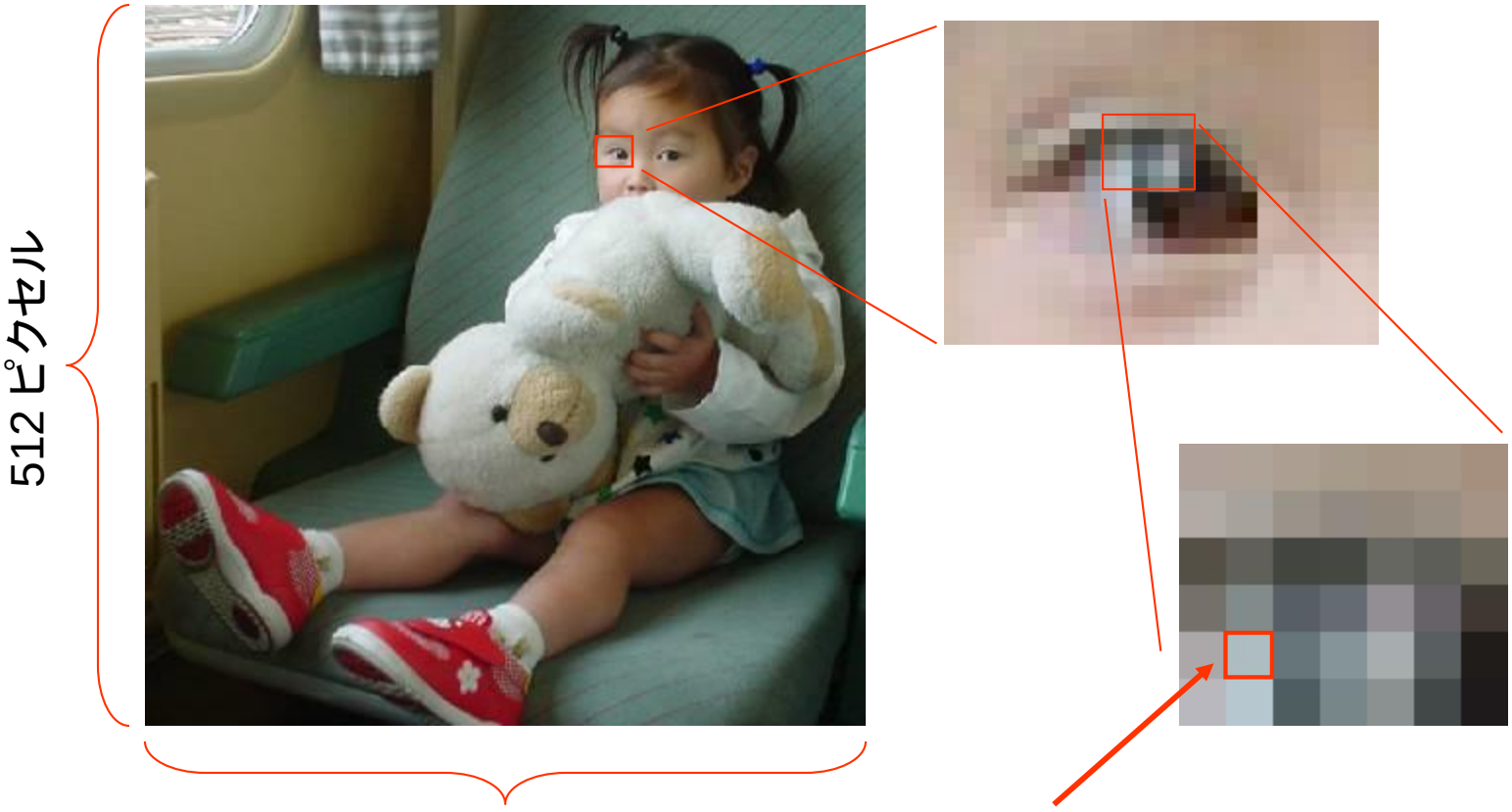
デジタル画像とは

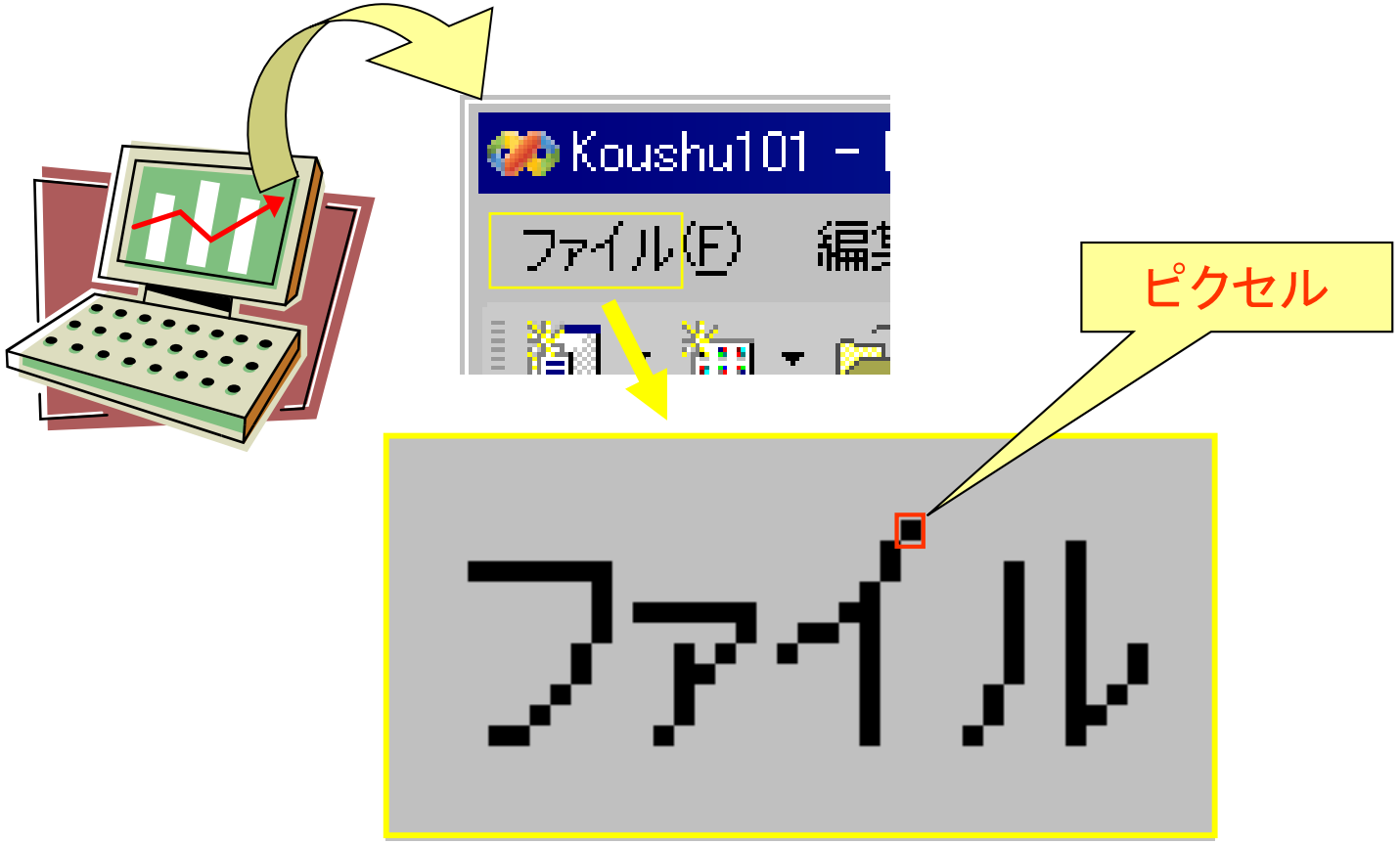
復習



ピクセル
(画素，絵素，pixel)

コンピュータの画面 ＝ デジタル画像

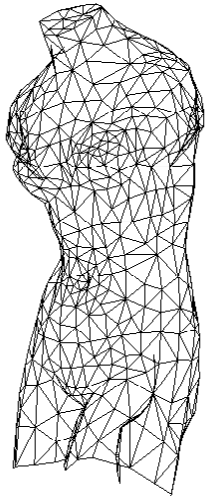
復習



かなり古いパソコンやノートパソコン	横1024ピクセル x 縦768ピクセル	XGAサイズ
古いパソコンやノートパソコン	横1280ピクセル x 縦1024ピクセル	SXGAサイズ
最近のパソコン	横1920ピクセル x 縦1200ピクセル	WUXGAサイズ

3次元CGのおおまかな流れ

復習



モデリング

物体形状データ

3次元座標で表されるデータ

レンダリング

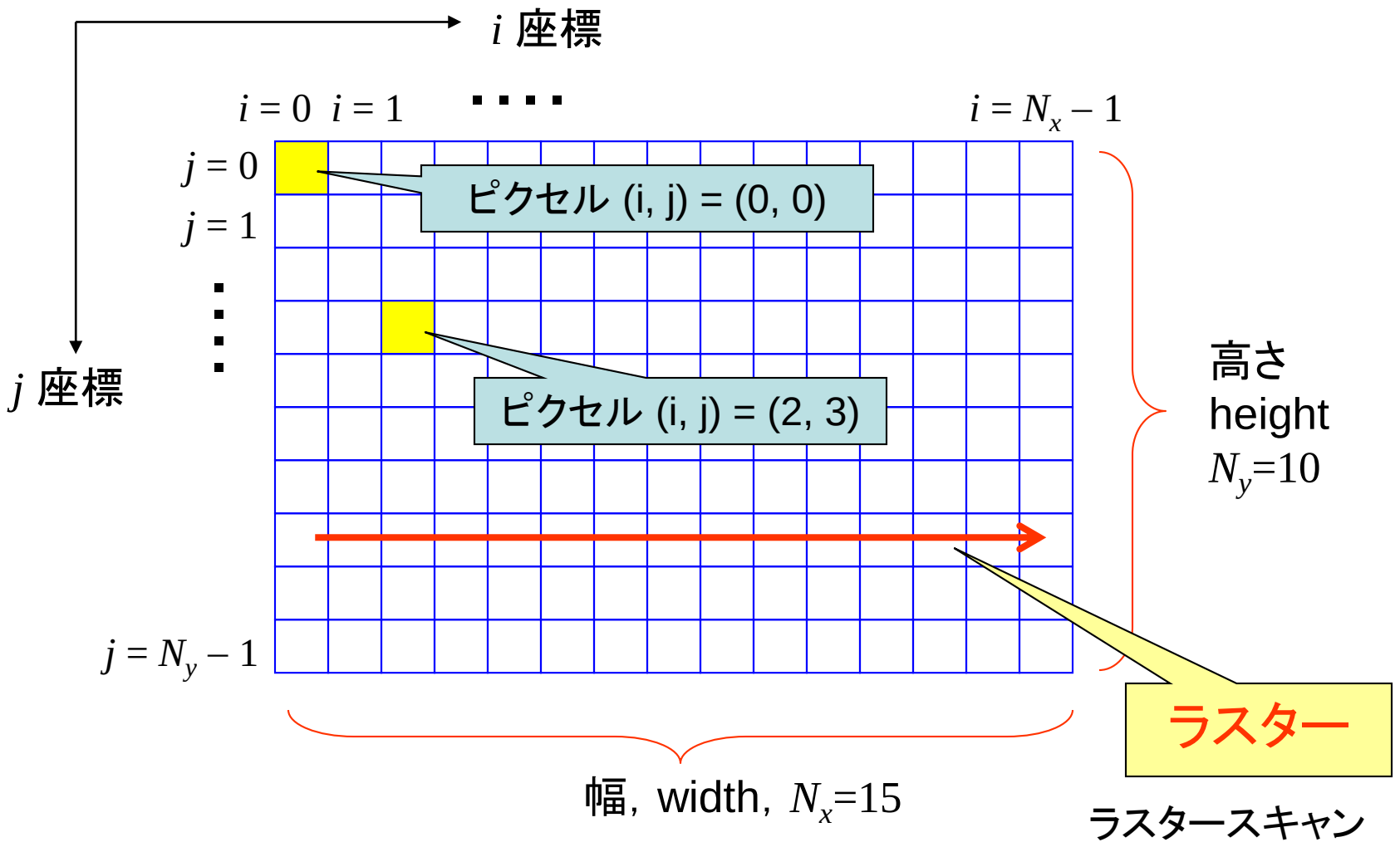
デジタル画像
(ラスター画像)

2次元座標で表されるデータ

出来上がるのはデジタル画像！

復習

ラスター画像


i 座標
i = 0 i = 1
i = N_x − 1
j = 0
j = 1
j = N_y − 1
j 座標
ピクセル (i, j) = (0, 0)
ピクセル (i, j) = (2, 3)
高さ
height
N_y=10
ラスター
幅, width, N_x=15
ラスタースキャン

実行結果としくみ

復習

配列
image[i][j]

i=0 i=1 ・・・・

j=0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
j=1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

実行結果

```
.......0.......
.......0.......
.......0.......
.......0.......
.......0.......
000000000000000
.......0.......
.......0.......
.......0.......
.......0.......
続行するには何かキーを押してください . . .
```

注：基礎プログラミングではi方向が縦方向でj方向が横方向であったが，ここではその関係を逆転している．(この逆転はDrawCharImage()関数が行っている)

ちょっとお遊び･･･文字で画像表示

復習

```
#include    <stdio.h>

// 幅xsizeピクセル 高さysizeピクセルの画像データimage[][]を文字で描く関数
void DrawCharImage(unsigned char* image, int xsize, int ysize);

#define WIDTH   15      // 画像の幅(ピクセル数)
#define HEIGHT  10      // 画像の高さ(ピクセル数)
int main(void)
{
    unsigned char image[WIDTH][HEIGHT]; // 画像データ配列の宣言
    int i, j;                           // ループ変数

    for (i = 0; i < WIDTH; i++)         // 2重ループで画像を一旦ゼロにクリアする
    {
        for (j = 0; j < HEIGHT; j++)
            image[i][j] = 0;
    }

    for (i = 0; i < WIDTH; i++)         // 横線を引く
    {
        image[i][HEIGHT/2] = 1;         // 横線はちょうど真ん中で引く
    }

    for (j = 0; j < HEIGHT; j++)        // 縦線を引く
    {
        image[WIDTH/2][j] = 1;          // 縦線もちょうど真ん中で引く
    }

    DrawCharImage((unsigned char*) image, WIDTH, HEIGHT); //画像描画関数呼び出し
}
```

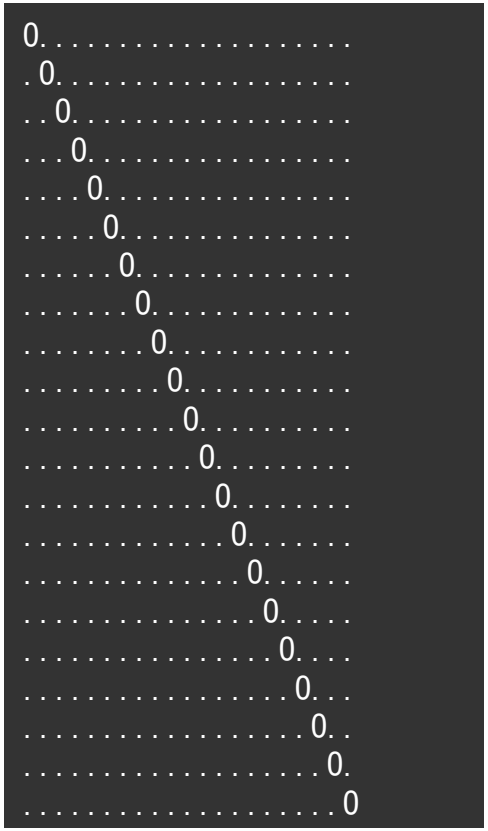
考え方
(1) すべてを一旦0にクリア
(2) 必要なピクセルだけを1をセット

今日の課題

復習

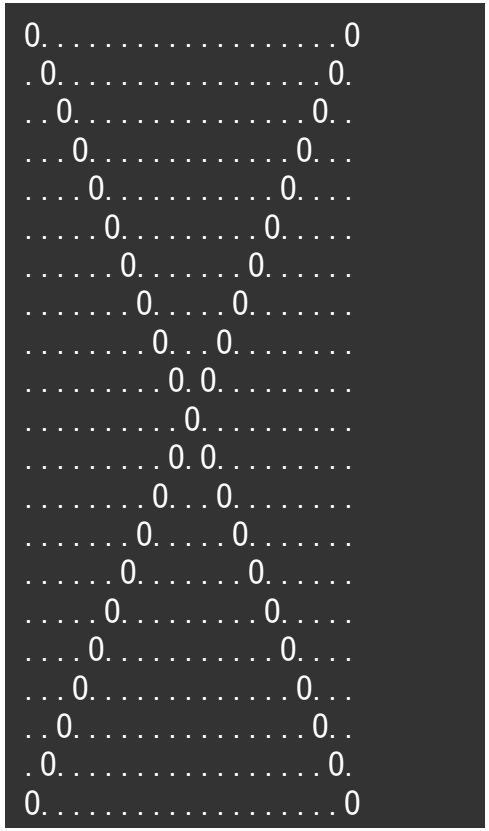
DrawCharImage()関数のソースをそのまま用いて，次のような文字による画像を出力するmain()プログラムを作成せよ．

基本課題1　必ず提出



21 x 21 ピクセル

発展課題1　できれば提出



21 x 21 ピクセル

提出方法

- 基礎プロと同じく，Wordにソースと実行結果(画像)を貼り付ける．
- 関大LMSでWordのファイルをアップロードする．

基本課題1 正解例

正解例1

```
for (i = 0; i < WIDTH; i++)          // ゼロにクリアする
{
    for (j = 0; j < HEIGHT; j++)
        image[i][j] = 0;
}

for (i = 0; i < WIDTH; i++)  // 2重ループで縦横の座標値が等しければ1にする.
{
    for (j = 0; j < HEIGHT; j++)
        if (i == j)
            image[i][j] = 1;
}
```

ループ回数 : 21 × 21 = 441 回

ループ回数 : 21 × 21 = 441 回

合計
441+441=882

正解例2

```
for (i = 0; i < WIDTH; i++)          // ゼロにクリアする
{
    for (j = 0; j < HEIGHT; j++)
        image[i][j] = 0;
}

for (i = 0; i < WIDTH; i++)
{
    image[i][i] = 1;      // 斜め線の縦横の座標値は同じ
}
```

ループ回数 : 21 × 21 = 441 回

ループ回数 : 21 回

合計
441+21=462

配列
image[i][j]

	i=0	1	2	3	・・・										
j=0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
j=1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
j=2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
j=3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

基本課題1 さらに良い解答

正解例1

ループ回数 : 21 × 21 = 441 回

```
for (i = 0; i < WIDTH; i++)          // ゼロにクリアする
{
    for (j = 0; j < HEIGHT; j++)
        image[i][j] = 0;
}

for (i = 0; i < WIDTH; i++)  // 2重ループで縦横の座標値が等しければ1にする.
{
    for (j = 0; j < HEIGHT; j++)
        if (i == j)
            image[i][j] = 1;
}
```

ループ回数 : 21 × 21 = 441 回

合計
441+441=882

正解例3

ループ回数 : 21 × 21 = 441 回

```
for (i = 0; i < WIDTH; i++)
{
    for (j = 0; j < HEIGHT; j++)
        if (i == j)
            image[i][j] = 1;    //座標値が同じなら1
        else
            image[i][j] = 0;    //座標値が異なるなら0
}
```

合計
441

基本課題1 良くない例(≒不正解)

```
for (i = 0; i < WIDTH; i++) // ゼロにクリアする
{
    for (j = 0; j < HEIGHT; j++)
        image[i][j] = 0;
}

for (i = 0; i < WIDTH; i++)
{
    image[i][i] = 1;
}

for (j = 0; j < HEIGHT; j++)
{
    image[j][j] = 1;
}
```

```
for (i = 0; i < WIDTH; i++) // ゼロにクリアする
{
    for (j = 0; j < HEIGHT; j++)
        image[i][j] = 0;
}

for (i = 0; i < WIDTH; i++)
{
    for (j = 0; j < HEIGHT; j++)
        image[i][i] = 1;
}
```

```
for (i = 0; i < WIDTH; i++) // ゼロにクリアする
{
    for (j = 0; j < HEIGHT; j++)
        image[i][j] = 0;
}

j = 0;
for (i = 0; i < WIDTH; i++)
{
    image[i][j] = 1;
    j++;
}
```

危険な不正アクセス！

```
for (i = 0; i < WIDTH; i++)
{
    for (j = 0; j < HEIGHT; j++)
        image[i][j] = 0;
}

for (i = 0; i < WIDTH; i++)
{
    image[i][i] = 1;
}

for (j = 0; j < HEIGHT; j++)
{
    image[WIDTH][j] = 1;
}
```

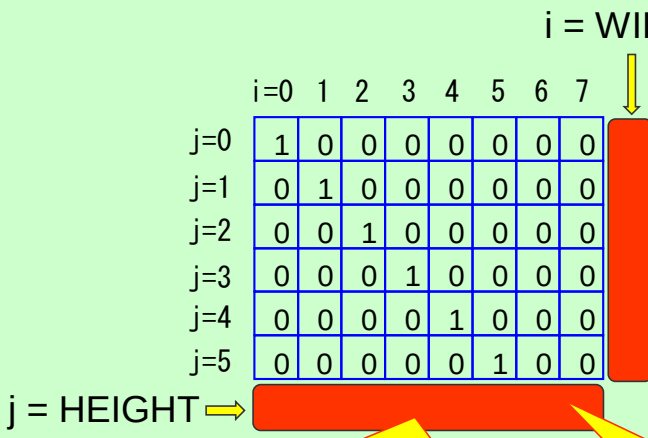
```
for (i = 0; i < WIDTH; i++) // ゼロにクリアする
{
    for (j = 0; j < HEIGHT; j++)
        image[i][j] = 0;
}

for (i = -1; i < WIDTH; i++)
{
    image[i][j] = 1;
    j++;
}
```

j=0に戻していない
ループ開始時 j=21

i=-1からループスタート??!

実行結果が正解と一致するのは単なる偶然！

```
#define WIDTH   8  // 画像の幅(ピクセル数)
#define HEIGHT  6  // 画像の高さ(ピクセル数)
配列 image[WIDTH][HEIGHT]
```



存在しない配列要素！ → 不正アクセス

運が良い時 ⇒ 何も問題は起きない
(ただし正解と一致したのは偶然)

問題が起きるときの症状
- ✓ 途中でプログラムが停止しているが，一見正常終了
- ✓ 不正アクセス等のエラーメッセージで終了
- ✓ 突然，コンソールウィンドウ(黒ウィンドウ)が閉じる

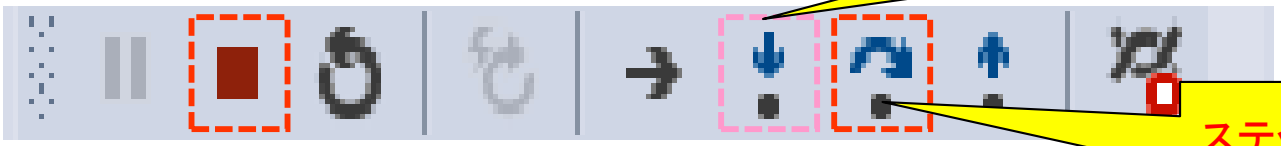
今後はデバッガを利用せずに課題に解答するのは困難！

デバッガの利用

ツールバーでデバッグ
(1)　ツールボタンで右クリックして，デバッグを選ぶ
(2)　デバッグツールバーからボタンを選ぶ

特定の行からデバッグを始める
ソースプログラムの特定の行をポイントしておいて，
右ボタンで「カーソル行の前まで実行」メニューを用いる

(a)ステップオーバー	次の行を実行
(b) ステップイン	次の行を実行
(c) デバッグの中止	デバッグをやめる



(c)　(b)　(a)

ステップインは関数内部に入って1行づつ実行する

ステップオーバーは関数内部に入らずに1行づつ実行する

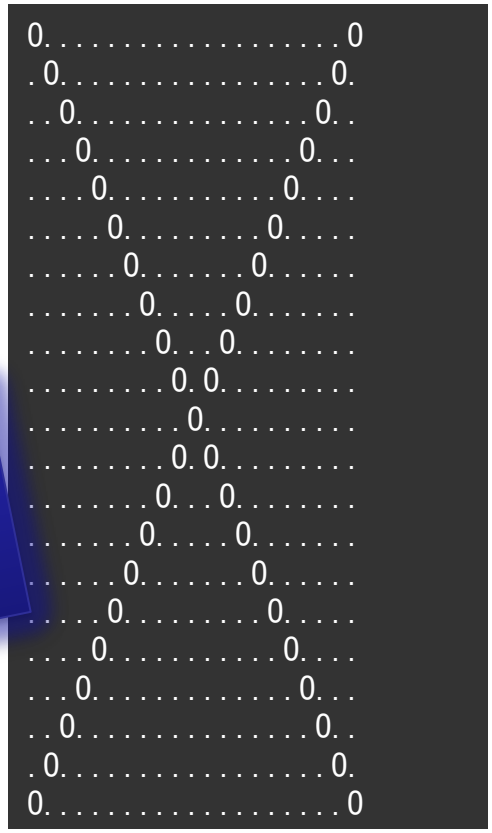
発展課題1の解答

考え方1
(1) 2重ループしてすべてを一旦0にクリア
(2) 必要なピクセルだけに1をセット

```
//・・・クリアの2重ループ(省略)

for (i = 0; i < WIDTH; i++)
{
    image[i][i] = 1;
}

for (i = 0; i < WIDTH; i++)
{
    image[WIDTH - 1 - i][i] = 1;
}
```

ループ回数
21+21 = 42

プログラムの効率が重要



```
//・・・クリアの2重ループ(省略)

for(i = 0; i < WIDTH; i++)
{
    for(j = 0; j < HEIGHT; j++)
    {
        image[i][i] = 1;
        image[HEIGHT-1-i][i] = 1;
    }
}
```

ループ回数
21×21 = 441

ムダ！

```
//・・・クリアの2重ループ(省略)

for (i = 0; i < WIDTH; i++)
{
    image[i][i] = 1;
    image[WIDTH - 1 - i][i] = 1;
}
```

良くできました!

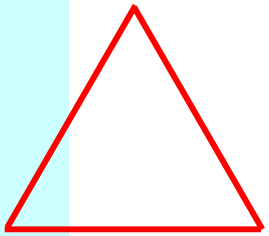
発展課題1の解答2

復習

考え方2
(1) 2重ループして条件を満たすピクセルだけに1をセット

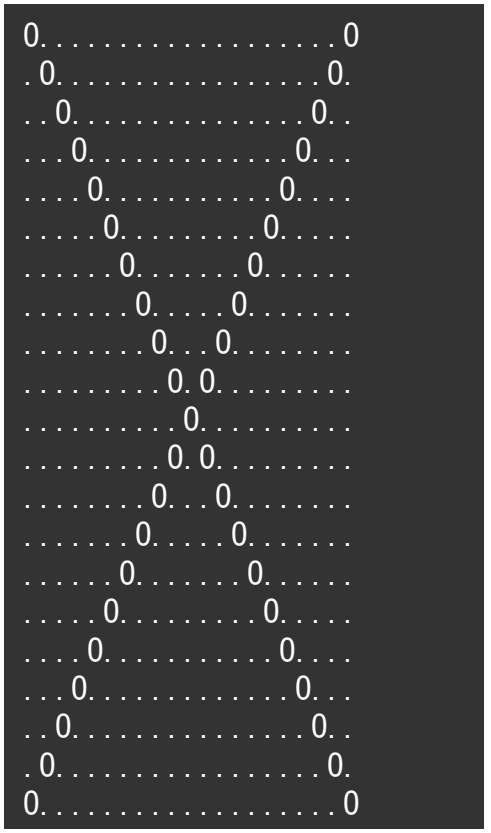
```
for(i=0;i<WIDTH; i++)
{
    for(j=0;j<HEIGHT;j++)
        image[i][j] = 0;
}

for(i=0;i<WIDTH;i++)
{
    for(j=0;j<HEIGHT;j++)
    {
        if(i==j || i+j==(WIDTH-1))
            image[i][j] = 1;
    }
}
```

ムダ！



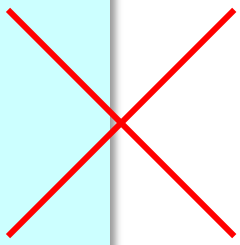
```
for(i=0;i<WIDTH;i++)
{
    for(j=0;j<HEIGHT;j++)
    {
        if(i==j || i+j==(WIDTH-1))
            image[i][j] = 1;
        else
            image[i][j] = 0;
    }
}
```

良くできました!

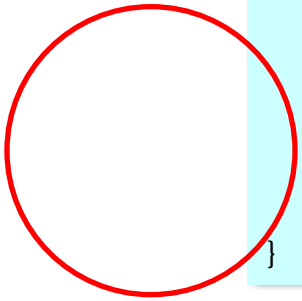


良いソースと悪いソース

```
int main(void)
{
    unsigned char image[WIDTH][HEIGHT]; // 画像データ配列の宣言
    int i, j; // ループ変数
    for (i = 0; i < WIDTH; i++)  // 2重ループで画像を一旦ゼロにクリアする
    {
        for (j = 0; j < HEIGHT; j++) image[i][j] = 0;
    }
    for ( i= 0; i < WIDTH; i++)  // 横線を引く
    {
        image[i][i] = 1;          // 横線はちょうど真ん中で引く
    }
    for ( i= 0; i <HEIGHT ; i++) // 横線を引く
    {
        image[i][HEIGHT-i-1] = 1;// 横線はちょうど真ん中で引く
    }
    DrawCharImage((unsigned char*) image, WIDTH, HEIGHT); //画像描画関数呼び出し
}
```



誤ったコメントが付いている場合，今後は0点になる！



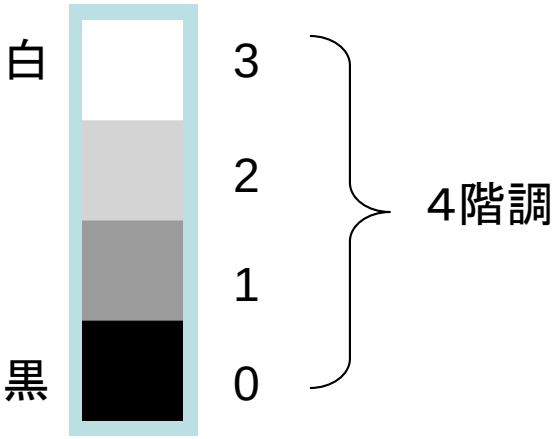
```
int main(void)
{
    unsigned char image[WIDTH][HEIGHT]; // 画像データ配列の宣言
    int i, j;                    // ループ変数
    for (i = 0; i < WIDTH; i++) // 2重ループで画像を一旦ゼロにクリアする
    {
        for (j = 0; j < HEIGHT; j++) image[i][j] = 0;
    }
    for (i = 0; i < WIDTH; i++) // 行数分ループ
    {
        image[i][i] = 1;         // 左上から右下に線を引く
        image[i][WIDTH-i-1] = 1;// 右上から左下に線を引く
    }
    DrawCharImage((unsigned char*) image, WIDTH, HEIGHT); //画像描画関数呼び出し
}
```

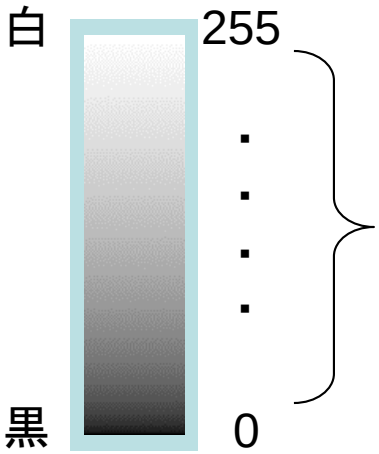
グレースケール画像

カラー画像

グレースケール画像

階調 ＝ 1つのピクセルの明るさの段階





0, 1, ・・・, 255
階調値, ピクセル値, グレーレベル

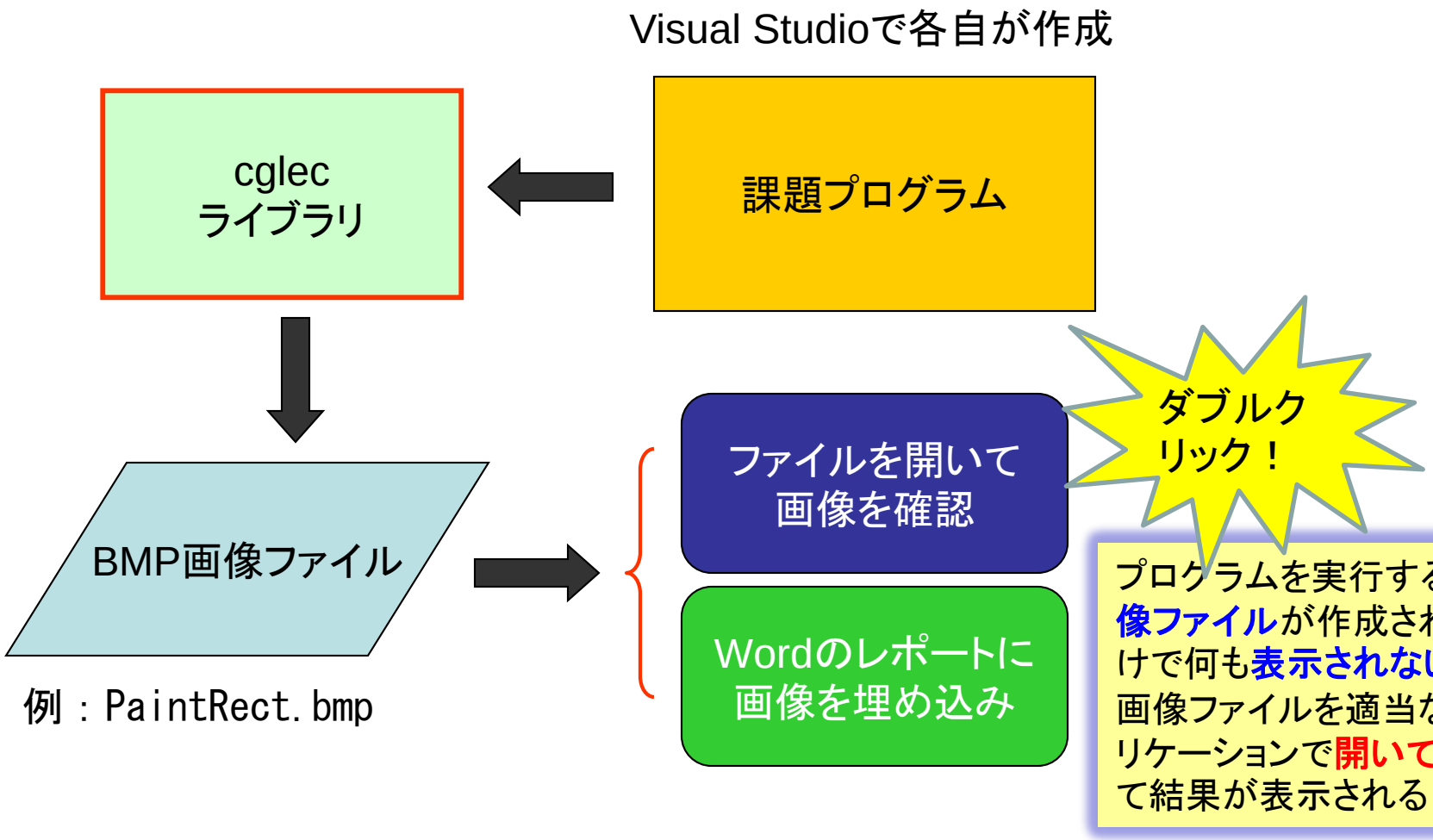
1ピクセルが1バイトで表現できる

$256=2^8$ → 8ビットグレースケール

「深さ8ビットのグレースケール」と表現する場合もある

グレイスケール画像を描くCGプログラミング

cglecライブラリ　：　この授業用に開発された2次元CG用ライブラリ



cglecの利用準備：構造体

ー　オブジェクト指向の第一歩　ー

いくつかの変数をまとめて管理する変数

例えば

複素数型

実部と虚部のfloat型の２変数からできている

日付型

年，月，日を表すint型の３変数からできている

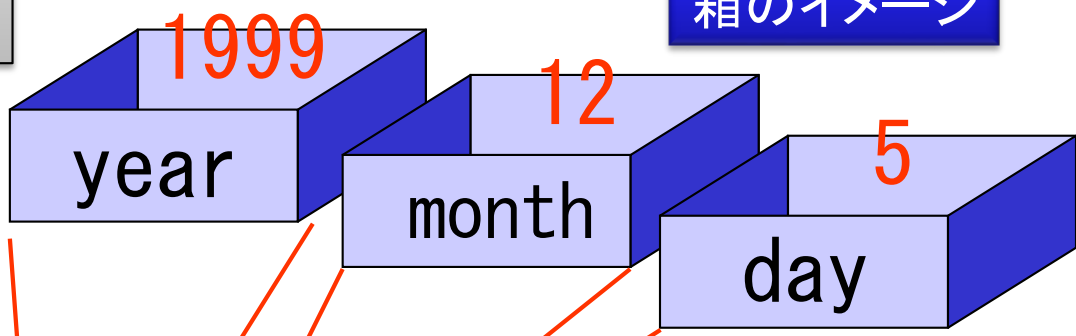
学生身体データ型

学籍番号(int型変数)
名前(文字列変数)
誕生日(日付型構造体)
身長(float型変数)
体重(float型変数)

例えば，課題が
「A君の誕生日をa，B君の誕生日をbとし，どちらが早く生まれたか比べなさい・・・」
日付aや日付bを表す変数があると便利
⇒実際には日付は複数の変数でできている

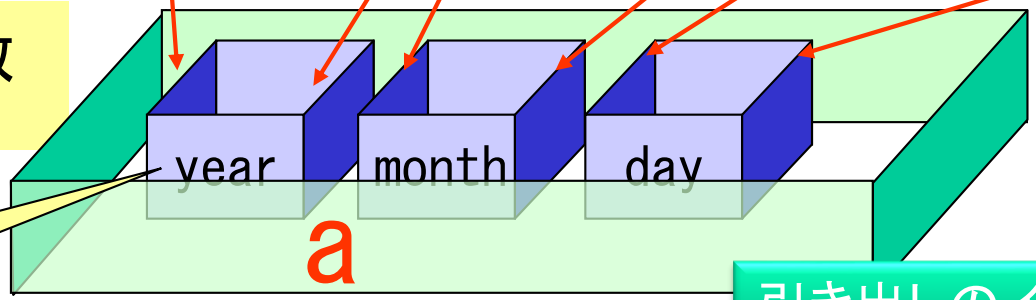
構造体変数

箱のイメージ



普通の変数
（ここでは年・月・日）

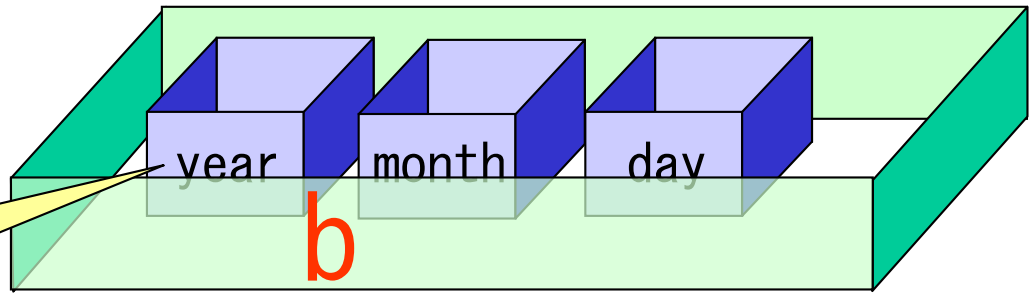
構造体変数
（ここでは日付）



引き出しのイメージ

構造体変数aの中のyear
プログラム中では a. year

ピリオド



構造体変数bの中のyear
プログラム中では b. year

ピリオド

同じ構造をした「引き出し」変数を構造体変数と呼ぶ

構造体のプログラム例(1) ー 日付型

キーワード
struct

構造体内の変数をメンバ変数と呼ぶ

Date型の構造体の定義 ➡ Dateは変数ではなく設計図

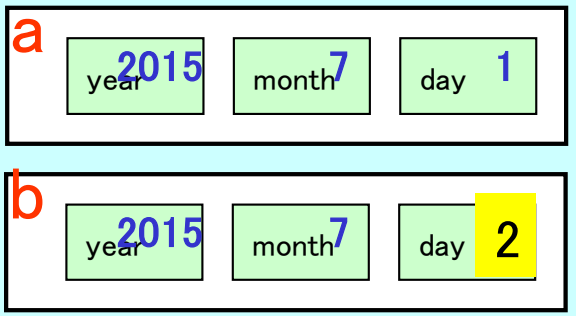
```
#include <stdio.h>

struct  Date
{
      int year;
      int month;
      int day;
};

void main()
{
     Date a, b;
     a.year = 2015;   a.month = 7;    a.day = 1;
     b = a;
     b.day = 2;
     printf("初めの日付は%d年%d月%d日, ", a.year, a.month, a.day);
     printf("次の日付は%d年%d月%d日¥n", b.year, b.month, b.day);
}
```

Date型の構造体変数aとbの宣言

a.year は構造体変数aのメンバ変数yearを意味する

構造体変数aをbに代入

b.day は構造体変数bのメンバ変数dayを意味する



初めの日付は2015年7月1日, 次の日付は2015年7月2日

構造体のプログラム例(2)

```
struct  Date
{
    int year;
    int month;
    int day;
};

void main()
{
    Date a = {1998, 6, 28};
    Date b;
    b = {1998, 6, 29};
    Date c[20];
    c[15].year = 1998;
    c[15].month = 6;
    c[15].day = 28;

    printf("今日は%d年%d月%d日¥n", c[15].year, c[15].month, c[15].day);
}
```

構造体変数を宣言すると同時に初期化

同時ではないのでダメ エラー

Date型構造体の配列の宣言

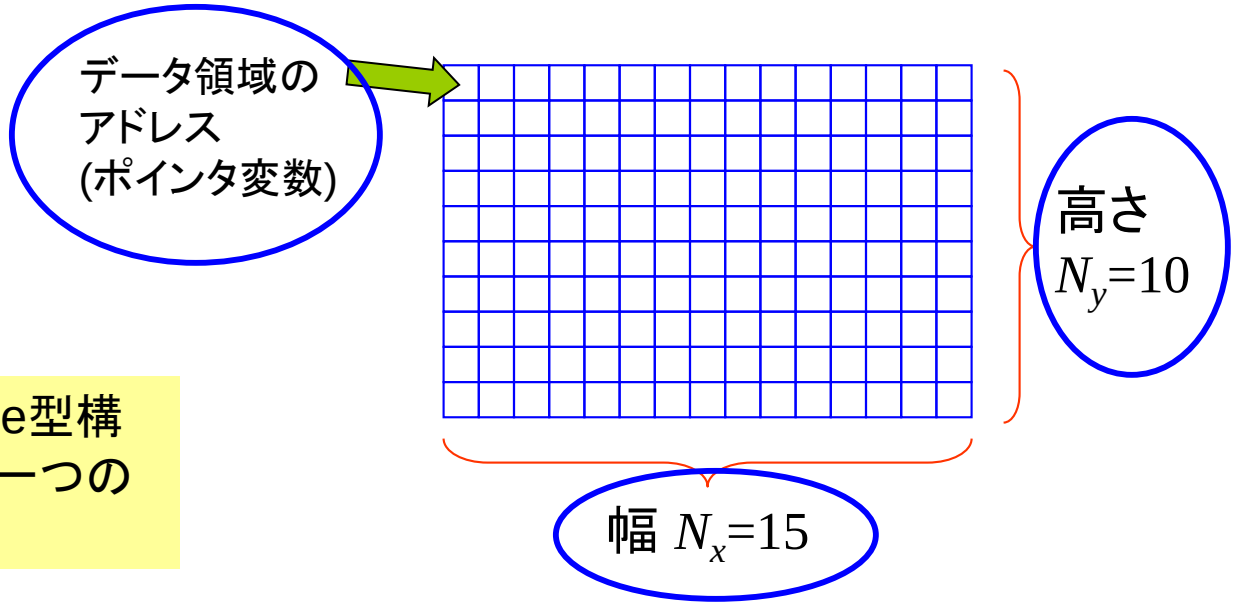
Date型構造体配列のインデックス15の要素に値を代入

※この例では配列の添え字15には特に意味はない

画像を管理する構造体の定義

```
struct Image
{
    unsigned char* Data; // データ領域へのポインタ
    int Nx;              // 横方向ピクセル数
    int Ny;              // 縦方向ピクセル数
};
```

この構造体はヘッダファイルcglec.hの中で定義されている



一つのImage型構造体変数が一つの画像を表す

複数の画像を用いる時に便利

cglecライブラリを使って 8ビットグレースケール画像を描く

Example2-1

```
#include    "cglec.h"
#define WIDTH   400
#define HEIGHT  400

int main(void)
{
    unsigned char data[WIDTH][HEIGHT];

    Image img = { (unsigned char*) data, WIDTH, HEIGHT };

    CglSetAll(img, 0);                  // imgをグレイレベル0でクリアする
    int x, y;

    for (y = 100; y < 150; y++)
    {
        for (x = 100; x < 200; x++)
            data[x][y] = 255;           // グレイレベル255(白)の四角形描画
    }

    for (y = 200; y < 250; y++)
    {
        for (x = 300; x < 350; x++)
            data[x][y] = 127;           // グレイレベル127(灰色)の四角形描画
    }

    CglSaveGrayBMP(img, "PaintRect.bmp");
}
```

cglecライブラリのヘッダファイル読み込み
(Image構造体の定義もここにある)

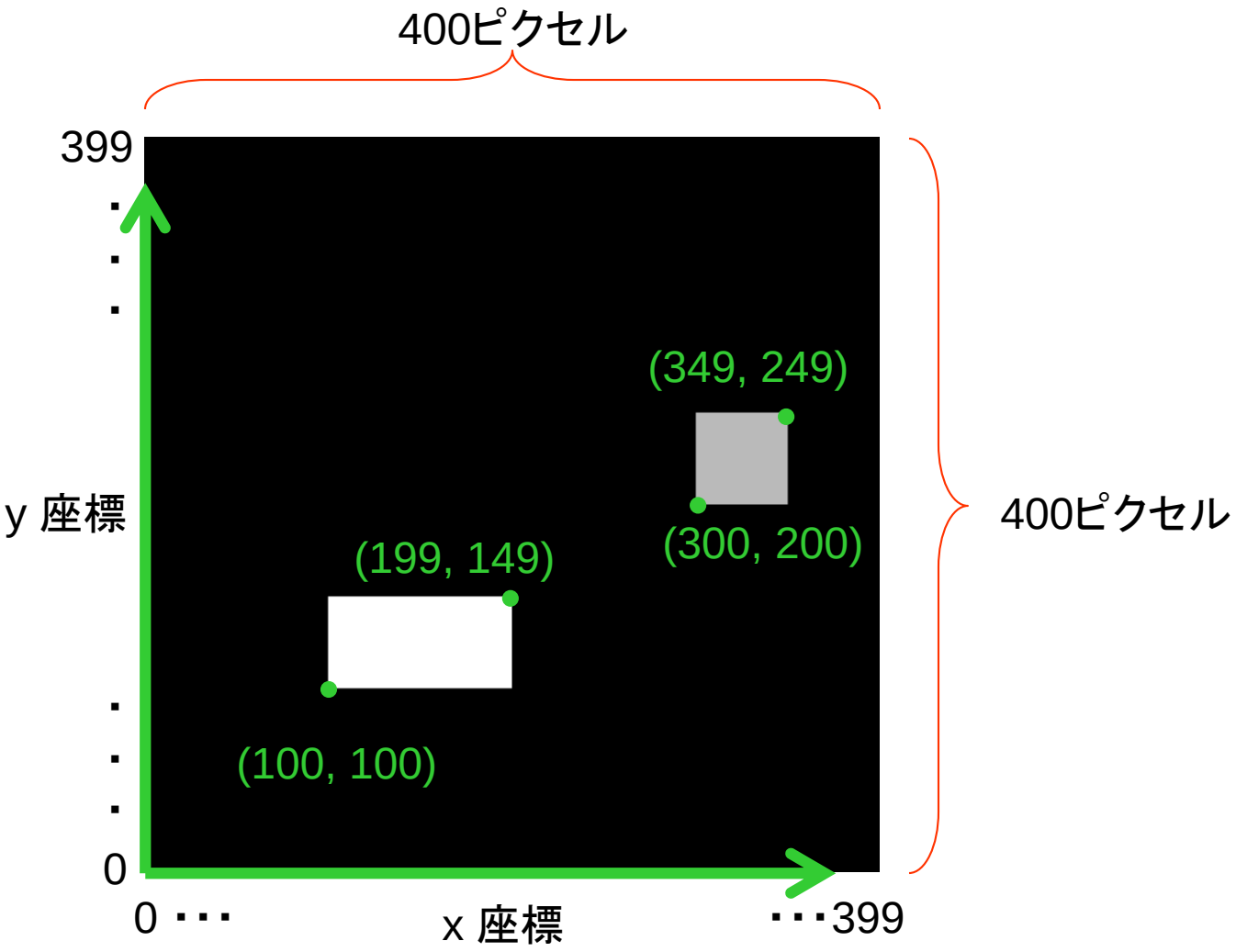
データ領域の確保

Image型構造体変数のimgを宣言して初期化

cglecライブラリのCglSetAll()関数を呼び出す

cglecライブラリのCglSaveGrayBMP()関数を呼び出す

実行結果(PaintRect.bmpファイル)



cglecライブラリの関数

```
void CglSetAll(Image img, unsigned char level)
```

画像全体を一つのグレーレベルで塗りつぶす関数
img　　塗りつぶす画像を示すImage型構造体変数
level　グレーレベル

```
void CglSaveGrayBMP(Image img, const char* fname)
```

画像をBMPファイルとして保存する関数
img　　保存する画像を示すImage型構造体変数
fname　ファイル名(拡張子は「.bmp」にしておくこと)

プロジェクトフォルダ
各自が作成したソースファイル(〇〇〇.cpp)があるフォルダ

cglecライブラリを使うには

LMSから，ファイルcglec111.zipをダウンロードし，それを解凍して中にある次の3つのファイルをVisual Studioのプロジェクトフォルダに入れておく

- cglec.h
- cglec.lib　⇒ この2つはコンパイル(ビルド)時にのみ必要なファイル
- cglec.dll　⇒ プログラム実行時に必要なファイル

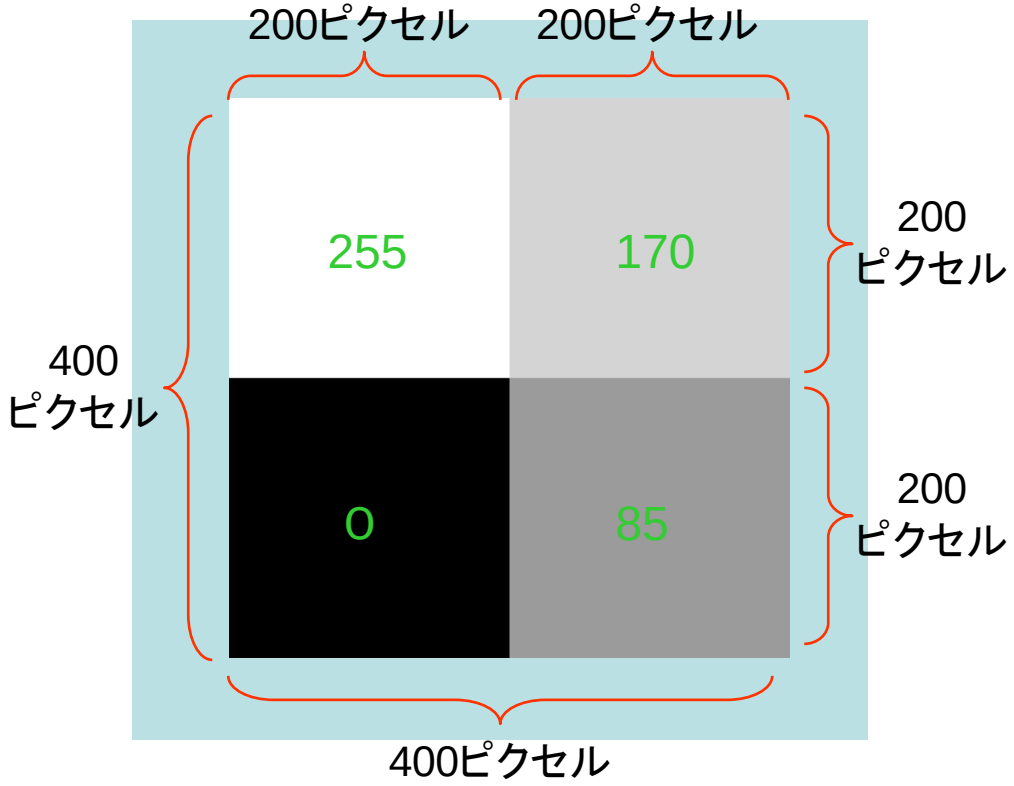
注) 新たにプロジェクトを作成するたびにそのプロジェクトフォルダーに3つのファイルをコピーしておく必要がある．(いちいちコピーしなくても良い様に設定できるが，少し設定が難しい)

(注) 必ず最新のcglec (cglec111.zip)を使用してください．Rel.1.1.0以前のcglecではガンマ値が1.0であるため，この結果と一致しません．

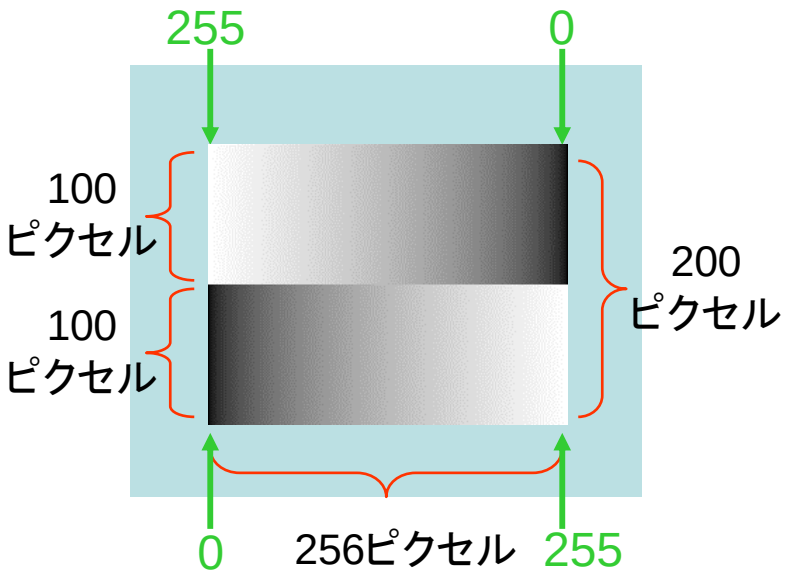
今日の課題

次のようなグレースケール画像を作り出すプログラムを作成せよ．
（緑色の数字はグレーレベル）

基本課題2　必ず提出



発展課題2　できれば提出



※階調が1レベルづつ連続的に変化すること

提出方法　Wordファイルに以下を貼りつけてLMSにアップ
(1)ソース，(2)実行結果(画像)