情報処理 II No.4

レポートの書き方、 gnuplot の紹介、 Fortran の文字列処理

桂田 祐史

1995 年 5 月 19 日

今回までで、図形描画の話はひとまず終りにします。レポートの書き方に関する注意と、グラフを描くためのコマンド gnuplot の紹介、ある課題を解くために必要な Fortran の文字列処理機能の解説をしますが、基本的には課題を解くための実習時間にしたいと考えています。

1 レポートについて

今回の範囲では、プログラミング上の難しさはないでしょう。その分色々実験してみて下さい。(曲線の例を探したい人は、微積分や幾何学のテキストや数学公式集[1]などを見ましょう。)

1.1 提出の手続き

今回までに出された問題のうち、 3 問以上 (出来る人はなるべく毛色の変わった問題をたくさん) を解いて、 5 月 25 日 (木) 午後 6 時までに提出して下さい。桂田 (6716B や 6701 号室にいることが多い) が不在の時は、数学科資料室 (6606 号室) の野町さん or 中里さんに渡して下さい。

1.2 レポート作成上の注意

例年こちらの注意を全く無視したレポートが散見されます。今年はそういうことがないようにして下さい。以下にあげる注意を無視した場合は減点することを考えています。

- プログラム・リストに名前を添えただけのものではなく、きちんとした報告書になっていることを要求します。
- 表紙をつけ、タイトル（情報処理 II 第 1 回レポート）、学年、番号、氏名、ユーザー名[2]、解いた問題の番号、提出年月日を明記して下さい。
- 自分でプログラムを書いた場合はプログラムのリスト、実験した場合は実行結果のプリント・アウト、また実行結果を得るのに用いた入力データ (もしあれば) を添えて下さい。他人がレポートを読んで、結果を再現出来るだけの情報をまとめて下さい。
- 何を解いたのか、プログラムではなく文章で、はっきり説明して下さい。例えば関数のグラフを描いた場合は、どの関数の、どこからどこまでの範囲のグラフを描いたのかを明記して下さい。プログラムを読まなくても分かるようにして下さい。

[1] 例えば、森口・宇田川・一松著、岩波 数学公式 I (微分積分・平面曲線)、岩波書店。

[2] “ee480??” というログイン時に入力する文字列のこと。

- どうやって解いたのかを説明して下さい。自分でプログラムを一から書きあげた場合はもちろん、例題プログラムを修正した場合も、どこをどのように修正したのか書いて下さい。
- 短くても構いませんから、結果についての分析や入力データ（例えばグラフを描く場合の区間の分割数など）を選んだ理由などの考察をそえてください。（「…となるはず」、「…となった」、「予想通りであった」、「おかしくなったのは○○のためと考えられる」、「…となることがわかった」などなど。）

採点の際に、君達のホームディレクトリィにおいてあるファイルを桂田が見ることがあります。レポートのために作ったプログラム等のファイルは 5 月中は消さないで下さい。

2 Fortran の文字列処理 — 問題 2-5 を解くために

問題 2-5 を解くにはどうしたら良いでしょうか？これは基本的には “mapdata” というファイルを読んで、そこに含まれている座標 (経度と緯度) を適当に変換すれば良いわけです。難しいのは、座標が書いてある行と、線を区切るための空白ラベル “" "” だけのある行と、二種類の行があって、どちらのタイプの行であるかは、それを読んでみるまで分からないようになっているということです。空白ラベルだけの行を “read(*,*) x,y” のように数値データを読むようにして扱うとエラーが生じてしまいます。

この問題を解決するには、最初に行データを文字列として読み込んでから、それがどちらのタイプの行であるか調べて、それに応じて処理を選ぶようにしなければなりません。それを Fortran プログラムで実現するには、 Fortran の文字列処理機能を勉強する必要があります。

2.1 文字変数

Fortran では、整数 (型名は integer) や実数 (浮動小数点数。型名は real = real*4, double precision = real*8, real*16 等)、複素数 (型名は complex = complex*8, complex*16, complex*32 等) などの数の他にも、文字 (型名は character) を使うことが出来ます。

これまでも、 write 文で引用符 “"” や単一引用符 “’” で囲まれた文字定数を使ってきました。

```
write(*,*) ’ こんにちは、おひさしぶりです。’
write(*,*) " Fortran で漢字が使えるなんて、世の中変わりました。 "
```

しかし、文字定数だけでは不便です。 Fortran では文字変数や (parameter 文で定義する) 名前のついた文字定数も使えますので、マスターして大いに利用して下さい。

まず宣言の仕方ですが、「長さの指定」が必要です（文字というよりは、文字列と言った方がいいのかもしれません）。それは宣言文の文字変数の名前の後にアスタリスク（星印） “*” と 1 以上の整定数を指定します。長さ 1 の文字変数を宣言する場合は “*1” は省略できます。

```
character name*20,c,telno*12
```

2.2 基本操作

代入文による値の設定　数の場合と同様に “文字変数名 = 文字式” で OK です。なお parameter 文の中で値を設定する場合も同様です。

```
parameter (msg = ’ 今日も楽しく勉強しませう’)
name = ’ かつらだ まさし’
```

read 文による値の設定　これも数の場合と同様です。

```
write(*,*) ’ 電話番号を教えて下さい。’
read(*,*) telno
```

比較　“.eq.”, “.ne.”, “.lt.”, “.le.”, “.gt.”, “.ge.” 等を使って「文字関係式」が作れます。 短い方の文字列の末尾に空白を補って、同じ長さにしてから判定します。 “.eq.” では等しいかどうかを判定します。大小関係は ‘ ’, < ‘0’ < ‘1’ < ‘2’ < ... < ‘9’ < ‘A’ < ‘B’ < ... < ‘Z’ のようになっています。

注意 1:　他の多くの言語のように、動的に長さの変化する文字列を表現することは出来ません。余った部分は空白 ‘ ’ で埋められます。例えば “name*20” と宣言した場合 “name=’Katsurada’” と長さ 9 の文字列を代入すると、 “name” の内容は “Katsurada ” と末尾に 11 個の空白が付いたものになります。また、長さが不足した時は、入り切らない文字はカットされます。例えば “tleno*12” と定義している時に “telno=’1234-567-8901’” と長さ 13 の文字を代入すると、最後の “1” が落ちて、 telno の内容は “1234-567-890” となります。

注意 2:　日本語の文字 (漢字やひらがな等) は一文字の長さを 2 とします (“The テレビジョン” という文字列の長さは $3 + 1 + 2 \times 6 = 16$)。幸い (?) 現在の環境では kterm に表示した時に、日本語文字は普通の英数字の 2 倍の幅に見えるので、「文字列の長さ」は画面に表示した時の幅だけ必要になると、覚えればいいでしょう。

2.3　内部ファイル

問題 2-5 を解くプログラムで 1 行の内容を文字変数 “line” に読み込んで、それが 2 つの数値データを含む行だと分かったとして、その値を実数型の変数 “keido”, “ido” に設定するには、「内部ファイル」という機能が使えます。具体的には

```
read(line,*) keido,ido
```

とすれば OK です。つまり “read” の読み込み元として、標準入力 (“*” で指定), 装置番号 (0 以上の整数で指定) 以外に、文字変数が使えるわけです。

同様に “write” 命令の出力先にも文字変数を指定することが出来ます。

2.4　その他便利な機能

部分文字列　“telno(1:3)” のようにすると、文字変数 telno の 1 文字目から 3 文字目までが取り出せます (部分列名)。

連結演算子　“//” という演算子で文字列を連結できます。

```
s=’ 私の名前は’ // name // ’ で、電話番号は’ // telno // ’ です。’
```

index　“index(s1,s2)” s1 の中から s2 を探し、見つかれば位置を、そうでなければ 0 を返す。

index(’ABCDEF’, ’CD’) は 3 を返す。

index(’ABCDEF’, ’GH’) は 0 を返す。

index(’ABCDEFCDEFCDEF’, ’CD’) も 3 を返す。

その他の組み込み関数

len(s) 文字式 s の長さを返す。

ichar(c) 文字 c の文字コードを返す。

char(i) 整数 i を文字コードとする文字（長さ 1 ）を返す

2.5 例題プログラム

“getsample” とすると “hint2-5.f” というファイルが手元にコピーされます。これは問題 2-5 のヒントとなるプログラムです。上に説明した文字変数の使い方の例として参考にして下さい。コンパイルして実行すると “mapdata” を読み込んで “mapdata.new” というファイルを作成します。これは経度・緯度のデータを直角座標系（北極を $(0,0,R)$, 南極を $(0,0,-R)$、緯度と経度が共に 0 の点を $(R,0,0)$ としたものです。ここで R は地球の半径を km 単位で表した値のつもりです。）のデータに変換して、 y, z 座標だけを出力したものです。

こうして出来たデータを mgraph にかければ、 No.2 のプリント 6 ページの「透明な地球」が表示されます。

```
cat mapdata.new | mgraph -b | xplot
```

問題 4-1: 上の例題プログラムに次のいずれかの改良をしてみよ。 (1) (不透明な地球) 裏側も描いてしまって透けて見えるのを直す。 (2) (まわる地球) 地球上のある点を経度、緯度で指定して、それが図の真ん中に来るようにする。

3 gnuplot

前回までに一変数関数のグラフや平面曲線の描く方法を説明しましたが、これだけだと詰まらないので、二変数関数のグラフ等を描くのに便利なコマンド **gnuplot** を少しだけ紹介しておきます。

起動 次の二つの方法があります。

1. gnuplot
2. gnuplot コマンド・ファイル名

終了 exit または quit コマンドを使います。

コマンドファイルの読み込み load コマンドを使います。

load "ファイル名 "

タイトルの設定 set title "タイトル文字列 "

一変数関数のグラフ plot コマンドを使います。

1. plot x の式
2. plot コンマで区切った x の式のリスト

例えば plot x,x**2,x**3 とすると 3 つの関数 $x \mapsto x$, $x \mapsto x^2$, $x \mapsto x^3$ のグラフを描く。

グラフの表示範囲の指定 二つの方法があります。

1. plot を使う際に一々指定する。
   (a) “[左端の座標: 右端の座標]” としてグラフの x 軸の表示範囲が指定できます。
   (b) “[左端の座標: 右端の座標] [下端の座標: 上端の座標]” としてグラフの x 軸, y 軸の表示範囲が指定できます。
   例えば `plot [-10:20] [-10:200] x**2`
2. 前もって範囲の指定をしておく。
   `set xrange`[左端の座標: 右端の座標]
   `set yrange`[下端の座標: 上端の座標]

グラフの再描画 replot コマンドを使うとグラフの再描画ができます。

二変数関数のグラフ splot コマンドを使います。
`splot x と y の式`
例えば $f(x,y) = x + y$ のグラフを描くには `splot x+y`

パラメーター表示による曲線の表示 まず `set parametric` をしておいてから
x 座標を表す `plot t` の式,y 座標を表す `t` の式
もとに戻すには `set noparametric` とする。
例えば `set parametric ; plot cos(2*t),sin(5*t) ; set noparametric` とすると？

電卓としての利用 print の後に式を書きます。

print sin(2.0/3.0*pi)	$\sin(\frac{2\pi}{3})$
print 1,2**2	$(1+2i)^2$
print abs(2,4)	$\lvert(2+4i)\rvert$

ユーザー定義の関数 関数名 (引数)= 式
例えば `f(x,y)=x**2-y**2` とします。
（それから `splot [-1:1] [-1:1] f(x,y)` のような使い方ができます。）

データのプロット[3] `plot "ファイル名"`
`plot "ファイル名" with line`

印刷用のデータ作成 次のようにして描画先を画面ではなく、ファイルにします。

1. `set terminal postscript`
2. `set output ファイル名`

こうして作成したファイルは lpr でプリンターに送るだけで印刷できます。描画先を画面に戻すには `set terminal x11` として下さい。
次のようにすると “mygraph.ps” というファイルが出来ます。

```
splot [-1:1] [-1:1] x**2 + y**2
set terminal postscript
set output "mygraph.ps"
replot
set terminal x11
```

ktermで `lpr mygraph.ps` とすると関数 $(x, y) \mapsto x^2 + y^2$ のグラフが印刷できます。

問題 4-2: gnuplot を使って何か 2 変数関数のグラフを書いてみなさい。