Guide to Use BMPMini for BMP Image Processing

Introduction

BMPMini is a lightweight C++ class for reading, processing, and writing BMP images. This guide explains how to use its functions effectively.

---

1. Loading an Image

Creating a BMPMini Object and Reading an Image

```
image::BMPMini bmp("input.bmp");
```

This **creates an object** named bmp that automatically reads:

- The **BMP header** (width, height, color channels, etc.)
- The **pixel data** from input.bmp

Mandatory: You must create a `BMPMini object` with an image filename before using any other functions.

---

2. Getting Image Details

Retrieving Image Metadata

```
auto img = bmp.get();
```

This function returns:

- img.width → Image width (pixels)
- img.height → Image height (pixels)
- img.channels → Number of color channels (1 = grayscale, 3 = RGB)
- img.data → Pointer to pixel data

Example Usage:

```
std::cout << "Width: " << img.width << ", Height: " << img.height
          << ", Channels: " << img.channels << std::endl;
```

---

3. Extracting Pixel Data

Extracting RGB Channels from a 24-bit Image

```
unsigned char *R = new unsigned char[img.width * img.height];
unsigned char *G = new unsigned char[img.width * img.height];
unsigned char *B = new unsigned char[img.width * img.height];
bmp.getRGBChannels(R, G, B);
```

This separates the **Red, Green, and Blue** channels from an RGB image.
Use Case: Needed when applying image processing on separate color channels.

Extracting Grayscale Data from an 8-bit Image

```
unsigned char *Gray = new unsigned char[img.width * img.height];
bmp.getGrayChannel(Gray);
```

Retrieves pixel intensity values for an **8-bit grayscale image**.
Use Case: Required for processing monochrome images.

---

4. Writing Processed Images

Saving an RGB Image

```
bmp.writeRGB("output.bmp", R, G, B, img.width, img.height);
```

Writes the modified RGB image to a new BMP file.

Saving a Grayscale Image

```
bmp.writeGreyScale("output_gray.bmp", Gray, img.width, img.height);
```

Writes a processed grayscale image to a BMP file.

---

5. Cleaning Up Memory

After processing, always free allocated memory:

```
free(R);
free(G);
free(B);
free(Gray);
```
