２回目
画像ファイルの読み書き

pgm形式

- 本講義では，pgm（Portable gray map）画像を主に利用する

- Raw形式に近いデータ
- UNIXにおける標準画像フォーマット

前回の復習

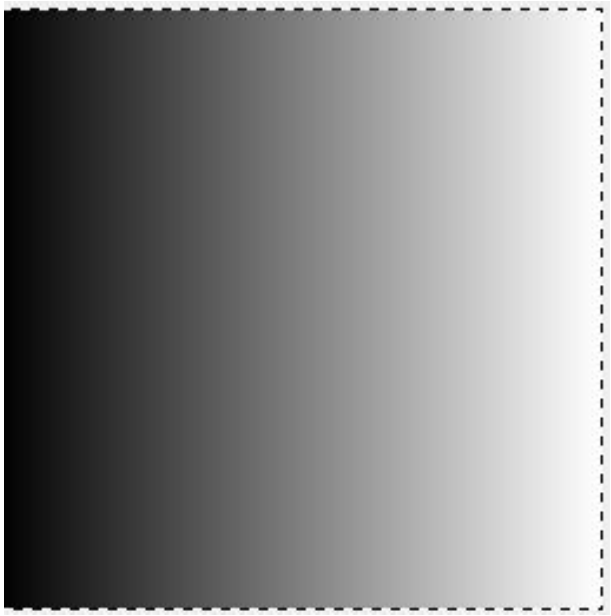
- バイナリファイルに

```
for(int i=0;i<256;i++){
	for(int j=0;j<256;j++){
		fputc((unsigned char)j,f p);
	}
}
```

とファイル出力するとraw形式の画像ファイルができた

画像データの確認

- GIMPでファイルを開く
  - 画像の種類をインデックスとする
  - 幅,高さをどちらも256とする



Raw形式画像が作成できた

Raw形式画像の問題点

- 画像を開く際に、画像の種類，階調や大きさを指定していた

- 一般的な画像は、そのような指定は必要ない

画像データにサイズや階調の情報が埋め込まれている

pgmフォーマット

P5 ← pgm形式

#Image ← #コメント・・・読み飛ばされる

256 256 ← サイズ

255 ← 階調

(0,0)の階調値 (1,0)の階調値・・・(255,0)の階調値 (0,1)の階調値 (1,1)の階調値・・・(255,1)の階調値・・・ (0,255)の階調値 (1,255)の階調値・・・(255,255)の階調値

課題

```
//pgm画像形式ファイルの書き出し。
//コピペはしないで自分で入力すること
if((fp=fopen("out.pgm","wb")) == NULL){
        printf("FILE OPEN FAILED");
        return(0);
}
fprintf(fp,"P5¥n");
fprintf(fp,"#Image¥n");
fprintf(fp,"256  256¥n");
fprintf(fp,"255¥n");
for(int i=0;i<256;i++){
        for(int j=0;j<256;j++){
                fputc((unsigned char)j,fp);
        }
}
fclose(fp);
```

Windowsにおける改行コードは　¥n

入力し実行したら、out.pgmをGIMPによって開く。

参考：Bmpフォーマット

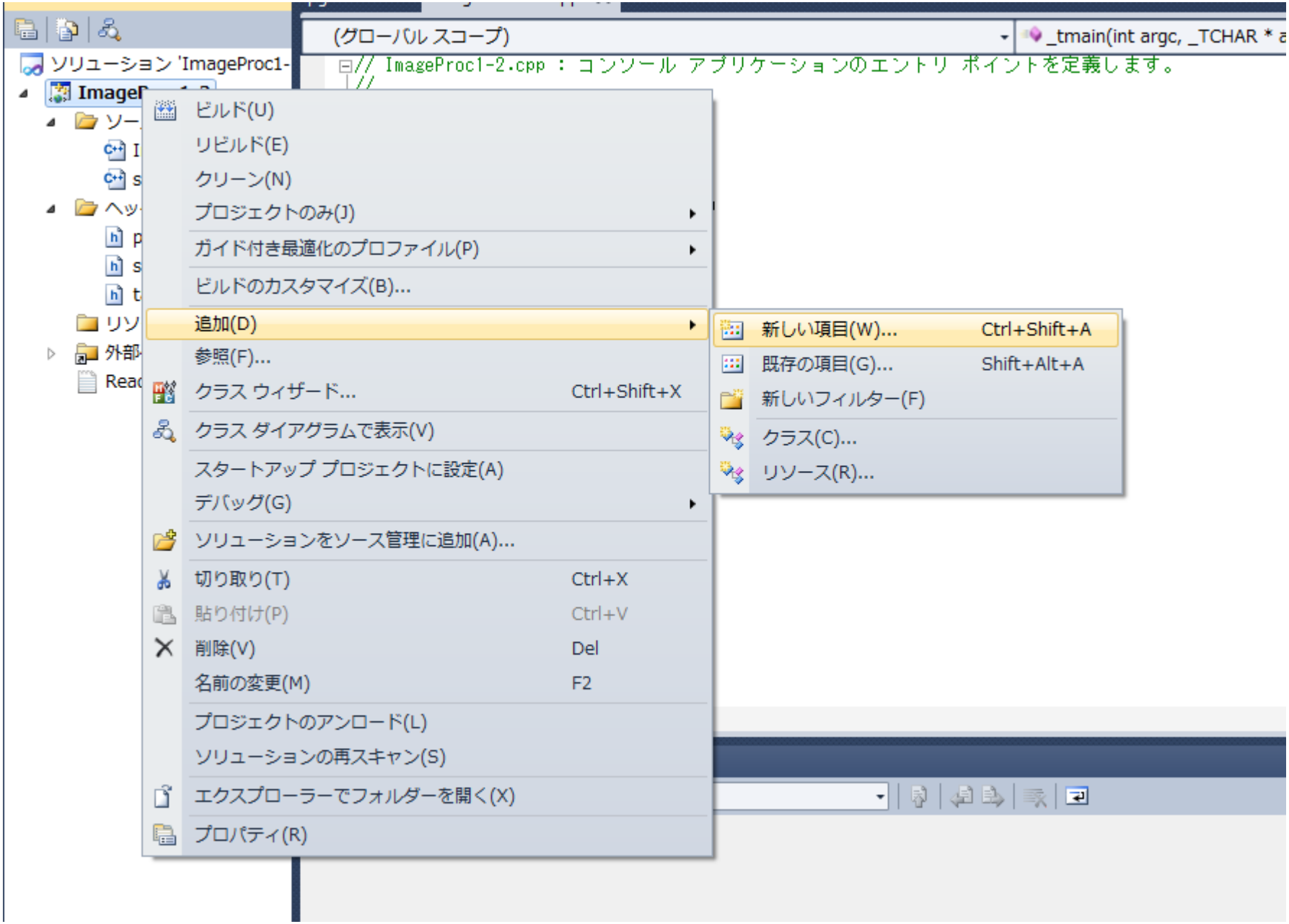
BMP ファイル体構造

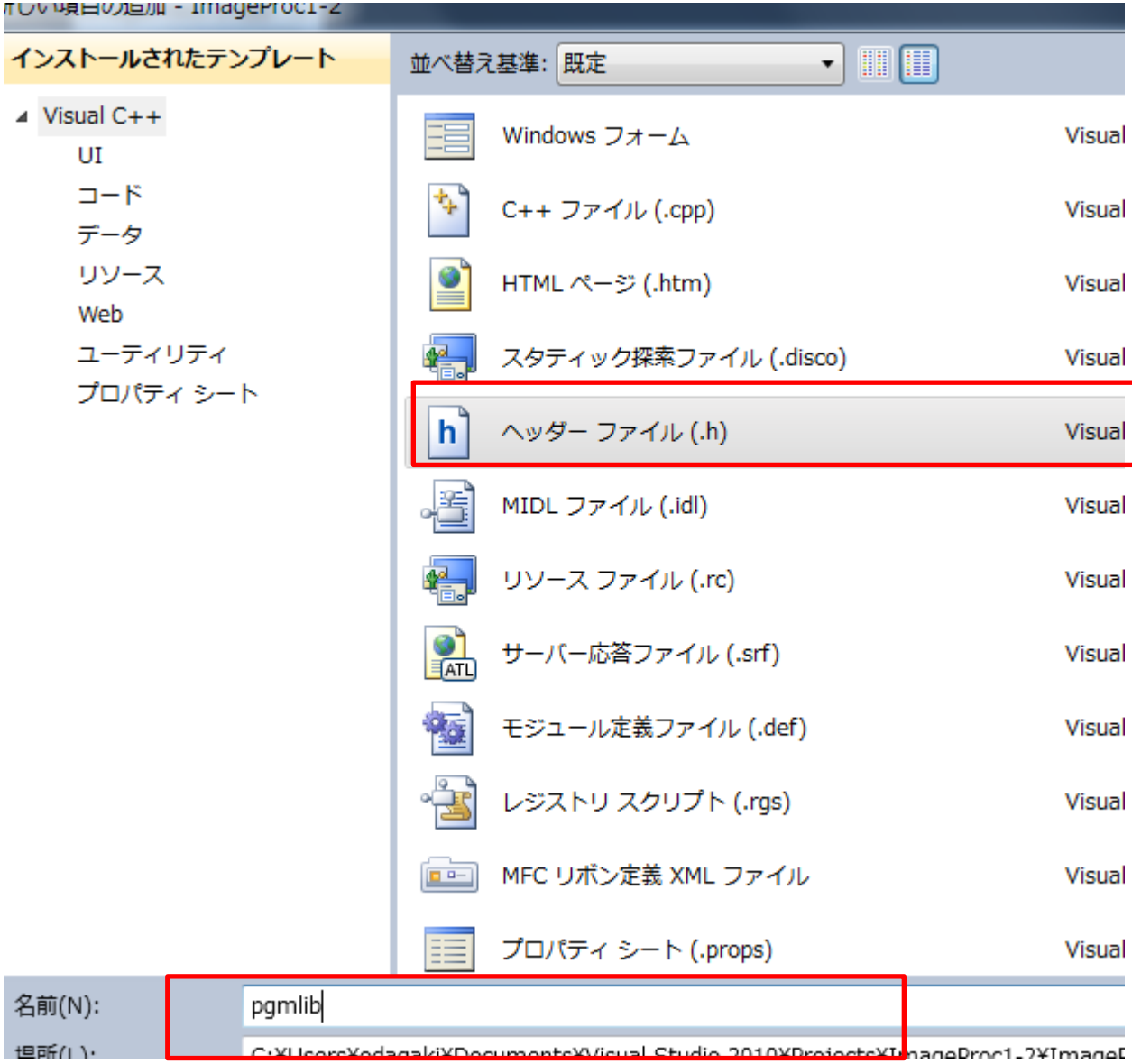
	Windows Bitmap
ファイルヘッダ	BITMAPFILEHEADER (14bytes)
情報ヘッダ	BITMAPINFOHEADER (40bytes)＋カラーパレット(無い場合もある)
画像データ	可変長

画像ファイルの入出力ライブラリの作成

- 毎回、上述のファイル出力関数を実装していると手間を要する

- ファイル入出力や画像コピー等の関数をライブラリ化して、作業の簡略化を図る


(グローバル スコープ)
_tmain(int argc, _TCHAR * a
// ImageProc1-2.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
ソリューション 'ImageProc1-
ビルド(U)
リビルド(E)
クリーン(N)
プロジェクトのみ(J)
ガイド付き最適化のプロファイル(P)
ビルドのカスタマイズ(B)...
追加(D)
参照(F)...
クラス ウィザード... Ctrl+Shift+X
クラス ダイアグラムで表示(V)
スタートアップ プロジェクトに設定(A)
デバッグ(G)
ソリューションをソース管理に追加(A)...
切り取り(T) Ctrl+X
貼り付け(P) Ctrl+V
削除(V) Del
名前の変更(M) F2
プロジェクトのアンロード(L)
ソリューションの再スキャン(S)
エクスプローラーでフォルダーを開く(X)
プロパティ(R)
新しい項目(W)... Ctrl+Shift+A
既存の項目(G)... Shift+Alt+A
新しいフィルター(F)
クラス(C)...
リソース(R)...


インストールされたテンプレート
並べ替え基準: 既定
Visual C++
UI
コード
データ
リソース
Web
ユーティリティ
プロパティ シート
Windows フォーム
C++ ファイル (.cpp)
HTML ページ (.htm)
スタティック探索ファイル (.disco)
ヘッダー ファイル (.h)
MIDL ファイル (.idl)
リソース ファイル (.rc)
サーバー応答ファイル (.srf)
モジュール定義ファイル (.def)
レジストリ スクリプト (.rgs)
MFC リボン定義 XML ファイル
プロパティ シート (.props)
Visual
名前(N):
pgmlib

void load_image(int n, char name[]);

複数の画像ファイルを読み込み、配列に格納する関数

- int n : 読み込む画像ファイルの番号
- char name[] : 読み込む画像ファイル名

定数宣言

```
/* 定数宣言 */
#define MAX_IMAGESIZE      1280 /* 想定する縦・横の最大画素数 */
#define MAX_BRIGHTNESS      255 /* 想定する最大階調値 */
#define GRAYLEVEL          256 /* 想定する階調数(=最大階調値+1) */
#define MAX_FILENAME       256 /* 想定するファイル名の最大長 */
#define MAX_BUFFERSIZE     256 /* 利用するバッファ最大長 */
#define MAX_NUM_OF_IMAGES    5 /* 利用する画像の枚数 */
```

大域変数の宣言

```
/* 大域変数の宣言 */
/* 画像データ image[n][x][y] */
unsigned char
image[MAX_NUM_OF_IMAGES][MAX_IMAGESIZE][MAX_IMAGESIZE];
/* image[n] の横幅・縦幅 */
int width[MAX_NUM_OF_IMAGES], height[MAX_NUM_OF_IMAGES];
```

- 大域変数・・・関数の外側で変数を宣言することによって、異なる関数間で共用する変数として扱うことができる。
- 多用しすぎるとバグの要因になるので、極力使わないほうが望ましい。

関数の上部

```
/* 階調画像を入力する関数 */
void load_image( int n, char name[] )
/* n:画像番号，name[]:ファイル名（""のときはキーボード入力）*/
/* 横幅，縦幅はそれぞれ width[n], height[n] に代入する */
{
   char file_name[MAX_FILENAME]; /* ファイル名用の文字配列 */
   char buffer[MAX_BUFFERSIZE];  /* データ読み込み用作業変数 */
   FILE *fp; /* ファイルポインタ */
   int max_gray; /* 最大階調値 */
   int x, y; /* ループ変数 */
```

ファイルオープン

```
/* 入力ファイルのオープン */
  if ( name[0] == '¥0' ){
     printf("入力ファイル名 (*.pgm) : ");
     scanf("%s",file_name);
  } else strcpy( file_name, name );
  if ( (fp = fopen( file_name, "rb" ))==NULL ){
     printf("その名前のファイルは存在しません. ¥n");
     exit(1);
  }
```

- 引数name[0]が空（ヌル文字）だったら、読み込むファイル名をキーボードより入力させる
- 引数name[0]が空（ヌル文字）でない場合は、文字列nameを変数 file_nameにコピーする

P5文字列の検索

```
/* ファイルタイプ(=P5)の確認 */
  fgets( buffer, MAX_BUFFERSIZE, fp );
  if ( buffer[0] != 'P' || buffer[1] != '5' ){
     printf("ファイルのフォーマットが P5 とは異なります．¥n");
     exit(1);
  }
```

- fgets(buffer, MAX_BUFFERSIZE, fp);によってファイルから改行コードまでを読み込み、配列変数bufferに代入する
- Bufferのゼロ番目の要素が’P’でない、または一番目の要素が’5’でない場合はpgmファイルでないと判断してプログラムを終了させる

画像サイズの読み込み

```
/* width[n], height[n] の代入（#から始まるコメントは読み飛ばす）*/
  width[n] = 0;    height[n] = 0;
  while ( width[n] == 0 || height[n] == 0 ){
     fgets( buffer, MAX_BUFFERSIZE, fp );
     if ( buffer[0] != '#' )
        sscanf( buffer, "%d %d", &width[n], &height[n] );
  }
```

- n番目の画像ファイルのサイズを取得する
- fgets(buffer, MAX_BUFFERSIZE, fp);によって1行ずつデータを読み込み、もし、コメント(‘#’)で始まっていない行であれば、sscanf(buffer, “%d %d”, &width[n], &height[n]);により、画像の幅と高さを取得する

画像の最大階調の取得

```
/* max_gray の代入（#から始まるコメントは読み飛ばす）*/
  max_gray = 0;
  while ( max_gray == 0 ){
     fgets( buffer, MAX_BUFFERSIZE, fp );
     if ( buffer[0] != '#' )
        sscanf( buffer, "%d", &max_gray );
  }
```

- sscanf(buffer, “%d”, &max_gray);によって最大階調値を読み込む

画像情報の表示

```
/* パラメータの画面への表示 */
  printf("横の画素数 = %d, 縦の画素数 = %d¥n", width[n], height[n] );
  printf("最大階調値 = %d¥n",max_gray);
  if ( width[n] > MAX_IMAGESIZE || height[n] > MAX_IMAGESIZE ){
     printf("想定値 %d x %d を超えています. ¥n",
        MAX_IMAGESIZE, MAX_IMAGESIZE);
     printf("もう少し小さな画像を使って下さい. ¥n");
     exit(1);
  }
```

最大階調値

```
if ( max_gray != MAX_BRIGHTNESS ){
    printf("最大階調値が不適切です. ¥n");  exit(1);
  }
```

画素の読み込み

```
/* 画像データを読み込んで画像用配列に代入する */
  for(y=0;y<height[n];y++)
     for(x=0;x<width[n];x++)
        image[n][x][y] = (unsigned char)fgetc( fp );
  fclose(fp);
  printf("画像は正常に読み込まれました. ¥n");
```

- image[n][x][y] = (unsigned char)fgetc(fp);によって画像から1画素ずつデータを読み込み配列に格納してゆく

利用方法

- 呼び出したいプログラムの上部に

  #include "pgmlib.h“

  と記述し、ライブラリをインクルードする

- 以後の回では，プロジェクトを作成するたびに、このヘッダーファイルをコピーして、ソースファイルと同じフォルダに張り付ける

利用方法

- main関数（例１）

```
load_image(0,"");
copy_image(0,1);
save_image(1,"");
```

- main関数（例２）

```
load_image(0,"");
copy_image(0,1);
save_image(1,"out.pgm");
```

各画像の画素値は，三次元配列image[n][x][y]に格納されている

メモ

- Unsafeの対応策

- ファイルメニューから『プロジェクト → プロパティ』を選択し、左ツリーから順に、『構成プロパティ → C/C++ → プリプロセッサ』を選択、右ペインの『プリプロセッサの定義』欄
  に **;_CRT_SECURE_NO_DEPRECATE** を追加する