Read PBM

Submit Solution

Create the `pbm.h` and `pbm.cpp` files that allow you to use the following class:

```
struct BinaryImage{
    int W;
    int H;
    std::vector<uint8_t> ImageData;
};
```

and the member function:

```
bool BinaryImage::ReadFromPBM(const std::string& filename);
```

The method must read a binary bitmap (PBM) **8-pixels per byte** from file, store data into the `BinaryImage`, and return `true` or `false` depending on whether the image was read correctly or not.

The PBM file consists of:

1. A *magic number* to identify the type of file. The magic number of a PBM image is the sequence of two characters `P4`.
2. The character `'\n'`, that is the Line Feed (LF), that is the character 10 (0x0A), that is a C new line. Beware that this cannot be the pair `"\r\n"`.
3. An optional comment identified by a `'#'` character followed by any sequence of characters and ending with the `'\n'` character. During reading, it is possible to verify if the `'#'` character is present, otherwise this field is not present.
4. The width of the image (W), formatted as a sequence of ASCII characters in decimal.
5. The character `' '`, that is a space.
6. The height of the image (H), formatted as a sequence of ASCII characters in decimal (*i.e.* the number of rows).
7. The character `'\n'`.
8. A raster of H rows, in order from top to bottom. Each row is W bits, **packed 8 to a byte**, with don't care bits to fill out the last byte in the row. Each bit represents a pixel: **1 is black, 0 is white**. The order of the pixels is left to right. The order of their storage within each file byte is most significant bit to least significant bit. The order of the file bytes is from the beginning of the file toward the end of the file.

Note that in the PBM format `0` is white and `1` is black and the same convention must be used in the `BinaryImage`.

`W` and `H` are respectively the width and the height of the image, `ImageData` is a vector of `uint8_t` with the image data. Each byte of the vector contains the values of 8 pixels of the input image. If the width of the image is not a multiple of eight, the last byte of each row is 0-padded.

You can extend the `BinaryImage` class as you prefer.