「情報処理」1年文I/IIクラス11-12　＃5

久野　靖*

1995.11.13

0　本日の目標

さて、今回と次回でいよいよ「お絵描き」をやります。(Webページを作らせろという要望が多いですが、絵がないと見栄えがしないでしょう? だからお絵描きが先なわけです。) 本日の目標は次の通り。

- ファイルの整理とディレクトリについて知る。
- 枝分かれのあるプログラムについてより理解する。
- xpaintによるお絵描き。

ところで、そろそろPascalの参考書が欲しいですか? 一応、2つ挙げておきますので、投資してもいいと思う人はどうぞ。

- 川合 慧、Pascal入門、共立出版 — 丁寧に論理的に書かれていますが、その論理的なのが理系っぽくていやだという人にはなじみにくいかも知れません。
- 戸川隼人、ザ・Pascal、サイエンス社 — すごく易しく書かれているので、上の理系っぽいのが恐い人にはおすすめです。反面、きっちり全部説明してくれないと、という人には向かないでしょう。

1　ファイルの整理/＊

そろそろ、`ls`で見るとファイルが沢山あって気分が悪いという人もいそうですね? そこでファイルの整理方法を学んで頂く。

*筑波大学大学院経営システム科学専攻

1.1 コマンド入力の技/＊

その前に! これから Kterm の窓で多数コマンドを打ち込むわけだけど、

- 前とほとんど同じコマンドを打ち込む場合
- コマンドをちょっとだけ間違ってしまいやり直す場合

に使うワザを紹介しておこう。それは…「^P」を押すと前のコマンド行が出てきて (数個前なら^P を好きなだけ繰り返す。行き過ぎたら^N で戻る)、その後^B や^F で直したい場所に行き、^D などでいらない字を消し、追加する字はそのまま打ち込み (つまり Mule とおんなじ!)、最後に [RET] を打つとそのまま実行させられる。これでコマンドが多くなっても安心でしょう?

1.2 cp、mv、rm/＊

まず、ファイル整理の基本コマンドは次の通り。

```
cp ファイル１ ファイル２ -- ファイル１の内容をファイル２にコピー
mv ファイル１ ファイル２ -- ファイル１の名前をファイル２に変更
rm ファイル１            -- ファイル１を消す
```

これらのコマンド名はそれぞれ「copy」「move」「remove」の略ということになっている。これで、いらないファイルを消しているファイルは適当な名前に変更できるし、似たようなプログラムを複数書く場合には適当なファイルをまずコピーしてそれを元に直すといった技も使える。

1.3 ディレクトリ/＊

しっかし、ファイルが沢山になってくると、不要なのを消して名前を整理したくらいでは済まない。そこで、いくつかのファイルを 1 つの名前の下に移してまとめる、といったことをしたい。そのような単位を「ディレクトリ」という。実は! 皆様が現在ファイルを置いているところもディレクトリであり、login するとまずそのディレクトリが利用可能になる。ここを特に「ホームディレクトリ」という。それ以外に、以下で説明するようにファイル整理用の副ディレクトリ (サブディレクトリ) をいくつでも作れる。ディレクトリを作ったり消したりするには次のコマンドによる。

```
mkdir ディレクトリ名    --- ディレクトリを作る
rmdir ディレクトリ名    --- ディレクトリを消去
```

例えば「mkdir pascal」を実行した後 ls で見ると「pascal/」というものが見える。このように、ディレクトリは名前の最後に「/」をつけて表示される。

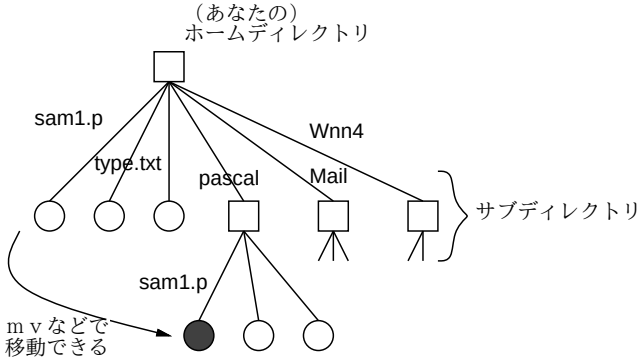

図 1: ディレクトリによるファイルの整理

次に、mv を繰り返し使って Pascal のプログラムファイルをこの下に移してしまおう。

```
% mv sam1.p pascal/sam1.p
% mv sam3.p pascal/sam3.p
....
```

このように、ディレクトリの下のファイルを指定するにはディレクトリ名の後に「/」で区切ってファイル名を書く。これをやった後で ls を見ると、移したファイルは一見なくなってしまったように見えるが、実は pascal ディレクトリに移っただけである。その状況を見るには

```
% ls pascal
```

のように、ディレクトリ名を指定して ls を実行すればよい。またファイルの中身を見たり Mule で指定するのもこれからは「pascal/sam1.p」のようにディレクトリ名つきで指定することになる。さらに沢山プログラムが増えたら? その時は「pascal/easy/sam3.p」「pascal/advance/sam9.p」のようにディレクトリの中にさらにディレクトリを作って細かく整理することもできる。このほか、メモなどのテキストファイルは text というディレクトリに入れるなどいろいろな整理法が考えられますね?

演習 1 ☆ 自分のファイルを上で説明したコマンドを利用して適当に整理してみよ。ディレクトリも活用すること。

演習 2 △ 電子メールメッセージなどは `Mail` ディレクトリの下に保管されるようになっている。その下にどんな風に保管されているか、`ls` と `less` を使って調べてみよ。[1][2]

2 枝分かれ処理の復習/＊

では例によって、前回の演習のレビューをやってみょう。

a. 2 つの数 (実数) を読み込み、大きい方 (両方同じ値ならその値) を打ち出す。

これもアプローチとして 2 通りある。まず、図 2 を見てほしい。つまり、

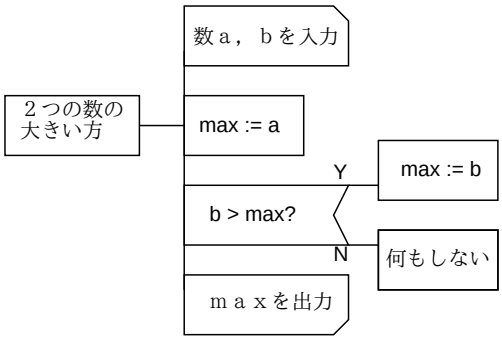

図 2: 2 つの数の大きい方 (1)

「`max`」という変数にとりあえず a を入れ、もし b の方がこれより大きいようなら改めて b を入れ直す。これをプログラムにすると次のようになる。

```
program sam5a(input, output);
var a, b, max: real;
begin
  write('a = '); readln(a);
  write('b = '); readln(b);
  max := a;
  if b > max then max := b;
  writeln('larger value = ', max:8:4)
end.
```

実行例も示しておこう。

[1] 単にこのファイルを `lwp` で打てば印刷できるわけですね。
[2] ただし MH か mh-e を使っている人でないとうまく行きません。

```
% pc sam4a.p
% a.out
a = 10.5
b = 12.3
larger value =  12.3000
%
```

ところで、「別解」として図 3 のような PAD も考えられますね? この方

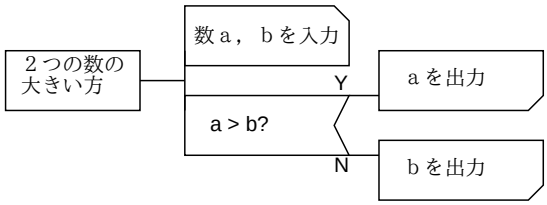

図 3: 2 つの数の大きい方 (2)

が簡単でよさそうですか?

```
program sam5a1(input, output);
var a, b: real;
begin
  write('a = '); readln(a);
  write('b = '); readln(b);
  if a > b then
    writeln('larger value = ', a:8:4)
  else
    writeln('larger value = ', b:8:4)
end.
```

しかし、この方法はやや一般化に弱点がある。順序は前後するが、c の問題を次に考えてみる。

c. 3 つの数 (実数) を読み込み、最大値 (どれよりも小さくない値) を打ち出す。

これを先の (2) の方針でやると、if で枝分かれした中でさらに枝分かれする必要がある (図 4)。これを Pascal にすると次の通り。

```
program sam5c1(input, output);
var a, b, c: real;
begin
```

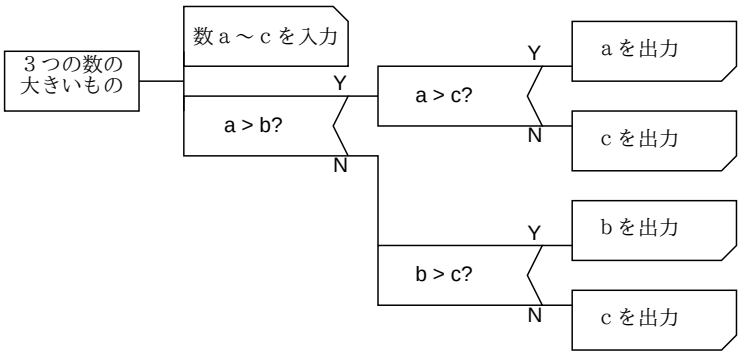

図 4: 3 つの数の大きいもの (2)

```
  write('a = '); readln(a);
  write('b = '); readln(b);
  write('c = '); readln(c);
  if a > b then
    if a > c then
      writeln('larger value = ', a:8:4)
    else
      writeln('larger value = ', c:8:4)
  else
    if b > c then
      writeln('larger value = ', b:8:4)
    else
      writeln('larger value = ', c:8:4)
end.
```

ところで、最初の方針 (max に最大のをとっておく) だとどうだろう? 図 5 のようになる。そして Pascal に直すと次のようになる。

```
program sam5c(input, output);
var a, b, c, max: real;
begin
  write('a = '); readln(a);
  write('b = '); readln(b);
  write('c = '); readln(c);
  max := a;
  if b > max then max := b;
  if c > max then max := c;
  writeln('larger value = ', max:8:4)
```

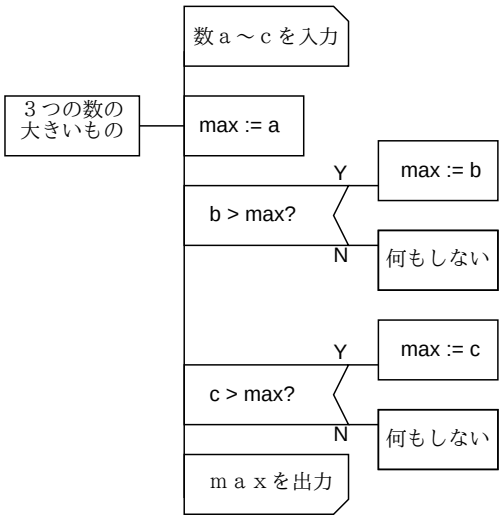

図 5: 3 つの数の大きいもの (1)

```
end.
```

こんどはどちらがいいですか?

3 if-then-else if-then ... の連鎖/*

一般に、枝分かれの中でさらに枝分かれというのは頭がウニになりやすいのでなるべく避けた方がいい。ただし、次のような形だけは理解しやすいので積極的に活用するとよい。

```
if 条件1 then
  動作1
else if 条件2 then
  動作2
else if 条件3 then
  動作3
else
  動作X
```

これは、「条件 1 を調べて、成り立っていれば動作 1 を、そうでなければ条件 2 を調べて、成り立っていれば動作 2 を、そうでなければ条件 3 を調べて、成り立っていれば動作 3 を、そうでなければ動作 X を実行する」と読める。これなら十分分かりやすい。なお、このパターンでは明らかに条

件の数はいくつあってもよいし、また「それ以外」の場合の動作 X が「何もしない」なら最後の else 以下も省略してよい。

この「if-then-else if-then ... の連鎖」の形はとても役にたつので、PAD でもこれに対応する箱が用意されている。それを図 6 に示した。

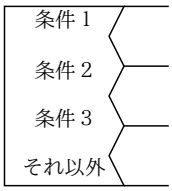

図 6: 複数選択の PAD の箱

これを使って問題 b を書くとわかりやすい (もちろん、課題の方ではまだこれを教えてなかったので、if の中の if でやって頂いたはずである)。

b. 1 つの数 (整数) を読み込み、正、負、零に応じて 1、-1、0 を打ち出す。(ヒント: 枝わかれ先の中でまた枝わかれする。)

この PAD 図は図 7 のようになるだろう。これを Pascal にすると次のよう

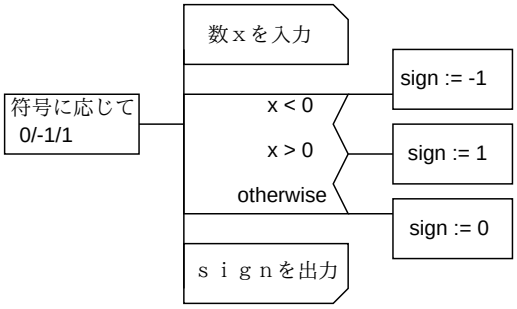

図 7: 符号に応じて 0/1/-1

になる。

```
program sam5b(input, output);
var x, sign: integer;
begin
  write('x = '); readln(x);
  if x < 0 then
    sign := -1
  else if x > 0 then
```

```
      sign := 1
    else
      sign := 0;
    writeln('sign value = ', sign:2)
  end.
```

どうですか、見やすいでしょう? もっとも、これは複数の if 文を組み合わせているという点では皆様が作った (であろう) 版と本質的に変わりはない。

要は、プログラミングも剣道や書道と同じで「自分が使いこなす型」を身につけ、「ここはこの型だ!」と反射的に応用できるようになればうまくなるのでして、教官はその「お手伝い」にヒントをあげているわけ。

4 時分秒問題の 2 つの考え方と begin-end/☆

さて、前に「2 つの時刻の差を秒数で求める」というのをやった。でもやっぱり「何時間何分何秒」で表示させたいですね? まず考えるのは、除算と剰余を使う方法だろう。PAD を図 8 に示す。これを Pascal にすると

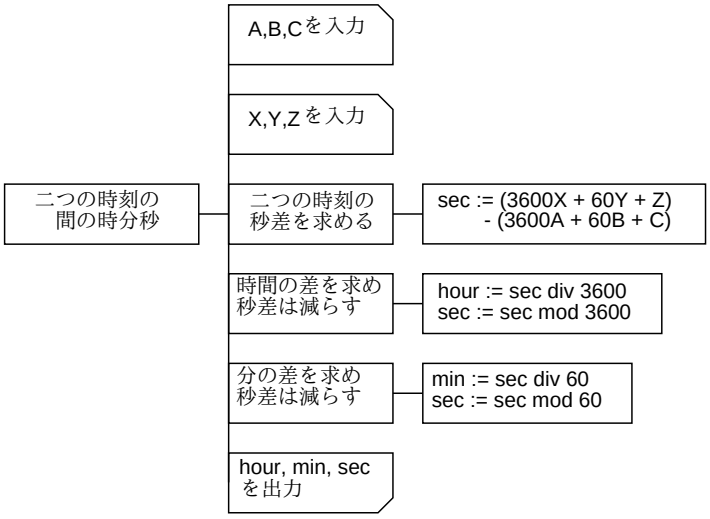

図 8: 時間差問題 (1)

次の通り。

```
program sam5d(input, output);
var a, b, c, x, y, z, hour, min, sec: integer;
begin
```

```
  write('a = '); readln(a);
  write('b = '); readln(b);
  write('c = '); readln(c);
  write('x = '); readln(x);
  write('y = '); readln(y);
  write('z = '); readln(z);
  sec := (3600*x + 60*y + z) - (3600*a + 60*b + c);
  hour := sec div 3600;
  sec := sec mod 3600;
  min := sec div 60;
  sec := sec mod 60;
  writeln(hour:1, '時間', min:1, '分', sec:2, '秒')
end.
```

ところで、せっかくもとのデータが時分秒になってるのだから、全部秒に直してしまうなどという力ずくをやらなくても、秒ごと、分ごと、時ごとに引いてもいいはずである。ただし、例えば「10 秒から 23 秒を引く」といった引けない場合は「分から借りてくる」必要がある。それを行うような PAD を図 9 に示す。

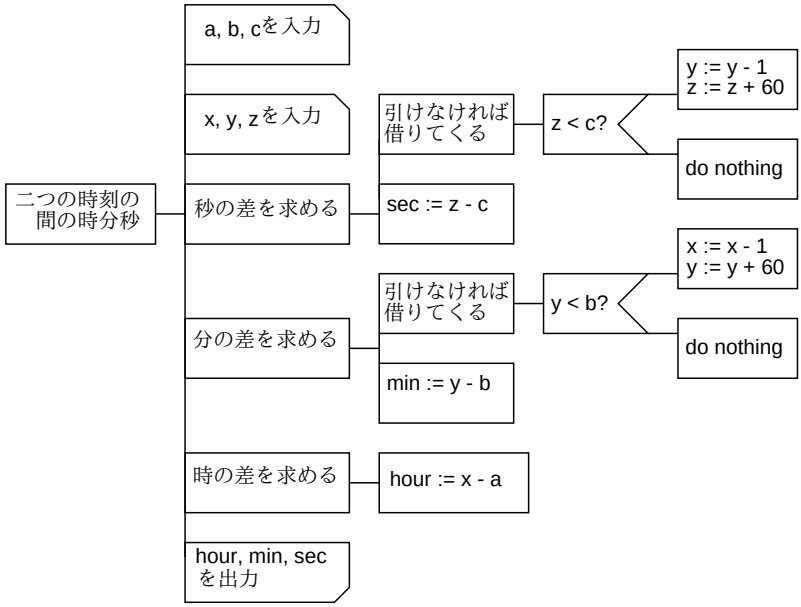

図 9: 時間差問題 (2)

こちらを Pascal に直すときには、ちょっと注意が必要である…というのは、条件の箱の下に「複数の文」がぶら下がっている。こういう場合は

その複数の文を全体として

```
begin 文 ; 文 ; … ; 文 end
```

のように囲んでやらないといけない。begin と end というのはいわば、複数の文をまとめて「1 つの文だよ」ということにする「スーパーかっこ」である。ではプログラムを示そう。

```
program sam5d1(input, output);
var a, b, c, x, y, z, hour, min, sec: integer;
begin
  write('a = '); readln(a);
  write('b = '); readln(b);
  write('c = '); readln(c);
  write('x = '); readln(x);
  write('y = '); readln(y);
  write('z = '); readln(z);
  if z < c then begin y := y - 1; z := z + 60 end;
  sec := z - c;
  if y < b then begin x := x - 1; y := y + 60 end;
  min := y - b;
  hour := x - a;
  writeln(hour:1, '時間', min:1, '分', sec:2, '秒')
end.
```

else の前と同様、end の前にも「;」は不要である (難しいですね? 次回にもう 1 回やりますので)。

演習 3 次の動作を行う PAD を作成せよ。

a. ☆ 2 つの数値 (整数) を読み込み、小さくない順になるように、並べかえて出力せよ。

b. △ 3 つの数値 (整数) を読み込み、小さくない順になるように、並べかえて出力せよ。

c. △ 西暦を年号 (昭和とか平成とか) 表示に変換せよ。明治より前は扱わなくてよい。

d. △ 年号表示を西暦に変換せよ。明治より前は扱わなくてよい。年号の種類は適当に数値化して入力する (例えば明治=1、大正=2 など) ように設計せよ。

演習 4 上の PAD を Pascal にして動かせ。

5　お絵描きについて

さて、いよいよお絵描きのお話に行きましょう。まず最初に説明しなければならないのは、計算機でお絵描きをする時には「ペイント系」と「ドロー系」の 2 つの系統があって、これらは全然違っているということである。この名前の由来は、Apple Computer という会社が Macintosh を最初に世に出した時一緒に出した 2 つのお絵描きソフト MacPaint と MacDraw から来ている。

考え方からいくとペイント系の方が分かりやすい。つまり、皆様が見ている画面というのはブラウン管の上で「様々な色の点の集まり」でできている。だから、それぞれの点に色々な色を割り当てれば任意の絵が作れますよ、というのがペイント系ツールの基本姿勢なわけである。とりあえず、今日は「ペイント系」のソフトである `xpaint` を体験して頂く。

5.1　xpaint 入門

`xpaint` を起動するには Kterm の窓で

```
% xpaint &
```

と打ち込む。しばらくすると、図 10 のような窓が現われる。

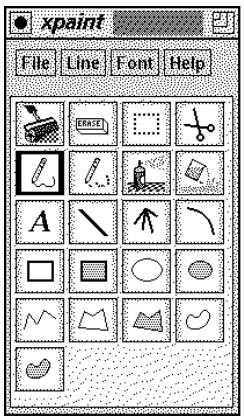

図 10: xpaint のツールパレット窓の様子

これを「ツールパレット」と呼ぶ。さて、以下では当面、特に述べない限りマウスの左ボタンを使用すること。キャンバスを開くには、「File」メニューの上でボタンを押してメニューを出し、中から「New With Size」を選ぶ。すると、幅と高さと拡大率を打ち込む窓が出て来る。

ここで、あんまり大きい絵は原理が分かりにくいし描きにくいので、練習用に幅と高さは 64、拡大率は 8 を入力して欲しい (どれも既に値が入っているところをマウスでクリックして [DEL] キーでもとの値を消して新しい値を打ち込む)、最後に OK のボタンをつつくと図 11 のような四角い窓が現われる。

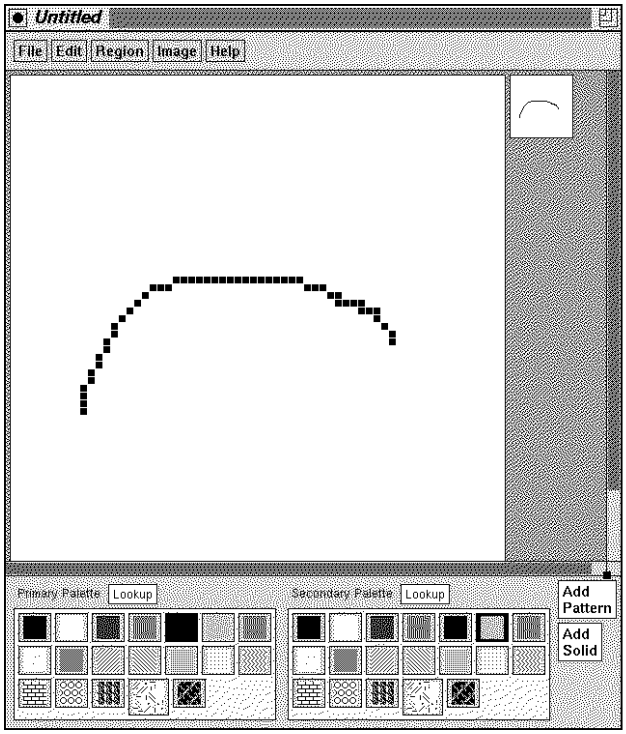

図 11: xpaint のキャンバス窓の様子

キャンバス窓の下には 2 つの「色パレット」があるがが、これは左側が鉛筆や筆の色、右側がスクリーントーンの色と思えばよい。とりあえず両方のパレットとも模様ではない色 (で互いに違うもの) をクリックして選んでおくこと。

つぎに、ツールパレットから「鉛筆」マークを選び、キャンバスの上でどれかのボタンを押して動かす。すると、鉛筆の動いた後の点に色がついて残る。鉛筆の太さを変えたい時はパレットの「Line」メニューを使えばできる。

もっと太い線を描きたい場合にはパレットから「ローラ」を選んで使う。ローラの形はいく通りかに変えられるが、変えたい場合にはローラのパレットを右ボタンでつつくとメニューが出て、その中から「Select Brush」を選べばよい。

ローラの隣にあるのは「消しゴム」で、これは間違って書いてしまったところを白に戻すのに使う。消しゴムもローラと同様形が変えられる。「スプレー」はやってみれば分る通りスプレーペイントのような効果がある。「バケツ」は、閉じた形を作ってその中でバケツをこぼすと色が行き渡る。やってみて「失敗した!」と思ったときは、キャンバスの「Edit」メニューの「Undo」を選ぶと 1 回ぶんだけは操作を元に戻すことができる。

あと、直線や四角などの図形のパレットは idraw と同じように使えるが、ただしペイント系なので一度描いてしまったものは動かしたり消したりできない。なお、図形で中が灰色のものは、描いた後で中を色やパターンで塗る。その色やパターンはキャンバス窓の下右側のパレットで選ぶ。文字は文字が描けるが、ただし xpaint では漢字は使えない (どのみち文字はあまり重要でない)。

さて、キャンバス窓の色パレットについてだが、もっと別な色を使いたい場合には右下の「Add Solid」と書かれたボタンを押す。すると図 12 のような窓が現われるので、ここで円形のうち望む色のところを左ボタンでクリックするか、またはスライドレバーのところでマウスの中ボタンを押して動かし、望む色が出た所で「OK」をクリックすればその色がパレットに追加されて利用可能になる。その他の機能については、「Help」ボタンなどを利用して調べて欲しい。

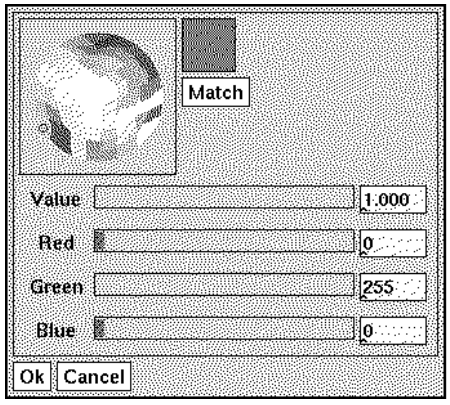

図 12: xpaint の色選択窓の様子

最後に、絵が完成して保存する時はキャンバス窓の「File」メニューから「Save As」を撰ぶ。すると、ファイル名を打ち込む窓が開く。ここで、絵を保存するときの様々なファイル形式が選べるが、当面は「GIF」(Grafics Interchange Format) と呼ばれる形式を使っておく。標準では TIFF のボタンが選ばれているはずなので、「GIF」のボタンを押して選び、それから「Save In File:」の入力欄にカーソルを持っていって「`test.gif`」のよ

うにファイル名を打ち込んで [RET] するか「OK」をつつく。GIF ファイルの場合、ファイル名の最後は必ず「.gif」で終わらせておくこと。

5.2 絵のプリントアウトと利用

できたファイルを見るのには「xv」というプログラムを使うとよい。使い方は

```
xv test.gif  &
```

のようにファイル名を指定して起動するだけである。すると、画面上に窓ができて作成したイメージが表示される。これを大きくしたりプリントしたければこの窓をマウス右ボタンでつつく。そうすると図 13 のような制御パネルが現れる。ここで「Image Size」のメニューをつついて「Double

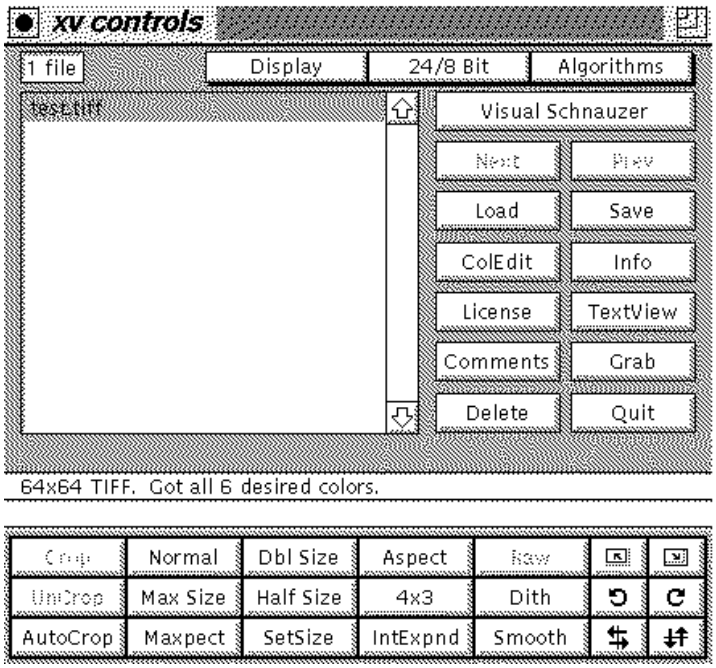

図 13: xv の制御パネルの様子

Size」を選ぶと倍に大きくなったりする。また印刷したければ次の手順による。

1. 制御パネルの「Print」ボタンをつつく。
2. プリンタ名打ち込み窓が現れるので、「lpr」の後に空白をあけてから「-Plw11」などのようにおなじみのプリンタ指定を打ち込む。

3. 「GrayScale」ボタンをつつく (GrayScale とは灰色中間色つきの意味。残念ながらカラープリンタはこの部屋にはないので…)。
4. 「xv postscript」という窓が現れるが、そこで紙の方向 (Portrate — 縦づかい、Landscape — 横づかい) の好きな方、紙のサイズ (A4 にするべきでしょうね) を選んで「OK」をクリックする。

なお、用が済んで xv を終らせるには制御パネルの「Quit」をつつく。

さて、絵が作れて印刷できるようになった。ではこれを何に使おうか? とりあえず、窓の背景模様にしよう。それには

```
xloadimage -onroot test.tiff
```

を実行すればよい。login したらいつでもこの絵を背景模様にしたい場合は、.xsession ファイルの xsetroot の行を消して (消すのがいやなら先頭に「#」を入れれば無視されます)、代わりに

```
LD_LIBRARY_PATH=/usr/X11R6/lib  xloadimage -onroot test.tiff
```

という行を入れればよい (ファイル名は自分のに適宜変更すること)。

演習 5 ☆ xpaint を使って、背景模様用の絵と、自分のトレードマークないしシンボルマークとを作成せよ。(面倒なら 1 つで両方兼ねてもよい。) そして、できた絵を xv で 8 倍拡大くらいにしてプリンタ出力してみよ。また、背景模様は.xsession を変更して login 時に自動設定させよ。なお、今回やった 64 × 64 ドットでは物足りないならもっと大きくしてもいいが、大きい絵ほど美しく作るのは大変なのでそのつもりで。大きくした場合、拡大率は減らさないと画面に入り切らない。

A　本日の課題 5A

本日の課題は演習 1(どんな風に整理したか書く)、演習 3 の a のみ (PAD 図だけでもいい。できて動いた人はプログラムも)、演習 5 の絵の出力 (好きなもの 1 枚でよい) を提出してください。レポート番号は 5A、アンケートは次の通り。

- Q1. if による枝分かれについて、納得しましたか? まだだめですか?
- Q2. 絵を描いてみてどのような感想を持ちましたか? 想像していたのとどう違いましたか?
- 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 5B

次回までの課題は、背景とトレードマークの完成 (トレードマークのみ印刷出力)、および演習 3 と 4 の選択肢 2 つ以上 (a を含めてもよい) を完成させる (PAD 図と Pascal と実行例) です。レポート番号は 5B、アンケートは次の通り。

- Q1. プログラムについて「なんにも知らない状態」から現状までの自分の変化 (どんなことならできるようになったとか、計算機に対する見かたの変化とか) を簡単に述べてください。
- Q2. xpaint でうまく絵を描くための「コツ」は自分なりにいえばどういうことですか?
- Q3. その他、感想、要望、質問があればどうぞ。