「情報処理」1年文I/IIクラス9-10　＃4

久野　靖*

1994.11.14

0　本日の目標

アンケート回答を頂いた中に「自分はプログラムを書くつもりはまったくない」と豪語されてる方がいらっしゃいました。それはそれで構いませんが、プログラムが書けなければ絶対単位は出ませんのでそのつもりで。この科目は本来プログラミングを教えるということで頼まれているのでして、他の内容はすべて「余録」であることをご理解ください。ともあれ、本日の目標は次の通り。

- 電子メールの送受ができるようになる。
- 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の練習

1　電子メールの送受

今回は、loginした後の様子がいつもとちょっと違うことに気がつかれましたか? そう、郵便箱の色が違いますね。実は、郵便箱は郵便(電子メール)が来ている時に知らせてくれるという機能を持っている。これからはloginした時郵便が来ているようなら、すみやかに読むように。

1.1　電子メールの基本概念

まず、電子メールとは基本的には図1のように、Muleなどで書いたテキストファイル(英語でも日本語でもいい)を特定の相手に向かって送る機能のことである。受け取った方はそのメッセージ

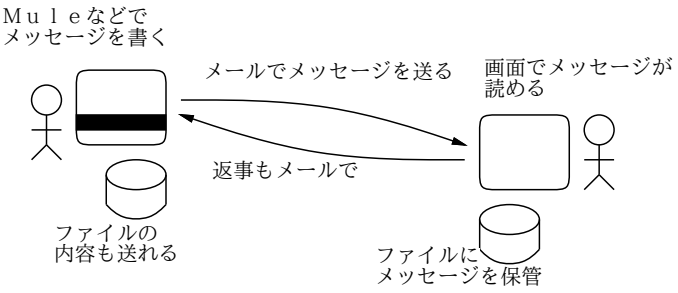

図 1: 電子メールの概念

を読み、必要なら返事を(やはり電子メールで)送ることができる。電子メールの特徴は次の通り。

*筑波大学大学院経営システム科学専攻

- 相手がその場にいなくても、メールを送っておけばあとで読んでもらえる。要するにお手紙と同じに使える。
- 電子的に送るので、相手が別の場所や外国などにいても、瞬時に配送される。当面皆様は海外と文通するわけではないが、たとえば私に質問があればメールでくだされば私は東大に来ない日でも地元で読んで返事が出せます。
- プログラムやその他のデータなど、計算機のファイルになっている情報なら簡単に送れる。だから友人にプログラム課題のコピーを送るのは簡単ですが、悪のりしないこと。

さて、メールを送る方はただ送ればいいだけだが、受け取った方は沢山のメッセージをどう整理するか工夫する必要がある。以下我々は MH と呼ばれるシステムを用いてこれを行う。MH では、メールが届いたあとまず「未決」(inbox) と呼ばれるフォルダ (メールをためておく場所) に取り込み、そこで順に見て整理する。具体的には、見てもういらないメッセージは消してしまい、保存しておきたいメールは種別ごとに別のフォルダに分けて整理する。(ずっと未決に置いておくのはずぼらなのでおすすめしない。) もちろん、後で各フォルダに保存したメッセージを再度見たり別のフォルダに移したり改めて消したりすることもできる (図 2)。

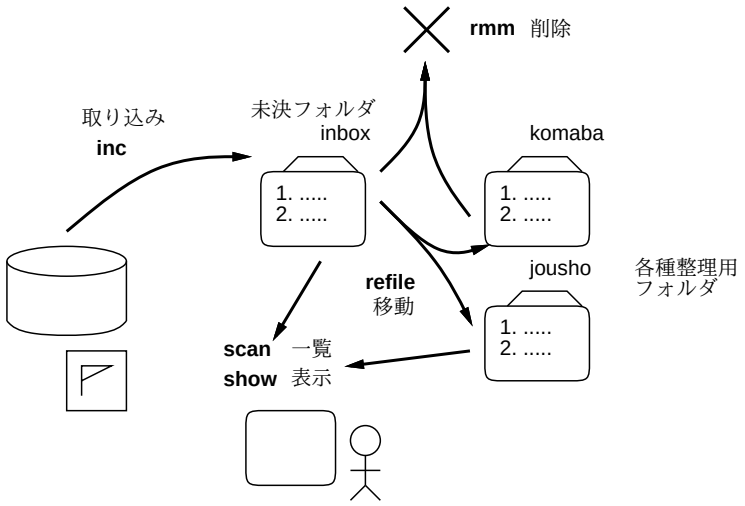

図 2: MH によるメールの扱い

では前置きはそれくらいにしてやってみよう。

1.2 電子メールを取り込む

電子メールが来ている場合には、まず来ているメールを未決 (inbox) フォルダに移す。それには

```
% inc      --- メッセージの取り込み
```

と打つ。本当に始めてメールを使う時は「`inbox` フォルダがないけど作るか?」と聞かれるので「`y[RET]`」と答えるように。

`inc` を実行すると新たに未決フォルダに入ったメッセージの一覧が表示されるが、それ以外にいつでも

```
% scan      --- メッセージの一覧表示
```

と打つことでそのフォルダにあるメッセージの一覧を見ることができる。実は未決に限らず、いくつかフォルダを作った場合

```
% scan +フォルダ名      --- フォルダ指定しての表示
```

と打つことで指定したフォルダに切り替えてその一覧を見ることができる。その後で改めて inbox に切り替えたければ

```
% scan +inbox
```

とする。フォルダ名の前には常に「+」をつける必要があるので注意。inc を実行して新しいメッセージを取り込む時はフォルダは自動的に inbox に切り替わる。

1.3 メッセージを読む

さて、いよいよメッセージを読むには

```
% show 番号      --- メッセージを表示
```

とする。番号は scan などで表示されるメッセージ番号で、省略した場合は inc の直後なら最初に取り込んだメッセージが、また show の後なら最後に見たメッセージが再度表示される。いちいち番号を覚えているのが面倒な場合のため、

```
% next     --- 次のメッセージを表示
% prev     --- 前のメッセージを表示
```

があり、それぞれ最後に見たメッセージの次ないし前のメッセージを表示する。

メッセージの表示には less が自動的に使われるので、1 メッセージ読み終わったら「q」で less を終わらせる必要があることに注意。

1.4 メッセージの整理

メッセージを整理する場合には次の 2 つを使う。

```
% rmm 番号                      --- メッセージの削除
% refile 番号 +フォルダ      --- メッセージを別のフォルダに移す
```

メッセージ番号を省略すると、最後に表示したメッセージが削除ないし移動される。削除はそのメッセージを取っておかなくてよい場合に使う。refile はメッセージを別フォルダに保存する。例えば友人ごと、科目ごとなどに対応したフォルダを作るなどの流儀がふつうだろう。なお、これまでに存在しないフォルダを指定すると「そのフォルダを作りますか?」と聞いてくるので「y[RET]」と答える。

演習 1 inc で自分宛のメールを取り込み、scan で一覧を出してみる。また show で各メッセージの内容を読む。

演習 2 いらなそうなメッセージを rmm で消す。また残りのメッセージを refile でこの授業用のフォルダ (例えば+jouhou といった名前がいいかな?) に移す。「scan +フォルダ名」でそのフォルダに切り替え、show で再度メッセージを見てみる。

1.5 メッセージを送る

読み方は分かったので、次は送る方へ進もう。それには、comp(compose — 組み立ての意味) コマンドを使う。

```
% comp      --- メッセージの送信
```

を実行すると、その Kterm の窓の中が Mule に切り替わり、なおかつ先頭の方に

```
To:
Cc:
Subject:
-----
```

という内容が入っている。ここで、To:のすぐ後ろにカーソルを持っていってメールのあて先を書く。あて先は、このシステム内の人であればユーザ名を書けばよいが、他のシステムの人であれば

```
ユーザ名@tansei.c.u-tokyo.ac.jp
```

などのようにシステムのところ番地を正確に指定しなければならない。複数の人にまとめて送りたい場合には「,」で区切って何人ものあて先を書いてよい。Cc:(Carbon Copy) の後には何も書かなくてよいが、例えば自分のユーザ名を書けば送ったメッセージの「控え」が自分にも転送される。

次に、Subject:(主題) の後にそのメッセージの主旨を 10 文字程度で書く。ここのシステム内あてなら漢字を使ってもよいが、外部あての場合は漢字はやめておいた方がよい (伝達が保証されない)。

そして、「-----」のところは変更しないこと。その後に自分の書きたい内容を (今度はいくらでも漢字を使ってよい) 書く。例えば次のような具合である。

```
To:g00002
Cc:g00001
Subject:test...
-----
This is a test. テストです。
```

なお、本文を改行 ([RET]) を入れずにどんどん打って行くと画面の端まで来たところで継続行マーク (「\」) が出て折り返されるが、Unix ではこのような長い行はトラブルの原因になる (例えば文字化けしたりちょん切れたりする) ので、必ず自分で適宜改行を入れて、継続行マークが出ないようにする。日本語ワープロに慣れている人は特に注意すること。

メッセージが完成したら、これまで通り^X^S で保存し^X^C で Mule を終わらせる。すると

```
what now?
```

と聞かれるので、「send」と応答すると無事メッセージが送られる。気が変わってやめたければ「quit -delete」と応答する。

ところで、メッセージは新たに書くことよりも、来たメッセージに返事を出すことの方が多い。その場合には comp の代わりに

```
% repl     --- 返事を出す
```

を使うと、最後に show で表示したメッセージに返事を出せる。その場合は To:や Subject:はもとのメッセージから自動的にセットされるのでわざわざ打ち込まないでも済む。また、来たメッセージが自動的に引用されるので、適宜いらない所は削って使うこと。

演習 3 自分で自分あてに電子メールを送ってみよ。しばらく待って郵便箱の色が変わるのを確認し、取り込んで読んでみよ。

演習 4 OK なら、前後左右の隣に座っている人に了解を得てメッセージを送ってみよ。またもらったメッセージの返事を書いてみよ。いらないメッセージは消し、保存したいものは適宜フォルダにしまうこと。

1.6 実名等の登録

さて、メールなどを送るようになると、その先頭部分に自分の実名などが入って欲しいですね? (最初はユーザ名しか登録されてない。) それには、chfn コマンドを使う。

```
% chfn
....
Name [g00001]: Komaba Tarou
Password:(パスワードを入力)
NIS entry changed...
%
```

もちろん、ちょっと席を離れている間に勝手に変えられると困るのでパスワードによる確認がある。あと、パスワードと同様一度に沢山の人が変えようとすると失敗したり、うまく変更できてもメールの上に効果が現われるのに 1 日くらい掛かることがあるなどの欠点がある、まあ気長に変えておいて欲しい。

あと、メッセージの末尾に自分の名前等 (signature — サインのこと) を入れる人も多い。その場合には自分のホームディレクトリに「.signature」というファイルを作っておいて、これに例えば

駒場太郎/g00001@komaba.c.u-tokyo.ac.jp

のように名前等を入れておき、適宜挿入して欲しい。メールハンドラによっては自動的に挿入してくれるが、ここでは当面「^Xi^/.signature[RET]」で挿入するものとする。

2 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。

2.1 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。

2.2 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。

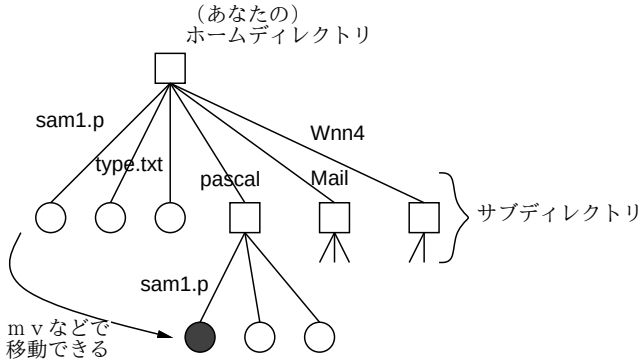

図 3: ディレクトリによるファイルの整理

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。

演習 5 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 6 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 を使って調べてみよ。

3　枝分かれ処理の復習

では例によって、前回の演習 4 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。

これも枝分かれは必要ですが、アプローチとして 2 通りある。まず、図 4 を見てほしい。つまり、

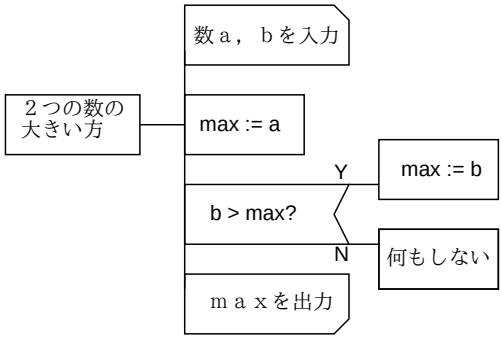

図 4: 2 つの数の大きい方 (1)

「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 sam4a(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。

```
% 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 5 のような PAD も考えられますね? この方が簡単でよさそうですか?

```
program sam4a1(input, output);
var a, b: real;
begin
  write('a = '); readln(a);
  write('b = '); readln(b);
  if a > b then
```

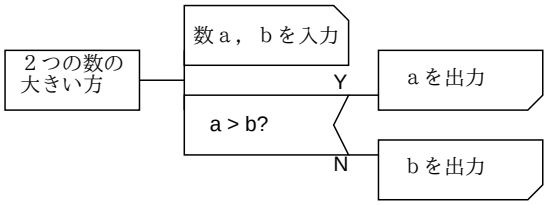

図 5: 2 つの数の大きい方 (2)

```
    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 6)。こ

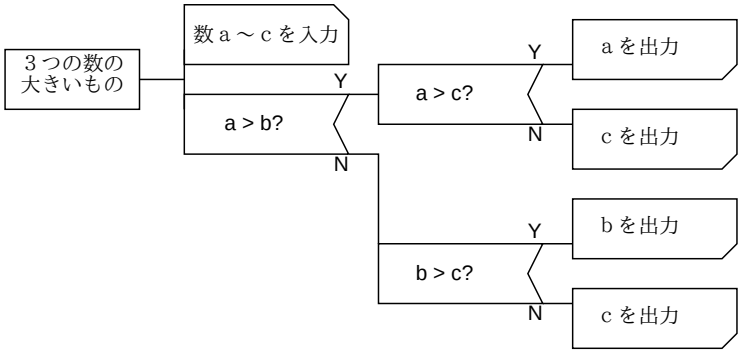

図 6: 3 つの数の大きいもの (2)

れを Pascal にすると次の通り。

```
program sam4c1(input, output);
var a, b, c: real;
begin
  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 7 のようになる。そして

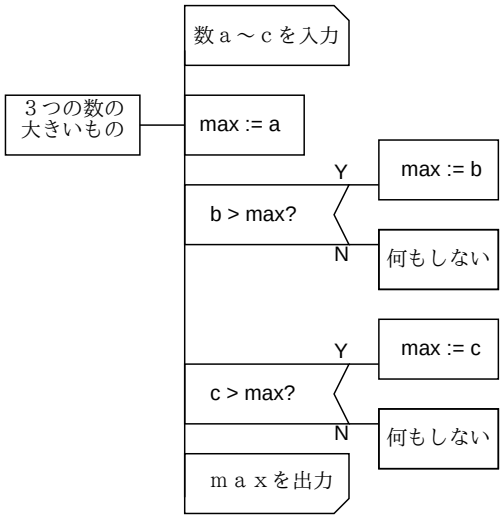

図 7: 3 つの数の大きいもの (1)

Pascal に直すと次のようになる。

```
program sam4c(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)
end.
```

こんどはどちらがいいですか?

4 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 8 に示した。

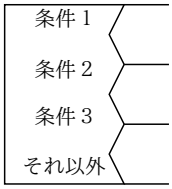

図 8: 複数選択の PAD の箱

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 9 のようになるだろう。これを Pascal にすると次のようになるだろう。

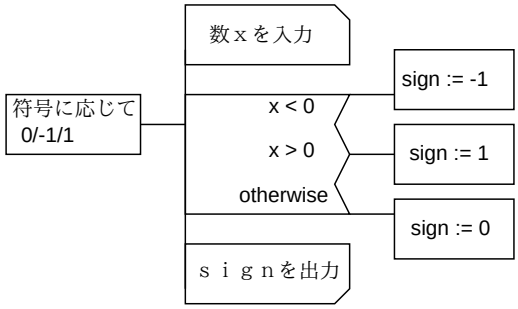

図 9: 符号に応じて 0/1/-1

```
program sam4b(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?

5 時分秒問題の 2 つの考え方

さて、「2 つの時刻の差を時分秒表す」問題も例解を示そう。まず PAD を図 10 に示す。これを

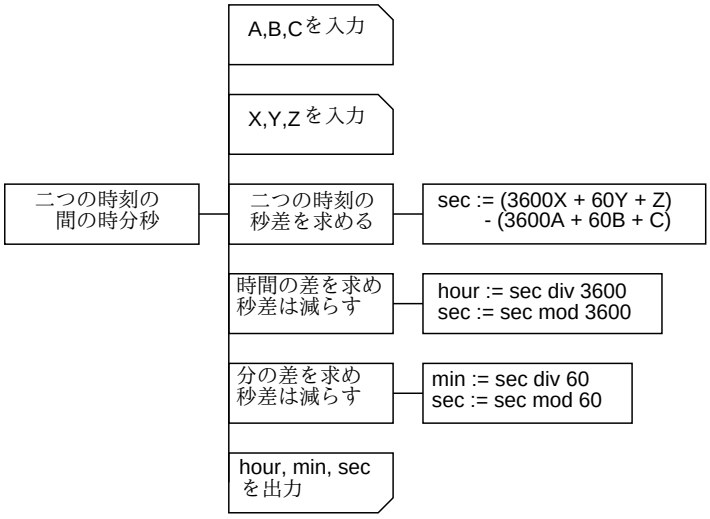

図 10: 時間差問題 (1)

Pascal にすると次の通り。

```
program sam5a(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, 'hour ', min:1, 'minutes ', sec:2, 'sec')
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 11 に示す。

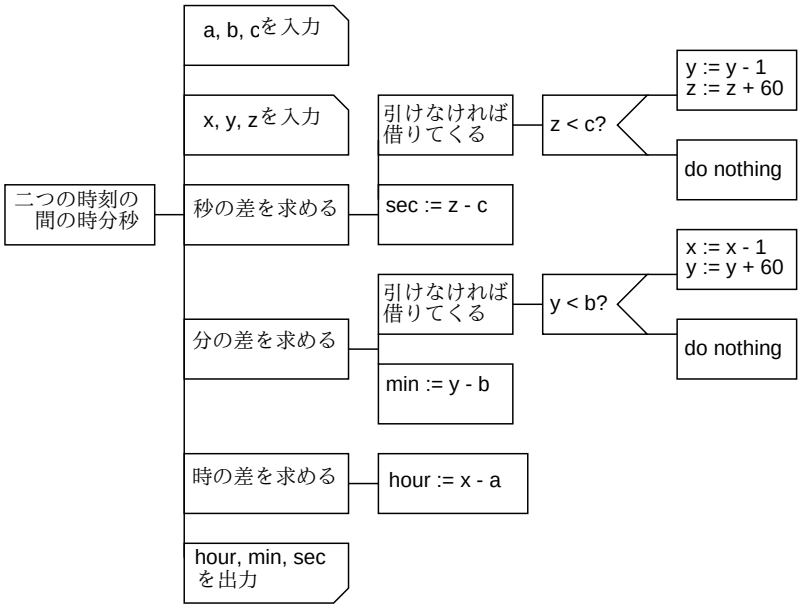

図 11: 時間差問題 (2)

演習 7 図 11 を Pascal のプログラムに直して動かせ。なお、if の枝に複数の動作を書く場合には全体を `begin` と `end` で囲んで「1 つの文」にする必要があることに注意。

演習 8 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 9 上の PAD を Pascal にして動かせ。

A 本日の課題 4A

本日の課題提出は、せっかく習ったのだから電子メールでお願いします。あて先は「`kuno`」です。内容はすべてアンケートで、次のものを書いてください。

- Q1. 電子メールを使った感想はいかがですか?
- Q2. ファイルをどのように整理しましたか?
- Q3. ディレクトリ Mail の下はどうなってましたか?
- Q4. その他、感想、要望、質問があればどうぞ。

Subject: には必ず「Report 4A」と入れてください。また、本文では Mule の「継続行マーク」が出ないように、適当なところで改行を入れてください。

B　次回までの課題 4B

次回 21 日は駒場際の後かたづけのため午前中のみ休講との指示を頂いていますが、本科目は午後なので通常通り実施されます。次回までの課題 4B は以下の通りとします。

- 本日の演習 7。プログラムのプリントアウトのみでよい。
- 本日の演習 8。ただし、a または b の好きな方、および c または d の好きな方の計 2 つでよい。
- 本日の演習 9。ただし演習 8 でやったものでよい。プログラムのプリントアウトと、実行例の画面ハードコピーの対で出すこと。
- 以下に述べる演習 10。どんなやりとりをしたかの概要を日本語で簡潔にまとめて提出してください。

演習 10 Rex Albi さんは米国の人で、たまたま日本人の Pen Pal を求めてらっしゃいます。私からお願いしたところ、e-mail の練習相手を快く引き受けてくださいました。つきましては、皆様それぞれ Rex さんに電子メールを書いてやりとりしてみてください。もちろん、英語でですよ。最低、こちらから何か (自己紹介プラスてきとーな話題を書いて) 送って、向うから返事をもらえること。それ以上のやりとりはノリに応じて判断してください。彼のアドレスは次の通り。

```
mrex@ix.netcom.com
```

アンケートは以下の通りです。

Q1. 「アルゴリズム」と「PAD」と「Pascal プログラム」の違いについて、自分なりに整理して書いてみてください。

Q2. 課題に対する感想と今後の要望をお書きください。

課題は、授業開始時刻までに、1F レポートボックスに提出してください。

C　参考文献について

そろそろ、Pascal の本が欲しいでしょうね? 本屋さんにいけば山と並んでいるので、ぱらぱらとめくってスタイルが気に入ったのを買って頂いて大丈夫です。(Pascal 言語そのものは比較的標準化されてるので、どれでも言語が違うという騒ぎにはなりにくい。) ただし TurboPascal とか ObjectPascal とか別の語が先にくっついてるのは避けた方が無難です。

また、Mule ですが今のところこれ専門の本はないので…一応、私の同僚のを推薦しておきます。「大木敦雄、入門 NEmacs、アスキー、1993」これで結構役に立ちますが、NEmacs と Mule で違うところもあります。なお、「入門 Mule」もまもなく出るのですが、こちらは説明されているかな漢字変換システムが Canna で、ここで使っている Wnn と違います。両方買ってあげれば彼も喜ぶでしょうけど…

また、環境設定全般については手前味噌ですが「久野禎子・久野　靖、UNIX の環境設定、アスキー、1993」があります。辞書登録などについても載っていますので。