# 「情報処理」1年文I/IIクラス11-12　＃9

久野　靖*

1995.12.11

## 0　本日の目標

さて「プログラミングが嫌いな人」に朗報ですが、今回の内容でプログラミングの「必修部分」を終ります。というのは、冬休みレポートの課題をこなすのには今回の内容までででも可能だからです。冬休みレポートというのは悪い知らせだという人も当然いるでしょうが、今回 (それもちょっとだけです!) を除いては冬休みレポートの〆切 (1/29 の授業開始時刻までとします) まで「次回までの課題」を出しませんのでその代わりだと思ってやってください。

冬休みレポートの結果は成績づけにおいて重視されますので、必ず出すようにお願いします。提出のない場合は単位を放棄したものとみなします。一方、レポートの採点に当たってはプログラムそのものよりレポートがちゃんと書けているかどうかを重視しますので、プログラムは「一応」書ければ大丈夫です。それよりレポートの構成をしっかりして日本語 (英語で書きたい人は英語) をまともに書いてください。

次回以降は、当日課題 (と 1/29 以降の次回までの課題) はプログラミングものとそれ以外のものの選択可能にしますので、資料や説明のプログラミングに関する部分は無視しても構いません (が、プログラミングものではできるだけレポートをやる上でヒントになる内容をやります)。

では本日の内容は次の通り。

- 「配列」について理解する。
- プログラムによるお絵描きについて理解する。

## 1　前回の問題の解説/＊

まず 2 分探索による求根から。図 1 に PAD を再掲しておく。そして、

*筑波大学大学院経営システム科学専攻

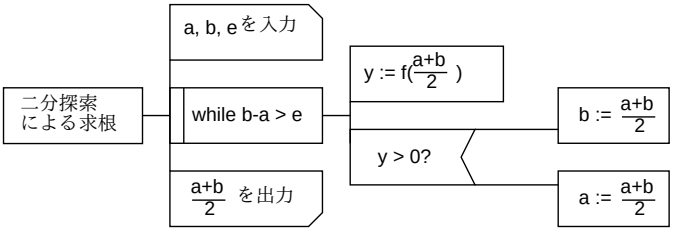


図 1: 根の 2 分探索の PAD

Pascal は次の通り。PAD には $x$ という変数はないが、Pascal にした時は一旦 $\frac{a+b}{2}$ を $x$ とおいて、それを使う方が見やすいですね。

```
program sam9a(input, output);
var a, b, e, y, x: real;
begin
  write('a = '); readln(a);
  write('b = '); readln(b);
  write('e = '); readln(e);
  while b-a > e do begin
    x := 0.5 * (a + b);
    y := x * x - 2.0;  ←※
    if y > 0 then b := x else a := x
  end;
  writeln('root = ', 0.5 * (a + b):8:5)
end.
```

もちろん、「3 の立方根」の場合は※のところを「`x * x * x - 3.0`」とすればいいわけだ。実行例も示しておこう。

```
% a.out
a = 0
b = 2
e = 0.000001
root =  1.41421
```

次は「$N$ 個の数の合計に加えて平均も求める」だが、平均を求めるためには何個のデータを入力したかも数える必要がある。

```
program sam9b(input, output);
var x, sum, ave : real;
    n: integer;
```

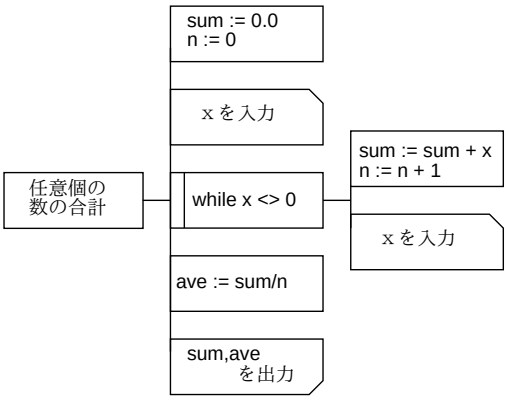


図 2: $N$ 個の数の合計と平均

```
begin
  sum := 0.0; n := 0;
  write('x = '); readln(x);
  while x <> 0.0 do begin
    sum := sum + x; n := n + 1;
    write('x = '); readln(x)
  end;
  ave := sum / n;
  writeln('sum = ', sum:8:3, ' ave = ', ave:8:3)
end.
```

次に 2010 年までの複利計算だが、これは while の条件を変えただけ。

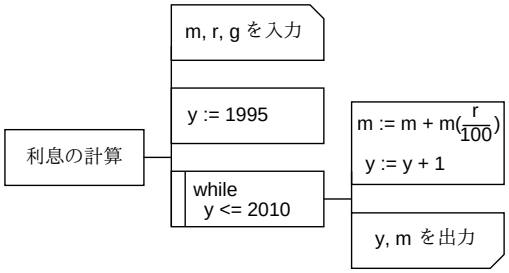


図 3: 2010 年までの複利計算

```
program sam9c(input, output);
var y: integer;
    m, r, g: real;
```

```
begin
  write('money amount  = '); readln(m);
  write('interest rate = '); readln(r);
  write('goal amount =   '); readln(g);
  y := 1995;
  while y <= 2010 do begin
    m := m + (m*r/100.0); y := y + 1;
    writeln('year ', y:4, ' amount = ', m:8:4)
  end
end.
```

2010 年も印刷したいのだから、while の条件が「<=」になることに注意。次の「1 から $N$ まで出力」はさすがに PAD は略してプログラムだけ示す。まず while 版から。

```
program sam9d(input, output);
var n, i: integer;
begin
  write('n = '); readln(n);
  i := 1;
  while i <= n do begin
    writeln(i:4); i := i + 1
  end
end.
```

やはり for を使った方がずっと短い。

```
program sam9e(input, output);
var n, i: integer;
begin
  write('n = '); readln(n);
  for i := 1 to n do writeln(i:4)
end.
```

演習 6 の回答例は紙面の都合から次回に回します。代わりに、最大公約数を Pascal に直したものを掲げておこう。

```
program sam9e(input, output);
var x, y, z: integer;
begin
  write('x = '); readln(x);
  write('y = '); readln(y);
```

```
  while x <> y do begin
    writeln(x:0, ’と’, y:0, ’の最大公約数は’);  ←※
    if x > y then begin z := x; x := y; y := z end;
    y := y - x
  end;
  writeln(x:0, ’だよん。’)
end.
```

※のところはせっかくだからどう計算が進んでいくのか見るために追加してみた。実行例も見よう。

```
% a.out
x = 24
y = 18
24 と 18 の最大公約数は
18 と 6 の最大公約数は
6 と 12 の最大公約数は
6 だよん。
%
```

## 2　配列/☆

前回には、ループを使って一般に $N$ 個のデータを次々に読み込んで処理したり、生成して出力することを学んだ。しかし、プログラムの中で扱っているデータは「現在高」とか「合計」など単一の数値で、それが 1 つの変数に入っているだけだった。それで済むのならそれでもよいが、計算機の中に $N$ 個のデータを蓄えて扱いたい場合もある。例えば、次の例題を考えてみよう。

a. $N$ 個のデータ (0 が来たらおしまい) を読み込み、読み込んだのとは逆の順に出力せよ。$N$ は最大でも 200 とする。

ここで $N$ がたとえば 4 個という風に決まっていれば話は簡単で、次のようなあほみたいなプログラムを書けばよい (PAD は略した)。

```
program sam11a(input, output);
var x1, x2, x3, x4: real;
begin
  write(’x1 = ’); readln(x1);
  write(’x2 = ’); readln(x2);
  write(’x3 = ’); readln(x3);
```

```
    write('x4 = '); readln(x4);
    writeln(x4:8:3);
    writeln(x3:8:3);
    writeln(x2:8:3);
    writeln(x1:8:3)
end.
```

しかしこれが100個とか200個になると、とても書く気にならないだろうし、一般に$N$個ということになるとどうするのだろう?

ここで、もし変数に「添字」が使えたとすれば、この問題は次のように書くことができるだろう。「$x_1 \sim x_{200}$までの変数を用意しておく。データを次々に読み、$i$番目のデータを変数$x_i$に入れる。0が来たらその手前を$N$番目として、$x_N \sim x_1$の順に出力する。」

こういうことをするために、Pascalでは「添字つきの変数」を使う機能が用意されている(プログラミング言語の用語では「配列」(array)、つまり何やらならんだもの、というコトバを使うのが伝統である)。もちろん、プログラムを打ち込んでいる時に小さい字は使えないので、添字は「[、]」で囲んで、たとえば$x_i$だったら`x[i]`、$a_{n-1}$だったら`a[n-1]`のように現す。変数宣言の方も、例えば次のようにする。

```
a: array[0..50] of integer;
x: array[1..200] of real;
```

ここで「`0..50`」とか「`1..200`」は使える添字の範囲を指定していて、「`of integer`」とか「`of real`」は1つひとつの「箱」が整数か実数かを指定している。配列は図4のように普通の変数と同じ「箱」が必要なだけ「並んでいる」ものと考えてよい。

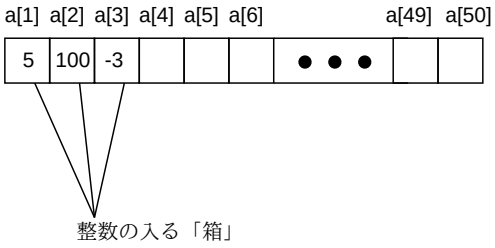


図 4: 配列の概念

これを使えば、先の問題は図5のPADと、次のようなPascalプログラムで書くことができる。

```
program sam9f(input, output);
```

```
var n, i: integer;
    d: real;
    x: array[1..200] of real;
begin
  n := 0;
  write('data = '); readln(d);
  while d <> 0.0 do begin
    n := n + 1; x[n] := d;
    write('data = '); readln(d)
  end;
  for i := n downto 1 do writeln(x[i]:8:3)
end.
```

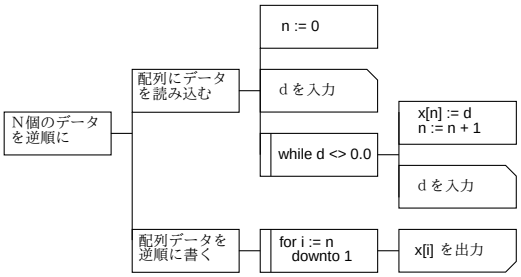


図 5: $N$ 個の数を逆順にする PAD

なお、最後の方に出てくる for 文は、最初に i を n にして、そこから 1 つずつ減らしながら 1 になるまでループを実行するのでしたね。

**演習 1** ☆ 上のプログラムを打ち込んでそのまま動かせ。

**演習 2** △ $N$ 個のデータを読み込み (0 が来たらおわり)、平均値と平均値より大きいものの個数を打ち出すプログラムを作れ (PAD を図 6 に示しておく)。

**演習 3** △ $N$ 個のデータを読み込み (0 が来たらおわり)、平均値と平均値に一番近いデータ値を打ち出すプログラムを作れ。[1]

[1]「一番近い」を判定するには「絶対値」を使うと便利。式 $x$ の絶対値は abs($x$) で求められる。

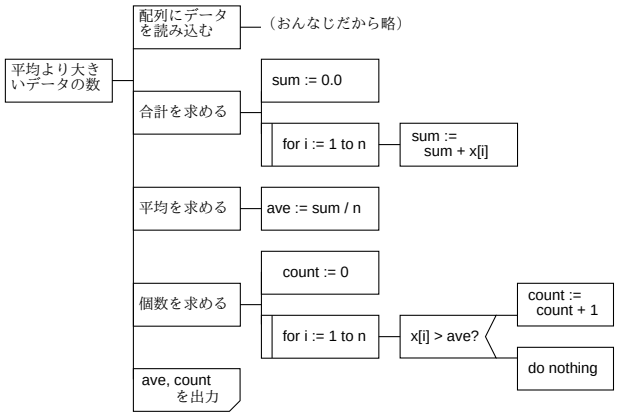


図 6: 平均より大きいデータの数を数える PAD

# 3　2 次元配列と PPM 形式ファイル

## 3.1　2 次元配列/＊

さて、線形代数 (その言葉を聞きたくないという人はすいません) などでは、行列の要素を指定するのに 2 つ以上の添字を使いますね ($x_{ij}$ のように)。Pascal の配列でも同様に、添字を 2 つ以上つけることができる。その場合には `x[i, j]` のように添字どうしをカンマで区切って書く。また、変数宣言のところでも

```
x: array[1..20, 1..100] of integer;
```

のように範囲を複数ならべてカンマで区切る (範囲は添字ごとに違っていてよいことに注意)。これは、図 7 のように整数の箱が縦横に並んでいるものと考えればよい。一般に添字の数を「次元」といい、先に出て来た添字が 1 つの配列を 1 次元配列、ここに示した 2 つのものを 2 次元配列とよぶ。(なお、3 次元以上の配列はややこしいのでめったに使われない。)

さて、2 次元配列を使った手頃な例題として、プログラムによるお絵描きをやろう。前に、計算機画面というのは様々な色の「点」から成っているという話をしましたね? そして画面は平面 (2 次元) だから、これを 2 次元配列に対応させて考えればちょうどよい。ひとつひとつの点は、赤 (red)、緑 (green)、青 (blue) の 3 色の蛍光体を適当な明るさで光らせることで様々な色になるのである。

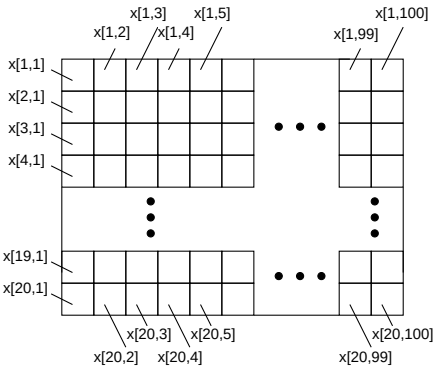


図 7: 3 つの配列でカラーの絵を表す

### 3.2 PPM 形式ファイル/☆

次に、Pascal プログラムの内部で作った絵をどうやって画面で見るかを説明しよう。それには、プログラムに「PPM 形式」の出力を出させて、それを `ppmtogif` というプログラムでおなじみ GIF に変換してから見る。PPM 形式というのはとても簡単で、次の形をしている。

```
P3 幅 高さ 255
赤 緑 青
赤 緑 青
赤 緑 青
...
```

ここで「赤 緑 青」とあるのは、1 つの点の赤、緑、青の光り方の強さを表す 0〜255 の数値である。たとえば

```
P3 10 5 255
255 0 0
255 0 0
255 0 0
...          ←全部で 150 行
```

という内容のファイルは「幅 10 ドット、高さ 5 ドットの真っ赤な絵」を表している。

**注意!** 表示可能な色の種類は 256 × 256 × 256 色…であるかのような気がしますが、実はそれができるのは高価なシステムだけです。我々が使っているシステムでは同時に表示可能な色の数は 256 色で、しかも自分のプログラムだけで 256 使うと Mosaic などが使える色がな

くなってしまうので、プログラムで使う色の数はせいぜい 100 色程度にとどめるのがよいでしょう。

**演習 4** ☆ Mule で上の内容と同じもの (色合いは適宜変更してもよい) を作って「`test.ppm`」というファイルに書き出せ。同じ行をコピーするには、

- 行の先頭で`^K^K`
- そのままコピーしたい回数だけ`^Y`

をやると速い。続いて Kterm の窓で

```
% ppmtogif test.ppm >test.gif
% xv test.gif &
```

により xv で表示させてみよ (小さくてよく見えない場合には窓の大きさを引っ張って大きくすれば拡大される)。納得したら、一部の色合いを変化させてみよ。たとえば最初の 10 個の点を真っ青にしてみよ。なお、「コマンド >ファイル」というのはコマンドの出力を指定したファイルに書き込んで保存することを指定している。

ファイル上で各点を指定すると、それは行方向に並んでいる (つまり最初の 10 個の点はいちばん上の線、次の 10 個は 2 番目の線、…となっている) ことに注意。

### 3.3 プログラムによるお絵描き/☆

さて、今度はエディタではなくプログラムで絵を描いてみる。そのためには、赤/緑/青それぞれの明るさを 2 次元配列 red、green、blue に入れることにする。つまり図 8 のように配列 3 つに 3 原色それぞれの明るさの値 0〜255 を入れておき、それを「合成」すると 1 枚のカラーの絵になるわけである。たとえば右下隅の点を「真っ黒」にするには、`red[1,1]`、`green[1,1]`、`blue[1,1]` をすべて 0 にすればよい。[2]

プログラムでは、最初に全体を「真っ白」にしてから必要な部分だけ適当に変更する。図 9 に「ななめ線の絵を描く」プログラムの PAD を示す。まず、絵の全体は 300 × 200 の大きさにすることにして (あまり大きいとファイルが大きく出力に時間が掛かる)、`red`、`green`、`blue` という 2 次元配列を絵の全体に対応させる。最初に全体を白くするが、それには for ループの中にさらに for ループをいれて、その中で `red`、`green`、`blue` の

[2]図形を描くときには普通は右下隅が原点になる方が分かりやすいのでそうしてある。

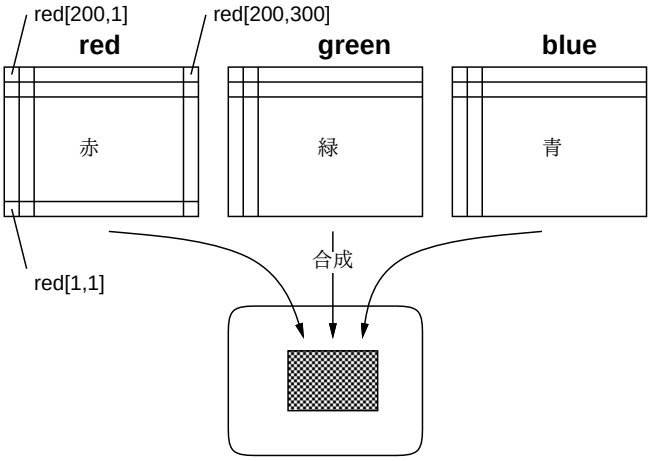


図 8: 3 つの配列でフルカラーの絵を表現

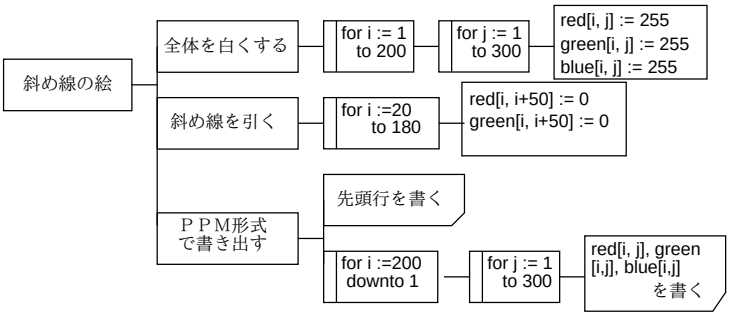


図 9: 斜め線の絵を描く PAD

各要素を 255 にすることでできる (白はすべての色が最も明るく光っている状態。おわかりかな?)。斜め線は線の上に対応する場所を適当な色に書き換えることで描くことができる。簡単のため、角度は 45 度にしてある (ここでは赤と緑を 0 にしたので、色は「真っ青」になる)。

最後に、red、green、blue の内容を書き出すが、最初に PPM ファイルの 1 行目 (大きさ指定) を書き、あとは再び 2 重の for ループの中で各要素の値を書き出せばよい。PPM では上の方から順に書く必要があるのに注意。これをプログラムにしたものを次に示す。

```
program sam9g(input, output);
var i, j: integer;
    red, green, blue: array[1..200, 1..300] of integer;
begin
  for i := 1 to 200 do
    for j := 1 to 300 do begin
      red[i,j] := 255; green[i,j] := 255; blue[i,j] := 255
    end;
  for i := 20 to 180 do begin
    red[i, i+50] := 0; green[i, i+50] := 0
  end;
  writeln('P3 300 200 255');
  for i := 200 downto 1 do
    for j := 1 to 300 do
      writeln(red[i,j]:1, ' ', green[i,j]:1, ' ', blue[i,j])
end.
```

このプログラムは入力データは全くなく、出力は PPM 形式ファイルのみなので、次のようにして出力をそのまま ppmtogif に接続して GIF 形式に変換する。その後はこれまで通り xv を使って結果を眺める (xv の画面の様子を図 10 に示す)。

```
% a.out | ppmtogif >test.gif
(色数に関するメッセージが出る)
% xv test.gif &
```

なお、「a.out | ppmtogif ...」というのは a.out プログラムの出力を ppmtogif の入力に接続して処理させることを意味している。

**演習 5** ☆ 斜め線の例題プログラムを打ち込んで動かせ。成功したら、どんな風にでもいいから少し違った絵になるようにプログラムを直して動かせ。

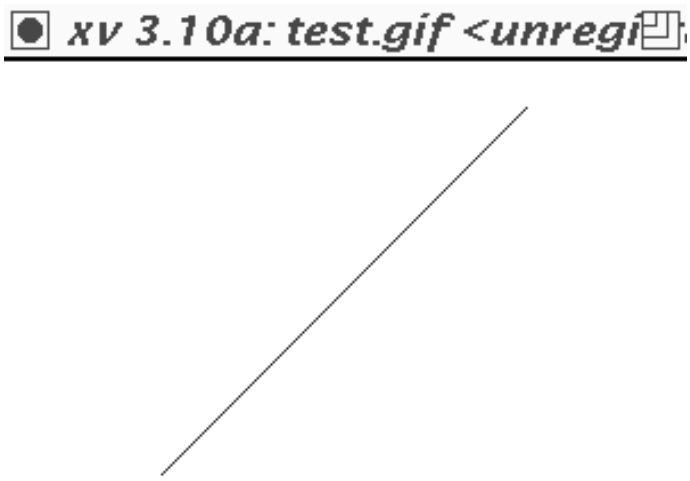


図 10: 斜め線の絵

**演習 6** △ 斜め線のプログラムを参考にして、次のいずれかのプログラム (PAD と Pascal) を作って動かせ。

a. ×印を描く。
b. 長方形か正方形 (枠でも塗りつぶしたものでもよい) を描く。[3]。
c. 45 度以外の斜め線を描く。
d. 円 (枠でも塗りつぶしたものでもよい) を描く。

これだけだと物足りないという人もいそうなので、末尾に冬休みの課題を予告しておく。〆切は 1 月 29 日になるので、たっぷり楽しんで頂きたい。

## A　本日の課題 9A

本日の課題は演習 1〜3 のうちどれか 1 つの実行例、演習 5(直したもの) の Pascal プログラムと生成した絵の印刷出力を提出してください。久々に紙で提出して頂きますので、アンケートとレポート番号氏名等を忘れないように。レポート番号は 9A です。アンケートは次の通り。

Q1. 「配列」についてどう思いましたか? または、配列を使ったプログラムを打ち込んで動かして見た感想でもいいです。

[3]ヒント: 最初に真っ白にしているのは「真っ白な長方形を描いている」と思っても良い

Q2. プログラムで絵を描くのは明らかに xpaint や idraw よりめんどくさいですが、では利点があるとしたらどんなことだろうと予想しますか? それとも利点はないと思いますか?

Q3. その他、感想、要望、質問があればどうぞ。

## B 次回までの課題 9B

次回までの課題は、演習 6 のどれか 1 つだけで、PAD と Pascal、および実行結果の絵の印刷出力を提出してください。レポート番号は 9B です。アンケートは次の通り。

Q1. 配列について理解しましたか？しなかったとしたら、どの辺に問題がありますか? したとしたら、どの辺がポイントだと思いましたか?

Q2. その他、感想、要望、質問があればどうぞ。

## C 冬休みの課題 1R

冬休みの課題は次の通り。

> 今回の内容を参考に「美しい」絵を出力するプログラムを設計し作成せよ。

何をもって「美しい」と考えるかは各自に任されるが、自分のプログラミングの腕前から見て不当に易しい内容を選択しないこと。レポートは必ず A4 版の用紙を用い、以下の構成を取ること。

1. 表紙。レポート課題番号 (1R)、学籍番号、氏名、提出日を記すこと。
2. 方針。問題を解くにあたって、どのようなことを考えどのような方針を立てたか。
3. 設計。プログラムの構造など (PAD 図はもちろんだが、PAD 図では把握しにくい情報があればそれも分るように工夫する)。
4. 回答。この場合にはプログラムと、生成された絵を印刷出力したもの。
5. 考察。この課題をやったことで、新たにどのようなことが分ったか検討し考察する。今から考えるとどうした方がよかった (反省)、どう感じた (感想)、などの内容も含めてよいが、反省と感想が大部分というのでは困る。

6. 付録。何かつけ加えたい内容があれば。(今回はあまりなさそうですね…)

この課題 1R は、1 月 29 日の講義開始までに、1F のレポートボックスに提出すること。また、これと並行して、プログラムが生成した絵の GIF ファイルを「`WWW/report1r.gif`」というファイルに置くこと (互いに鑑賞できるようにしましょう)。