＜お役立ち情報＞

★数学テスト問題作成には,Writer （Math） が便利!

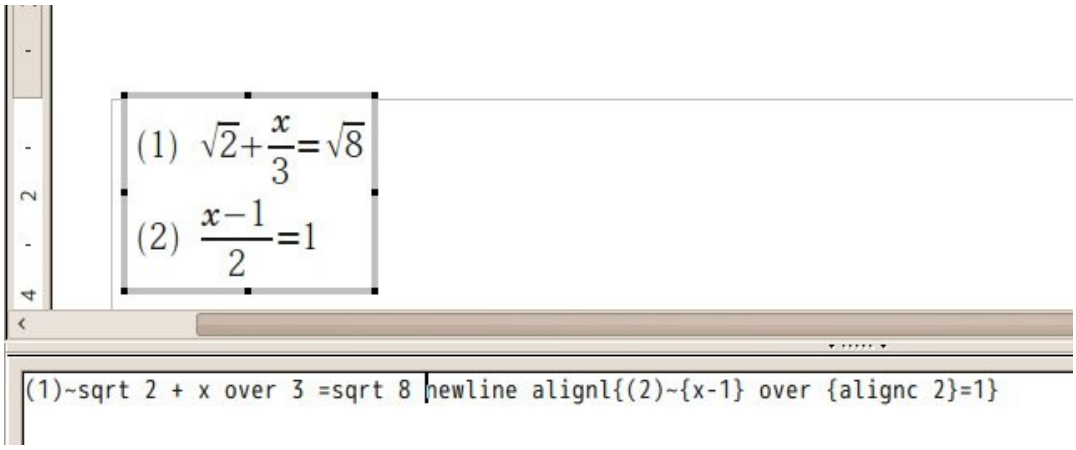
数式を扱うのは TeX が定番と思いますが、案外中学校では知られてませ ん。というのも、中学校ではそんなに難しい数式は必要でなく、せいぜい分数やルートがその形でかければいいわけで、例えばWordについている数式エディ ターでも可能なわけです。

しかし、操作性は非常に悪く、テスト問題を作成するのはとても面倒です。

そこで登場するのが LibreOffice の Writer（Math）ですが、これは便利で、TeX 風に

sqrt 2 みたいに書いたりできます。

分数なら　x over 3 といった具合です。TeXとは少し違いますが、むしろ、こちらの方が便利です。テスト問題作成が劇的に時間短縮できます。



★OS は Ubuntu がおすすめ

Ubuntu とは（Ubuntu Japanese より抜粋）

Ubuntu（ウブントゥ） とは、コミュニティ により開発されているオペレーティングシステムです。ラップトップ、デスクトップ、そしてサーバーに利用することができます。Ubuntu には、家庭・学校・職場で必要とされるワープロやメールソフトから、サーバーソフトウェアやプログラミングツールまで、あらゆるソフトウェアが含まれています。

Ubuntu は現在、そして将来に渡って**無償**で提供されます。ライセンス料を支払う必要はありません。Ubuntu をダウンロードすれば、友達や家族と、あるいは学校やビジネスに、完全に無料で利用できます。

私たちは、**新しいデスクトップおよびサーバーを 6 ヶ月ごとにリリースする**ことを宣言しています。これにより、オープンソースの世界が提供する最新の優れたアプリケーションを常に利用できるようにしています。

Ubunut を新しいバージョンにアップグレードする場合も、常に無償です。

Ubuntu は **1 枚の CD で提供**され、その中に完全に動作する環境が含まれています。さらに、オンラインでソフトウェアを追加することができます。

グラフィカルインストーラにより、**素早く簡単にインストールして使い始めることができます**。標準的なインストールにかかる時間は 25 分未満です。

一度システムをインストールすれば、インターネット、ドローイング、グラフィックス、そしてゲームといったアプリケーションが**すぐに使えるようになります**。

「Ubuntu」の意味

Ubuntu は、アフリカの単語で「他者への思いやり」や「皆があっての私」といった意味を持ちます。Linux ディストリビューションである Ubuntu は、Ubuntu の精神をソフトウェアの世界に届けます。

※ 注：Windows 用ソフトも走ります。（Wine などを利用）

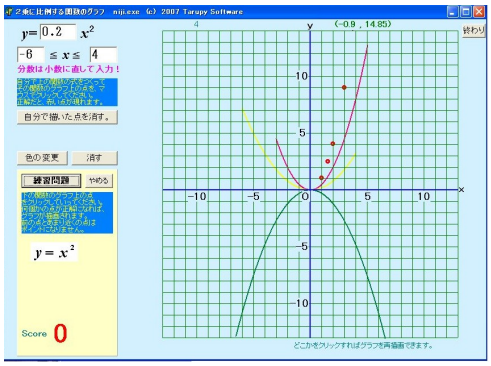
DVD 収録ソフトウェア利用方法

(1) 回転体シミュレーション kaiten.exe を使ってみよう。

おみやげ CD 内の kaiten.exe を実行してください。

(2) 一時関数 Func.exe をやってみてください。
（左下の、黄色い部分の問題、人によって数値は異なります）

おみやげ CD 内の Func.exe を実行してください。



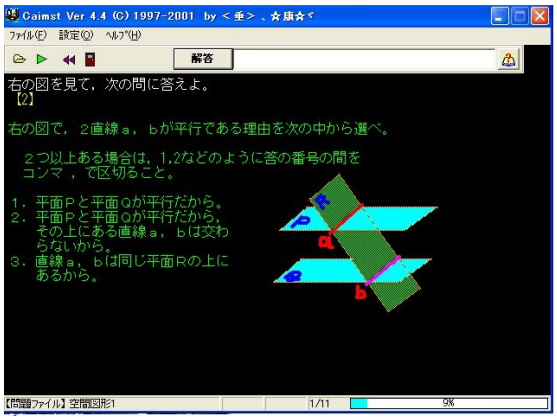
(3) 2 乗に比例する関数、niji.exe で練習問題をやってみましょう。

おみやげ CD 内の niji.exe を実行してください。

(4) Ruby が使える環境なら、円周率の無限表示 pi.rb や
鶴亀算 tsurukame.rb などを動かして見ましょう。
1.DOS 窓（console）を開く
2.CD に移動　　cd "CD の場所(Q:など)"
3.ruby で各スクリプトを実行
例　Q>ruby pi.rb　　・・・・円周率が表示されます。　ctrl+c で止まります。
Q>ruby tsurukame.rb　・・・・鶴亀算

(5) ドリルソフト caimst を立ち上げて、数学のドリルを試してみましょう。また、簡単な問題を自作してみましょう。

1.caimst を立ち上げる。
CD 内の caimst.exe を実行



2.CAIMST コースウェア内の教材を実行

3.簡単なコースウェアをつくってみる

例
3
1+2=
/

・上記のような問題をつくり、テキストファイルとして保存（拡張子は PRB）
・これを CAIMST で呼び出す。

(6) JavaScript で、簡単な練習問題をつくってみましょう。

CD 内の tashizan.html などを参考に
下のページにもあります。

（これを元に書き換えていくといいです。）

JavaScript　<正負の計算>

```
==========================================================
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 <meta http-equiv="Content-Script-Type" content="text/javascript">
 <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
 <title>正負の計算（和）</title>
 <script type="text/javascript">
<!--
function getRandom() {
        return Math.floor(Math.random() * 9+1);
}
// -->
 </script>
</head>
<body bgcolor="#ffffff">
<h2>正負の計算（和）</h2>

<hr>
<script type="text/javascript">
<!--
//質問の数
var n = 5;
var a, b;
var correct = 0;
 //出題する
for (var i = 1; i < n+1; i++) {
        a = getRandom()-5;
        b = getRandom()-5;

    while(a==0){
        a = getRandom()-5;
    }
```

```
        if(a>0 && b>0)
                msg = a+"+"+b+"=";
        if(a>0 && b<0)
                msg = a+"+("+b+")=";
        if(a<0 && b>0)
                msg = "("+a+")+"+b+"=";
        if(a<0 && b<0)
                msg = "("+a+")+"+"("+b+")=";

           ans = prompt(msg,"");

           if (ans == a+b) {
        document.write('('+i+') 正解!<br>')
                   correct++;
           }
        else{
        document.write('('+i+') 残念!<br>')
        }
}

//正解率を表示する
document.write('<HR>');
document.write("<h2>正解率:",
                   Math.round(correct/n *100),
                   "%</h2>");
document.write("<p>", n, "問中", correct, "問正解", "</p>");
// --></script><a
 href="seifu-wa.html">もう一度やる</a><br>
<a href="JHSmath.html">終わる</a><br>
</body>
</html>
```

<方程式文章題>

```
==========================================================================
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Script-Type" content="text/javascript">
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
  <title>ame1</title>

  <script type="text/javascript">

<!--
function getRandom() {
        return Math.floor(Math.random() *
9+1);
}
function getRandom2() {
        return Math.floor(Math.random() *
4+2);
}

// -->
  </script>
</head>
<body bgcolor="#ffffff">
<h2><span style="color: rgb(0, 153, 0);">一次方程式</span> <span
 style="color: rgb(0, 0, 0);">&lt;文章題２&gt;</span></h2>
<hr>
<script type="text/javascript">
<!--
//質問の数
var n = 3;
var a, b,eq,c,d,e;
var correct = 0;

//出題する
ans="###";
sol="???";
for (var i = 1; i < n+1; i++) {

        a=getRandom();
        b = getRandom();
        c = getRandom();
        d = getRandom();

        while(a<=b || c<=a ||(b+d)%(c-a)!=0){
         a=getRandom();
         b=getRandom();
         c=getRandom();
```

```
         d=getRandom();
        }

        seikai=(b+d)/(c-a);

        eq=a+"x+"+b+"="+c+"x-"+d;


         document.write('('+i+') 子供にアメを '+a+'個ずつ配ると '+b+'個あまり、'+c+'個ずつ配ろうとすると、'+d+'個たりない。<br>子供は何人いますか。<BR><BR>');

          ans = prompt("子供の人数を x 人として、方程式を作りなさい。","");

          if(ans == eq) {
            sol=prompt("OK! 次に  "+eq+"  を解きなさい。","x=");
              if (sol=="x="+seikai){
                document.write('<span style="color: rgb(200, 200, 0);">正解! よくできました。</span><br><br>');
                   correct++;
              }
              else{
              document.write('<span style="color: rgb(0, 153, 0);">残念!  でも、式を立てるところまではあってたねえ。</span><br><br>');
              alert('正解は 式は '+eq+'  解は '+seikai+' です。頑張りましょう!');
              }
          }
          else{
             sol=prompt("残念！ 正解は  "+eq+"  です。ではこれを解きなさい。","x=");
                if (sol=="x="+seikai){
                  document.write('<span style="color: rgb(0, 153, 0);">正解! でも、式を立てられるようにしよう！</span><br><br>');
                  alert('正解は 式は '+eq+'  解は '+seikai+' です。頑張りましょう!');
                    correct++;
                }
                else{
                document.write('<span style="color: rgb(0, 128, 255);">残念!  基本からしっかりやろう！</span><br><br>');
                alert('正解は 式は '+eq+'  解は '+seikai+' です。頑張りましょう!');
                }
          }
}

//正解率を表示する
document.write('<HR>');
document.write("<h2>正解率:",Math.round(correct/n *100),"</h2>");
document.write("<p>", n, "問中", correct, "問正解", "</p>");

// -->
</script><a href="ame1.html">もう一度やる</a><br>
<a href="eqquiz1.html">終わる</a><br>
</body>
</html>
```

==========

個別学習シート IWSheet　利用方法

==========

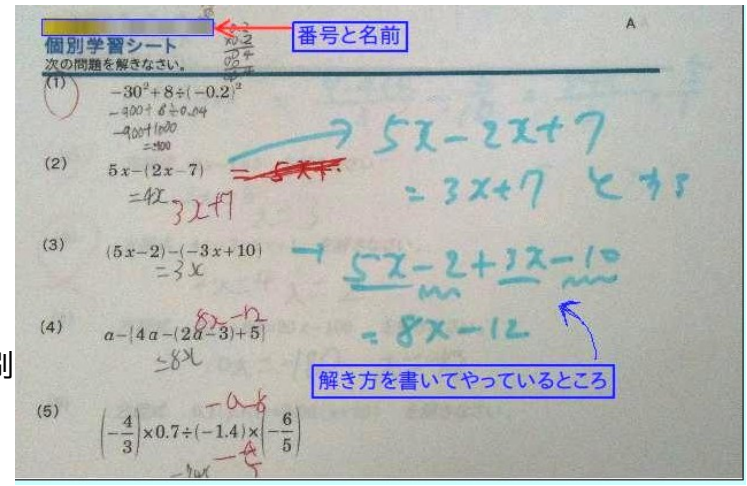
生徒の進度に合わせて個別学習シートを作成する ツールです。

印刷環境があれば、個別学習シート(IWSheet) を試してみましょう。

CD 内の　IWS_usage.doc 参照

下のページにもあります。

<概要>



A4縦の用紙に、上半分に問題、下半分に解答を、生徒ごとにそれぞれの進度に応じて個別の問題を５題ずつ 印刷できます。各生徒の名前や座席の列も同時に印刷されまで、教室で配布するのに便利になってます。

１回目は全員同じ問題でスタートしますが、２回目以降は、各自が前回に間違った問題を含めて計 ５題出題されますので、生徒によって個別の問題を出題することになります。(自己採点です)

また、数式の印刷もできるので、数学での利用に便利になってます。

1000	ダミー	列	成績		
1001	安部マリア	A			
1002	磯野カツオ	B			
1003	垂井　剛	B			
:	:	:			

OpenOffice(または、lLibreOffice)の calc 用ワークブックです。
Calc 専用の calc basic というマクロ言語を利用してますので、Excel では動作しません。

<準備>

[準備物] パソコン、OpenOffice または、LibreOffice、プリンタ(大量に高速で印刷できる、レーザープリンタがおすすめ)、A4 用紙、IWSheet.ods(インターネットで検索すれば出てきます。FCAI,Vector などに登録されています。)

[準備]　1. calc で IWSheet を開き、DATA シートに生徒番号、生徒名、座席列名、

を入力する。()

2.Questシートに問題を作成する。
A列 問題番号　A1は全問題数(自動カウント)
B列 問題　数式はcalc形式で入力する。
(例 分数 2over3. 平方根 sqrt(2)
C列 解答
※問題も解答も、印刷時に用紙からはみ出さない範囲で入力する。

Questシートの例

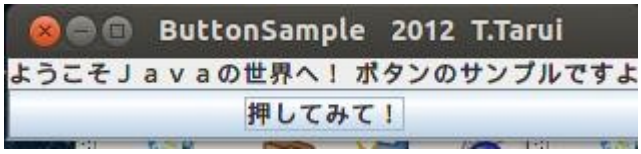
No	問題	解答
1	2a-b+a-3b	-a-4b
2	方程式~~x over 2+1 =1 over 3~~を解きなさい。	x=-4 over 3
3		
:		

3.checkシートを開いて、正誤入力をする。
・回収したシートを見て、正解の問題に１以上の数値を入力する。
・全員入力が終われば、データ埋め込みボタンを押す。
・DATAシートに、成績が追加されたことを確認して、checkシートのデータを消去する。
・適当な名前をつけてデータを保存する。

4.Print Outシートを開く
・学年、組、印刷する生徒番号の範囲を指定する。
注:この時の生徒番号は、座席列でソートされた順に上から数えた番号になる。
・印刷ボタンを押して印刷実行。
・カッターで、上下半分に切断する。

入門 Java アプリ

[1] ボタンとイベント処理



ボタンを押すとラベルが変わるクラスです。　ButtonSample.java

```
import java.awt.*;     //awt は Abstract Window Toolkit といって、Java の GUI をサポートするパッケージ。これをインポート
import java.awt.event.*;  //ボタンを押したらどうする、などのイベント処理のパッケージ
import javax.swing.*;     //awt を拡張してもっと便利なメソッドなどを加えたパッケージ、かな？
import javax.swing.JButton;  //swing で設定されているボタン
import javax.swing.JFrame;   //swing で設定されているフレーム（Window のこと）

public class ButtonSample extends JPanel implements ActionListener{
  JLabel label;     //ラベル用変数
  JButton btn;      //ボタン用変数
  public ButtonSample(){
    label=new JLabel("ようこそJavaの世界へ！ ボタンのサンプルですよ");     // 元のラベル
    btn=new JButton("押してみて!");  //ボタンの生成
    btn.addActionListener(this);    //イベントリスナー設定
    setLayout(new BorderLayout());  //ボタンやラベルの配置を Border レイアウトに
    add(label,BorderLayout.NORTH);    // ラベルを上に
    add(btn,BorderLayout.CENTER);     // ボタンを真ん中に
  }

  public void actionPerformed(ActionEvent e){     //イベント処理のメソッド
    if(e.getSource()==btn){           //ボタンが押されたら
       label.setText("＼( ^ o ^ )／");
    }
  }

  // 以下 main メソッド (お決まりの形)

   public static void main(String[] args) {
      JFrame frame = new JFrame("ButtonSample   2012  T.Tarui");  //ウィンドウタイトル
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);    //×で窓を閉じられる
```

ように

```
        ButtonSample bs = new ButtonSample();   //ButtonSample のインスタンス bs の生成
        frame.add(bs,BorderLayout.CENTER);      //bs を borderLayout 形式のレイアウトでこの frame のセンターに配置
        frame.pack();                           // frame をフィットさせる？　よく分かりません(^^);
        frame.setVisible(true);                 //　この frame を見えるようにする
    }

}
```

これを、「1」の3で説明したように、コンパイルして実行すると 上のような Window が現れて、ボタンを押すとラベルが変わります。

[2] ボタンを押すと直線を描画

ボ タンを押すと直線を描くクラスです

BtnAndLine.java　　→　コンパイルすると　→　　BtnAndLine.class

```
import java.awt.*;      //awt は Abstract Window Toolkit といって、Java の GUI をサポートするパッケージ。これをインポート
import java.awt.event.*;//ボタンを押したらどうする、などのイベント処理のパッケージ
import javax.swing.*;   //awt を拡張してもっと便利なメソッドなどを加えたパッケージ、かな？
public class BtnAndLine extends JPanel implements ActionListener {
    Ar area;            // area という変数を、Ar というクラス（Frame 用のクラスで下で定義している）のインスタンスとする
```

```
    JButton btn,btn2;    // ボタンを2つつくる

    int z;       //変数 z を宣言・・・これが1なら直線を引く。0なら消す。
    public BtnAndLine(){
        JPanel p = new JPanel(); //ボタン用パネル
        btn = new JButton("直線");btn2=new JButton("消す");  //ボタン2つ
        btn.addActionListener(this);btn2.addActionListener(this);  //ボタン
を押したらそれがわかるようにする。
        p.setBackground(Color.cyan);        //p という名前をつけたパネル
（Frame の下の部分でボタンを置く）を水色に
        p.add(btn);p.add(btn2);   //パネル p に2つのボタンを置く
        area = new Ar();
        setLayout(new BoxLayout(this,BoxLayout.Y_AXIS));  //縦に配置
        add(area);   //Frame の上部に area というパネルを置く
        add(p);     // p という y パネルを置く

    }

     //Action(ボタンを押すなど)が行われた時の処理
    public void actionPerformed(ActionEvent e) {
      if (e.getSource()==btn){    //ボタンを押したという情報が入ったら
        z=1;                        //z に 1 を代入
        setBackground(Color.blue);  //グラフィックの初期値　何色でもよいが、こ
れがないとなぜか線蛾引けない???
      }                           //これで悩んでたが、どうもグラフィックスの初期値のよう
なものが必要みたい。
      else if (e.getSource()==btn2){
         z=0;                     // z に 0 を代入
         setBackground(Color.green); //グラフィックの初期値 これがないと線が
消せない。上のと違う色でないとだめのようだ??
        }
    }

    class Ar extends JPanel {   //パネル　Ar の定義
        Ar(){
            setBackground(Color.white);      //背景色
```

```
            setPreferredSize(new Dimension(300,200));   //Frame のサイズ
        }
        public void paintComponent(Graphics g){  // paint コンポーネントを準備（グラフィックスを使うのに必要みたい）
            super.paintComponent(g);        // paintComponent を継承の上位クラスとする、かな？
            if (z==1) {                     //もし z=1 なら線を引く
               g.drawLine(50,50,200,150);
            }
        }
    }


    // main クラス
    public static void main(String[] args) {
      JFrame.setDefaultLookAndFeelDecorated(true);
      JFrame frame = new JFrame("BtnAndLine    2012 T.Tarui");
//Wiodow をつくる
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  //×ボタンで W 終われるようにする。
      BtnAndLine e = new BtnAndLine();                  //上のクラス BtnAndLine を生成
      frame.add(e,BorderLayout.CENTER);                 //frame を BorderLayout のレイアウトにする
      frame.pack();
      frame.setVisible(true);
    }

}
```

コマンドコンソール（ターミナル）でコンパイル、実 行する方法　　※ 統合開発環境 eclipse などを利用する方法もあります。

コンパイル　→　javac BtnAndLine.java

実行　　→　java BtnAndLine

[3] キーボード から値を入力

square.java　入力した数値を2乗するだけのクラスです。square.class

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class square extends JPanel implements ActionListener {
 JTextField nyuuryoku;    //入力フィールド
 JLabel nijou;          //ラベル
 double x;             // 入力値
 public square(){
    setBackground(Color.white);
    nyuuryoku=new JTextField(10);
    nijou=new JLabel();
    nyuuryoku.addActionListener(this);
    setLayout(new GridLayout(2,2));
    add(new JLabel("x=?"));add(nyuuryoku);
    add (new JLabel("x^の2乗は ")); add(nijou);
 }

 public void actionPerformed(ActionEvent e){
    try{
       x=Double.parseDouble(nyuuryoku.getText());
       }
    catch(NumberFormatException error){x=0.0;}
    keisan();
 }
```

```
  void keisan(){
     double kekka = x*x;
     nijou.setText(String.valueOf(kekka));
 }
    public static void main(String[] args) {
       JFrame frame = new JFrame("2乗を求める");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       square h = new square();
       frame.add(h,BorderLayout.CENTER);
       frame.pack();
       frame.setVisible(true);

    }

}
```

Ruby

■ Rubyの紹介

Rubyとは、簡単に言えば、純粋なオブジェクト指向のスクリプト言語の名前です。つまり、C++みたいにコンパイラではなくて、いわゆるインタープリタなわけです。Rubyは他の多くの言語と違い、日本人がつくりました。UNIX上で開発されましたが、今では、MS Windows等のプラットフォーム用もあります。詳しくはご本家のRuby のWeb　Pageをご覧下さい。

私は数学の教師ですが、授業でたとえば平方根や円周率の近似値を扱う時に、画面に多くの桁を表示させたいとよく思って ました。そういう場合は、これまではU−Basic等を利用していました。U−Basicはもちろん非常に優れた言語ではありますが、Rubyは言うまで もなく構造化言語であるという点、また、上記のようにオブジェクト指向であるという点、単純で非常に馴染みやすい文法等、今後の事を考えるといろんな面で 利用価値が高いと思います。また、なによりもうれしいのは、U−Basicもそうではありますが、フリーソフトウェアであるという点です。インタープリ ターでもあるし、プログラミング初心者にとっても、最適の入門用言語にもなり得ます。

また、Window（XWindow や MS-Windows）を使う場合は、幸い Tk というツール（Tcl/tk の Tk）がありますので、これを利用できます（Ruby/Tk)。最近は Ruby/Tk を扱うページも随分増えましたので、助かります。このページでもいくらかサンプルを紹介したいと思います。（他のページにあるものを少し変えただけですが）

■ 円周率の計算

以下は、Ruby のパッケージの sample に入っていた円周率の計算です。

```
k, a, b, a1, b1 = 2, 4, 1, 12, 4
while TRUE
  # Next approximation
  p, q, k = k*k, 2*k+1, k+1
  a, b, a1, b1 = a1, b1, p*a+q*a1, p*b+q*b1
  # Print common digits
  d = a / b
  d1 = a1 / b1
  while d == d1
    print d
    $stdout.flush
    a, a1 = 10*(a%b), 10*(a1%b1)
    d, d1 = a/b, a1/b1
  end
end
```

たった、これだけのプログラムで、

3141592653589793238462643383279502884197169399375105820974944
5923078164062862089986280348253421170679821480865132823066470
9384460955058223172535940812848111745028410270193852110555964

```
462294895493038196442881097566593344612847564823378678316527 1
201909145648566923460348610454326648213393607260249141273724 5
870066063155881748815209209628292540917153643678925903600113 3
053054882046652138414695194151160943305727036575959195309218 6
117381932611793105118548074462379962749567351885752724891227 9
381830119491298336733624406566430860213949463952247371907021 7
........
```

このように、永遠に計算を続けてくれます。

(参考)----------

単に

```
print 4*Math..atan2(1,1), "¥n"
```

とすれば、

```
3.141592654
```

と表示してくれます。（※　"¥n" は改行の意味）

ちなみに　atan2(x,y) というのは　Ruby で　arctan(x/y) を求める事です。

π／4=arctan(1)=1−1／3+1／5−1／7・・・

なんてなるんですねえ。これは、ちょっと微積の知識と二項定理の知識があれば
なんとか分かると思います。これを使ってるわけです。

また、Ruby に組み込まれている　PI　という定数を使って、

```
include Math
print PI,"¥n"
```

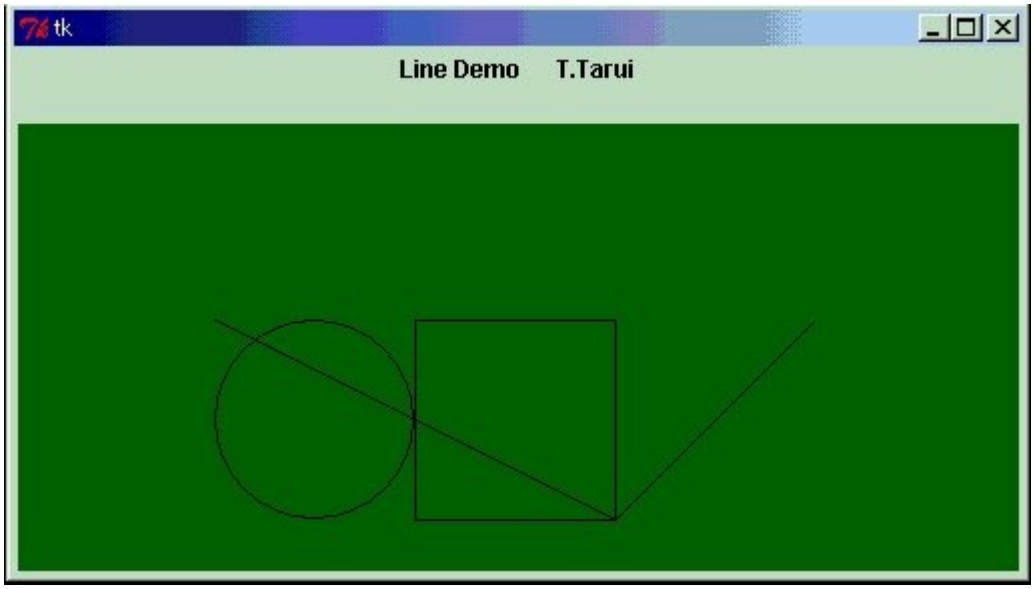
でも同じ結果になります。これが一番簡単ですね。(^ ^)

Ruby/Tk　（Ruby でグラフィックス）

1. LineDemo

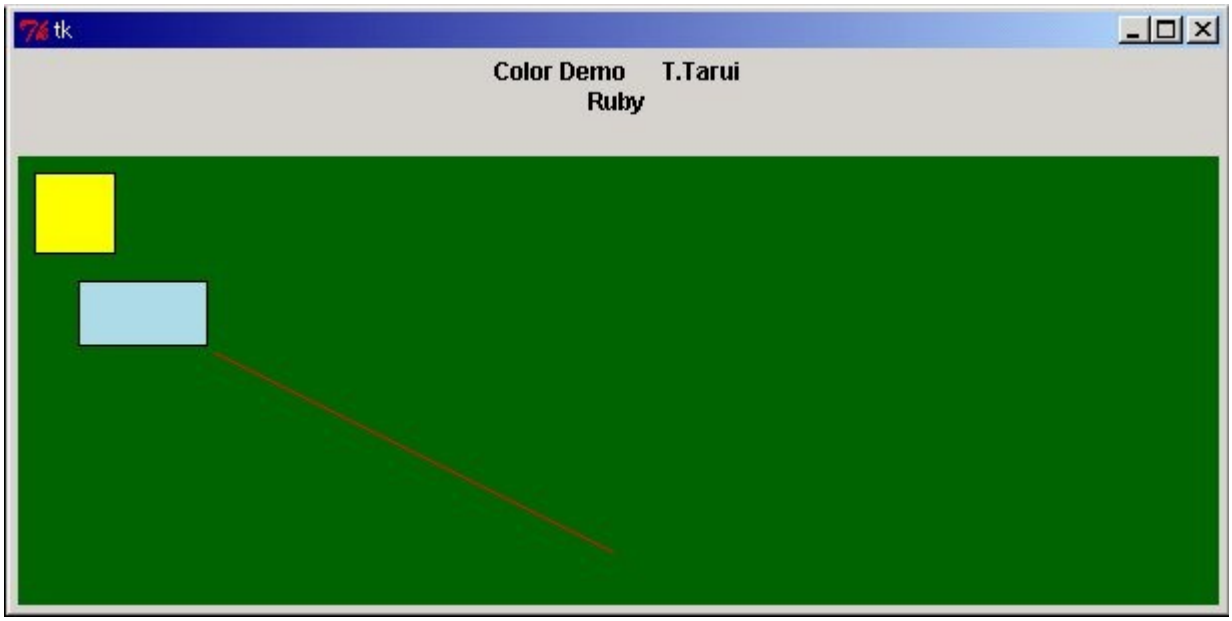
```
require "tkclass"

#テキストを配置

TkLabel.new{
  text "Line Demo      T.Tarui¥n"
  pack
}


$c = Canvas.new                    #キャンバスの初期化
$c.pack
```

```
$c.width 500
$c.background "darkgreen"

Line.new($c,100,100,300,200)          # 直線
Line.new($c,300,200,400,100)          #   直線
Oval.new($c,100,100,200,200)          #   円(楕円)
Rectangle.new($c,200,100,300,200)   #   正方形(長方形)
Tk.mainloop
```



2.Color Demo

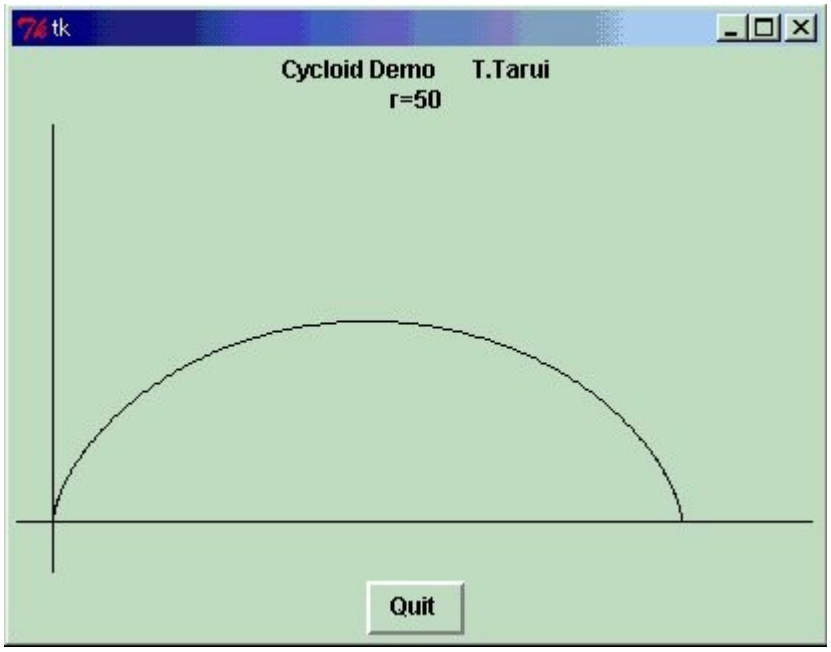
```
require "tkclass"

TkLabel.new{
  text "Color Demo      T.Tarui¥nRuby¥n"
  pack
}


canvas = TkCanvas.new

canvas.background "darkgreen"
canvas.width 600

TkcRectangle.new(canvas, '1c', '2c', '3c', '3c','outline' => 'black', 'fill' => 'LightBlue')

TkcRectangle.new(canvas,10,10,50,50,'outline' => 'black', 'fill' => 'Yellow')
TkcLine.new(canvas,100,100,300,200,'fill' => 'red')

canvas.pack
Tk.mainloop
```



3.Cycloid Demo

```
##########################################
# Cycloid Demo      Dec.10 2000 T.Tarui #
##########################################

require "tkclass"

#   ラベル

TkLabel.new {
  text "Cycloid Demo      T.Tarui¥n"+"r=50"
  pack
}

#   サイクロイド描画

r=50

$c = Canvas.new
$c.pack

$c.width 400

Line.new($c,0,200,400,200)
Line.new($c,20,0,20,400)

x0=0
y0=0

for i in 1..360

 x=r*(3.14*i/180-Math.sin(3.14*i/180))
 y=r*(1-Math.cos(3.14*i/180))
```

```
   Line.new($c,x0+20,200-y0,x+20,200-y);
  x0=x
  y0=y
end
#   Quitボタン
TkButton.new {
 text 'Quit'
  command proc{
    exit
  }
 pack('side'=>'bottom')
}
Tk.mainloop
```



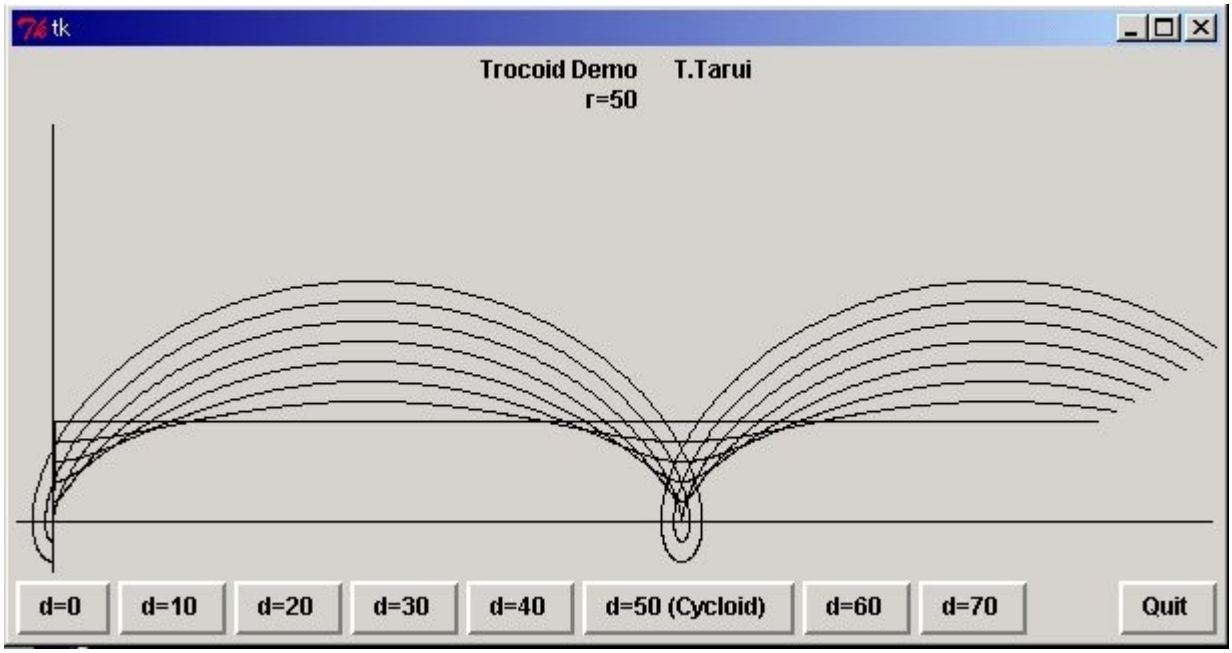
4.Trocoid Demo

```
##########################################
# Trocoid Demo     Dec.10 2000 T.Tarui #
##########################################
require "tkclass"
#   ラベル
TkLabel.new {
   text "Trocoid Demo      T.Tarui¥n"+"r=50 "
   pack
```

```
}
# 描画領域作成
$r=50
$c = Canvas.new
$c.pack
$c.width 600
Line.new($c,0,200,600,200)
Line.new($c,20,0,20,500)

#　ボタン
TkButton.new {
 text 'd=0'
  command proc{
    $d=0
    prot
  }
 pack('side'=>'left')
}
TkButton.new {
 text 'd=10'
  command proc{
    $d=10
    prot
  }
 pack('side'=>'left')
}

TkButton.new {
 text 'd=20'
  command proc{
    $d=20
    prot
 }
 pack('side'=>'left')
}

TkButton.new {
 text 'd=30'
```

```
  command proc{
    $d=30
    prot
  }
 pack('side'=>'left')
}
TkButton.new {
 text 'd=40'
  command proc{
    $d=40
    prot
  }
 pack('side'=>'left')
}
TkButton.new {
 text 'd=50 (Cycloid)'
  command proc{
    $d=50
    prot
  }
 pack('side'=>'left')
}
TkButton.new {
 text 'd=60'
  command proc{
    $d=60
    prot
  }
 pack('side'=>'left')
}
TkButton.new {
 text 'd=70'
  command proc{
    $d=70
    prot
  }
 pack('side'=>'left')
}
#   Quitボタン
TkButton.new {
```

```
 text 'Quit'
  command proc{
    exit
  }
 pack('side'=>'right')
}

#   描画

def prot


x0=0
y0=0

for i in 1..600

  x=$r*3.14*i/180-$d*Math.sin(3.14*i/180)
  y=$r*1-$d*Math.cos(3.14*i/180)

   Line.new($c,x0+20,200-y0,x+20,200-y);

  x0=x
  y0=y

end

end


Tk.mainloop
```



==========

ショートプログラムでいろいろな言語入門

==========

■ 分数100桁小数表現

まず、FCAI の広瀬さんが Basic でそのアルゴリズムを示してくれました。

1.Basic

```
------------------------------------
  10   dim Q(20)
  20   input "a=";A
  30   input "b=";B
  40   for I=0 to 20
  50   Q(I)=int(A/B)
  60   R=A-Q(I)*B
  70   A=R*100000
  80   next I
  90   for J=0 to 20
 100    print Q(J)
 120    next J
 130    end
------------------------------------
```

5桁ずつまとめてるのがすごいです。
UBasic なら1行で書けるそうです。

2 C

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

int a,b,r,i,q[20];

int main(void)
{
a=1;
while(a!=0){
 printf("a=");
 scanf("%d",&a);
 printf("b=");
 scanf("%d",&b);

 for (i=0;i<=20;i++){
   q[i]=a/b;
   r=a-q[i]*b;
   a=r*100000;
```

```
 }
  printf("%d.",q[0]);
  for (i=1;i<=20;i++){
  printf("%d",q[i]);
  }
  printf("¥n");
}
}
```

広瀬さんが Logo 坊で書いてくれました。
Logo 坊 は 兼宗さん（大阪電通大）による Logo です。MS Windws 用と MSDOS 用があります。
（現在は「ドリトル」という言語に発展しています。）

3. LogoB

手順は　分数小数
　　絵を元へ
　　（書け　”分数を小数にする。"10 "/ "7 "の形で入力。）
　　変数は　"X　読んだリスト
　　変数は　"A　（最初　:X）
　　変数は　"B　（最後　:X）
　　　　続けて書け　整数　(:A / :B)
　　　　続けて書け　".
　　　　変数は　"R（余り　:A :B）
　　　　変数は　"A（:R * 100000）
　　繰り返せ　100「
　　　　続けて書け　整数　(:A / :B)
　　　　変数は　"R（余り　:A :B）
　　　　変数は　"A（:R * 100000）
　　　　」
　　書け　”
終り

4.VB

次は広瀬さんによる、Visual Basic 版(MSWindows 用)です。

```
Private Sub Command1_Click()
    a = Text1.Text
    pos = InStr(a, "/")
    p = Mid(a, 1, pos - 1)
    q = Mid(a, pos + 1)
        Print p ¥ q; "."
        p = (p Mod q) * 100000
    For i = 1 To 100
        Print p ¥ q
        p = (p Mod q) * 100000
    Next i
End Sub

Private Sub Command2_Click()
    End
End Sub
```

次は私が Linux で Perl でやってみました。
コマンドライン引数を利用してます。

5.Perl

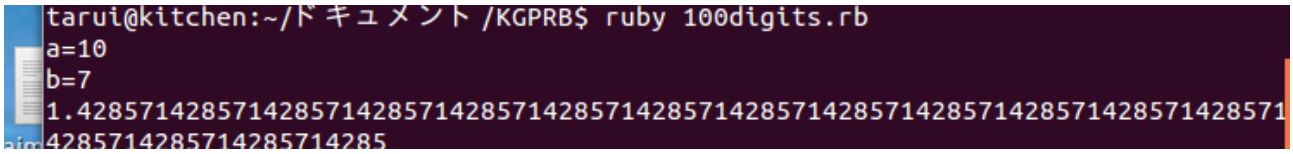
```
#!/usr/bin/perl

$a=$ARGV[0];
$b=$ARGV[1];


for ($i = 0;$i <= 20;$i++){
    $q[$i]=int($a / $b);
    $r=$a - $q[$i]*$b;
    $a=$r*100000;
}


print "$q[0].";
for ($i = 1;$i <= 20;$i++){
    print "$q[$i]";
}
print "¥n";
```

---

6. tcl

```
#!/bin/sh
# the next line restarts using wish ¥
exec wish "$0" "$@"

set a [lindex $argv 0]
set b [lindex $argv 1]

for {set i 0} {$i <= 20} {incr i} {
set q($i) [expr $a / $b]
set r [expr $a - $q($i)*$b]
set a [expr $r*100000]
}

puts -nonewline $q(0)
set dot .
puts -nonewline $dot

for {set i 1} {$i <= 19} {incr i} {
puts -nonewline $q($i)
}
puts $q(20)
exit
```

7.Ruby




```
#!/usr/local/bin/ruby

q = Array.new(0)
a=1
while a!=0
   print "a="
   a=readline().to_i

if a!=0
      print "b="
      b=readline().to_i

 for i in 0..20
   q[i]=a / b
```

```
    r=a-q[i]*b
    a=r*100000
  end
 print q[0],"."
 for i in 1..20
   print q[i]
 end
 print "¥n"
end
 print "¥n"
end
```

8.Lisp

次は、高橋さんによる Emacs Lisp 版です。

(1) 再帰による方法

```
(defun bunsu (a b n)
  (setq sho (int-to-string (/ a b)))
  (setq amari (* (mod a b) 100000))
   (cond  ((equal n 0)  (bunsu amari b (+ n 1)))
        ((<  n  20) (concat  sho (bunsu amari b (+ n 1))))
        ((=  n  20) nil)))
```

この文末で、C-x,C-e として、関数 bunsu　をインストール。

(bunsu 22 7 0)の文末で C-x,C-e として評価すると、結果が下に出ます。

(2) while を使った繰り返し

```
(defun bunsu3  (a b)
 ; (interactive "P")
  (setq a (* (mod a b) 100000))
  (setq ans ())
  (setq n  0)
 (while (< n 21)
   (setq sho (int-to-string (/ a b)))
   (setq ans (concat ans sho))
   (setq a (* (mod a  b) 100000))
    (setq n  (1+ n)))
  (message "Ans = %s"  ans))
```

■ 累乗

1.basic

```
 10   '累乗
  20   input "a=";A
  30   input "b=";B
  40   X=1
  50   while B<>0
  60   if B@2=1 then X=X*A
  70   B=B¥2
  80   A=A*A
  90   wend
 100    print "a^b=";X
 110    end
```

2.Perl

```
#!/usr/bin/perl
$a=$ARGV[0];
$b=$ARGV[1];
$c=$a;
$d=$b;
$x=1;
while ( $b !=0){
  if ($b%2==1) {$x=$x*$a};
  $b=$b/2;
  $a=$a*$a;
}
print "$c^$d=$x¥n"
```

3.Ruby

```
tarui@kitchen:~/ドキュメント/KGPRB$ ruby power.rb
a=2
b=10
2^10=1024
```

```
#!/usr/local/bin/ruby
print "a="
a=readline().to_i
c=a
print "b="
b=readline().to_i
d=b

x=1

while b!=0
  if b%2==1
     x=x*a
  end
  b=b/2
  a=a*a
end
print c
print "^"
print d
print "="
print x
print "¥n"
```

下のほうの、print の繰り返しは不細工ですので、printf を使うほうがすっきりします。

参考　http://www.eonet.ne.jp/~tarcom/

H24.11.15(木)　　関西学院大学理工学部　数学科教育法(C)

ゲストスピーカー　神戸市立押部谷中学校　垂井　剛