2.3　多次元配列

(1)多次元配列の考え方

旧来の C 言語，C++では，配列を要素とする配列を多次元配列としており，たとえば，

```
int A[4][30];
A[3][2] = 87;
```

と記述することになっていますが，C#では，他の多くの言語と同様，以下のように自然な表現となっています。

```
int[,] A　=　new　int[4,30];
A[3,2] = 87;
```

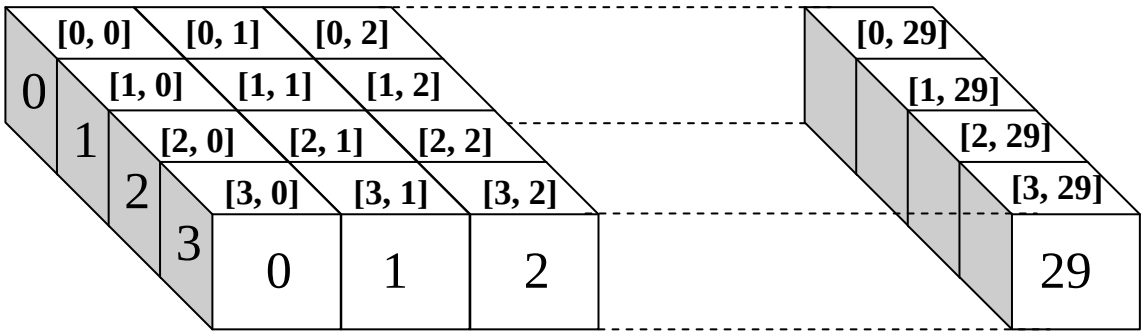


図 2-8　2 次元配列

(2)数値地図

2 次元配列の代表例として，XY のメッシュ番号をインデックスとし，標高値を要素値とする数値地図を挙げることができます。

例えば，以下のように XY 座標をメッシュで分割し，各マス目に標高値が入るようなデータです。

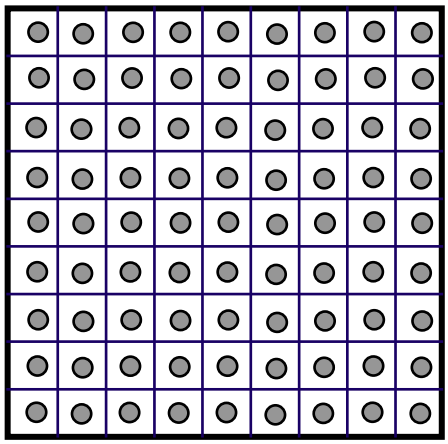


図 2-9　メッシュデータのイメージ

国土地理院の 50m メッシュ数値地図は，約 50m（正確には緯度経度で異なる）メッシュの縦 200×横 200 のマス目に標高値×10 の整数値が入っています。

ファイルの形式は提供されている CD に付属する説明書に委ねますが，大まかにいうと，先頭 1 行がタイトル及び制御データ，2 行目以降に番号と 5 桁刻みの標高値が入っています。南西から東方向に 200 個の標高値データが並び，これが北方向に並びます。

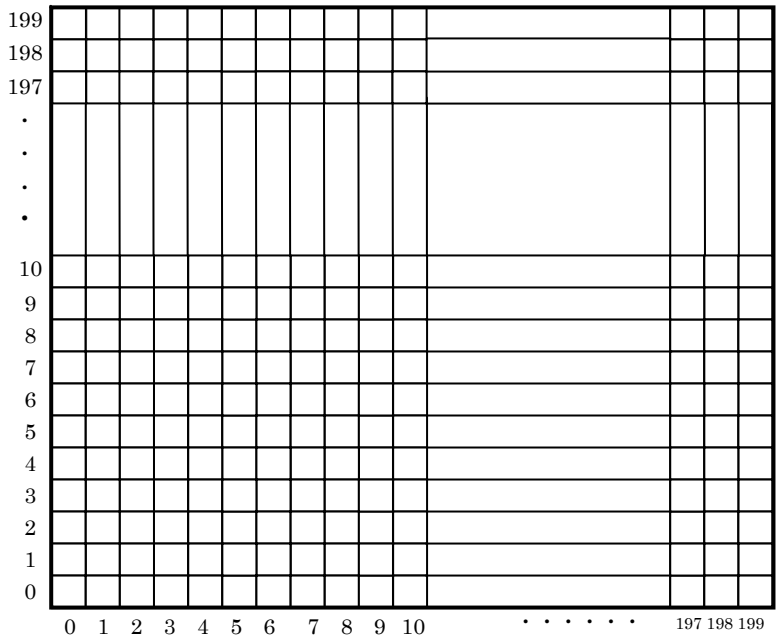


図 2-10　国土地理院 50m メッシュ数値地図のデータの順序

国土地理院の数値地図(MEM)ファイルを読み込んで，色地図として画面に表示するプログラムを Program 2-12 に示します。

プログラムでは，ビットマップ型の変数に描画しておき，それを**描画用オーバライド** OnPaint で描画します(注)。

なお，先頭レコードのタイトル，制御データ，2 行目以降の先頭レコード番号等は無視し，表示に必要な標高データのみ使用しています。

(注) Onpaint で描画しないと，画面切替え等のタイミングでフォームの再描画が行われません。

[Program 2-12] 数値地図データの読み込みと色地図表示

Form に button1(Text：数値地図読込み)と openFileDialog1 を貼り付けます。以下は実行例です

プログラムリスト１

```
private double [,] pbJR=new double[201,201];
private string pbA;
private Bitmap image;
// 描画用オーバライド
protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e); e.Graphics.DrawImage(image,0,0);
}
private void 描画()
{
  double MX = 標高最大(); double MN = 標高最小();
  double DX = (MX - MN) / 255; if(DX < 0.000001) DX = 255;
  Graphics g=Graphics.FromImage(image);
  for(int i = 1; i <= 200; i++)
      for(int j = 1; j <= 200; j++)
      {  int C = (int)((pbJR[201 - j, i] - MN) / DX);
         Brush brush = new SolidBrush(Color.FromArgb(C, C, C));
         g.FillRectangle(brush, j - 1, i - 1, 1, 1);
       }
}
private double 標高最大( )
{
  double MX = pbJR[1, 1];
  for(int i = i; i <= 200; i++)
     for(int j = 1; j <= 200; j++)
                    if(MX < pbJR[i, j]) MX = pbJR[i, j];
  return MX;
}
private double 標高最小( )
{
  double MN = pbJR[1,1];
  for(int i = 1; i <= 200; i++)
     for(int j = 1; j <= 200; j++)
                    if(MN > pbJR[i, j]) MN = pbJR[i, j];
  return MN;
}
```

プログラムリスト２

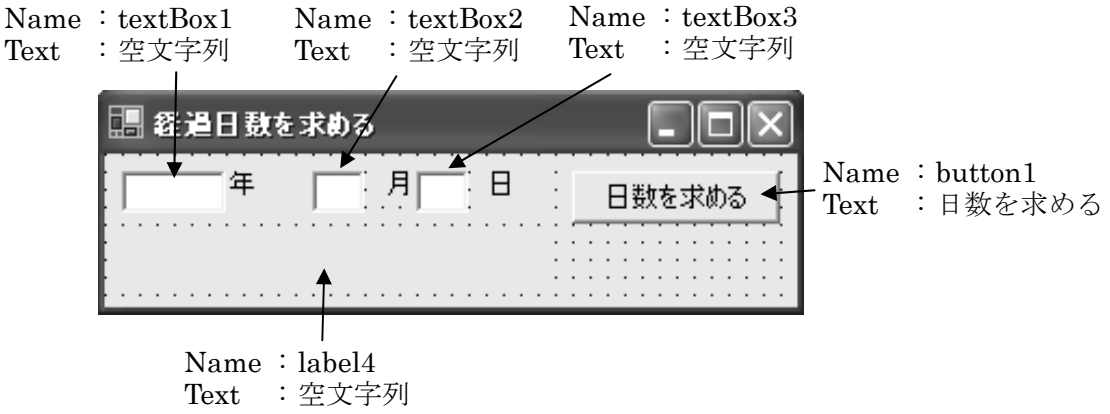
```
private void openFileDialog1_FileOk(object sender,
                  System.ComponentModel.CancelEventArgs e)
{
  StreamReader DTS;
  int ii, pbll;
  string FName = openFileDialog1.FileName;
  if(FName == "") return;
  DTS  = new StreamReader(FName, System.Text.Encoding.Default);
  pbA  = DTS.ReadLine();              // 制御データの無視
  pbll = 0;
  string DT = DTS.ReadLine();         // １行読込み
  while(DT != null)
  {
      ii = 5; pbll++;
      for(int j = 1; j <= 200; j++) // 5桁ずつ区切って
      {                             // 西→東にデータを設定
        ii += 5;
        pbJR[pbll, j] = int.Parse(midStr(DT, ii, 5));
      }
      DT = DTS.ReadLine();          // １行読込み
  }
  DTS.Close(); 描画();
  this.Invalidate();         // OnPaint を強制的に引き起こす
 }
private string midStr(string DT, int ist,int N)
{
    string S = ""; int k = ist - 1;
    for(int i = 1 ; i <= N; i++) S += DT[k++];
    return S;
}
private void button1_Click(object sender, System.EventArgs e)
{
    openFileDialog1.ShowDialog();
}
private void Form1_Load(object sender, System.EventArgs e)
{
    image = new Bitmap(200,200);
}
```

(3)経過日数

多次元配列は，何らかの条件で値が異なるような表データとして用いられることも多くあります。

ここで示す例は，うるう年と通常年のときの月の日数の違いを表で表した例です。

[Program 2-13] 経過日数




```
private int[,] 月日数 = new int[,]
  {{31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
   {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}};
private int うるう年判定(int Y)
{
  return (Y % 4 == 0 && Y % 100 != 0 || Y % 400 == 0)? 1 : 0;
}
private int 経過日数(int Y, int M, int D)
{
  int i; int 日数 = D; int K = うるう年判定(Y);
  for (i = 1; i < M; i++) 日数 += 月日数[K, i - 1];
  return 日数;
}
private void button1_Click(object sender, System.EventArgs e)
{
  int Y = int.Parse(textBox1.Text);
  int M = int.Parse(textBox2.Text);
  int D = int.Parse(textBox3.Text);
  label4.Text = 経過日数(Y,M,D).ToString();
}
```