「情報処理」1年文I/IIクラス9-10　＃3

久野　靖*

1994.11.7

0　本日の目標

前回アルゴリズム、プログラム、PAD、Pascalを一通りやりましたが、駆け足だったので結構苦しかったと思います。今回はこれらを一通り復習し、あとそろそろ計算機で日本語が使えるようになりたいですね? 本日の目標は次の通り。

- 簡単なプログラムのPADが書けるようになる。
- 簡単なプログラムをPascalに直して動かせるようになる。
- Muleで日本語を打ち込めるようになる。

1　復習: PAD図の書き方

PAD図はアルゴリズムを記述する方法の1つで、あとでプログラムになりやすく、しかも人間にとって見てわかりやすいようにデザインされている。その要点は、

- 計算処理は長方形の中に、入力と出力はそれぞれ右上すみと右下すみの欠けた長方形の中に書く。
- 順番に行う処理は、縦に線を引いてこれに接して上から順に書く。
- ある処理をより詳しく記述 (詳細化) する時は、その箱から右に線を引き出してそこに描く。

では、前回の課題4を再掲し、そのPAD図を見ていただく。ただその前に注意して欲しいが、PADの描き方は人によって千差万別だから、これと同じものだけが正解というわけではない。正しいPADは何通りも書ける。その中で、いかに「美しい」のを描くかはセンスの問題。あなたの美的感覚を存分に発揮して頂きたい。

a. 二つの数a、bを入力し、その合計を出力する。

b. あるものの長さがXメートルYセンチZミリの形で表されていたとして、それをセンチ単位に換算する。(例えば1メートル10センチ5ミリ→110.5のように。)

c. 同じ日の2つの時刻A時B分C秒とX時Y分Z秒を読み込み、その間の時間を秒単位で出力する。

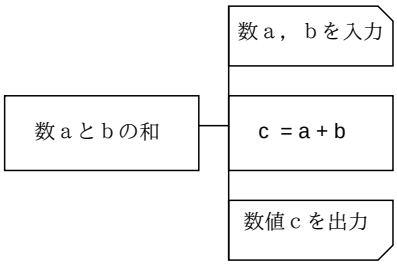

図 1: 「二つの数の合計」の PAD

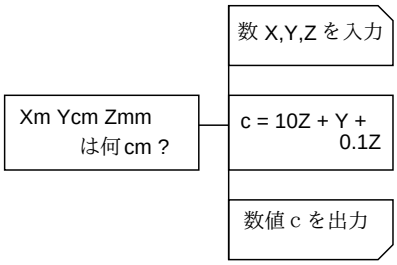

図 2: 「cm への換算」の PAD

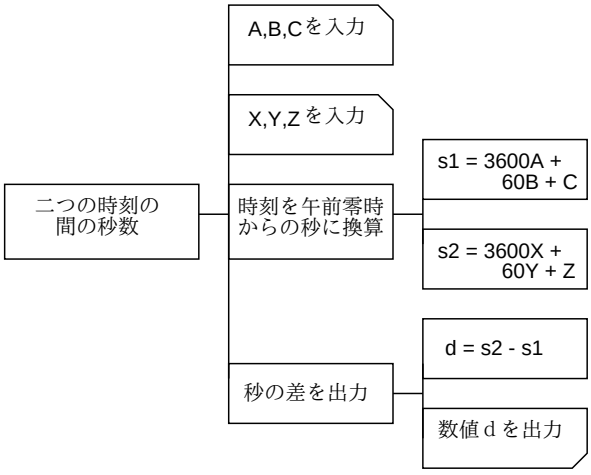

図 3: 「2 つの時刻の差」の PAD

自分で書いたものとくらべて、どうでしたか? 特に「c」が議論のあるところだと思うが。いちいち「秒に換算」だの「秒の差を出力」だの書かなくてもいきなり本体の計算や出力を並べておいた方が簡単でいいと思う人もいるかも知れない。(1) あなたがいつでも (1 年後でも) その PAD を見て「これは秒に換算してるのだな」と即解るくらい頭脳に自信があって、(2) その PAD をあなたしか見ない、ならばそれでもいいが、凡人の場合には多少まだるっこしくてもこれくらい解りやすく描いておく方がよい (それにどのみち他人である私が見ることになる)。

2 復習: Pascal プログラムの書き方

前回はあんまりちゃんとやらなかったので、Pascal プログラムの記法についても再度まとめておく。

2.1 Pascal プログラムの構造

Pascal プログラムは全体として次の形をしている。

```
program 名前 (input, output);
var 変数宣言; 変数宣言; …
begin 文; 文; … ; 文 end. ←最後の「.」に注意!
```

なお、名前とは英字 (アルファベット) で始まる、英字と数字の列である。(後で面倒を避けるため、とりあえず小文字だけを使っておいて欲しい。) 「…」と記してあるのは、変数宣言や文は 1 個以上何個あってもいいことを示している。

ところで、この形は一見前回の例とずいぶん違って見える。実は、名前や文字列 (後述) や演算子 (後述) の途中以外のところでは自由に空白、改行をいれてよい。だからまったく同じプログラムでもその見え方はかなり自由に工夫できる (再びセンスが問題になる)。

2.2 変数宣言

Pascal では入力した値や計算の結果を保持する「場所」のことを「変数」と読んでいる。(電卓のメモリーのようなもの。) 変数を使うには、変数宣言を書かなければならない (宣言、というのはつまり「使うからそのつもりでね」と計算機に教えることさっ)。その形は次の通り。

```
名前, …, 名前: 型
```

名前の規則は上述の通り。「型」というのは、その変数つまり「場所」にどんな種類の値を入れるかを示している。当面は次の 2 つの型のみを使用する。

```
real    --- 実数 (少数点つきの数値)
integer --- 整数 (少数点なしの数値)
```

整数は実数に含まれるのだから、全部実数で済ませればいいと思うでしょう? ところがどっこい。

- 切捨ての割り算、剰余などの演算は整数でしか使えない
- 整数の演算の方が計算機での実行が速いことが多い
- 整数の値の方が少ないビット数で記憶できる

*筑波大学大学院経営システム科学専攻

だから、整数と実数は適材的所で使い分けるように (センスの問題)。

ところで、プログラムの中には変数宣言は複数書け、1 つの変数宣言の中で複数の変数を宣言できる。だから

```
var x, y: integer;
```

でも

```
var x: integer;
    y: integer;
```

でも同じこと。どちらにするか? これもまたまたセンスの問題。一般的には、同じように使う変数は一緒にした方が見やすいだろう。ところで、「var」は変数宣言の並びの始まりを示すものだから、1 回だけしか書いてはいけないことに注意! 以下では変数として宣言した名前のことを「変数名」という。

2.3 文と値

文とは、プログラムの個別の動作を行う単位のこと。おおむね、PAD 図の様々な「箱」に対応する。今のところ、次の文だけ使っている。

```
readln(変数名, …, 変数名)  --- 値の読み込み
writeln(値, …, 値)         --- 値の出力
write(値, …, 値)           --- 〃、ただし行かえなし
変数名 := 値               --- 値を変数に書き込む
```

もちろん、「…」のところは変数名や値が 1 つでもいいことを示す。

ところで、write と writeln の「値」のところは

```
値:幅
値:幅:桁数
```

の形をしていてもよい。「幅」が指定されると、その値の出力を指定された幅 (文字数) に納めるように努力する。また、「桁数」は値が実数の時のみ使え、「幅」のうち何文字を少数点以下の表示に使うかを指定する。桁数を指定しないと、表示はすべて

```
1.234567e+08
8.765432e-01
```

のようなものになる。これは指数形式といい (関数電卓に見られる)、要は 1.234567×10^8 とか 8.765432×10^{-1} という意味である。ひどく大きい (小さい) 数値を扱う場合にはいいけれど、普段は 123456700.00 とか 0.8765432 とか表示してもらった方が嬉しいので、適宜幅と桁数を指定した方がよい。

2.4 値ないし式

さて、それでは「値」とは何だろう。プログラミング言語の世界では、値を書き表すための記述を「式」と呼ぶので、以下ではすべて「式」という用語に統一させて頂く。

'…'　　--- 文字列。今のところ write でメッセージ用にのみ使う
3.14 等 --- 実数定数。その値 (3.14 など) を表す
10 等　 --- 整数定数。その値 (10 など) を表す
変数名　--- 変数に入っている値を持ってくる
演算子 式 --- 演算した結果を表す (単項演算子)
式 演算子 式　--- 演算した結果を表す (2 項演算子)

単項演算子としては「-」(符号の反転) がある。2 項演算子としては、次のものがある。

+　　--- 足し算。整数用と実数用。
-　　--- 引き算。整数用と実数用。
*　　--- かけ算。整数用と実数用。
/　　--- 割り算。実数用。答えは必ず実数。
div　--- 整数除算。整数専用。
mod　--- 剰余。整数専用。

加減算より乗除算は「強い」ので、「2 + 3 * x」ではかけ算がまず実行されて、その結果に 2 を足す。それでまずい場合とか解りにくい場合には適宜「()」を使ってやる。
　実数用の演算子に整数の値を渡すと、自動的に実数に変換してくれるが、逆はだめ。その場合には次を使う。

trunc(実数の値)　--- 切捨てた整数の値をとる
round(実数の値)　--- 丸めた整数の値をとる

2.5　復習: Pascal プログラムの例

　では、先の「a」をプログラムにしてみる。

```
program sam2a(input, output);
var a, b, c: integer;
begin
  write('A> '); readln(a);
  write('B> '); readln(b);
  c := a + b;
  writeln('a + b = ', c:2)
end.
```

これが「sam2a.p」というファイルに入っていたとして、これを実行させる様子を示す。

```
%  pc sam2a.p
% a.out
A> 20
B> 10
a + b = 30
%
```

なお、この「%」のところはコマンドプロンプト (皆様の場合には「xss25{g00001}1: 」などとなっている部分) なので打ち込まないように。ここでは変数を整数にしたので、整数しか計算できない。ところで、同じアルゴリズムでも図 4 のように設計してもよい。その場合にはプログラムは次のようになる。

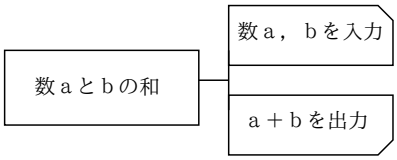

図 4: 「二つの数の合計」の PAD(その 2)

```
program sam2d(input, output);
var a, b: integer;
begin
  write('A> '); readln(a);
  write('B> '); readln(b);
  writeln('a + b = ', a + b:2)
end.
```

つまり、writeln の「値」のところに演算子を使っても全く問題はない。どちらがいいと思うかは、再びあなたの好みとセンスによる。ただし! PAD をなおさずプログラムだけこう直すというのはいけない。PAD と合っていないプログラムは後で始末に困るので (わかります?)。

演習 1 まだやってない人は、図 2 と図 3 の PAD 図をもとに Pascal プログラムを作成し、動かせ。

演習 2 (ゆとりのある人)「時刻の差」で、差が秒数というのはあんまり嬉しくない。差も x 時間 y 分 z 秒のように出るように直した PAD を作り、プログラムにして動かせ。(ヒント: 整数除算と剰余を活用する。)

3 判断と分岐

さて、ここまでに習ったのだとプログラムは上から順に処理していって、下まで来ておしまい、ですね? これでは大したバリエーションはできない。そこで次に、判断と分岐を習っておこう。例えば、次の問題を考える。

例題 3 数値 (実数) を入力し、その絶対値を出力する。

このような問題では、もらった値が「負かどうか」で処理を分けなければなりませんね?

PAD では、条件による処理の分岐は右側がぎざぎざの箱で表す。上の問題に対する PAD の例を示そう。このように、ぎざぎざの箱の中に条件を書き、その右に条件が成り立った場合 (Y) と成

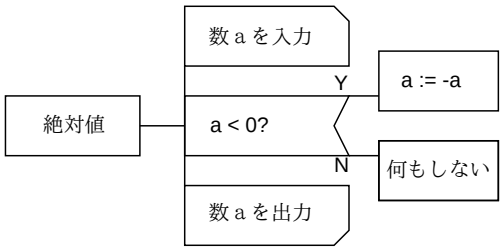

図 5: 「絶対値」の PAD

り立たなかった場合 (N) の処理を分けて描く。上の場合には、a の値が負だったら反転するので、

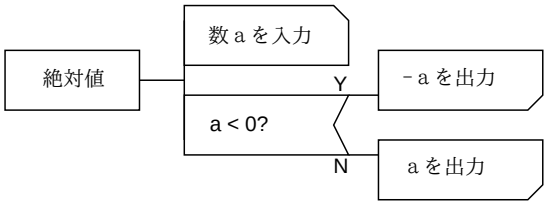

図 6: 「絶対値」の PAD(その 2)

もともと正の場合には何もすることがない。そこで「何もしない」と書いてあるわけだ。ところで、もう 1 つ別の考え方の PAD も示しておこう。これでもいいわけですね。あなたはどちらが好みか?

さて、今度はこれらを Pascal にする。そのために、さっき習った Pascal の規則に次のことを追加する。

- 2 項演算子には次のもの (比較演算子) もある。

  ```
  =     --- 等しい
  <>    --- 等しくない
  >     --- より大きい
  >=    --- 以上である
  <     --- より小さい
  <=    --- 以下である
  ```

  これらはすべて真偽値 (`true`=「はい」、`false`=「いいえ」のどちらかの値) を返す。

- 文の種類に次のもの (if 文) を追加する。

  ```
  if 式 then 文
  if 式 then 文 else 文
  ```

  「式」は真偽値を表すものでないといけない。その働きは、「式」の値が `true` であれば then の次の文を実行し、そうでなければ (あれば)else の次の文を実行する。だから、PAD で片方が「何もしない」ような枝わかれは上、それ以外は下の形に対応する。枝の中には「文」が 1 つしか書けないことに注意せよ。複数書きたい場合には下記参照。

- 文の種類に次のもの (複合文) を追加する。

  ```
  begin 文; … ; 文 end
  ```

  つまり、沢山の文でもその全体を begin-end で囲むと 1 つの「文」として扱われるので、if の枝に書くことができる。

では、上の 2 つの PAD を Pascal にしたものを示しておく。

```
program sam3a(input, output);
var a: real;
begin
  write('A> '); readln(a);
  if a < 0 then a := -a;
  writeln('absolute value = ', a:8:5)
end.
```

```
program sam3b(input, output);
var a: real;
begin
  write('A> '); readln(a);
  if a < 0 then
    writeln('absolute value = ', -a:8:5)
  else
    writeln('absolute value = ', a:8:5)
end.
```

実行結果はどっちも同じだから、片方だけ。

```
% pc sam3a.p
% a.out
A> -3.21
absolute value =  3.21000
% a.out
A> 1.235
absolute value =  1.23500
%
```

演習 3 絶対値の例題の、好きな方を打ち込んで動かせ。

演習 4 次の問題を解く PAD を設計せよ。

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。
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。)
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。

演習 5 演習 4 の PAD を Pascal にして動かせ。

4 日本語の入力

さて、これまで 3 回でだいぶキーボードになじんだと思うので、そろそろ日本語も打ってみよう。まず断わっておくが、皆様の端末についている「かな」キーは使用しない。ワープロでかな入力している人には泣いて頂くことになるので、ご勘弁頂きたい。以下すべてローマ字入力を前提とさせて頂く。[1]

4.1 ローマ字入力

日本語を打つには、Mule の窓で「Control-＼」を打つ。すると、窓の左下すみに「あ」と現われるはずである。これは「ローマ字モードに入ったよ」ということを意味している。ローマ字モードで再度「Control-＼」を打つと元の英字モードに戻る。

[1] ここで説明するのは、「Wnn」(うんぬ) と呼ばれるかな漢字変換系と Mule 上で動作する「たまご」と呼ばれるインタフェースを組み合わせたものである。Unix ではほかに「sj3」と「かんな」と呼ばれるシステムが広く使われているが、大筋はどれでも同様に操作できる。ただしカスタマイズなどの細かいところは違っているので注意。

ここで今度は「watasiha」と打ち込むと、画面には「わたしは」と出る。つまり、打ち込まれたキーから直ちにローマ字かな変換がなされて、変換結果のかなが画面に出るわけである。なお、ローマ字というのはまっかな嘘で、本当のローマ字なら「watasi wa」ですよね? つまり、かなを打ち込むための便宜的手段としてローマ字「のようなもの」を打つというだけだと割り切って頂きたい。例えば次のような変な便宜がある。

```
n     → ん (普通)
n'    → ん (普通)
N     → ん (うーん)
nn    → ん (いやなら止めることができます。)
-     → ー (長音の記号です。)
xa    → ぁ
xi    → ぃ
…
fa    → ふぁ
…                (あとは試して見てください。)
```

さて、打っている間その部分が「|」(たて棒) で囲まれているのにお気づきだろう。この状態を「フェンスモード」といい、まだ入力が完成していないことを示す。打ち込みたいのがひらがなであれば、ここで [RET] キーを打つとフェンスが取れて (「確定」という) 打ち込みが完了する。

4.2 漢字変換

では、漢字にしたい場合は…? その時は、フェンスモードの状態で [SP](スペースバー) を押す。すると、かな漢字変換サーバと接続され、漢字への変換が起動される (最初だけちょっと遅い)。しばらく待つと、漢字への変換が行われ、「私は」になる。この原理は単純で、計算機の中に読みと漢字表記の対応を収録した国語辞典のようなもの (変換辞書) があって、この中から読みのあてはまるものを探してきて置き換えてくれるわけである。

最初に出てきたものでよければ [RET] で確定するか、単に次の言葉を打ち始めるだけでよい (自動確定)。しかし日本語には同音異語が沢山あるので、1 発で成功しないことも多い。その時はさらに [SP] を押すと「渡しは」のように別の候補がでてくるので、次々に見て行けばよい。行きすぎた時は…? その時は^P で前の候補に戻れる。(実は [SP] の代わりに^N でも同じ。)

ところで、場合によってはいくら候補を見て行っても目的の漢字が出て来ないことがある。それには以下の 3 つの場合がある。

1. 区切り損ない: 例えば「不自由」を出そうとして「ふじゆう」と打ったのにこれを計算機が「ふじ/ゆう」と間違って分けてしまうと、「ふじゆう」→「富士-友」→「不治-夕」→…のように、いつまでも別の「ふじ」が出て来るだけになる。この場合には、区切り方を調整しないといけない。それには

   ```
   ^O  --- 文節のばし
   ^I  --- 文節縮め
   ```

   を使う。例えば上の場合は^I を 1 回打つと区切りが「ふ/じゆう」のように変わるので、あとはうまく行くはずである。

2. 後の節の選択: 上でぶじに「ふ/じゆう」になったあと「ふ」が「不」になったとして、「不-事由」になってしまったとしよう。「事由」を「自由」に直したいが、このまま [SP] を使っても先頭の「不」が変わってしまう。その場合は

^F --- 先の区切りへ行く
^I --- 前の区切りへ行く

を使う。この例では^F でカーソルを「事由」の所へ移してから [SP] を使えばよいことになる。

3. 辞書にない: 自分の名前などは固有名詞だから、もともと変換辞書に入っていないかも知れない。その時は、(1) まずそれぞれの漢字を含む単語を考えて変換し、いらない字は削ってしまってとにかく問題の漢字を作る。(2) 次回に備えてそれを辞書に登録してやる。

辞書への登録については少し後で説明する。

4.3 その他の字種

以上で「ひらかな」と「漢字」の入れ方はわかったが、では「カタカナ」は? それは、実は漢字変換すると最初にでる「直前の」候補は変換内容を全部カタカナにしたものなので、一度 [SP] で変換してすぐ^P で前の候補を出すとカタカナになる。しかしそれでも不便だから、「コンピュータ」などよくあるカタカナ語は変換辞書に入っていて普通に変換するとでてくる。だから自分の使うカタカナ語で変換辞書にないものが見つかったらすぐ登録してしまうことをすすめる。

普通の英字や数字は? それは、Control-＼でローマ字モードを抜ければ入れられますね?

ところで、英字モードの字は画面上では漢字の半分に見えるが、ローマ字モードで記号などを打ち込むと漢字と同じ幅の記号が出てくる。実は英字モード (1 文字を 8 ビットで表現) で使っている英数字や記号すべてと同じものが日本語モード (1 文字 16 ビット) の文字セットにも含まれている。そして、Mule をや Kterm をはじめ多くのソフトが 16 ビットの文字を 8 ビットの文字の倍の幅に表示し、[2]たとえば「A」と「A」が実は同じ文字だということをちゃんと理解してくれない (別のものだと思って扱う)。人間が見るぶんにはまあどちらでも用は足りるが、計算機に食べさせる場合には注意すること。たとえば Pascal のプログラムでは文字列の内部以外に 16 ビットの文字を使ってはいけない!

演習 6 以下の文章をできるところまで打ち込んでみよ。行かえとかは適当でよい。[3]

> 特別急行「富士」は、戦前のもっとも豪華な列車だった。
>
> 洋食堂車があり、一等寝台車があり、最後尾には展望車が連結されて、富士山をかたどったテール・マークが誇らしげに架けられていた。
>
> 展望車のデッキに立って、群れ集まった見送り人に鷹揚に会釈する高官、固い表情の軍人たち。腰にサーベルの警官や白手袋の駅長もいて、東京駅の 8 番線は物々しかった。
>
> 小学生の私は、それを遠くから眺めていた。「富士」を見たくて、幾度も東京駅へ行った。展望車には後光がさしているようで、眩しかった。
>
> 宮脇俊三、「シベリア鉄道 9400 キロ」より

[2]このため、8 ビットの文字を「半角」16 ビットの文字を「全角」と呼ぶ人が多いが、あくまでもソフトの勝手で半分の幅に表示しているだけなのだから本当はおかしい。

[3]日本語のかっこは [と] を打てば出る。「・」は「z/」で出る。

4.4 辞書登録

単語登録とは、先にも述べたように変換辞書に載っていない単語を追加することである。この場合、追加は各ユーザごとの個人辞書になされるので、他人と登録がぶつかる心配はない。一方、一度登録してしまえば南棟の Unix を使っている限りそれはずっと有効である。だから変換できない語やカタカナ語が見つかったらそのつど、すぐに登録することを勧める。

登録するには、まずその語を何とか別の方法で入力する。漢字については、その字を含んだ別の単語を考えてそれを変換して出し、いらない漢字は削ってしまうなどする。たとえば先の「鷹揚」だと、「三鷹」と「揚げる」から作るなどする。どうしても思いつかなければあまたある「ワープロ漢字辞典」の「JIS コード」が引けるものを購入して文字の JIS コードを調べ、

```
[ESC]Xjis[RET]xxxx[RET]
```

で入力する (xxxx のところに JIS コードを打つ)。

問題の語がともかくできたら、まずその先頭にカーソルを持ってきて`^W` を押し (ここで Mark Set と表示される)、次に登録する語の直後まで (つまりその語の文字数ぶん) カーソルを移動する。この状態で、

```
[ESC]Xtoroku-r[RET]
```

と打つと登録モードになり、最下行に語が表示され、読みを聞いてくる。読みはローマ字入力で普通に打ち込み、最後に `[RET]` する。次に登録辞書を聞いてくるがこれはただ `[RET]` する。その後品詞を聞いてくるので、自分の文法知識の範囲で正しく答えると、登録が完了するはずである。

演習 7 先の演習で出なかった語や自分/知り合いの名字、名前など変換できない語を最低 10 個以上探して登録せよ。

演習 8 自分の好きな日本語の歌を 1 つ撰び、その歌詞を打ち込め。ただし、できる限りタッチタイプするように。

A 本日の課題 3A

今日は「演習 1」(既に今日までに出してしまった人は「演習 2」) のプログラムと「演習 6」の成果、そして「演習 7」で登録した語のリストを lwp でプリントアウト (ハードコピーではない!) して提出してください。レポート番号は 3A です。アンケートは次の通り。

Q1. 簡単なプログラムなら書けるようになりましたか? あるいは、もう少しでなりそうですか? あるいは、簡単すぎてつまんないですか?

Q2. 日本語入力は始めてですか? もしそうなら、Mule/Wnn/たまごのシステムをどう思いましたか? 始めてでなければ、これまでに使ったシステムと比べて優れている/劣っていると思う点を列挙してください。(たぶん慣れていないので「最悪」に思えたでしょうけど。)

Q3. 本日の全体的な感想と今後の要望をお書きください。

B 次回までの課題 3B

次回授業開始までに「演習 4 の a〜c のうち 1 つ以上 (できれば全部)」、「演習 5(演習 4 でやったものでいい)」、「演習 8」を提出してください。PAD 図はレポート用紙に手書き、プログラムはファイルのプリントアウトと実行例のハードコピーの組で出してください。演習 8 はファイルのプリントアウトでお願いします。レポート番号は 3B です。アンケートは次の通り。

Q1. 前回とおんなじですが…プログラミングの課題はどれくらい大変でしたか? PAD 図を描くのと、Pascal に直すのと、打ち込んで動かすのとで掛かった手間の比率はどうですか?

Q2. 英字のタッチタイプとローマ字入力のタッチタイプとでは違いますか? 違うとすればどんな風に? また、どれくらいタッチタイプできるようになりましたか?

Q3. 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、次回授業開始時刻までに、レポートボックスに提出してください。