assignments. All of the previous assignments’ functionality will remain intact.

Programming Requirements

Make changes as described below.

**Update** `PPM.h` and `PPM.cpp`

The following methods must be added to the `PPM` class declaration in `PPM.h` and implemented in `PPM.cpp`.

- `bool operator==( const PPM& rhs ) const;` Returns true if `*this` has the same number of pixels as `rhs`. Otherwise returns false.
- `bool operator!=( const PPM& rhs ) const;` Returns true if `*this` has a different number of pixels than `rhs`. Otherwise returns false.
- `bool operator<( const PPM& rhs ) const;` Returns true if `*this` has a fewer number of pixels than `rhs`. Otherwise returns false.
- `bool operator<( const PPM& rhs ) const;` Returns true if `*this` has a greater number of pixels than `rhs`. Otherwise returns false.
- `bool operator>( const PPM& rhs ) const;` Returns true if `*this` has a greater number of pixels than `rhs` or equal number of pixels. Otherwise returns false.
- `PPM& operator/( const double& rhs ) const;` Creates a new `PPM` object with the same meta data (height, width, max color value) as `*this`. Sets the channel values in the new object to the product of the channel values for `*this` and the value of `rhs`. If the value is greater than max color value, set to max color value. If the value is less than 0, set to 0. Returns `*this`.
- `PPM& operator/( const double& rhs ) const;` Creates a new `PPM` object with the same meta data (height, width, max color value) as `*this`. Sets the channel values in the new object to the sum of the channel values for `*this` and the value of `rhs`. If the value is greater than max color value, set to max color value. Returns the new object.
- `PPM& operator*( const double& rhs ) const;` Creates a new `PPM` object with the same meta data (height, width, max color value) as `*this`. Sets the channel values in the new object to the difference of the channel values for `*this` and `rhs`. If the value is less than 0, set to 0. Returns the new object.
- `PPM& operator-( const double& rhs ) const;` Creates a new `PPM` object with the same meta data (height, width, max color value) as `*this`. Sets the channel values in the new object to the product of the channel values for `*this` and the value of `rhs`. If the value is greater than max color value, set to max color value. If the value is less than 0, set to 0. Returns the new object.

**Update** `image_menu.h` and `image_filters.cpp`
Implement the following functions in a new file \texttt{image_filters.cpp}. Put the declarations in \texttt{image_menu.h}. The functions should use input image 1 as the left hand operand. If the right hand operand is a \texttt{PPM} object, use input image 2. If the right hand operand is a numeric value, use \texttt{getDouble} to ask the user for the value to use. If the operator does not change the left hand operand, assign the result into the output image.

- \texttt{void plusEquals(ActionData& action_data);} Modifies input image 1 by adding input image 2 to it.
- \texttt{void minusEquals(ActionData& action_data);} Modifies input image 1 by subtracting input image 2 from it.
- \texttt{void timesEquals(ActionData& action_data);} Modifies input image 1 by multiplying it by the double obtained by calling \texttt{getDouble} with a prompt of "Factor?".
- \texttt{void divideEquals(ActionData& action_data);} Modifies input image 1 by dividing it by the double obtained by calling \texttt{getDouble} with a prompt of "Factor?".
- \texttt{void plus(ActionData& action_data);} Sets output image to be the sum of input image 1 and input image 2.
- \texttt{void minus(ActionData& action_data);} Sets output image to be the difference of input image 1 and input image 2.
- \texttt{void times(ActionData& action_data);} Sets output image to input image 1 times the double obtained by calling \texttt{getDouble} with a prompt of "Factor?".
- \texttt{void divide(ActionData& action_data);} Sets output image to input image 1 divided by the double obtained by calling \texttt{getDouble} with a prompt of "Factor?".

**Update** \texttt{image_menu.h} and \texttt{image_output.cpp}

- \texttt{void readUserImage2(ActionData& action_data);} Like \texttt{readUserImage1}, but stores into input image 2.

**Update** \texttt{controllers.cpp}

The following functions will require updates to their implementations.

- \texttt{void configureMenu(MenuData& menu_data)} add the new actions with the names and descriptions listed below.

### Table of New Commands

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read2</td>
<td>readUserImage2</td>
<td>&quot;Read file into input image 2.&quot;</td>
</tr>
<tr>
<td>&quot;+&quot;</td>
<td>plus</td>
<td>&quot;Set output image from sum of input image 1 and input image 2.&quot;</td>
</tr>
<tr>
<td>&quot;+=&quot;</td>
<td>plusEquals</td>
<td>&quot;Set input image 1 by adding in input image 2.&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;-&quot;</td>
<td>minus</td>
<td>&quot;Set output image from difference of input image 1 and input image 2.&quot;</td>
</tr>
<tr>
<td>&quot;-=&quot;</td>
<td>minusEquals</td>
<td>&quot;Set input image 1 by subtracting input image 2.&quot;</td>
</tr>
<tr>
<td>&quot;*&quot;</td>
<td>times</td>
<td>&quot;Set output image from input image 1 multiplied by a number.&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;*=&quot;</td>
<td>timesEquals</td>
<td>&quot;Set input image 1 by multiplying by a number.&quot;</td>
</tr>
<tr>
<td>&quot;/&quot;</td>
<td>divide</td>
<td>&quot;Set output image from input image 1 divided by a number.&quot;</td>
</tr>
<tr>
<td>&quot;/=&quot;</td>
<td>divideEquals</td>
<td>&quot;Set input image 1 by dividing by a number.&quot;</td>
</tr>
</tbody>
</table>

**Update** \texttt{Makefile}

This file must now also include a rule for \texttt{clean}. The following commands should work correctly.

- \texttt{make hello} - builds the hello program
- \texttt{make questions_3} - builds the questions_3 program
- \texttt{make ascii_image} - builds the ascii_image program
- \texttt{make image_file} - builds the image_file program
- \texttt{make ppm_menu} - builds the image_file program
- \texttt{make all} - builds all programs
- \texttt{make} - builds all programs (same as \texttt{make all})
- \texttt{make clean} - removes all .o files, and all executable programs

**Additional Documentation**

- \texttt{C++ Reference}
- \texttt{Examples from class}
- \texttt{Digital Image Processing on Wikipedia}
Sample PPM Images

- Monet’s Lilies
- Van Gogh’s Starry Night
- Monet + Van Gogh
- Monet - Van Gogh
- Monet *= 1.5
- Van Gogh /= 2.0

Show Off Your Work

To receive credit for this assignment, you must

- complete the unit tests available in CodeGrinder
- use git to add, commit and push your solution to your repository for this class.

Additionally, the program must build, run and give correct output.

Extra Challenges (Not Required)

- Create additional operators.