モジュール設計

現時点のプログラムの構造

- main.cに全ての変数と関数を定義している
  - 可読性低い，毎回全てコンパイル必要，秘匿性低い

```
#include ...
#define ...
構造体の定義
関数プロトタイプ宣言

int main(int argc, char *argv[])
{
  ...
}

iioLoadFile関数の定義
iioSaveFile関数の定義
iioMallocImageBuffer関数の定義
iioFreeImageBuffer関数の定義
ipCopy関数の定義
```

iioLoadFileにしか関係しない定数

ファイルの読み書きや
領域の確保解放に関係するもの

画像処理（画素値の編集）に関係するもの

モジュール設計

- 機能ごとに独立したファイルに分割する
  - 可読性の向上，分割コンパイル可能，情報の秘匿

main.c
画像ファイルをロードする関数を呼び出す
画像処理関数を呼び出す
画像ファイルをセーブする関数を呼び出す
バッファを解放する

img_io.c
画像ファイルのロード関数の定義
画像ファイルのセーブ関数の定義
画素値用バッファを確保する関数の定義
画素値用バッファを解放する関数の定義

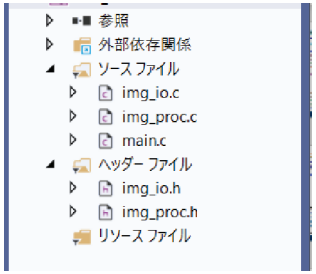
img_proc.c
画像処理のための関数の定義

モジュール設計の方針

- main
  - 各ヘッダファイルのinclude
  - main関数
- img_io
  - PIXEL構造体およびIMAGE構造体の定義
  - iioLoadFile，iioSaveFile関数
  - iioMallocImageBuffer，iioFreeImageBuffer関数
- img_proc
  - ipCopy関数
  - その他の画像処理関数

【作業課題1/11】モジュール化

- main.cと同フォルダに以下のcファイルとヘッダファイルを新規に作成し，プロジェクトに追加
  - img_io.c
  - img_io.h
  - img_proc.c
  - img_proc.h



Visual Studioの場合，
こうなればOK

【作業課題2/11】モジュール化

- img_io.hの作成
  - main.cから，以下の構造体定義と関数プロトタイプ宣言を移動させる

main.c

```
typedef struct {
   unsigned char r;
   unsigned char g;
   unsigned char b;
} PIXEL;

typedef struct {
   int    xsize;
   int    ysize;
   int    level;
   PIXEL **pBuffer;
} IMAGE;

int  iioLoadFile(IMAGE *pImage, const char *fname);
int  iioSaveFile(IMAGE *pImage, const char *fname);
void iioMallocImageBuffer(IMAGE *pImage);
void iioFreeImageBuffer(IMAGE *pImage);
```

img_io.h

```
#ifndef 識別子
#define 識別子

   ここに移動

#endif
```


【作業課題3/11】モジュール化

- img_io.hのインクルードガード
  - #ifndef, #define, #endifを忘れずに
  - 識別子は以下のルールに従うこと

本課題の識別子の命名規則：
　拡張子を除いたファイル名_学籍番号

学籍番号123456Aの人がimg_io.hの識別子を作成する場合の例:
　IMG_IO_123456A

img_io.h

```
#ifndef 識別子
#define 識別子

   ここに移動

#endif
```


【作業課題4/11】モジュール化

- img_io.cの作成
  - 適切なヘッダファイルをincludeする

img_io.c

```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "img_io.h"
```

（Warning抑制のためのもの）
malloc関数に関係するヘッダ
FILEに関係するヘッダ
str関数に関係するヘッダ

【作業課題5/11】モジュール化

- img_io.cの作成
  - 適切なヘッダファイルをincludeする

img_io.c

```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "img_io.h"
```

先に作成したimg_io.hをinclude
【補足】
< >で囲むと，プリプロセッサはまずシステムで定められた場所にヘッダファイルがあるか探す．
" "で囲むと，プリプロセッサはまずカレントフォルダ内にヘッダファイルがあるか探す．
自作ヘッダファイルは" "で囲むとよい

【作業課題6/11】モジュール化

- img_io.cの作成
  - 適切な定数をdefineする

img_io.c

```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "img_io.h"

#define LINEMAX 100
```

LINEMAXはiioLoadFile関数でしか使用しないので，main.cからimg_io.cに移動する．

【作業課題7/11】モジュール化

- img_io.cの作成
  - 以下4つの関数定義をmain.cから移動

main.c

```
int iioLoadFile(IMAGE *pImage, const char *fname)
{ ....
}

int iioSaveFile(IMAGE *pImage, const char *fname)
{ ....
}

void iioMallocImageBuffer(IMAGE *pImage)
{ ....
}

void iioFreeImageBuffer(IMAGE *pImage)
{ ....
}
```

img_io.c

```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "img_io.h"

#define LINEMAX 100
```

ここに移動

【作業課題8/11】モジュール化

- img_proc.hの作成
  - main.cから以下の関数プロトタイプ宣言を移動させる
  - インクルードガード忘れずに

main.c

```
void ipCopy(IMAGE* p_in_image, IMAGE* p_out_image);
```

img_proc.h

```
#ifndef 識別子
#define 識別子

ここに移動

#endif
```

【作業課題9/11】モジュール化

- img_proc.cの作成
  - 適切なヘッダファイルをincludeする

img_proc.c

```
#include "img_io.h"
#include "img_proc.h"
```

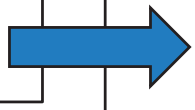
先に作成したimg_io.hとimg_proc.hをinclude
【注意】
img_proc.cではiioMallocImageBuffer関数を呼び出す必要があるため，img_io.hも必要になる．

【作業課題10/11】モジュール化

- img_proc.cの作成
  - 以下の関数定義をmain.cから移動

main.c

```
void ipCopy(IMAGE* p_in_image, IMAGE* p_out_image)
{
   ....
}
```



img_proc.c

```
#include "img_io.h"
#include "img_proc.h"
```

ここに移動

【作業課題11/11】モジュール化

- main.cの修正
  - 以下の2つのヘッダファイルを追加でincludeする

main.c

```
....

#include "img_io.h"
#include "img_proc.h"

int main(int argc, char* argv[])
{
   ...
```

main.cではiioFreeImageBuffer関数やipCopy関数を呼び出す必要があるため，どちらのヘッダも必要になる．

モジュール化の確認

- これでモジュール化が完了したので，実行する
  - これまでと同じ結果が得られるか確認する
  - また，main.cがいかにコンパクトになったか確認する

main.c
画像ファイルをロードする関数を呼び出す
画像処理関数を呼び出す
画像ファイルをセーブする関数を呼び出す
バッファを解放する

img_io.c
画像ファイルのロード関数の定義
画像ファイルのセーブ関数の定義
画素値用バッファを確保する関数の定義
画素値用バッファを解放する関数の定義

img_proc.c
画像処理のための関数の定義

作業課題の目安: 10分