「情報処理」1年文I/IIクラス11-12　＃12

久野　靖*

1995.1.8

1　本日の目標

レポート1Rの〆切まで1週間となりましたが、進捗はいかがですか。本日は後半はおもにレポートのためのプログラムの質問受け付けに使おうと思うのですが、その前に「そっけないPascalプログラムにお化粧を施す」話をしようと思います。あと、手続きを使って「構造のある絵」を作る話もしましょう。本日の内容は次のとおり。

- WWWのCGIを使って自分の(簡単な)プログラムにユーザインタフェースをつける。
- 手続きの意味の復習と構造化グラフィクス

2　WWWとCGI

2.1　CGIとは?

HTMLを使ってWWWのページを作成する、というのは前にやっていただいたが、HTMLを使って作ったページは結局、絵とかは入れられるけど「見るだけ」でページの中身が動いてくれるわけではない。

しかし「WWWでページの中身を動かす」という仕組みも存在する。それは、次のような枠組みになっている。

- HTMLでページを書くときに「入力欄」のようなものを記述できる。
- さらに、その「入力欄」の内容を読んで処理するプログラムを指定できる。
- 「入力欄」に値を埋めた後で「提出」というボタンをクリックすると、埋めた値が先に指定した処理プログラムに送られる。

*筑波大学大学院経営システム科学専攻

- 処理プログラムでは、値を受けとって処理し、**プログラムの方で新しい HTML ページを生成して返す**ことができる。

この間のデータの流れを図 1 に示す。ここで、WWW サーバとプログラ

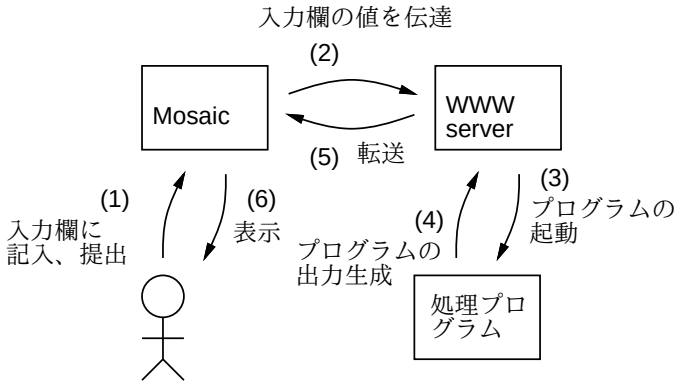

図 1: CGI の処理のながれ

ムの間で値を受け渡す約束ごとを「Common Gateway Interface」ないし「CGI」、処理プログラムのことを「CGI プログラム」または「CGI スクリプト」と呼ぶ。

なぜ前回やった「スクリプト」が出てくるかというと、CGI の約束を処理するプログラムを全部自分で書くのは結構面倒なので、約束を処理するコマンド群が用意されていて、これをシェルスクリプトから呼び出して処理することが多いからである。

2.2 HTML とフォーム

では例として、一番最初にやった Pascal の例題「摂氏華氏変換」を CGI プログラムにしてみよう。まず、入力欄のある HTML ファイルを作成する。それには、これまでに出てこなかった HTML のタグ`<FORM> ... </FORM>`を使う。

- `<FORM>`タグは`<FORM METHOD=POST ACTION="スクリプト">`という形になる。ここで「スクリプト」は CGI スクリプトの名前。なお、CGI スクリプトはファイル名の末尾が「`.cgi`」で終っている必要がある。
- そこから`</FORM>`までの間に入力欄を複数書ける。
- 入力欄の間のテキストの整形や行かえなどは、これまでに学んだ普通のタグを使って指定する。

- 最もよく使う入力欄は「<INPUT TYPE="text" NAME="名前">」という形のもの。これで任意の文字列を打ち込む入力欄ができる。「名前」は、CGI スクリプトの側でどの値がどの入力欄のものかを区別するために用いる。
- 提出ボタンは「<INPUT TYPE="submit" VALUE="ラベル">」というタグで作る。「ラベル」はボタンに表示される文字列。

ごたくばかりでは面白くない。とりあえず、摂氏華氏変換のためのフォームつきページを作ってみよう。

```
<HTML>
<HEAD>
<TITLE>摂氏華氏変換</TITLE>
</HEAD>
<BODY>
<H1>摂氏→華氏の換算をします。</H1>

<FORM METHOD=POST ACTION="temppost.cgi">
<P>華氏の温度: <INPUT TYPE="text" NAME="ftemp"></P>
<P><INPUT TYPE="submit" VALUE="計算!"></P>
</FORM>
</BODY>
</HTML>
```

これをたとえば「tempform.html」という名前で用意し、自分の「WWW」ディレクトリの下に置く。(今日出てくるファイルはすべて「WWW」の下に置くので、最初に「cd WWW」してから作業をするとよい。自分のホームページからこのページがたどれるようにするには、ホームページの HTML ファイル (index.html) に次のようなリンクを入れておけばいいのですね? (思い出してくださいよ!)

```
<A HREF="tempform.html">摂氏華氏変換のページへ行く</A>
```

で、そのページを表示させたところの様子を図 2 に示す。ここで入力欄に適当な数値を打ち込んで、「計算!」ボタンをクリックすると CGI スクリプトが起動される。

2.3 CGI スクリプトの内容

次に、CGI スクリプト temppost.cgi を見てみよう (これも WWW ディレクトリに置くこと!)。

図 2: フォームのあるページの表示

```
(1)  #!/bin/sh
(2)  PATH=/usr/local/httpd_ncsa/www/bin:/home/ユーザ名/WWW/:$PATH
(3)  echo 'Content-type: text/html'
(3)  echo ''
(4)  eval `cgiparse -form`
(5)  tempcalc <<EOF
(6)  $FORM_ftemp
(7)  EOF
```

なお、数字は説明のために入れたものです。説明は次の通り。

(1) シェルスクリプトの先頭行。前回やった通り。

(2) コマンドのありかを指定する。前回やった「探索パス」のこと。なお、「ユーザ名」のところには自分のユーザ名を入れること。これは、自分のコマンドが実行できるようにするため。

(3) CGI スクリプトの約束ごとで、プログラムが生成する出力の形式 (ここでは HTML) を指定する。

(4) 入力欄を受け取る処理をしてくれるコマンドを起動する。ここだけ「'」(シングルクォート) ではなく「`」(バッククォート) を使うので注意。

(5) 自分で作った Pascal のプログラムの `a.out` ファイルを適当な名前に変更して指定。また、「以下 EOF とある行までが入力だよ」とい

う指定がついている。**注意!: EOF の行は、その後に空白がついているとうまく行かないので気をつけるよーに。**

(6) cgiparse を使うと入力欄の名前 (HTML ファイルの<INPUT TYPE="text" NAME="名前">で指定したもの) に対応して「$FORM_名前」という変数が作られ、この変数が入力欄に打ち込まれた値を保持している。ここではプログラムに入力するデータは 1 つの数値だけ。

この内容を先の HTML ファイルに指定した「temppost.cgi」という名前で保存し、WWW ディレクトリに置くとともに

```
chmod ugo+x temppost.cgi
```

で実行可能にしておく必要がある。

2.4 CGI から呼び出す Pascal プログラム

では最後に、Pascal プログラムを示そう。

```
program tempcalc(input, output);
var c, f: real;
begin
  readln(f);
  c := (5.0 * (f - 32.0)) / 9.0;
  writeln('<HTML><HEAD><TITLE>tempcalc</TITLE></HEAD><BODY>');
  writeln('<H1>華氏の温度', f:5:2, '度は</H1>');
  writeln('<H1>摂氏では', c:5:2, '度です。</H1></BODY></HTML>');
end.
```

要するにほとんど最初の例題そのままだが、次の点が違う。

- データは入力欄で記入してもらうので、プロンプトを出す必要はなく、いきなり readln で読む。
- 出力は HTML のページの形に整える。

これを「pc tempcalc.p」などとして翻訳し、そのあとできたファイルを「mv a.out tempcalc」によりあるべき名前に変更する (すべて WWW ディレクトリにないといけないのに注意!)。ではこれですべて完成したので、動かしたところを図 3 に示す。

図 3: CGI の出力ページ

2.5 CGI による絵の生成

例題が 1 つではつまらないので、今度は絵を生成するプログラムを CGI として動かしてみよう。つまり、

- 中心の x 座標、y 座標、半径
- 色の RGB 値

を入力欄から打ち込み、指定した位置、大きさ、色を持つ円を描くわけである。こんどは Pascal プログラムを先に示す。

```
program sam12a(input, output);
var i, j: integer;
    red, green, blue: array[1..200, 1..300] of integer;
    x, y, rad, r, g, b: integer;

procedure point(x, y, r, g, b: integer);
begin
  if (1 <= x) and (x <= 300) and (1 <= y) and (y <= 200) then begin
    red[y,x] := r; green[y,x] := g; blue[y,x] := b
  end
end;

procedure fillrect(x0,y0,w,h,r,g,b:integer);
var i, j: integer;
begin
  for i := y0 to y0+h do
    for j := x0 to x0+w do point(j, i, r, g, b)
```

```
end;

procedure fillcircle(x0,y0,r0,r,g,b:integer);
var i, j: integer;
begin
  for i := y0-r0 to y0+r0 do
    for j := x0-r0 to x0+r0 do
      if (i-y0)*(i-y0) + (j-x0)*(j-x0) <= r0*r0 then point(j,i,r,g,b)
end;

procedure clearcanvas;
var i, j: 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;
end;

procedure writecanvas;
var i, j: integer;
begin
  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]:1)
end;

begin
  clearcanvas;
  readln(x); readln(y); readln(rad);
  readln(r); readln(g); readln(b);
  fillcircle(x, y, rad, r, g, b);
  writecanvas;
end.
```

手続き群は前にやったプログラムのままで、メインプログラムは位置、半径、色などを入力から順次読み込んで円を 1 つ描くだけ。これをコンパイルして、実行形式ファイルを WWW ディレクトリに「cieclecalc」という名前で置く。次に、CGI スクリプトは次の通り:

```
#!/bin/sh
PATH=/usr/local/httpd_ncsa/www/bin:/home/kuno/WWW/:$PATH
echo 'Content-type: image/gif'
echo ''
eval `cgiparse -form`
circlecalc <<EOF | ppmtogif87 2>/dev/null
$FORM_x
```

```
$FORM_y
$FORM_rad
$FORM_r
$FORM_g
$FORM_b
EOF
```

前と違うところは次の通り。

- CGI の返すデータ種別を image/gif と指定。
- Pascal プログラムの出力を ppmtogif87 に送って GIF に変換させる (なぜかコマンド名がいつもと違うので注意)。なお「2>/dev/null」というのは ppmtogif87 が「何色あったよ」と表示するのを抑制する。
- 入力データはさっきは数値 1 個だったが今度は 6 個。

最後にフォームの HTML を示す。

```
<HTML>
<HEAD>
<TITLE>塗りつぶした円を描く</TITLE>
</HEAD>
<BODY>
<H1>お好みの円を描きます</H1>

<P>X 座標、Y 座標、半径、赤、緑、青の明るさを指定してください。
</P>

<FORM METHOD=POST ACTION="circlepost.cgi">
<P>X 座標: <INPUT TYPE="text" NAME="x" SIZE=5>
   Y 座標: <INPUT TYPE="text" NAME="y" SIZE=5>
   半径:  <INPUT TYPE="text" NAME="rad" SIZE=5></P>
<P>色合い (0〜255 の数値で指定すること):</P>
<P>赤: <INPUT TYPE="text" NAME="r" SIZE=5>
   緑: <INPUT TYPE="text" NAME="g" SIZE=5>
   青: <INPUT TYPE="text" NAME="b" SIZE=5></P>
<P><INPUT TYPE="submit" VALUE="描く"></P>
</FORM>
</BODY>
</HTML>
```

これでデータを入力している様子を図 4 に示す。

図 4: フォームのあるページの表示

2.6 CGI のまとめ

さて、結構長かったので最後にまとめておこう。CGI を使って入力欄を処理する場合には、結局次の 3 種類のファイルが必要になる。

- HTML ファイル。これで入力欄を含むページを用意する。名前は「`.html`」で終ること。
- CGI スクリプト。これで出力データの形式を指定し、入力欄のデータを受けとって、実行プログラムを呼び出すとともにデータを引き渡す。名前は「`.cgi`」で終ること。
- 実行プログラム。Pascal でふつうに作成する。名前は何でもいい。

そして 3 つとも、WWW ディレクトリに置く必要がある。

演習 1 ☆ 上に示した例題のどちらか好きな方と同じ HTML ページを用意 し、CGI を動かせ。そのページは自分のホームページからリンクすること。Paslcal プログラムは例によって`~kuno/pascal/`にあります。

演習 2 ☆ それをちょっと改造してみよ。たとえば「2 つの数を入力欄から打ち込むとその合計を返す HTML ページを作る」「塗りつぶした四角ができる HTML ページを作る」など何でもよい。そのページは自分のホームページからリンクすること。

3 手続きと構造化グラフィクス/△

3.1 手続きの意味の復習

さて、ここまで 2 回くらい Pascal の「手続き」とそのバリエーションである「関数」についてやってきたが、重要なところなので再度別の角度から眺めて復習しておこう。

最初に説明した時には、手続きというのは「新しい命令を増やしている」、たとえば `fillcircle` という手続きを定義したら、その後ではいつでも `fillcircle` という命令が使えるようになっている、ということを説明した。しかし実際には、どんなことが起こっているのだろう?

ふつう、プログラムの実行は (ループはあるにせよ) 図 5 の左側のように、上から始まって一直線に進む。ところが、手続き (たとえば `fillcircle`) の命令が書いてあるところまでくると、実行の流れは右側の fillcircl と書かれたところ (つまり fillcircle の本体の各命令がある場所) にジャンプして、そこから実行が続けられる。そして、fillcircle の最後まで来ると…す

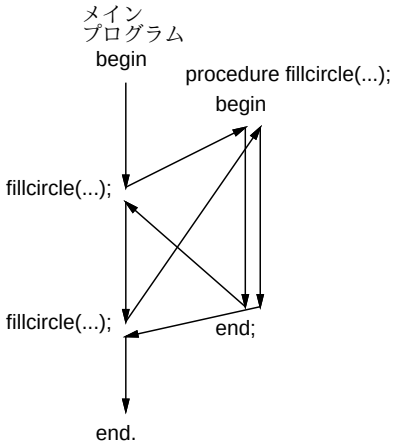

図 5: 手続きの実行のしくみ

ると、さっきジャンプしたところ (もちろん、ちゃんと覚えておくのだ) まで戻って、その続きを実行する。その後で、また別の `fillcircle` の参照箇所 (呼び出し、ともいう) に来るとまた fillcircle 本体のところにジャンプし、中を実行し終わるとさっきの続きに戻る。

このようにして、1 回書いたプログラムの部分 (fillcirlce の本体の命令) を何箇所からでも利用できる、というのが手続きの本質なのである。

3.2 手続きの階層と絵の構造

もちろん、このような呼び出しは 1 レベルではなく何段階にも渡って起きてよい。図 6 を見ると、main は A を 2 回呼び出し、A は B を 3 回呼び出し、B は C を 2 回呼び出すから…C の中は合計 12 回呼ばれることになる。

ところで、お絵描きの話に戻ろう。絵にはしばしば、階層構造が見られる。たとえば、自動車が描いてあったとすると、その自動車はタイヤ、窓、ボディといった「部品」から成っている。しかも、タイヤは同じものが 4 つついていたりする。このような、「全体としては 1 つの〜だが、中を詳しく見るといくつもの部品から成る」という構造が何レベルにも積み重なって、1 つの絵ができている (実は絵だけでなく現実世界の「もの」もだいたいそうなっている)。

そこで、このような絵を描くときに、1 つの「もの」の種類 (部品も「もの」である) を 1 つの手続きに対応させる。そうすれば、「もの」の種類の数だけ手続きを用意することで、かなり複雑な絵が描ける。どういうことか分かりますか?

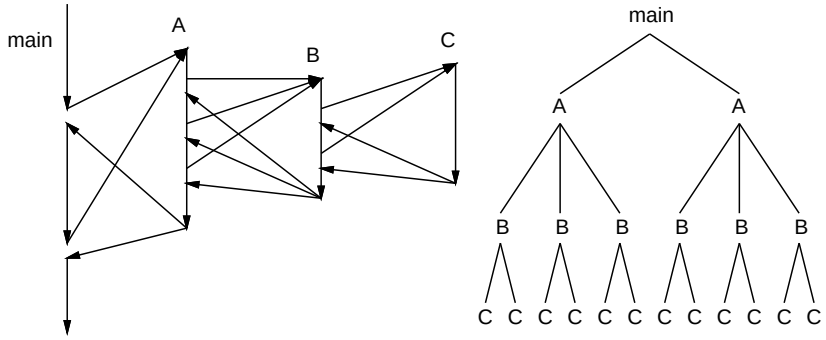

図 6: 複数レベルの手続き呼び出し

たとえば、「自動車」を描く手続きの中では「タイヤ」を描く手続きを 4 回呼ぶ。すると、自動車が 3 台あった場合にはタイヤは 12 回、つまりずいぶん何回も描かなければならないのだけれど、先に延べた手続きの原理を応用すれば、単に自動車を描く手続きを 3 回呼べば、その中でそれぞれタイヤは 4 回呼ばれるからほっといても 12 個のタイヤが描けるわけだ。

このように、絵の「構造」を考えてそれに対応した手続きを用意して絵を生成する手法を「構造化グラフィクス」などと呼ぶ (こともある。私はグラフィクスの専門家ではありませんが、駒場の計算機の先生がたの半分くらいはこの手の専門家ですので)。

3.3 例題: 家と人と車

注意: 以下の例題の絵は、あくまでも (絵の下手な) 私が適当に考えたものなので、無理に真似する必要はないことをお断わりしておく。

さて、絵の中に車 (家や人でもいいけど) をいくつも描くとして、それぞれの大きさとか位置とか色は描くごとに調整したい。そこで、絵そのものは適当な単位 (unit) を基準としてその倍数でデザインしよう。図 7 の左上に「1」と書かれた線があるが、この長さの倍数の長方形、三角系、円などを組み合わせて車、家、人を作ってみた。それぞれの絵の原点は左下隅にあるものとする。

これを Pascal の手続きにしたものを次に示す。なお、ボディや胴体や家の本体は他の部分と同じ色合いで、ただしちょっと「暗く」なるように色の明るさを全体に減らしてある。

```
procedure house(x0, y0, u, r, g, b:integer);
begin
  fillrect(x0+u, y0, u*6, u*4, trunc(r*0.8),trunc(g*0.8),trunc(b*0.8));
  fillrect(x0+u*2, y0, u*2, u*3, r, g, b);
  filltriangle(x0, y0+u*4, x0+u*8, y0+u*4, x0+u*4, y0+u*8, r, g, b)
```

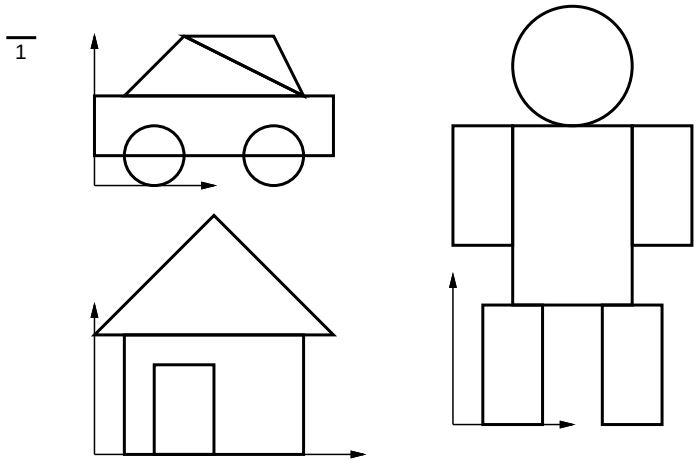

図 7: 家、人、車のデザイン

```
end;

procedure car(x0, y0, u, r, g, b:integer);
begin
  fillrect(x0, y0+u, u*8, u*2, trunc(r*0.7),trunc(g*0.7),trunc(b*0.7));
  fillcircle(x0+u*2, y0+u, u, r, g, b);
  fillcircle(x0+u*6, y0+u, u, r, g, b);
  filltriangle(x0+u, y0+u*3, x0+u*7, y0+u*3, x0+u*3, y0+u*5, r, g, b);
  filltriangle(x0+u*7, y0+u*3, x0+u*3, y0+u*5, x0+u*6, y0+u*5, r, g, b)
end;

procedure human(x0, y0, u, r, g, b:integer);
begin
  fillrect(x0+u*2, y0+u*4, u*4, u*6, trunc(r*0.6),trunc(g*0.6),trunc(b*0.6));
  fillrect(x0+u, y0, u*2, u*4, r, g, b);
  fillrect(x0+u*5, y0, u*2, u*4, r, g, b);
  fillrect(x0, y0+u*6, u*2, u*4, r, g, b);
  fillrect(x0+u*6, y0+u*6, u*2, u*4, r, g, b);
  fillcircle(x0+u*4, y0+u*12, u*2, r, g, b)
end;
```

なお、これら以外の手続きは先に出て来たので省略。これを呼び出して絵を描かせるだけのメインプログラムを示す。

```
begin
  clearcanvas;
  house(20, 40, 10, 200, 0, 150);
  human(120, 20, 7, 0, 150, 200);
  car(180, 30, 10, 100, 250, 100);
  writecanvas
end.
```

これでできた絵を図 8 に示す。

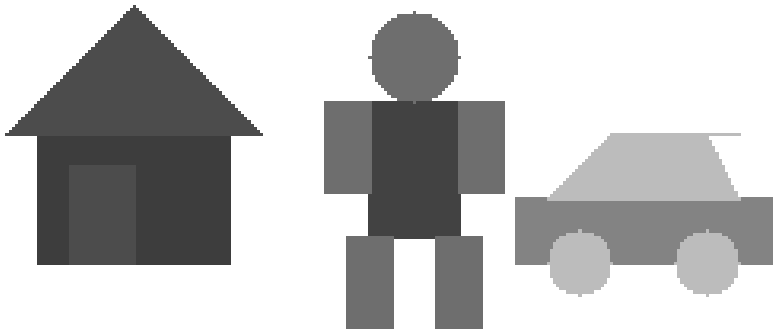

図 8: 家、人、車の絵

では次に、「家の戸口に人が立っている」のを作ろう。それには、家と人の手続きを呼び出す新しい手続き manandhouse を作ればよい。

```
procedure manandhouse(x0, y0, u, r1, g1, b1, r2, g2, b2:integer);
begin
  house(x0, y0, u*5, r1, g1, b1);
  human(x0+u*11, y0, u, r2, g2, b2);
end;
```

これを呼び出すメインプログラムは次の通り。

```
begin
  clearcanvas;
  manandhouse(20, 40, 3, 200, 150, 0, 100, 250, 0);
  manandhouse(100, 30, 4, 100, 50, 250, 200, 150, 0);
  car(200, 15, 10, 255, 100, 100);
  writecanvas
end.
```

絵も図 9 に示す。どうです、この調子でそれらしい絵が作れるでしょう? 最初は敷居が高いでしょうけれど、手続きはぜひ活用しましょう。なお、これらのプログラムのファイル (sam12b.p、sam12c.p) は例によって ~kuno/pascal/ の下にあります。

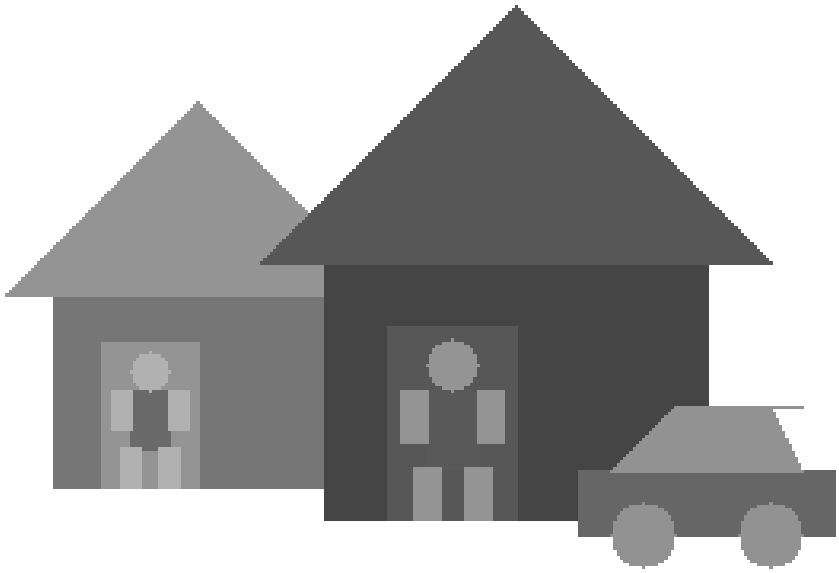

図 9: 家の戸口に人がいる絵

でも、レポートで私がデザインした絵が出て来たらさすがにがっかりしますから、絵のデザインは自分で考えてくださいね。あと、既に予告しましたが、1R の採点に当っては、レポートの文章がちゃんと書けていることを重視します (プログラムが動かないというのはさすがに困るけれど)。特に、プログラムリストだけしかない (あと 1 ページくらいおまけに感想が書いてある程度の) レポートは出さないように! (1 週間くらいなら待ちますから…) ではがんばってください。

A　本日の課題 12A

本日の課題は、演習 2 の「例題とは違う CGI のページを用意する」ことです。ページの作成が終ったらニュースに投稿してください。なお、そのページのどこかにアンケートの回答も入れてください。アンケートは次の通り。

- CGI についてどう思ったか、こういうものがどんな方面に利用できるか思いつくままに買いてください。
- (あれば) 感想と要望。