夏休みの課題

山本昌志*

2006 年 7 月 18 日

概 要

C 言語を使った数値計算の学習を進めている．そこでは，テイラー展開と線形代数をつかうことになる．そのために，それらの復習を夏休みの課題として課す．さらに，ソーティングのプログラムの作成をとおして，C 言語のプログラムになれて欲しい．

最後の課題は，円周率の値を 10 万桁まで計算するプログラムである．ただし，これは少し難しいので，選択課題とするので，自信のある者のみチャレンジせよ．円周率の値を 10 万桁まで計算するプログラムができたならば，他の課題は免除する．

1 数値計算の基礎

[**問 1**] テイラー展開の公式を導き出せ．公式を書くだけではだめである．3〜4 年生の数学の教科書に書かれているはずである．

[**問 2**] 問 1 の結果を利用して，関数 $f(x)$ を $x = a$ の周りでテイラー展開する式を示せ．

[**問 3**] 問 1 の結果を利用して，関数 $f(x)$ を $x = 0$ の周りでテイラー展開する式を示せ．これをマクローリン展開と言う．

[**問 4**] 問 1 の結果を利用して，関数 $f(x + \Delta x)$ を x の周りでテイラー展開する式を示せ．

[**問 5**] 次の 3 つの関数を，$x = 0$ の周りでテイラー展開，即ちマクローリン展開せよ．以下の 3 つの関係は，どうなっているか考察せよ．

$$e^x =$$
$$\sin x =$$
$$\cos x =$$

[**問 6**] テイラー展開の結果を利用して，$\theta = 31$[度] の時の $\sin\theta$ と $\cos\theta$ の値を小数点以下 6 桁の精度で求めよ．電卓を使っても良いが，三角関数の機能を使ってはならない．

*独立行政法人　秋田工業高等専門学校　電気情報工学科

[**問 7**] 以下の連立方程式を行列とベクトルで表現せよ．

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1N}x_N &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2N}x_N &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3N}x_N &= b_3 \\
&\vdots \\
a_{M1}x_1 + a_{M2}x_2 + a_{M3}x_3 + \cdots + a_{MN}x_N &= b_M
\end{aligned}$$

[**問 8**] 以下の連立方程式を行列とベクトルで表現せよ．

$$\begin{aligned}
x_1 + 2x_2 - x_3 + 3x_4 &= 1 \\
-2x_1 - 3x_2 - 5x_4 &= 2 \\
2x_1 + 2x_2 + 3x_3 + 5x_4 &= 3 \\
-x_1 + 3x_2 - 12x_3 &= 4
\end{aligned}$$

[**問 9**] 前問の連立方程式の解と係数行列の逆行列を求めよ．解と逆行列は掃き出し法を使うこと．

2 C 言語の練習

ソーティングとは，整列あるいは並び替えのことである[1]．プログラミングでは，数値を大きい順，あるいは小さい順に並び替える技法のことを言う．数値の並び替えは非常に重要な技法で，実際のプログラムではいたるところで使われいる．ソーティングでもっと重要なことは，処理速度である．高速な処理を目指していろいろなアルゴリズムが考えられている．ここでは，ソーティングの簡単なアルゴリズムを示し，C 言語のプログラムの学習を進める．

2.1 単純挿入法

ソーティングの中でも，最も簡単な単純挿入法のプログラムを作成する．有名な C 言語の本「NUMERICAL RECIPES in C」によると，これは経験を積んだトランプ師が使う方法と同じであるということである．順序がバラバラのトランプを並び替える場合，

- まず，2 枚目のカードを拾い，1 枚目と順序関係が正しい位置に置く．
- 次に 3 枚目のカードを拾い，最初の 2 枚と順序関係の正しい位置にそれを挿入する．
- 同じことを繰り返す．即ち，i 枚目のカードを拾い，最初の $i-1$ 枚のカードの順序関係の正しい位置にそれを挿入する．

[1] ここでの説明は，NUMERICAL RECIPES in C を参考にしている．

- 最後のカードを正しい位置に挿入したら，並び替えは完了である．

この単純挿入法を用いて，リスト 1 で作成された整数の配列 a[0]〜a[1023] を小さい順に並び替えよ．このリストの 19 行目以降に単純挿入法のプログラムを書く．**ヒント** 図 1 に単純挿入法のフローチャートを示す．

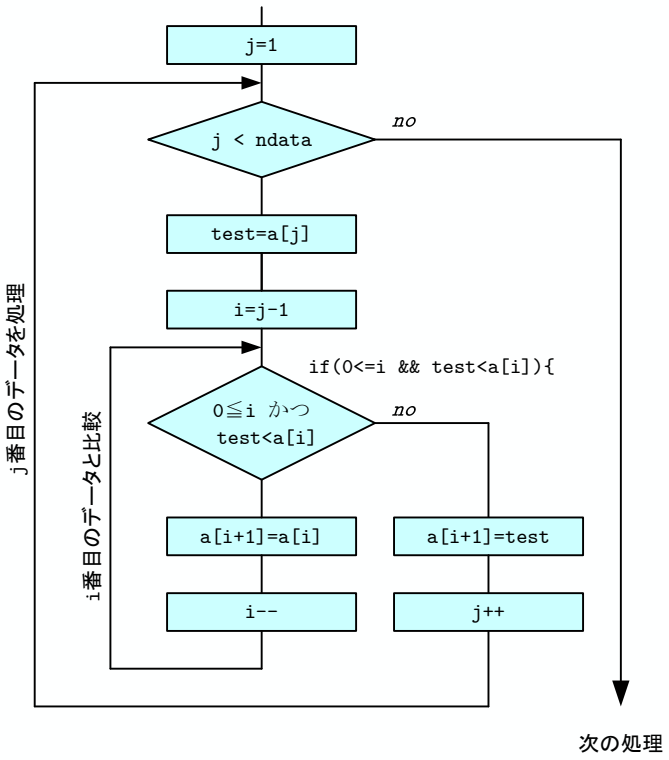

図 1: 単純挿入法のフローチャート．ndata はデータ数で，a[0]〜a[nadata-1] の配列に格納されている整数を小さい順 (昇順) に並べる．

リスト 1: 単純挿入法のプログラム

```
1  #include <stdio.h>
2  #include <stdlib.h>     /* 乱数発生のため       */
3  #include <time.h>       /* 時刻の関数を使うため */
4
5  int main(void){
6    int a[1024], i, j, ndata, test;
7
8    ndata=1024;
9
10   srand((unsigned int)time(NULL));     /* 起動毎に異なる乱数を発生させるため */
11
```

```
12    for(i=0; i<ndata; i++){
13      a[i]=rand();                          /* 配列a[i]に乱数の整数を設定 */
14    }
15
16
17
18    /*  これ以降に単純ソートと昇順に並んだ出力のプログラムを書く */
19
20
21
22
23
24
25
26
27    return 0;
28  }
```

2.2 shellソート

Shell ソート[2]は 1959 年に D.L.Shell が考案した方法で，単純挿入法を改良したものとなっている．バブルソート法は，隣同士を比較するが，Shell ソートでは，大きな h 飛ばしで比較する．ソートに時間のかかる大ききな数や小さな数は，一気に右や左に移動する．h 飛ばしで比較すると，

$$a[1] \leqq a[h+1] \leqq a[2*h+1] \leqq a[3*h+1] \leqq a[4*h+1] \leqq a[5*h+1] \cdots$$
$$a[2] \leqq a[h+2] \leqq a[2*h+2] \leqq a[3*h+2] \leqq a[4*h+2] \leqq a[5*h+2] \cdots$$
$$a[3] \leqq a[h+3] \leqq a[2*h+3] \leqq a[3*h+3] \leqq a[4*h+3] \leqq a[5*h+3] \cdots$$
$$\vdots$$
$$a[h] \leqq a[h+h] \leqq a[2*h+h] \leqq a[3*h+h] \leqq a[4*h+h] \leqq a[5*h+h] \cdots$$

と並び替える．この並び替えには単純挿入法をつかう．

そうして，とび幅 h をどんどん小さくし，最後は $h=1$ にすると並び替えが完了となる．この h の選び方にこつがあって，小さいほうから $1, 4, 13, 40, 121, \cdots$ と

$$h_{i+1} = 3h_i + 1 \qquad\qquad h_1 = 1$$

とするのが良いらしい．良いというのは早いと言うことである．最初に実行する一番大きな h は，データの個数の半分以下にする．

Shell ソートの手順は，次の通りである．

1. 最初の飛び幅 h を決める．データの個数の半分以下で最大の h_i を最初の飛び幅とする．
2. $i = 1, 2, 3, \cdots, h$ に対して，`a[i],a[h+i],a[2*h+i],a[3*h+i],`$\cdots$ を並び替える．
3. 次の `i++`して，並び替える．
4. 次の `h=(h-1)/3` にして，再度，並び替えを実行する．

リスト 1 の 19 行目以降に shell ソートのプログラムを書き，配列を小さい順 (昇順) に並び替えよ．

[2]この辺の説明は，www.rkmath.rikkyo.ac.jp/ kida/shellsort.htm を参考にしている．

3　数値計算の練習

3.1　オイラー法

オイラー法を使って，微分方程式を計算するプログラムを作成せよ．以下，詳細に説明しているので，これを良く読めばプログラムの作成ができるはずである．

3.1.1　計算方法

次の微分方程式

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \cos x \tag{1}$$

をオイラー法により計算する．ただし，初期条件は $x = 0$ の時 $y = 0$ とする．当然，これは数値計算するまでもなく，解は

$$y = \sin x \tag{2}$$

と分かっている．分かっているが，ここではコンピュータープログラムにより計算する．ここでの内容を良く理解すれば，通常解けない微分方程式でも，コンピューターにより計算できることが分かるだろう．

コンピューターで微分方程式を計算する方法を示す．微分方程式の解を $y = f(x)$ とする．すると，微分—正しくは導関数—は，

$$\frac{\mathrm{d}y}{\mathrm{d}x} \simeq \frac{\Delta y}{\Delta x} = \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{3}$$

と近似することができる．右辺の $\Delta x \to 0$ の極限が微分の値となる．したがって，元の微分方程式は

$$\frac{f(x + \Delta x) - f(x)}{\Delta x} \simeq \cos x \tag{4}$$

と書くことができる．これと，式 (1) から，

$$f(x + \Delta x) \simeq f(x) + \Delta x \cos x \tag{5}$$

である．これがオイラー法で微分方程式を解くときの基本の式である．Δx の値を小さくすると，この近似の精度が上がる．初期条件を上手に使うと，微分方程式の解の値が芋づる式にわかる．仮りに，$\Delta x = 0.001$ とすると，次のような手順で計算する．

(1)	$f(0.000) = 0$	これは初期条件である．
(2)	$f(0.001) = f(0.000) + 0.001 \times \cos(0.000)$	右辺は (1) の結果を利用する
(3)	$f(0.002) = f(0.001) + 0.001 \times \cos(0.001)$	右辺は (2) の結果を利用する
(4)	$f(0.003) = f(0.002) + 0.001 \times \cos(0.002)$	右辺は (3) の結果を利用する
(5)	$f(0.004) = f(0.003) + 0.001 \times \cos(0.003)$	右辺は (4) の結果を利用する
	$\vdots$	これを繰り返す

(1) は初期条件なので計算するまでもない．そして，(1) の結果を利用すると，(2) の右辺が計算でき，その左辺の値を決めることができる．同様に (2) の結果を利用すると，(3) の右辺が計算でき，その左辺の値が決められる．これを繰り返すのである．すると，$x = 0$ の初期条件からはじめて，任意の x まで $f(x)$ の値を求めることができる．元の微分方程式 (1) の近似解が求まったことになる．$\Delta x = 0.001$ は計算のステップ幅と呼ばれ，これを小さくするとさらに計算時間は必要になるが，計算精度が上がる．

例えば，これを 4000 回この計算を繰り返すと，微分方程式 (1) の解が $y = f(0.000)$ から $f(4.000)$ まで計算できる．すなわち，0〜4[rad] (229.1[度]) までの値が 0.001rad(0.057[度]) きざみで分かるのである．4000 回の計算なんか，コンピューターに取っ手は大したことはない．私の PC では，計算と表示に用した時間は，0.15 秒である．コンピューターの計算速度には，本当に驚かされる．

3.1.2 計算アルゴリズム

計算の原理が分かったので，プログラミング方法を示す．計算のフローチャートは図 2 のようになる．これにしたがって，プログラムを書けば良い．難しいことは何もない．

まずは，計算のステップ幅 Δx を決めなくてはならない．C 言語では

```
dx=4.0/4000;
```

と書く．プログラムではギリシャ文字は使えないので，ステップ幅を dx としている．解となる計算結果は後で利用することを考えて，配列に格納する方が良い．配列の先頭には，初期条件を格納する．すなわち，

```
x[0]=0.0;
y[0]=0.0;;
```

とするのである．次に i の値を 1〜4000 まで変えて，

```
x[i]=i*dx;
y[i]=y[i-1]+dx*cos(x[i-1));
```

を繰り返し計算する．このように計算回数が決まっている繰り返しには，for 文を使うのが一般的である．

```
for(i=1;i<=4000;i++){

   ここに繰り返したい内容を書く．

}
```

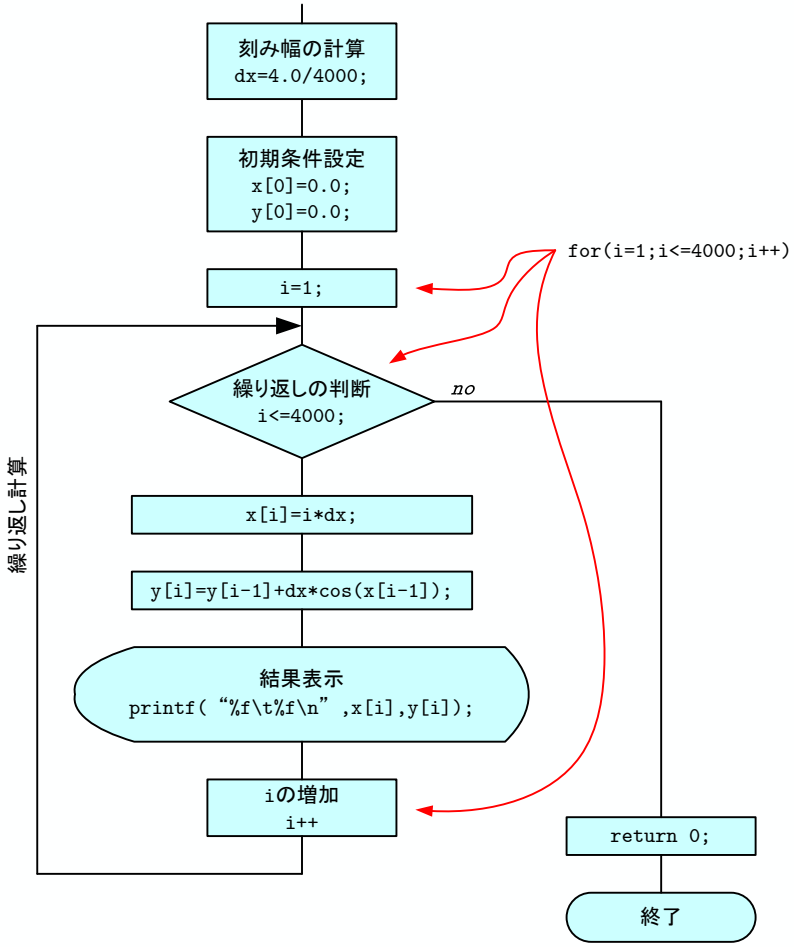

図 2: オイラー法のフローチャート.

4　円周率の計算 (上級者向選択課題)

これは上級者向の選択課題である．もし，このプログラムができたならば，他の課題は実施しなくても良い．これだけできれば十分である．

円周率 10 万桁まで精度良く計算するプログラムを作成する．これは，必須ではないが，興味のある者はプログラムを作成せよ．この部分は，山形大学の新関久一さんの「プログラミング演習 III」を参考にさせてもらっている．最初に，ヒントとして，長桁計算のプログラムを示す．

4.1 長桁計算

円周率を 10 万桁まで求めようとすると，長桁の計算が必要になる．しかし，C 言語の倍精度実数は 20 桁程度しか有効数字がない．そこで，長桁計算を考えることにする．

人間は時間と紙さえあれば，いくらでも長い桁の計算ができる．ただの計算なので，人間ができてコンピューターができないわけがない．人間と同じことをコンピューターにやらせれば良いのである．人間と同じように，コンピューターに長桁の筆算の計算をさせる．紙の代わりにデータは配列に記憶させるだけである．

これを最初から考えるのは，初心者には少し難しいので，リスト 2 にプログラムを載せておく．このプログラムでは，非常に大きな桁数の 2 つの整数を入力して，その和と差を計算することができる．

このプログラムでは，負の値は 10 の補数を用いている．もしある数が負の整数であれば，その絶対値の各桁を 9 から差し引いて，最後に 1 を加えている．この辺のところは，3 年生の電子計算機の授業で教えたはずである[3]．ただ，そのときは 2 進法を使っていたので，2 の補数だったが，考え方は同じである．

リスト 2: 長桁の整数の和と差

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #define N 32768
5  #define RADIX 10
6
7  void lf_scan(int n[]);
8  void lf_plus();
9  void lf_minus();
10 void lf_print(int n[]);
11 void lf_complement(int n[]);
12 void prt_bit(int n[]);
13 int a[N], b[N], Acc[N];
14
15 /*=============================================================*/
16 /*    main                                                     */
17 /*=============================================================*/
18 int main(void){
19
20   lf_scan(a);
21   lf_scan(b);
22   lf_plus();
23   lf_print(Acc);
24   lf_minus();
25   lf_print(Acc);
26
27   return 0;
28 }
29
30
31 /*=============================================================*/
32 /*    lf_scan                                                  */
33 /*=============================================================*/
34 void lf_scan(int n[]){
35   unsigned char key_in[N];
36   int i,l,flag=0;
37
38   scanf("%s",key_in);
```

[3] 2 進数では，(1) 各桁のビット反転 (2)+1 加算と教えた．これは，書く桁を 1 から差し引いて，1 を加える演算と同じである．

```
39
40    l=strlen(key_in);
41
42    if(key_in[0] == '-'){
43      flag=1;
44      for(i=1; i<l; i++){
45        key_in[i-1]=key_in[i];
46      }
47      l--;
48    }
49
50    for(i=0;i<l;i++){
51      n[i]=(unsigned int)key_in[l-1-i]-48;
52    }
53
54    if(flag==1)lf_complement(n);
55  }
56
57  /*================================================================*/
58  /*    lf_plus                                                     */
59  /*================================================================*/
60  void lf_plus(){
61    int i;
62
63    for(i=0;i<N;i++) Acc[i] = a[i]+b[i];
64
65    for(i=0; i<N-1; i++){
66      if(Acc[i] > 9)Acc[i+1]++;
67      Acc[i]%=RADIX;
68    }
69
70    Acc[N-1]%=RADIX;
71
72  }
73
74  /*================================================================*/
75  /*    lf_minus                                                    */
76  /*================================================================*/
77  void lf_minus(){
78
79    lf_complement(b);
80    lf_plus();
81
82  }
83
84  /*================================================================*/
85  /*    lf_print                                                    */
86  /*================================================================*/
87  void lf_print(int n[]){
88    int i,j,flag=0;
89
90    i=N-1;
91
92    if(n[i]>4){
93      flag=1;
94      lf_complement(n);
95      printf("-");
96    }
97
98    while(n[i]==0 && i>0) i--;
99    for(j=i;j>=0;j--)printf("%d",n[j]);
100
```

```
101     if(flag==1)lf_complement(n);
102
103     printf("\n");
104
105
106   }
107
108   /*================================================================*/
109   /*     complement                                                 */
110   /*================================================================*/
111   void lf_complement(int n[]){
112     int i;
113
114     for(i=0; i<N; i++) n[i]=9-n[i];
115
116     n[0]++;
117
118     i=0;
119     while(n[i]==10 && i < N){
120       n[i]=0;
121       n[i+1]++;
122       i++;
123     }
124   }
```

4.2 円周率の計算

4.2.1 円周率の計算方法

足し算と引き算の長桁の計算方法は分かった．わり算も同じである．後は自分で考えて，マチンの公式

$$\pi = 16\arctan\left(\frac{1}{5}\right) - 4\arctan\left(\frac{1}{239}\right)$$

とテイラー展開

$$\begin{aligned}\arctan(x) &= x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \frac{x^{11}}{11} + \frac{x^{13}}{13} - \frac{x^{15}}{15} + \cdots \\ &= \sum_{n=1}^{\infty}(-1)^{n-1}\frac{x^{2n-1}}{2n-1}\end{aligned}$$

を用いて，10 万桁を計算するプログラムを作成せよ．

4.2.2 10 万桁の値

計算結果はわかりやすいように，以下のように 10 万桁の値を表示させよ．これから分かるように，10 万桁の円周率は，$\pi = 3.1415926535897932\cdots0805655493624646$ となる．この表最後の値の 6 が 10 万桁目である．自分のプログラムの結果が正しいか否かの判定にこの表を使うと良い．

```
3.1415 9265 3589 7932 3846 2643 3832 7950 2884 1971 6939 9375
  1058 2097 4944 5923 0781 6406 2862 0899 8628 0348 2534 2117
  0679 8214 8086 5132 8230 6647 0938 4460 9550 5822 3172 5359
```

```
4081 2848 1117 4502 8410 2701 9385 2110 5559 6446 2294 8954
9303 8196 4428 8109 7566 5933 4461 2847 5648 2337 8678 3165
2712 0190 9145 6485 6692 3460 3486 1045 4326 6482 1339 3607
2602 4914 1273 7245 8700 6606 3155 8817 4881 5209 2096 2829
2540 9171 5364 3678 9259 0360 0113 3053 0548 8204 6652 1384
1469 5194 1511 6094 3305 7270 3657 5959 1953 0921 8611 7381
9326 1179 3105 1185 4807 4462 3799 6274 9567 3518 8575 2724
8912 2793 8183 0119 4912 9833 6733 6244 0656 6430 8602 1394
9463 9522 4737 1907 0217 9860 9437 0277 0539 2171 7629 3176
7523 8467 4818 4676 6940 5132 0005 6812 7145 2635 6082 7785
```

このあたりは長いので省略

```
0491 5378 8541 3909 4245 3169 1719 9876 2894 1277 2211 2946
4568 2948 6028 1493 1815 6024 9677 8879 4981 3777 2162 2935
9437 8110 0444 8060 7976 7242 9276 2495 1078 4153 4464 2915
0842 7645 2000 2042 7694 7069 8041 7758 3220 9097 0202 9165
7347 2515 8290 4630 9103 5903 7842 9775 7265 1720 8772 4474
0952 2671 6630 6005 4697 1638 7943 1711 9687 3484 6887 3818
6656 7512 7929 8575 0163 6341 1314 6275 3049 9019 1356 4682
3804 3299 7069 5770 1507 8933 7728 6580 3571 2790 9137 6742
0805 6554 9362 4646
```

4.3 課題提出要領

提出方法は，次の通りとする．

期限	9 月 4 日 (月) 20:00
用紙	A4
提出場所	山本研究室の入口のポスト
表紙	表紙を 1 枚つけて，以下の項目を分かりやすく記述すること． 授業科目名「計算機応用」 課題名「課題　夏休みの宿題」 5E　学籍番号　氏名 提出日
内容	ソースプログラムは，プリントアウト，手書き，いずれも OK とする