【第 2 回】
デジタルデータの理解と C 言語による画像データのハンドリング

人工衛星リモートセンシングによって撮影された画像データはデジタルデータである。よって、コンピューターで扱うことができ、様々な環境解析に応用することができるため、多数のアプリケーションソフトウエアが市販され、フリーウエアも存在している。これらのアプリケーションソフトウエアを駆使して、環境解析業務をこなすことが可能であるが、画像データの中身を知れば、ソフトウエアの高度な利用、操作上の様々な問題の解決、オリジナルな解析法の開発、等が可能になる。

君たちはソフトウエアを使っているか？　使われていないか？

そこで、この演習では C 言語を使って、画像データを自分で操作してみることにする。これができればデジタルデータを自由自在に扱いながら、煩雑な処理はアプリケーションソフトウエアに任せ、コンピューターを自分の手足にすることができる。

とはいえ、C 言語を習得するには日々のトレーニングの積み重ねが必要である。今回の演習は難しいかも知れないが、まずはやってみよう。プログラムを実際に動かしてみて、できる！という感触を掴んでほしい。

■　ステップ１：C プログラムの作成から実行までの手順

C プログラムを書いて、それをコンピューターで実行させるためには以下の手順が必要である。

ａ．エディターによるプログラムの作成
ｂ．コンパイルとリンク
ｃ．実行

ａ．エディターとはテキストファイルを作成するためのソフトウエア

ここでは Windows に付属のメモ帳を使います。“秀丸”はユーザーも多い有名なシェアウエアのソフトウエアです。

ｂ．作成した C プログラムを実行可能な形式にするためには、コンパイルとリンクという手順を経なければなりません。そのためにはソフトウエアが必要ですが、ここでは Borland C++Builder を使うことにします。

今後、使い続けたい方は下記に登録して自分でダウンロードしてください。

http://www.codegear.com/jp/downloads/free/cppbuilder

ｃ．実行

ここではコマンドプロンプトを開いて、コマンドラインから実行する方法を覚えましょう。コマンドプロンプトは UNIX ならば Shell に相当します。Windows のルーツである MS-DOS の時代はコマンドをキーボードから打ち込んで操作していました。

■　ステップ２：簡単な C プログラムの作成と実行

ａ．プログラムの作成

以下のプログラムをメモ帳で作成し、dekita.c としてセーブしましょう。

```
#include 〈stdio.h〉
main ()
{
        printf("はじめてできた！¥n");
}
```

注）C 言語の正式な書式については参考書で学んでください。

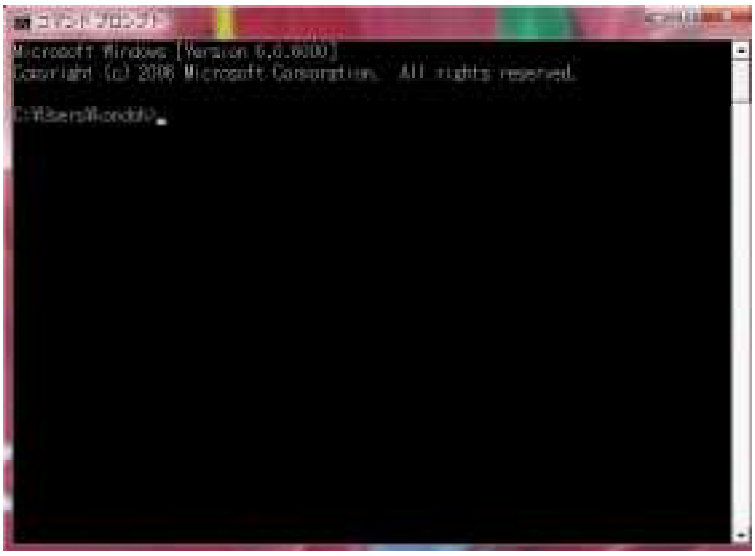
このプログラムは文字列を画面に表示しますが、ハードウエアを操作して画面の所定の位置に文字を表示する複雑な処理は簡単にはできません。そのために、printf() という関数が用意されていますが、この関数を使うためのパラメーター群が stdio.h に用意されているので、付加(include)すると覚えてください。

注）Ｃ言語では様々な関数が準備されていますが、それぞれ必要なヘッダーファイルは異なります。関数を一つ一つ覚えながら、必要なヘッダーファイルもその都度確認してください。

ｂ．実行形式への変換　－コマンドプロンプトの利用－

作成したプログラムは単なるテキストファイル[注)] です。そのままでは何の機能も発揮しません。そこで、実行形式の *.exe　に変換するプログラムが別に必要になります。ここでは、C++Builder を使います。

そのために､まずコマンドプロンプトの操作を覚えましょう｡コマンドプロンプトは[プログラム]－[アクセサリ]の中にあるはずです。起動してみましょう。



・黒っぽい枠が現れました。よく見ると、

C:¥Users¥kondoh〉_

とあり（これは使っている PC により異なります）、カーソルがブリンクしています。ここに、コマンドを入力することにより、様々な操作が可能になります。

・この場合、ルートの下の Users フォルダにさらに kondoh というフォルダがあり、現在ここにいることがわかります。

・フォルダはディレクトリとも呼

ぶ。

・自分の作業フォルダに移動してみましょう（現在いる場所はコンピューターによって異なります。必要なコマンドは cd（Change Directory）です。

cd ¥　ルート（最上階）に移動します。（cd /でもよい）
cd .. ひとつ上の階層に移動します。

作業フォルダに移動したら、dir コマンドを入力します。dekita.c があるはずです。存在を確認したら bcc32 dekita.c と入力しましょう。

Borland C++ 5.5.1 for Win32 Copyright（c）1993, 2000 Borland
dekita.c:
警告 W8070 dekita.c 6: 関数は値を返すべき(関数 main)
Turbo Incremental Link 5.00 Copyright（c）1997, 2000 Borland

このように表示されています。

関数は値を返すべき（関数 main）という警告が出ましたが、C 言語は関数の集合であり、すべての関数は戻り値を持ちます。main()も関数ですが、戻り値を指定しませんでしたので警告が出ましたが致命的なエラーではなかったので実行形式のファイルは生成されました。

dekita.c で main() を void main() に変更してもう一度実行形式のファイルを作成して見るとメッセージの内容が変わるはずです。

ｃ．実行

dir でフォルダの中身を確認すると、dekita.exe ができています。コマンドプロンプトで dekita と入力すると、文字が表示されるはずです。はじめてできましたか。

注）テキストファイルとバイナリーファイル
アルファベットや漢字などの文字が格納されているファイルがテキストファイルで、エディタやワープロで編集できます。数字やコードが 2 進数(あるいは 16 進数）で格納されているファイルがバイナリーファイルです。

■　ステップ３　画像データの形式

ａ．画像データとは何か。

ひとつの画素の明るさがデジタル値(DN:Digital Number)で表され、二次元の配列の中に格納されたデータ。DN をコンピューターで明るさ表示すると画像になります。

DN は 8 ビット、16 ビット、32 ビット等の二進数が使われます。たとえば、8 ビット、16 ビット整数では下記の値域の整数を扱うことができます。

unsigned char	符号なし 8 ビット整数	0-255
signed char	符号付き 8 ビット整数	-128 〜+127
unsigned short	符号なし 16 ビット整数	0 〜 65,535

signed short　　符号付き 16 ビット整数　　-32,768 ～ 32,767

型の宣言名はコンパイラによって異なるので注意。

ランドサットＴＭの場合、DN は符号なし 8 ビット整数で表される。よって、一つの画素は 0 から 255 の 256 段階の明るさを持つ。この DN は別に与えられている変換式により放射輝度に変換することも可能だが、DN のまま画像処理を行うことができる。

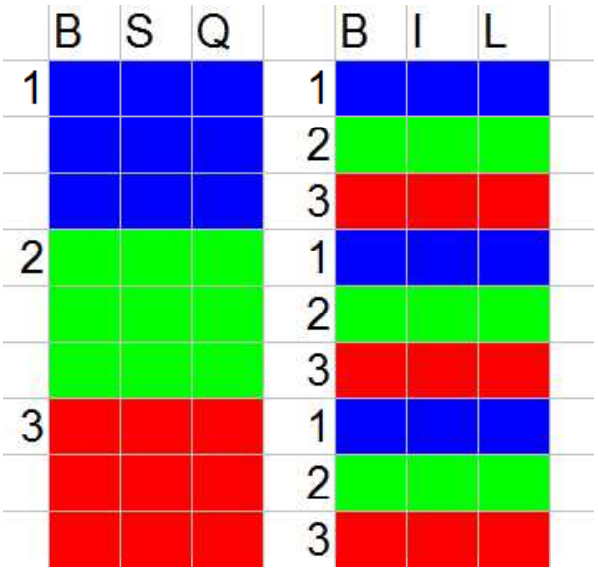
注）10 進数の 2 は二進数では 10 と表記されます。10 進数の 4 は二進数では桁があがって 100 となります。これは 3 ビットになり、3 ビットで表される最大の二進数 111 は 10 進数では 7 になります。よって、3 ビットでは 0 ～ 7 の 8 段階の整数を表すことができます。同様に、8 ビットでは 10 進数の 0 ～ 255 の 256 段階の整数を表すことができます。

b．画像の格納形式－ BSQ と BIL －

一つの画像は DN の二次元配列です。では、多数の波長帯で観測された多バンドのデータはどのように格納されているのでしょうか。

BSQ（Band Seaquential）：一つの波長帯（バンド）の画像を二次元の配列で表し、バンド分を連結したデータ

BIL（Band Interleived by Line）：ラインごとにバンド分のデータを並べたデータ。



左の図は 3 つのバンドを持つ 3 × 3 の画像の格納方法を表している。

BSQ ではバンドごとに完結した 3 つの画像データが連結されている。

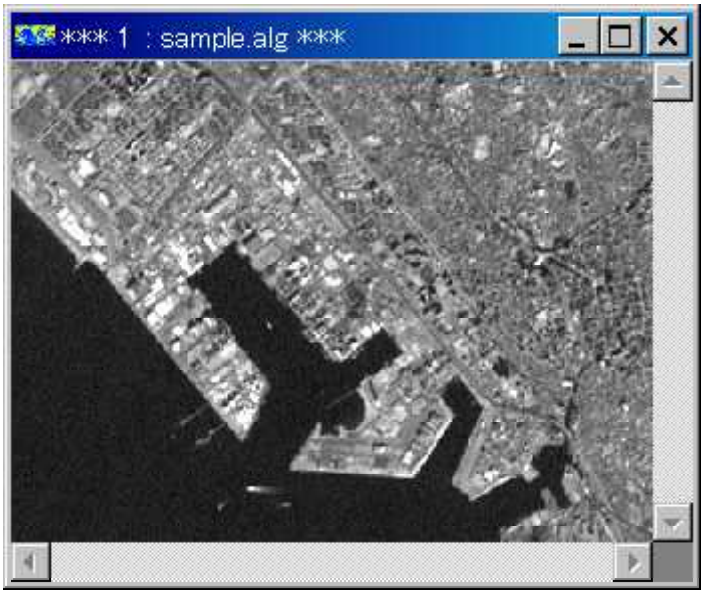
BIL では１ラインごとに全バンドのデータが並んでいる。かつて記憶装置として磁気テープが使われていた時は、連続的に読み込みながら画像化ができた。

現在はランダムアクセスのできるハードディスクにデータを格納するので、どちらを使っても良い。

そのほか、BIP 形式もある。

■　ステップ４：画像データの読み込みと値の表示

サンプル画像に格納されている DN をプログラムによって読んでみましょう。



・この画像は 2001 年 11 月 27 日撮影のランドサット 7 号 ETM+による白黒画像です。分解能は 15m で、このサンプル画像のサイズは 400 ピクセル× 300 ラインです。

・sample.bin という名前でデータを準備しました。ファイル名は sample、拡張子が bin で binary を意味しますが、これは開発者が勝手に命名しただけです。名前に惑わされず、画像データの本質を理解すること。

・１画素は 8 ビット（256 階調）、すなわち 1 バイトですので、画像の大きさは 400 × 300 × 1=120,000 バイトになります。ファイルの大きさを確認してください。デスクトップからは画像のプロパティーをみてください。サイズ 117KB（120,000 バイト）となっていますね。括弧の中が正確なサイズです。コマンドプロンプトで dir を実行したときもサイズが表示されます。

・このファイルを読み込んで、画素の値を表示するプログラムを作りましょう。その手順は：

・必要なヘッダーファイルのオープンとパラメータのセット
・ファイルオープン
・ライン方向のループの中で 1 行分のデータを読み込み
・ピクセル方向のループの中で 1 画素づつ値を表示
・1 ライン分の表示が終わったら改行
・おしまい

となります。では自分でプログラムを入力し、実行してみましょう。

```
/*****************************************************************************
   指定したライン・ピクセルのデジタル値を得る GETDN1.C
 *****************************************************************************/

#include        〈stdio.h〉   /* ここは別のプログラムのカット＆ペーストなので */
#include        〈stdlib.h〉        /* よけいなヘッダーファイルも入っているが、*/
#include        〈fcntl.h〉         /* 気にしない */
#include        〈sys¥types.h〉
#include        〈sys¥stat.h〉
#include        〈io.h〉
#include        〈dos.h〉
#include        〈process.h〉
#include        〈conio.h〉
#include        〈ctype.h〉
#include        〈math.h〉
#include        〈string.h〉
```

```
#define             SE                      stderr    /* stderr は画面のみ出力 */
#define             FILE_ERR                 -1
#define             NUMBER_OF_LINES         300
#define             NUMBER_OF_PIXELS        400

unsigned char     buffer[NUMBER_OF_PIXELS];

void      main(void)
{
           int          fi;
           int          i,j;

          if( (fi=open("sample.bin",O_RDONLY | O_BINARY))==FILE_ERR) {
                    fprintf(SE,"%s をオープンできません．¥n");
                    exit(-1);
          }

           for(i=0;i<NUMBER_OF_LINES;i++) {
                    read(fi,buffer,NUMBER_OF_PIXELS);
                    for(j=0;j<NUMBER_OF_PIXELS;j++) {
                              fprintf(SE,"%3d ",buffer[j]);
                    }
                    fprintf(SE,"¥n");
           }

}/* end of main */
```

ファイル名はプログラムの中に埋め込んでありますので、データが存在するフォルダの中に実行ファイル（*.exe）を作成し、実行してください。

読み込んだ値が画面に流れますが、途中で止めたいときは＾ S（CTRL キーを押しながら S を押す）で止めてください。

［応用］プログラムを改良して、ファイル名、画素の表示位置をキーボードから読み込み、結果を表示させるプログラムを作成しなさい。

【補足説明】

①配列

配列は最初に宣言しておくことにより、メモリーの中に領域が確保されます。

```
unsigned char     buffer[NUMBER_OF_PIXELS];
```

ここでは、buffer という一次元の配列を 8 ビット整数(0 ～ 255)で作ります。その大きさは NUMBER_OF_PIXELS で定義された数で、プログラムの先頭で定義されています。

②ファイルオープン

```
if( (fi=open("sample.bin",O_RDONLY | O_BINARY))==FILE_ERR) {
          fprintf(SE,"%s をオープンできません．¥n");
          exit(-1);
}
```

open という関数を使っています。その書式は、open(ファイル名, フラグ)であり、ここではファイル名は sample.bin となります（直接ファイル名をプログラムに埋め込むときは""で囲みます)。

フラグは機能を表す文字列、ここでは O_RDONLY、O_BINARY を"|"(or)で連結します。バイナリーファイルを読み取りのみでオープンします。

C では関数は戻り値を持ちます。もし、open に失敗したときは-1 を返します。よって戻り値が-1 だった場合はファイルオープンが失敗したことになります。

ファイルオープンに失敗した場合、プログラムを止めるために exit(-1)で実行を終了させます。

③画面へ出力

```
fprintf(SE,"¥n");
```

これは少し難しかった。printf("¥n")で結構です。printf()関数は括弧内をディスプレーに出力します。文字を""で囲めばそれが表示されますが、C では改行(¥n)もプログラマが書かなければなりません。

fprintf(SE,"¥n");は、SE すなわち stderr（これは画面です）に出力すること（ここでは改行のみ）を意味します。

④ for ループ

```
for(i=0;i<10;i++) {
        (処理)
}
```

このループでは、変数 i が 0 から始まり、(処理）を行いながら、10 を越えるまで繰り返し処理されます。i=9 まで処理されることに注意してください。

⑤#define による定数の定義

一度作ったプログラムは財産になります。変数等を一々プログラムに埋め込んでいくと、後で修正して応用することが難しくなります。よって、#define で定義しておけば、値の変更が容易になります。

■ ステップ５：多チャンネル画像データの読み込みと値の表示

この画像は ASTER が撮影した 2008 年 1 月 8 日の VNIR（可視・近赤外、分解能 15m）画像です。

400 ピクセル× 300 ラインの大きさで、緑・赤・近赤外の 3 バンドのデータが BIL 形式で格納されています。

このデータを読んでみましょう。データの名前は sample2.bin です。なお、MIRINKids 用のファイルも用意しましたので、表示して確認してください。

```
/*****************************************************************************
   BIL 形式のファイルを読む              read_bil.c
 *****************************************************************************/

#include        〈stdio.h〉
#include        〈stdlib.h〉
#include        〈fcntl.h〉
#include        〈sys¥types.h〉
#include        〈sys¥stat.h〉
#include        〈io.h〉
#include        〈dos.h〉
#include        〈process.h〉
#include        〈conio.h〉
#include        〈ctype.h〉
#include        〈math.h〉
#include        〈string.h〉

#define         SE                      stderr    /* stderr は画面のみ出力 */
#define         FILE_ERR                -1
#define         NUMBER_OF_LINES         300
#define         NUMBER_OF_PIXELS        400
#define         NUMBER_OF_BANDS         3

unsigned char   buffer[NUMBER_OF_PIXELS];

void    main(void)
{
        int         fi;
        int         i,j,band;

        if( (fi=open("sample2.bin",O_RDONLY | O_BINARY))==FILE_ERR) {
                fprintf(SE,"%s をオープンできません. ¥n");
                exit(-1);
        }
```

```
        for(i=0;i<NUMBER_OF_LINES;i++) {
                for(band=0;band<NUMBER_OF_BANDS;band++) {
                        fprintf(SE,"[L:%3d B:%1d]  ",i,band);

                        read(fi,buffer,NUMBER_OF_PIXELS);

                        for(j=0;j<NUMBER_OF_PIXELS;j++) {
                                fprintf(SE,"%3d ",buffer[j]);
                        }
                        fprintf(SE,"¥n");
                }
        }

}/* end of main */
```

このプログラムの要点は、BIL を読むための for ループの使い方です。変数は i,band,j でそれぞれライン数、バンド、ピクセル数を表します。

【難しいと感じたあなたへ】

sample3.bin として 9 ピクセル× 9 ライン× 3 バンドのファイルを用意しました。0 から始まり、242 まで 1 づつ増加（インクリメント）する数字が格納されており、243 の要素からなります。

これを BSQ ファイルとして読んでみましょう。（read_as_bsq.c）

```
/*************************************************************************
        BSQ として読む。
 *************************************************************************/

#include        <stdio.h>
#include        <stdlib.h>
#include        <fcntl.h>
#include        <sys¥types.h>
#include        <sys¥stat.h>
#include        <process.h>
#include        <dos.h>
#include        <io.h>
#include        <share.h>
#include        <math.h>
#include        <string.h>
#include        <ctype.h>
#include        <conio.h>
#include        <errno.h>

#define         FILE_ERR                -1
#define         SE                      stderr
#define           NUMBER_OF_LINES               9
#define           NUMBER_OF_PIXELS      9
#define           NUMBER_OF_BANDS               3

unsigned char   buffer[NUMBER_OF_LINES][NUMBER_OF_PIXELS];
```

```
void    main(void)
{
        int     fi;
        int     i,j,band;

        if( (fi=open("sample3.bin",O_RDONLY | O_BINARY))==FILE_ERR) {
                fprintf(SE,"%s をオープンできません. ¥n");
                exit(-1);
        }

/* BSQ として読む */

        for(band=0;band<NUMBER_OF_BANDS;band++) {
                fprintf(SE,"Band:%2d¥n", band+1);

                for(i=0;i<NUMBER_OF_LINES;i++) {

                        read(fi,buffer[i],NUMBER_OF_PIXELS);

                        for(j=0;j<NUMBER_OF_PIXELS;j++) {
                                fprintf(SE,"%4d ",buffer[i][j]);
                        }
                        fprintf(SE,"¥n");
                }
        }

}/* end of main */
```

次に BIL ファイルとして読んでみましょう。(read_as_bil.c)

```
/*************************************************************************
        BIL として読む。
 *************************************************************************/

#include        <stdio.h>
#include        <stdlib.h>
#include        <fcntl.h>
#include        <sys¥types.h>
#include        <sys¥stat.h>
#include        <process.h>
#include        <dos.h>
#include        <io.h>
#include        <share.h>
#include        <math.h>
#include        <string.h>
#include        <ctype.h>
#include        <conio.h>
#include        <errno.h>

#define         FILE_ERR                -1
#define         SE                      stderr
#define         NUMBER_OF_LINES                 9
#define         NUMBER_OF_PIXELS        9
#define         NUMBER_OF_BANDS                 3
```

```
unsigned char     buffer[NUMBER_OF_BANDS] [NUMBER_OF_PIXELS];

void      main(void)
{
          int          fi;
          int          i,j,band;

         if( (fi=open("sample3.bin",O_RDONLY | O_BINARY))==FILE_ERR) {
                    fprintf(SE,"%s をオープンできません. ¥n");
                    exit(-1);
         }

/* BIL として読む */

          for(i=0;i<NUMBER_OF_LINES;i++) {
                    fprintf(SE,"Line:%2d ¥n",i);

                    for(band=0;band<NUMBER_OF_BANDS;band++) {
                              fprintf(SE,"Band:%2d ",band);

                              read(fi,buffer[band],NUMBER_OF_PIXELS);

                              for(j=0;j<NUMBER_OF_PIXELS;j++) {
                                        fprintf(SE,"%4d ",buffer[band] [j]);
                              }
                              fprintf(SE,"¥n");
                    }
          }

}/* end of main */
```

違いがわかりましたか。for ループの部分の違いに注目してください。

RS/GIS 実習　－ C 言語の習得－　　近藤(2008.10.24)

【その１】　タンクモデルの計算

●単位の底面積を持つタンクを考える。
●底面に流出孔がある。
●最初、タンクに 100mm の水が入っているとする。
●孔から水が漏れだしてからの、タンクの水位の時間変化を求めよ。

[考え方]

```
        u;      /* 水位(mm) */
        v;      /* 漏れの速さ(mm3/sec) */
        dv;     /* dt 秒間の流出水量(mm3) */
        du;     /* dt 秒間の水位低下量(mm) */
        c;      /* 流出孔の水の漏れやすさ係数 */
        t;      /* 時間 */
        dt;     /* 時間の刻み幅 */
```

∂ u
水位の時間変化は　　-------- = -cu
∂ t

→水位 u の変化速度は、その時の水位に逆比例

u(t+Δ t) - u(t))
これを差分式で表すと、　---------- = -cu
Δ t

整理すると、u(t+Δ t)=u(t)-cu(t) Δ t

```
/*  タンクモデルの計算 */

#include <stdio.h>

void    main(void)
{
        double  u;      /* 水位(mm) */
        double  v;      /* 漏れの速さ(mm3/sec) */
        double  dv;     /* dt 秒間の流出水量(mm3) */
        double  du;     /* dt 秒間の水位低下量(mm) */
        double  c;      /* 流出孔の水の漏れやすさ係数 */
        double  t;      /* 時間 */
        double  dt;     /* 時間の刻み幅 */
        int     tm;

        u=100.0; /* 水位の初期値 */
        t=0.0;          /* 時間(秒) */
        dt=1.0;         /* 時間の刻み幅 */
        c=0.1;          /* 流出孔の水の漏れやすさ係数 */

        for(tm=0;tm<60;tm++) {
```

```
                printf("%6.3lf   %6.3lf¥n",t,u);

                u=u-c*u*dt;

                t+=dt;
        }
}
```

【その２】　タンクの側面にも穴が開いていたら

●-c*u*dt は微小時間 dt において底面から流出する水による水位変化
●側面の高さ h2 にも穴が開いていたらどうなるか。

```
/*   タンクモデルの計算  tank2.c */

#include <stdio.h>

void    main(void)
{
        double  u;      /* 水位(mm)  */
        double  v;      /* 漏れの速さ(mm3/sec)  */
        double  dv;     /* dt 秒間の流出水量(mm3)  */
        double  du;     /* dt 秒間の水位低下量(mm)  */
        double  c1;     /* 底面流出孔の水の漏れやすさ係数  */
        double  c2;     /* 側面の流出孔の水の漏れやすさ係数  */
        double  t;      /* 時間  */
        double  h2;     /* 側面流出孔の高さ  */
        double  dt;     /* 時間の刻み幅  */
        int     tm;

        u=100.0; /* 水位の初期値  */
        t=0.0;          /* 時間(秒)  */
        dt=1.0;         /* 時間の刻み幅  */
        c1=0.1;         /* 底面流出孔の水の漏れやすさ係数初期値  */
        c2=0.1;         /* 側面流出孔の水の漏れやすさ係数初期値  */
        h2=50.0; /* 側面流出孔の底面からの高さ  */

        for(tm=0;tm<60;tm++) {

                printf("%6.3lf   %6.3lf¥n",t,u);

                u=u-c1*u*dt;

/* この if 文に注意  */
                if(u>h2) {
                        u=u-c2*(u-h2)*dt;
                }

                t+=dt;
        }
```

}

【演習課題】　タンクが 2 段になっていたらどうなるか。

【その３】平衡状態のシミュレーション

　長さ 10cm の針金の一方を 10 ℃、もう一方を 0 ℃にしたら、針金の温度分布はどうなるか。ただし、針金は断熱材でくるまれているとする。

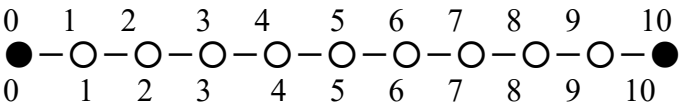
$$\frac{\partial^2 T}{\partial x^2} = 0$$

　差分式で表すと、

$$\frac{T(x-dx) - 2T(x) + T(x+dx)}{dx^2} = 0$$

よって、

$$T(x) = \{T(x-dx) + T(x+dx)\} / 2$$



```
/*　平行状態の計算 */

#include <stdio.h>
#include <math.h>

void    main(void)
{
        double  T[11];          /* 温度 */
        double  Tnew[11];       /* 反復後の温度 */
        double  dx;             /* ｘ方向の刻み幅 */
        double  er;             /* 修正量 */
        int             i;

        for(i=1;i<10;i++) {
                        T[i]=5.0;
        }

        while(1) {

                T[0]=10.0;
                T[10]=0.0;
                for(i=1;i<10;i++) {

                        Tnew[i]=(T[i-1]+T[i+1])*0.5;
                }
```

```
                er=0.0;
                for(i=1;i<10;i++) {
                        er+=fabs(Tnew[i]-T[i]);
                }
                if(er<0.1)              break;

                for(i=1;i<10;i++) {
                        T[i]=Tnew[i];
                }
                printf("%6.3lf¥n",er);
        }

        for(i=0;i<11;i++) {
                printf("%3d     %6.3lf¥n",i,T[i]);
        }

}
```

【その４】時間変化のシミュレーション

$$\frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2}$$

ここで、T：温度、t：時間、K：熱伝導定数

$$\frac{T(x,t+dt)-T(x,t)}{dt} = K * \frac{T(x-dx,t)-2T(x,t)+T(x+dx,t)}{dx^2}$$

よって、

$$T(x,t+dt) = (K*dt/dx^2)\ \{T(x-dx,t)-2T(x,t)+T(x+dx,t)\} + T(x,t)$$

同じ針金を使って、初期温度が０℃で、ある瞬間に左端を 10 ℃に固定した場合の温度の時間変化を求めてみよう。

```
/*   熱伝導の計算  */

#include <stdio.h>
#include <math.h>

void    main(void)
{
        double  T[11];  /* 温度  */
        double  Tnew[11];
        double  dx;     /*  x 方向の刻み幅  */
        int             i;
```

```
        double    K;
        double    tm;
        double    dt;

        dx=1.0;
        K=0.1;
        dt=0.1;

        for(i=0;i<11;i++) {
                        T[i]=0.0;
                        Tnew[i]=0.0;
        }

        tm=0.0;

        while(tm<100.0) {

                T[0]=Tnew[0]=10.0;
                T[10]=Tnew[10]=0.0;

                for(i=1;i<10;i++) {
                        Tnew[i]=(K*dt/dx/dx) * (T[i-1]-2.0*T[i]+T[i+1])+T[i];
                }

                printf("Time:%6.3lf¥n",tm);
                for(i=0;i<11;i++) {

                        printf("x=%4.1lf     T=%8.3lf¥n",(double)i*dx,Tnew[i]);
                }

                for(i=1;i<10;i++) {
                        T[i]=Tnew[i];
                }

                tm+=dt;
        }

}
```
