CSE 241 Programming Assignment 3

DUE

May 4, 2021, 23:55

Description

- This is an individual assignment. Please do not collaborate.
- If you think that this document does not clearly describes the assignment, ask questions before its too late.

In this assignment, you are going to extend the implementation in PA2.

Additional Functionality of ppmImage class

Implement the following (you can use friend functions or you can implement them as member functions if possible)

- `operator +`: Adds two ppmImage objects. Adds them pixel-by-pixel. They have to be the same size otherwise it returns an empty ppmImage object. (empty objects represents an image which has 0 number of pixels.). Color channels may reach to saturation. Don't go beyond the max value written in the image data. You can assume that both images will have the same max value.
- `operator -`: Similar to `operator +`. subtracts one image from the other. Color data cannot go below `0`.
- `operator <<`: Prints image details and image data to stdout. This is similar to saving the image to a file. Allows cascading.
- `operator ()`: Function-call operator. This operator takes three parameters. The first parameter is the index of the row number. The second parameter is the index of the column number. The third parameter is the color channel. It can either `1`, `2` or `3`. The upper left corner is `(0,0)`. This operator returns a reference to the pixel value.

Test Your Class

Write a main function and test all of the functions of your class. Create objects, read from file, write to a file. change individual pixels, read individual pixels etc...

Implement the following functions in order to test your class implementation

Although you can define and use other class internally in ppmImage class, for the following functions, you are not allowed to use those classes you defined. You can only use `ppmImage` class in these functions.

Standalone Functions

```
// returns 1 if the operation is successful. otherwise, returns 0.
// reads images from filename_image1 and filename_image2. Adds them and saves the resulting
↪  image to filename_image3
int test_addition(const string filename_image1, const string filename_image2, const string
↪  filename_image3);

// returns 1 if the operation is successful. otherwise, returns 0.
// reads images from filename_image1 and filename_image2. Subtracts filename_image2 from
↪  filename_image1 saves the resulting image to filename_image3
int test_subtraction(const string filename_image1, const string filename_image2, const string
↪  filename_image3);


// returns 1 if the operation is successful. otherwise, returns 0.
// reads images from filename_image1 and prints it to stdout
int test_print(const string filename_image1);
```

```
// Re-implement this using the function-call operator implemented.
// this function swaps the color values of every pixel in a given ppm image.
// this function does not create a new object but modifies the given one.
// if swap_choice is 1: swaps red and green
// if swap_choice is 2: swaps red and blue
// if swap_choice is 3: swaps green and blue
// if swap_choice is not 1, 2 or 3: no swaps (this does not mean that the operation is not
↪  successful. the function should return 1 in this case if everything is normal)
// returns 1 if the operation is successful. otherwise, returns 0.
int swap_channels(ppmImage& image_object_to_be_modified, int swap_choice);

// Re-implement this using the function-call operator implemented.
// creates and returns a copy of a new ppmImage object which stores only one color at each
↪  pixel. This simply takes the source and copies only one color information and stores it in
↪  a new object. The other color channels are simply going to be zeros.
//if color_choice is 1: red channel is preserved
//if color_choice is 2: green channel is preserved
//if color_choice is 3: blue channel is preserved
ppmImage single_color(const ppmImage& source, int color_choice);
```

Main Function

```
// Use this main function skeleton otherwise you will get zero

int main(int argc, char** argv)
{
 // check for number of command line arguments
 // the first argument is going to be choice number
 // the second argument is going to be a ppm_file_name1
 // the third argument is going to be ppm_file_name2  (this is optional)
 // the third argument is going to be ppm_file_name3  (this is optional)

 // if choice number is 1
 // check the existance of the optional arguments. If they are not given, exit
 int test_addition(ppm_file_name1, ppm_file_name2, ppm_file_name3);

 // if choice number is 2
 // check the existance of the optional arguments. If they are not given, exit
 int test_subtraction(ppm_file_name1, ppm_file_name2, ppm_file_name3);


 // The rest of the main is similar to PA2. Only the output file name changes.

 // if choice number is 3
 // read ppm_file_name1 using function read_ppm
 // swap red and blue channels
 // write the updated data to a file named "ppm_file_name2". use write_ppm function.

 // if choice number is 4
 // read ppm_file_name1 using function read_ppm
 // swap green and blue channels. use swap_channels function
 // write the updated data to a file named "ppm_file_name2". use write_ppm function.

 // if choice number is 5
 // read ppm_file_name1 using function read_ppm
```

```
// create a new object which only contains red channel data of the file read. use single_color
↪  function
// write the data of the new object to a file named "ppm_file_name2". use write_ppm function.

// if choice number is 6
// read ppm_file_name1 using function read_ppm
// create a new object which only contains green channel data of the file read. use
↪  single_color function
// write the data of the new object to a file named "ppm_file_name2". use write_ppm function.

// if choice number is 7
// read ppm_file_name1 using function read_ppm
// create a new object which only contains blue channel data of the file read. use
↪  single_color function
// write the data of the new object to a file named "ppm_file_name2". use write_ppm function.

}
```

Remarks

- Error checking is important.
- Your program should be immune to the whitespace before any user input.
- Do not submit your code without testing it with several different scenarios.
- Write comments in your code. Extensive commenting is required. Comment on every variable, constant, function and loop. (`10pts`)
- Do not use `#Define` and define macros. Instead use `constant` keyword and define constant variables. If you use macros, you will loose `5` points for each of them.
- Be very careful about the input and output format. Don't print anything extra(including spaces).

Turn in:

- Source code of a complete C++ program. Name of the file should be in this format: `<full_name>_<id>.cpp`. If you do not follow this naming convention you will loose `-10` points.
- Example: `gokhan_kaya_000000.cpp`. Please do not use any Turkish special characters.
- You don't need to use an IDE for this assignment. Your code will be compiled and run in a command window.
- Your code will be compiled and tested on a Linux machine(Ubuntu). GCC will be used.
- Make sure you don't get compile errors when you issue this command : `g++ -std=c++11 <full_name>_<id>.cpp`.
- A script will be used in order to check the correctness of your results. So, be careful not to violate the expected output format.
- Provide comments unless you are not interested in partial credit. (If I cannot easily understand your design, you may loose points.)
- You may not get full credit if your implementation contradicts with the statements in this document.

Late Submission

- Not accepted.

Grading (Tentative)

- `Max Grade : 100`.
- Multiple tests(at least `5`) will be performed.

All of the followings are possible deductions from `Max Grade`.

- Do **NOT** use hard-coded values. If you use you will loose `10pts`.
- No submission: `-100`. (be consistent in doing this and your overall grade will converge to `N/A`) (To be specific: if you miss `3` assignments you'll get `N/A`)
- Compile errors: `-100`.
- Irrelevant code: `-100`.
- Major parts are missing: `-100`.
- Unnecessarily long code: `-30`.
- Inefficient implementation: `-20`.
- Using language elements and libraries which are not allowed: `-100`.
- Not caring about the structure and efficiency: `-30`. (avoid using hard-coded values, avoid hard-to-follow expressions, avoid code repetition, avoid unnecessary loops).
- Significant number of compiler warnings: `-10`.
- Not commented enough: `-10`. (Comments are in English. Turkish comments are not accepted).
- Source code encoding is not `UTF-8` and characters are not properly displayed: `-5`. (You can use 'Visual Studio Code', 'Sublime Text', 'Atom' etc... Check the character encoding of your text editor and set it to `UTF-8`).
- Missing or wrong output values: `Fails the test`.
- Output format is wrong: `-30`.
- Infinite loop: `Fails the test`.
- Segmentation fault: `Fails the test`.
- Fails 5 or more random tests: `-100`.
- Fails the test: `deduction up to 20`.
- Prints anything extra: `-30`.
- Unwanted chars and spaces in output: `-30`.
- Submission includes files other than the expected: `-10`.
- Submission does not follow the file naming convention: `-10`.
- Sharing or inheriting code: `-200`.