Aluno: Juliano Felipe Prass da Silva. Correções de PID.

Foram colocados argumentos adicionais para teste. Ex.:

- Para formatar o BMP em MBT: PID.exe –write Exemplo.bmp;
- Para ler o MBT, quantizar e colocar em BMP: PID.exe –read PID_OUT.mbt;
- Para realizar os dois acima: PID.exe –full Exemplo.bmp.

Abaixo, seguem as principais alterações do código para corrigir os problemas principais encontrados no código durante a apresentação do trabalho. O código enviado tem mais alterações, no entanto, as listadas abaixo devem ser suficientes para o funcionamento como desejado. As linhas com “verde” na barra esquerda das imagens são as linhas alteradas.

Além da correção de bugs, foram retiradas funções sem uso na execução e retiradas cópias temporárias do mapa de pixels (Reduzindo consumo de memória).

Alteração no construtor do BMP que leva como parâmetro um MBT. Alterados “tipo”, do File Header; “biSize” e “biSizeImage” do Info Header. Por quê? Tais campos são utilizados com propósitos diferentes no MBT, então devem ser alterados para fazer sentido no BMP.

```
BMP::BMP(BITMAPFILEHEADER fil, BITMAPINFOHEADER inf, std::vector<Pixel> palette, MBT* i) //https://stackoverflow.com/que
   : PersistableIMG(i->getLines(), i->getColumns(), i->getRedBits(), i->getGreenBits(), i->getBlueBits(), i->getMap()),
     file_header(fil), info_header(inf), palette(palette) {

    file_header.Reserved2 = 0;
    file_header.Reserved = 0;
    file_header.Type = 0x4D42;
    info_header.biSize = 40;
    info_header.biSizeImage=0;

    if(palette.size() > 0){
        file_header.OffsetBits = 54 + (palette.size()*4);
        file_header.Size = BMP::BMP_file_size(palette.size(), no_lines, no_columns, 8);
        info_header.biBitCount = 8;
        info_header.biSizeImage=0;
    } else {
        file_header.OffsetBits = 54;
        info_header.biSize = BMP::BMP_file_size(palette.size(), no_lines, no_columns, 24);
        info_header.biSizeImage = info_header.biSize-54;
    }

    #ifdef LOG_PRINT
        std::cout << "BMP compatível com MBT\n";
    #endif // LOG_PRINT
}
```

Alterado a ordem em que os pixels eram lidos para o mapa de pixels do MBT. Antes estes eram copiados invertidos (horizontalmente), mas, no momento da escrita, o BMP também realiza a inversão. Agora a inversão só ocorre no BMP e a imagem resultante fica como deveria ser.

```
void MBT::readWithoutPalette(std::ifstream& InputStream){
    unsigned int lines = info_header.biHeight;
    unsigned int cols =  info_header.biWidth;
    unsigned int lines_and_cols = lines*cols;
    Image::pixelMap.clear();
    Image::pixelMap.reserve(lines_and_cols);

    unsigned int i,j;
    for (i=0; i<lines_and_cols; i++){ //Create "n" "empty" positions

    RGBTRI lineBuffer[cols];

    for(i=0; i<lines; i++){ //Read lines
        InputStream.read((char*)&lineBuffer, sizeof(lineBuffer));
        for (j=0; j<cols; j++){
            RGBTRI bgrP = lineBuffer[j];
            Image::pixelMap[i*cols+j] = Pixel(bgrP.rgbBlue, bgrP.rgbGreen, bgrP.rgbRed);
        }
    }
}
```

A matriz “buffer” estava com as dimensões invertidas: ColunasxLinhas ao invés de LinhasxColunas. A maneira com que estava antes causava problemas de acesso quando a imagem de entrada possuía mais linhas que colunas (e.g.: “Flores.bmp”). Outra alteração é a remoção do OMP: Operações de cópia de memória, então várias Threads não trariam muito ganho.

```
void MBT::writeWithoutPalette(std::ofstream& Output){
    ///Nao e generico, utiliza apenas formato 24 bits e 3 canais
    unsigned int cols = info_header.biWidth;
    unsigned int lines = info_header.biHeight;
    unsigned int i,j;
    RGBTRI buffer[lines][cols];

    Pixel p;
    for(i=0; i<lines; i++){
        for (j=0; j<cols; j++){
            p = Image::getPixel(i,j);
            buffer[i][j] = triBuilder(p);
        }
    }

    for(i=0; i<lines; i++){
        Output.write((char*)&buffer[i], sizeof(buffer[i]));
    }
}
```

No momento da escrita, o MBT ainda estava com o campo “File_Header.Type” como “BM”, o que fazia com que a operação de leitura fosse abortada (Assinatura errada do formato). Analogamente para o campo “Info_header.biSize”, onde o leitor de MBT chamava leitura errônea (Com paleta, quando deveria ser sem).

```cpp
int MBT::writeToFile(std::string FileName){
    std::ofstream Output(FileName,std::ofstream::binary); //Não verifica se ha arquivo já existente

    file_header.Type = 0x424D;
    info_header.biSize = 0;
    Output.write(reinterpret_cast<char *>(&file_header), sizeof(BITMAPFILEHEADER));
    Output.write(reinterpret_cast<char *>(&info_header), sizeof(BITMAPINFOHEADER));

    Output.flush();

    if (palette.size() == 0){ writeWithoutPalette(Output); }
    else { writeWithPalette(Output);}

    Output.close();

    #ifdef LOG_PRINT
        std::cout << "MBT escrito em arquivo.\n";
    #endif // LOG_PRINT
    return 0;
}
```