

# 質問に対する回答

# Q. 「まとめ」とは？

- 「課題」を達成できたのか否かの「結論」に相当する項目です.
- 「結論」でも良いと思います.

# Q. フローチャートは不要なのか？

「いまさら議論するのも馬鹿らしいけど、フローチャートなんぞはものの役に立たない」

「なぜそんなにフローチャートが嫌われているかというと、単なる時間の浪費というだけでなく、フローチャートはプログラマの思考を阻害するからだ。」

「「絶対ダメ」などとは言ってないと思うけど、あくまで「フローチャートが全く非効率で、実用的でない」ということなのだ。」

「実際、オブジェクト指向プログラムにおいて、全体の流れをフローチャートで書くのは困難です。」

全てのプログラムについてフローチャートを書かせる意味はあまりない  
組み込み系のような手続き型のプログラムはフローチャートを書く教育効果は高い

ご参考

<http://d.hatena.ne.jp/JavaBlack/20080719/p1>

<http://blog.livedoor.jp/dankogai/archives/51083212.html>

# Q. アルゴリズムの説明をどのようにすれば良いのか？

- 今回は、フローチャートを描くのは時間の無駄と判断し不要とした
- (次回以降は)どうしてもフローチャートを用いたほうが説明しやすいのであれば、フローチャートを用いても良い(PAD図でも良い)
- 利用する言語によらない、画像処理の方法を示すのが「アルゴリズム」とであると解釈して頂きたい

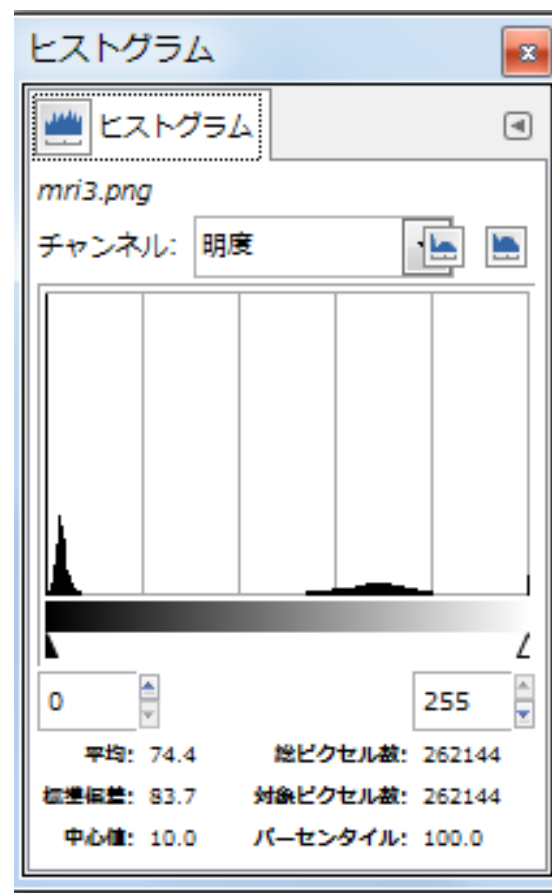
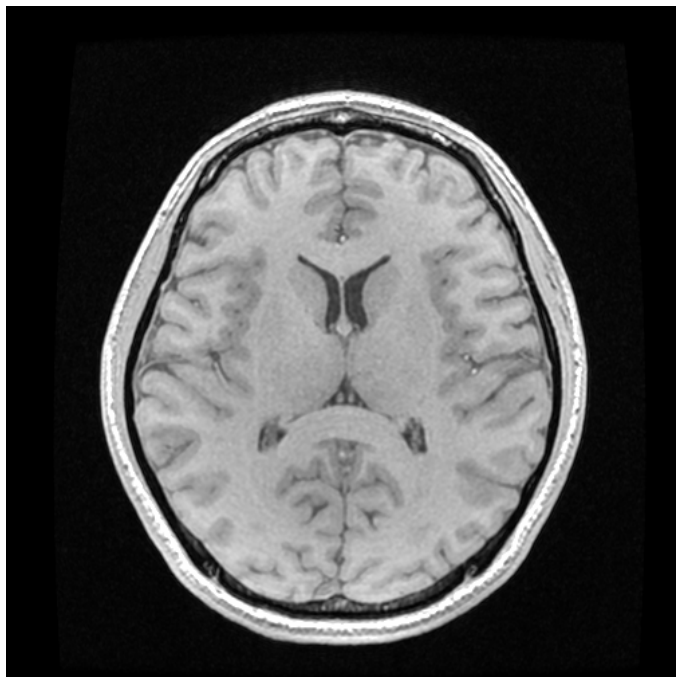
# 採点のポイント

- レポートの体裁をなしているか(5点)
- アルゴリズムが誰にでも伝わる内容となっているか(5点)
- 結果, 考察は十分であるか(5点)

# ヒストグラム

# ヒストグラムとは

- 横軸を輝度値，縦軸をその輝度値の出現頻度としたグラフ



# 目 的

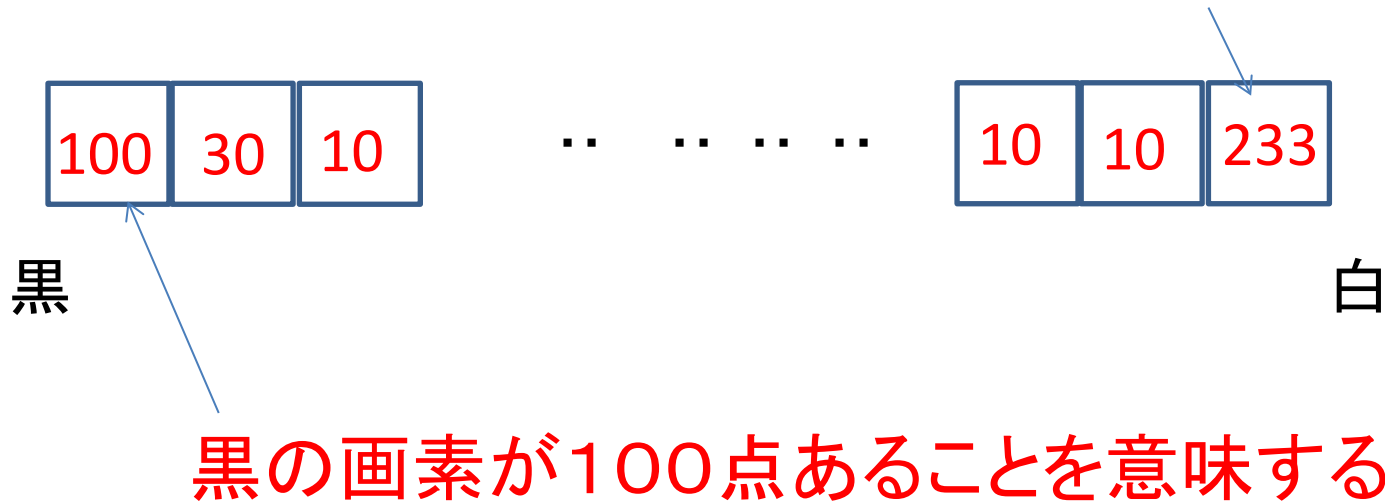
- ヒストグラムのデータをテキストファイルに出力する
- ヒストグラムのデータを画像として出力する
- 全面が黒, 全面が白およびグラデーションの画像をヒストグラム表示した結果を用いて, 結果が正しいか否かを確認せよ



# データの格納

- ヒストグラムの画素数を格納する変数(例えば) `int histogram[256]`を定義する

例: 配列 `histogram[]`




# 考え方

1画素ずつスキャンする



# 考え方

```
histogram[image[0][0][0]]++;
```



128	255	0	128
255	0	128	255
0	128	255	0
128	255	0	128

# 考え方

```
histogram[image[0][0][0]]++;
```



日本語訳: 配列histogramの  
要素番号image[0][0][0]の値を1足す



全ての画素のスキャンが終了したときの  
histogramが各輝度の画素数分布になる

# 課題

- ヒストグラムのデータをテキストファイルに出力せよ
- ヒストグラムのデータを画像データとして出力せよ

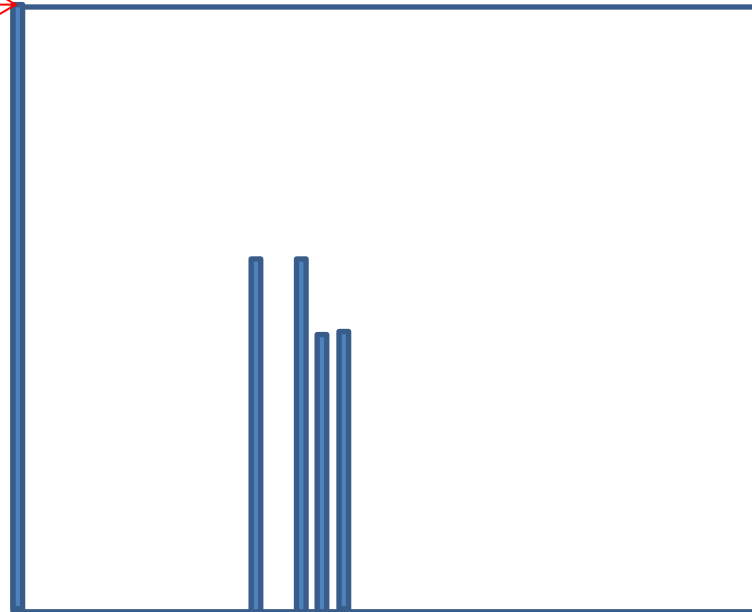
# 画像出力について

- 画像の横幅は256pixelに固定する
- 画像の縦幅は512pixelとする

例えば100だったら



縦は $100 * 5.12$  pixel  
で画像を出力する



# 画像出力について

- 配列変数histogramから最大値(max)を探索する
- ヒストグラムの値を $(512/\text{max})$ 倍して出力画像の縦の大きさとする



最大値(例えばint max)を検索したあと  
(int)  $(512 * \text{ヒストグラムの値} / \text{max})$ とする

# GIMPによるヒストグラム表示

- メニューバーの「色」→「色の情報」→「ヒストグラム」によって表示する
- 実行結果と比較して、実行結果の妥当性を確認せよ



