〈第1章〉
C言語入門

1

〔第1節〕 ディジタル画像

コンピュータで画像を扱うには，ディジタルにして扱う必要がある．ディジタルにするためには標本化と量子化をすることになる．コンピュータではメモリ上にデータを1つずつ格納していく．そのため連続的なデータを扱うことができない．連続的なものは一定間隔で区切ってその値を格納する．連続データをある一定間隔 Δx で区切り，その値をとっていくことを標本化という．1次元での標本化の様子を**図1-1**に示す．次に，とってきた値を格納するのは通常整数値であるので，小数点以下の値がある場合は，四捨五入などをして整数値にする．これを量子化という．1次元での量子化の様子を**図1-2**に示す．白丸で量子化した値が，四捨五入されて黒丸の位置の値で格納される．

2次元では，**図1-3**に示すようにディジタル化した画像は格子状に区切られる．格子の最小単位を画素（ピクセル）と呼ぶ．通常は，画素の中心を代表値として標本化し，四捨五入して量子化した値がそれぞれの画素に与えられる．ディジタル画像では，その数値に濃淡や色を付けて表現する．このディジタル画像をExcelで再現することができる．Excelの1つのセルを画素とみなし，縦と横の長さを合わせる．**図1-3**と同じように数字を入力すると**図1-4**のようになる．数字の入っているセルを選択し，「ホーム」タブ→「スタイル」グループ→「条件付き書式」→「カラースケール」から任意のスケールを選択すると，画素の値に応じて色を付けられる．2色のカラースケールで色付けした画像を**図1-5**に示す．

付属ソフトのdisplay.exeを使うとディジタル画像とExcelでの値の交換が可能となる．先ほどのセル範囲をコピーして，display.exeで「編集」メニュー→「Excelから貼り付け」を選択すると**図1-6**に示すように表示される．逆に，display.exeに表示したディジタル画像はExcelにコピーできる．**図1-7**に示すようにdisplay.exeに64×64画素のShepp-Loganファントムを表示し，「編集」メニュー→「テキストでコピー」を選択する．Excelの任意のセルを選択し，「貼り付け」をすると**図1-8**に示すように，セル範囲に数値が入る．このセル範囲に条件付き書式のカラースケールで色付けすると**図1-9**に示すようにShepp-Loganファントムの形が浮き上がってくる．このように医用画像で用いるディジタル画像は，Excelのセル範囲に値を入れて色付けしたものと同じと考えることができる．

〔第2節〕 整数画像と実数画像

画像の値は，通常，整数値で表される．しかし，画像処理や画像再構成などの処理を行うときの計算は実数で行われる．計算結果を整数値の画像に格納するとき，実数値を整数値に変換する．その変換では小数点以下を切り捨てるため，必ず丸め誤差が生じる．よって，処理を施した画像は，なるべく実数値のままで保存しておくとよい．ただし，画像を表示するソフトのほとんどが実数画像に対応していないので，画像を表示する際は整数画像にする必要がある．付属ソフトのdisplay.exeでは，実数画

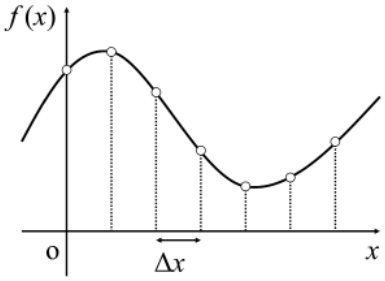

図 1-1　1 次元データの標本化

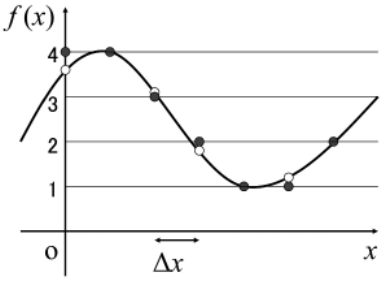

図 1-2　1 次元データの量子化

0	0	0	0	0	0	画素
0	1	2	2	1	0	
0	2	3	3	2	0	
0	2	3	3	2	0	
0	1	2	2	1	0	
0	0	0	0	0	0	

図 1-3　2 次元データ（画像）の標本化と量子化

図 1-4　Excel に数値を入力

図 1-5　条件付き書式のカラースケールを使って色付けしたセル

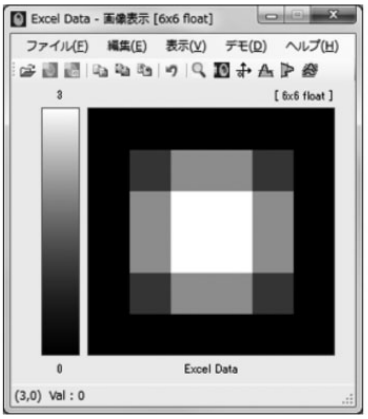

図 1-6　display.exe に Excel の数値を含んだセル範囲を貼り付けた様子

図 1-7　64×64 画素の Shepp ファントムの画像

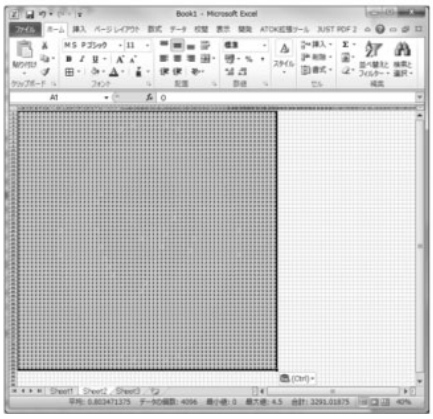

図 1-8　Excel に画像の値が貼り付いた様子

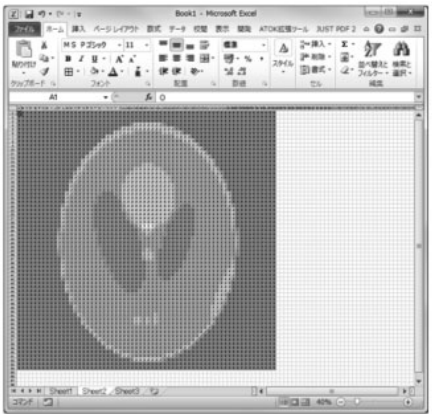

図 1-9　条件付き書式のカラースケールにより，
値に応じてセルに色を付けた様子

像を表示することができ，それを 2 バイト整数画像として保存する．整数画像として保存したい場合は，「ファイル」メニュー→「名前を付けて保存」からファイルの種類に「2 バイト整数（Short 型）画像ファイル」を指定して保存する．

整数画像では，2 バイト整数（short 型）が用いられ，実数画像では，4 バイト実数（float 型）が用いられる．整数画像から実数画像へは精度が上がるので，そのまま変換することができるが，逆の場合は精度が落ちるので，実数画像の値が 32767 より大きい場合は 32767 に，−32768 より小さい場合は −32768 にする．さらに小数点以下は四捨五入する．

〔第 3 節〕　数学と画像の座標系

数学の 2 次元座標系は，**図 1-10** に示すように原点が中心にきて，x 軸は右向きで y 軸は上向きとなる．それに対して，画像の 2 次元座標系は，画像の 2 次元データがメモリ上で左上から右下に順番に格納されているため，原点が左上となり，x 軸は右向きで y 軸は下向きになる．よって，画像の上で数学の式

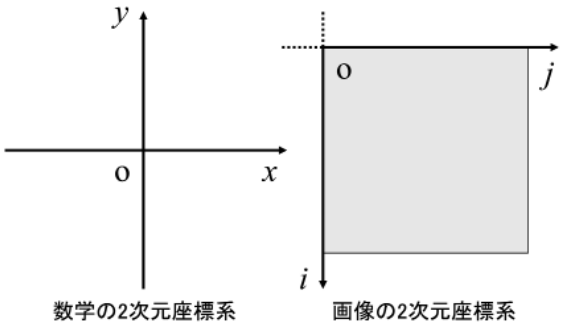

図 1-10　数学と画像の 2 次元座標系

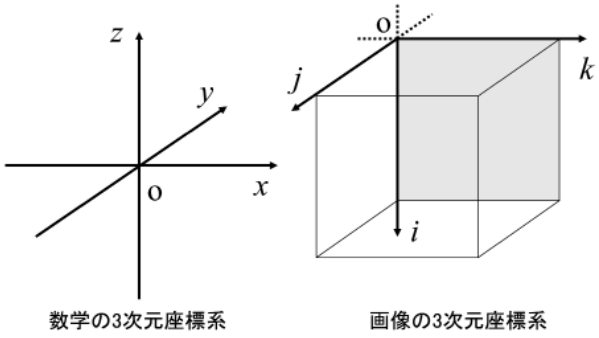

図 1-11　数学と画像の 3 次元座標系

を扱う場合，座標系の変換が必要となる．両者を変換する式は

$$\begin{cases} x = j - NX/2 \\ y = NY/2 - i \end{cases} \tag{1-1}$$

となる．ここで，(j, i) が画像の座標でそれに対応する数学の座標が (x, y) である．また，NX と NY はそれぞれ画像の幅と高さである．横方向は，画像は 0 から $NX-1$ となるが，数学の座標系では $-NX/2$ から $NX/2-1$ となり，マイナスの画素が 1 つ多くなる．一方，縦方向では，画像は 0 から $NY-1$ となるが，数学の座標系では $NY/2$ から $-NY/2+1$ となり，プラスの画素が 1 つ多くなる．

3 次元座標系においては，**図 1-11** に示す関係になる．座標系の変換式は

$$\begin{cases} x = k - NX/2 \\ y = NY/2 - j \\ z = NZ/2 - i \end{cases} \tag{1-2}$$

となる．ここで，(k, j, i) が画像の座標でそれに対応する数学の座標が (x, y, z) である．また，NX, NY と NZ はそれぞれ画像の幅，奥行きと高さである．

〔第 4 節〕　C 言語プログラムの作成

医用画像を扱う基本のプログラムを作成しながら C 言語を解説する．プログラムを作成しながら C 言語に慣れよう．

（1）2 次元の矩形画像を作成するプログラム

プログラム P1-01rect2d.c

（1.1）コメント

どのようなプログラムを作成するのかを記述しておくコメントを記入する．コメントはプログラムの内容とは関係ない解説や覚え書きなどを記述しておくもので，以下の 2 種類がある．

① コメントの始まり（/*）と終わり（*/）を指定する方法（複数行にまたがることができる）

/* 〜 */

② 行の途中から // 以降をコメントとする方法（その行のみ）

// 〜

【コード】P1-01rect2d.c　（グローバル領域）

```
/*
  2 次元の矩形画像を作成するプログラム
  p1-01rect2d.c
*/
```

（1.2）プリプロセッサ（include 文）

C 言語で準備されている標準的な関数を使うために，必要なコードがあらかじめ書かれている拡張子が「.h」のヘッダーファイルを include 文で指定する．

【コード】P1-01rect2d.c　（グローバル領域）

```
#include  <stdio.h>
#include  <stdlib.h>
```

（1.3）プリプロセッサ（define 文）

画像の幅，高さなど特別に指定する数値などを文字列に割り当てておく．実行用のファイルを作成するときに，その文字列を数値に置き換えてくれる．

【コード】P1-01rect2d.c　（グローバル領域）

```
#define  NX  256  // 画像の幅（x 方向）
#define  NY  256  // 画像の高さ（y 方向）
```

NX を 256 とし，NY を 256 とする．

// 以降はコメントである．

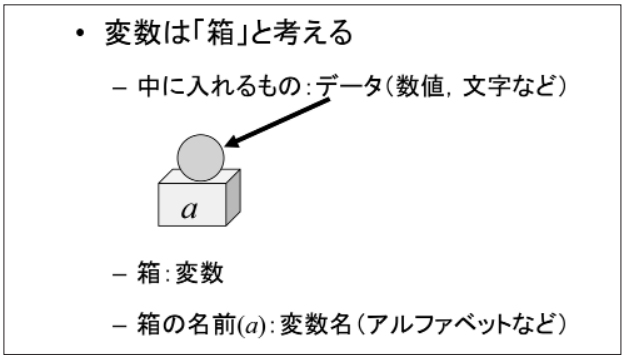

図 1-12　変数（Variable）

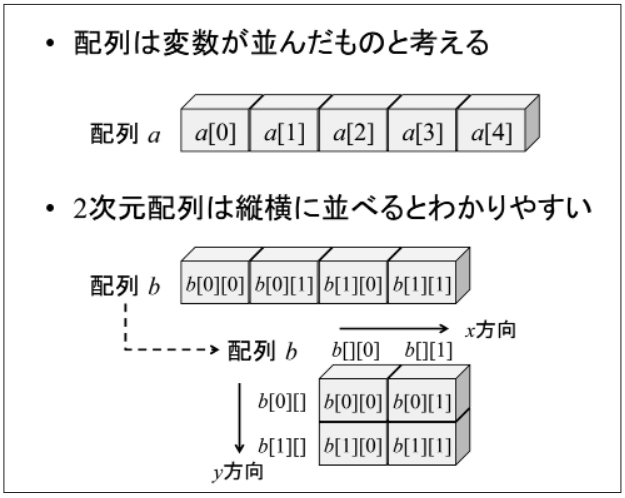

図 1-13　配列（Array）

（1.4）グローバル変数の宣言

プログラム全体で共通に使用する変数を宣言する．画像などの大きな配列を使用する場合は，グローバル変数として宣言した方がよい．

変数は，**図 1-12** に示すとおり，データを入れる「箱」のようなものと考える．配列は，**図 1-13** に示すとおり，変数が並んだものと考える．2 次元配列は縦横に並べると画像（行列のようなマトリクス）と対応できる．変数と配列の宣言は，以下の形で行う．

データ型　変数名；
データ型　配列名 [要素の数]；　（1 次元配列）
データ型　配列名 [要素の数][要素の数]；　（2 次元配列）
データ型　配列名 [要素の数][要素の数] [要素の数]；　（3 次元配列）

主なデータ型を**図 1-14** に示す．整数型は整数のみを扱えて，実数型は小数を扱うことができる．実数型の扱える値の範囲は，小数の精度の桁数と 10 のべき乗の大きさで示している．C 言語は 1 つの命

	データ型	サイズ	扱える値の範囲
整数型	char	1バイト	−128〜127
	short	2バイト	−32768〜32767
	int （long）	4バイト	−2147483648〜2147483647
実数型	float	4バイト	6〜7桁の精度で$10^{\pm 38}$の大きさ
	double	8バイト	15〜16桁の精度で$10^{\pm 308}$の大きさ

図 1-14　変数の種類（データ型）

令の最後を「;」で区切る．

【コード】P1-01rect2d.c （グローバル領域）

```
/* グローバル変数の宣言 */
char  g_fl[50];  // 出力用画像データのファイル名
int    g_d;  // 正方形の 1 辺の長さ（ピクセル）
float  g_img[NY][NX];  // 画像データのマトリクス
```

char 型の配列は文字列を扱う．
float 型（4 バイト実数型）の g_img という名前の 2 次元配列は，横が NX=256，縦が NY=256 の画像データを扱う．

（1.5）メイン関数（main 関数）の定義

メイン関数の中（｛と｝の間）でプログラムの流れ（アルゴリズム）を記述する．C 言語の関数では，型と引数を指定するが，今回のメイン関数では void 型で引数はなし（void）としている．詳細は関数を扱う節で説明する．関数の定義は命令ではないので，「;」は付けない．

【コード】P1-01rect2d.c （main 関数）

```
void  main( void )
{


}
```

これ以降の入力コードは，メイン関数の｛と｝の間に記述する．

（1.6）ローカル変数の宣言

メイン関数内でのみ使用する変数をメイン関数の最初に宣言する．宣言の方法はグローバル変数と同じである．

【コード】P1-01rect2d.c （main 関数）

```
int    i, j;
FILE  *fp;
```

整数型の変数 i と j は，繰り返しを行うときに回数を記録する変数として使用する．FILE 型は，ファイルへの入出力を行うときに使用する．

（1.7）画面への文字表示とキーボードからの入力

画面に文字を表示するときは printf 関数を用いる．使い方は，以下の通りである．

```
printf(" 表示する文字 ¥n");
```

ここで，¥n は改行を表す．

キーボードから文字列や数値を入力するときは scanf 関数を用いる．使い方は以下の通りである．

① 整数を入力する場合：

```
scanf("%d", &g_d);
```

%d が整数を意味し，& マークを付けて int 型の変数名を指定する．

② 文字列を入力する場合：

```
scanf("%s", g_fi);
```

%s が文字列を意味し，char 型の配列名を指定する．

【コード】P1-01rect2d.c （main 関数）

```
// プログラムの説明
printf("2 次元の矩形（正方形）画像を作成するプログラム ¥n¥n");

// プログラムで使用する変数の入力
printf(" 正方形の 1 辺の長さ（ピクセル）：");
scanf("%d", &g_d);

printf(" 出力用画像データのファイル名：");
scanf("%s", g_fl);
```

実行したときにどのようなプログラムかわかるように，プログラムのタイトルを表示する．このプログラムの実行時に入力する数値や文字列を変数や配列に入力する．そして，どのようなものを入力するのかわかるように画面に表示させる．

（1.8）画像データの初期化

繰り返しを利用して，画像データのすべての要素に 0 を入力する．繰り返しには for 文を使用すると便利である．for 文の書き方は以下のようになる．

```
for( 初期値 ; 条件 ; カウンタの増減 )
{
        処理；
}
```

{ と } の間の「処理」を繰り返すことになる．その繰り返しの流れを**図 1-15** に示す．2 種類示しているが，どちらも同じ流れを意味する．また，2 重ループの例を**図 1-16** に示す．画像に値を入力する場合は 2 重ループを利用し，画素に 1 つずつ値を入力させる．画像に対応させて考えると，**図 1-17** に示すように，j が x 方向，i が y 方向に対応し，j と i の値をそれぞれ 0 から $NX-1$ と $NY-1$ まで変化させて，座標 (j, i) の画素に 0 を代入している．