User Guide

BlockApps Photo Negative tool

Developed by,
Devarsh Patel
Software Engineer
dbpatel020500@gmail.com

Introduction

The purpose of this project is to create negative of a given uncompressed and 24-bits per pixel BMP formatted image in a non-destructive manner. The project is completely build using **C++** programming language without any 3rd party library. The project is compiled using build automation tool called **CMake**(version: 3.25).

Github: repo

Project Structure

root:

- build
    - blockapps <executable>
- images <sample input files>
    - blackbuck.bmp
    - snail.bmp
- lib <image class>
    - MImageClass.cpp <implementation>
    - MImageClass.h <header>
- CMakeLists.txt <cmake_file>
- main.cpp <driver>
- *.bmp <output>

Assumptions

1) Only supports uncompressed, 24 bits per pixel BMP formatted image files.
2) Executable `blockapps` in `build/` is compiled on MacOS System. For, windows and linux, please compile new executable using native CMake tool.

Commands

The executable supports 2 commands:

```
1) $ blockapps  --help
2) $ blockapps <input_filepath> <output_filename>
```

Note: the output file will be created in the present working directory.

Explanation

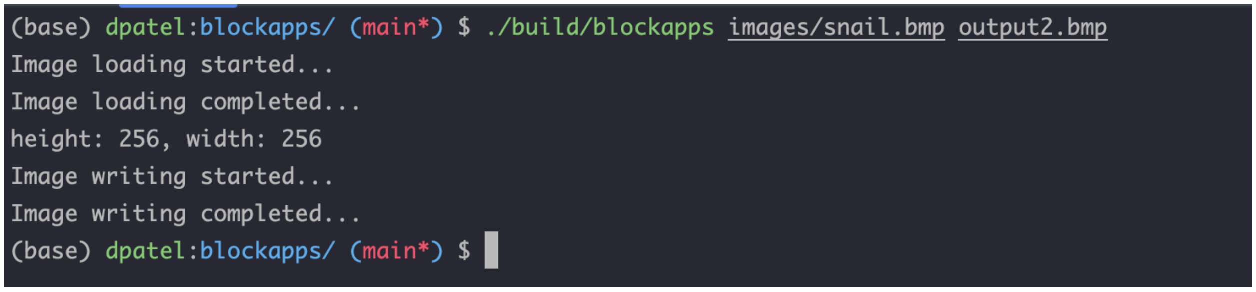
The project contains a dedicated classes for pixels and images. `Pixel` class provides the layout for each pixel in the image and `MImageClass` handles and process the BMP image file. This `MImageClass` provides following functionality:

1) open() : opens the image file and store the pixel values in the memory.
2) isValid() : validates the image stored in the memory.
3) negative() : produce a negative of the image using the maxValue from the image.
4) save() : save the class image into the output file.

Negative Calculation: <max_pixel_value> - <pixel_value>

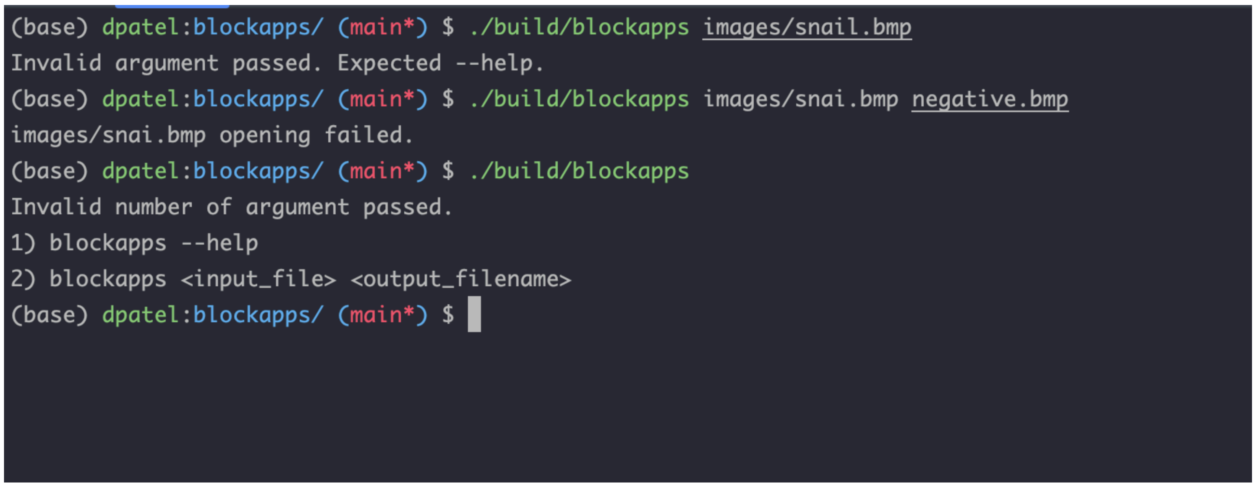
Note: C++ programming language is used here because of its versatility in different OS. Along with CMake , it becomes easy to build executables and deploy it.

Successful Execution



```
(base) dpatel:blockapps/ (main*) $ ./build/blockapps images/snail.bmp output2.bmp
Image loading started...
Image loading completed...
height: 256, width: 256
Image writing started...
Image writing completed...
(base) dpatel:blockapps/ (main*) $
```

Error Handling



```
(base) dpatel:blockapps/ (main*) $ ./build/blockapps images/snail.bmp
Invalid argument passed. Expected --help.
(base) dpatel:blockapps/ (main*) $ ./build/blockapps images/snai.bmp negative.bmp
images/snai.bmp opening failed.
(base) dpatel:blockapps/ (main*) $ ./build/blockapps
Invalid number of argument passed.
1) blockapps --help
2) blockapps <input_file> <output_filename>
(base) dpatel:blockapps/ (main*) $
```

The above screenshot shows few of the error handled by the executable. There are many more format errors implemented:

1) Compression format Error
2) 24-bits per pixel format Error
3) Validation Error

Result

The command line tools is tested on two BMP image files with no compression and 24-bits per pixel format. The image files can be found under images folder.

BMP Images (Input)	Negatives (Output)