「情報処理」1年文I/IIクラス9-10　＃3

久野　靖*

1995.10.30

0　本日の目標

前回アルゴリズム、プログラム、PADまでやって、宿題でPascalの打ち込みをお願いしまいしたが、今日はもう1回前に戻って再度説明します。その後、電子メールをやりましょう。文通の課題つきです! 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。
- 電子メールが使えるようになる。

1　復習: Pascalプログラムの打ち込みと実行/△

とにかくプログラムの打ち込みと実行はできないと困るので、ここでまとめておく。わかんない人は自主的に練習すること。

1. Muleを起動する。
2. `^X^Ftest01.p[RET]`などとして、ファイルを指定。Pascalプログラムのファイル名は最後が「`.p`」でないとだめ。
3. プログラムを打ち込む。例えば前回の例題だと:

```
  program sam1(input, output);     ←「,」、「;」を忘れないこと。
  var c, f: real;                  ←「:」と「;」を間違えない。
  begin
```

*筑波大学大学院経営システム科学専攻

```
    write('degree F> ');            ←「'」は「"」や「`」と
違う!
    readln(f);                      ←すべて「;」を忘れない
こと。
    c := (5.0 * (f - 32.0)) / 9.0;  ←かっこ対応に注意。
    writeln('degree C = ', c:5:2)   ←「:」と「;」を間違え
ない。
  end.                              ←「.」を忘れないこと。
```

4. 打ち込み終ったら^X^S で保存する。Mule の窓はまた使うかもしれないから置いておいてよい。
5. Kterm の窓を出して、そこで「cat test01.p[RET]」などとしてファイルができてることを確認。
6. 「pc test01.p[RET]」などによりコンパイルする。メッセージが何か出たら問題がある。それをよく見て間違いをさがし、また Mule の窓 (残してあるはずですね?) へ行き 3 からやりなおす。
7. 何もメッセージが出ずに終ったら「a.out[RET]」で実行開始する。
8. 温度の数値を聞いてくるので打ち込んで [RET]。
9. 結果が表示される。おつかれさま!

2 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。

勝手に「でたらめな」記法をでっちあげた人が結構いましたが、PAD 図の規則にはそれなりの根拠があるので、でたらめのままだと行きづまると思います。まじめに覚えましょう。

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。

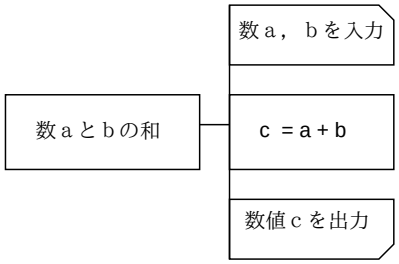

図 1: 「二つの数の合計」の PAD

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。)

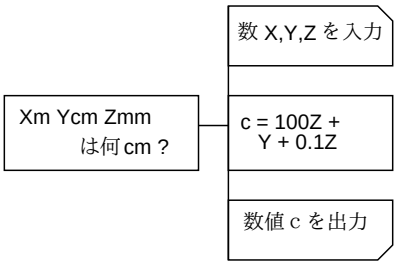

図 2: 「cm への換算」の PAD

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。

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

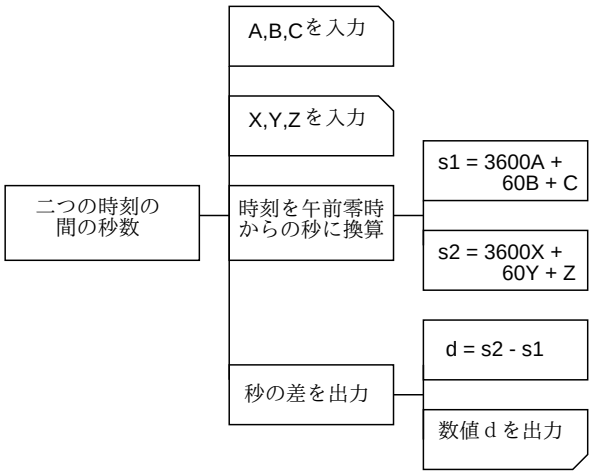

図 3: 「2 つの時刻の差」の PAD

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)。

3 Pascal プログラムの書き方/*

前回はプリントでいい加減な説明をしただけだったので、Pascal プログラムの記法についてきちんと説明しよう。

3.1 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)。

3.2 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

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。

3.3 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。

3.4 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
```

3.5 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
%
```

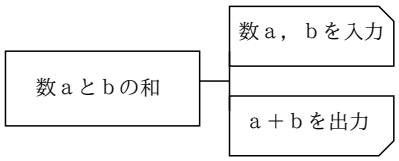

図 4: 「二つの数の合計」の PAD(その 2)

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。

```
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 のどちらか好きな方 (両方でもいい) について、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。)

演習 3 △ 次の問題をまず PAD にせよ。

a. 2 つの数 x、y を入力し、x を y で割った商と余りを出力。

b. 3 つの数 a, b, c を入力し、その平均を出力。

c. cm 単位の長さを入力し、XmYcmZmm の形で出力。たとえば 103.7 → 1m3cm7mm というふうに。(ヒント: まず、小数点つきの実数 X を切捨てて整数にするのに `trunc(`X`)` を使う。このため、整数型と実数型の両方の変数を使うことになる。その後は考えてみよう。)

演習 4 △ 演習 3 の各問題を Passcal に直して動かせ。

4 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。

4.1 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 5 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を読み、必要なら返事を

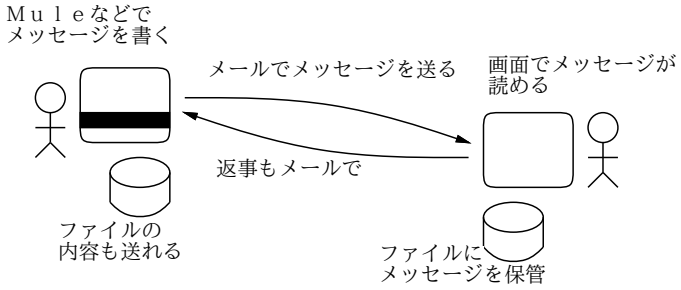

図 5: 電子メールの概念

(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。

- 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 6)。

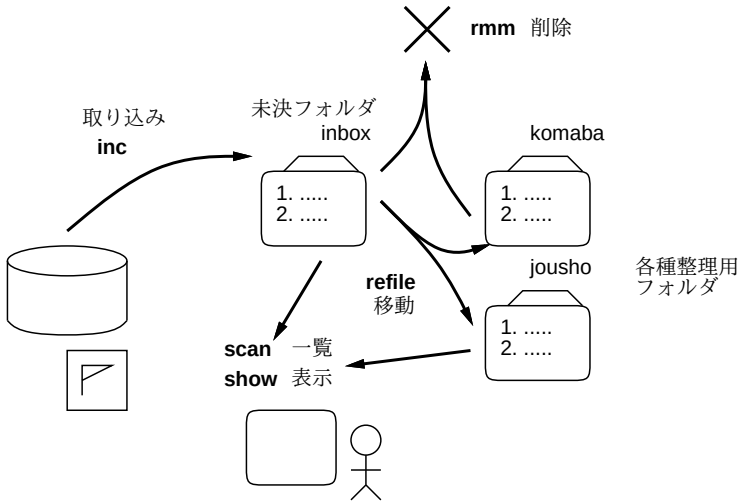

図 6: MH によるメールの扱い

では前置きはそれくらいにしてやってみよう。

4.2 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。

4.3 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。

4.4 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 5 ☆ 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 6 △ 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。

4.5 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 (来週漢字入力を習ったら、ここに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.
```

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[RET]`」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[RET]`」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 7 ☆ 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 8 △ 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。

4.6 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。

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」か「演習 3」) のプログラムから 1 つを 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. 電子メールのようなものは始めてですか? もしそうなら、(そうでなくても) どのような感想や意見を持ちましたか?

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

次回までに「演習 3 と 4 の a〜c のうち 1 つ以上」をやってください。1 つのプログラムについて、PAD(手描き)、Pascal プログラム (プリントアウト)、実行例 (画面ハードコピー) が必要です。

あと、皆様の電子メールの練習台になって下さる方を海外で募集しておきました。各自あて今日送ったメールの中に、それぞれ 3 名の「海外ペン

パル」のメールアドレスが記されているはずです。それらの人にメールを出し (もちろん英語!)、返事をもらってください (忙しくて返事の来ない人もいるでしょう。だから 3 名ずつ割り当てたわけです)。プリントアウトは不要で、アンケート Q2 に回答すればいいです。

アンケートともきちんと記名し (裏面先頭行にレポート番号日付氏名を書くのを忘れないこと!)、授業開始までに提出してください。あまり内職が多いようだと、〆切を繰り上げますので、次回授業中にやろうと思わないこと (説明を 10 分聞くのを内職に費やしてあとで自分だけで勉強したら 1 時間掛かるんですよ。損だと思いませんか。) レポート番号は 3B、アンケートは次の通り。

Q1. プログラムを書くという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. 海外ペンパルさんとどのような対話をしましたか。簡単に紹介してください。

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。