DO 文と 1 次元配列の練習問題

山本昌志*

2003 年 10 月 28 日

1　練習問題

以下、繰り返しの DO 文と 1 次元配列の練習問題を示します。プログラムを作成して、実行結果を確認すること。ただし、いろいろなレベルの問題が用意されています。各人のレベルに合わせて、問題のプログラムを作成すること。

1.1　教科書の問題

教科書の例題をプログラムして、内容を理解すること。ただし、教科書の解答のプログラムは、結果をラインプリンターに出力する場合です。皆さんは、ディスプレイに結果を出力するため、FORMAT 文を書き換える必要があります。それは、FORMAT 文の括弧の最初の部分でこれはラインプリンターの制御を表します。[1]。具体的には以下の部分を書き換えます。

- 教科書の P.52 の `FORMAT(1H1,'NINZU=',I5/` ⇒ `FORMAT('NINZU=',I5/`
- 教科書の P.52 の `*  1H,'HEIKIN NENREI=',F5.1)` ⇒ `*  'HEIKIN NENREI=',F5.1)`
- 教科書の P.61 の `FORMAT(1H1,'HEI',···//)` ⇒ `FORMAT('HEI',···//)`
- 教科書の P.61 の `FORMAT(1H ,I5)` ⇒ `FORMAT(I5)`

このラインプリンター制御文字に注意しながら、以下の問題を解きなさい。

1. 教科書の例題 3・1 のプログラムを作成しなさい。
2. 教科書の例題 3・2 のプログラムを作成しなさい。

1.2　ちょっとだけ応用

1. DO 文を利用して、1〜N まで足し合わせるプログラムを作成しなさい。
2. 一次元配列に、$1, 3, 5, 7, \cdot, N$ を格納し、それを足し合わせるプログラムを作成しなさい。

*国立秋田工業高等専門学校　電気工学科

[1]教科書の P.70〜71 に書いてある

1.3 ソーティング

ソーティングとは、整列あるいは並び替えのことである[2]。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います。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ます。ソーティングでもっと重要なことは、処理速度です。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ます。簡単なアルゴリズムを示しますので、数値の入ったデータを小さい順 (昇順) に並び替えなさい。

なお、その前に並び替えるデータの作成方法を示しておきます。以下のプログラムにより、0〜1 の乱数[3]が、配列名 DATA に 1024 個入れられる。RAND() が 0〜1 の実数を発生させる。SRAND により、初期値が決まる。この値を変えると発生される乱数が異なる。どんな値を入れても良い。

```
      PROGRAM MAKE RANDOM NUMBERS AND SORT

      INTEGER NDATA
      REAL DATA(1:1024)

      NDATA=1024

      CALL SRAND(20031028)

      DO 10 I=1,NDATA,1                                          10
         DATA(I)=RAND()
10    CONTINUE

      STOP

      END
```

1.3.1 単純挿入法

有名な教科書「NUMERICAL RECIPES in C」によると、これは経験をつんだトランプ師が使う方法と同じということです。順序がばらばらのトランプを並び替える場合、

1. 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おく。
2. 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。
3. 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。
4. 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。

という操作を行います。

これと同じことを先に示した 1024 個のデータについて行い、小さい順に並び替えなさい。ただし、ヒントとしてフローチャートを図**??**に示します。

[2] ここでの説明は、NUMERICAL RECIPES in C を参考にしている。
[3] 0 以上、1 以下の実数。ただし、数値及びその並びの順序はでたらめである。

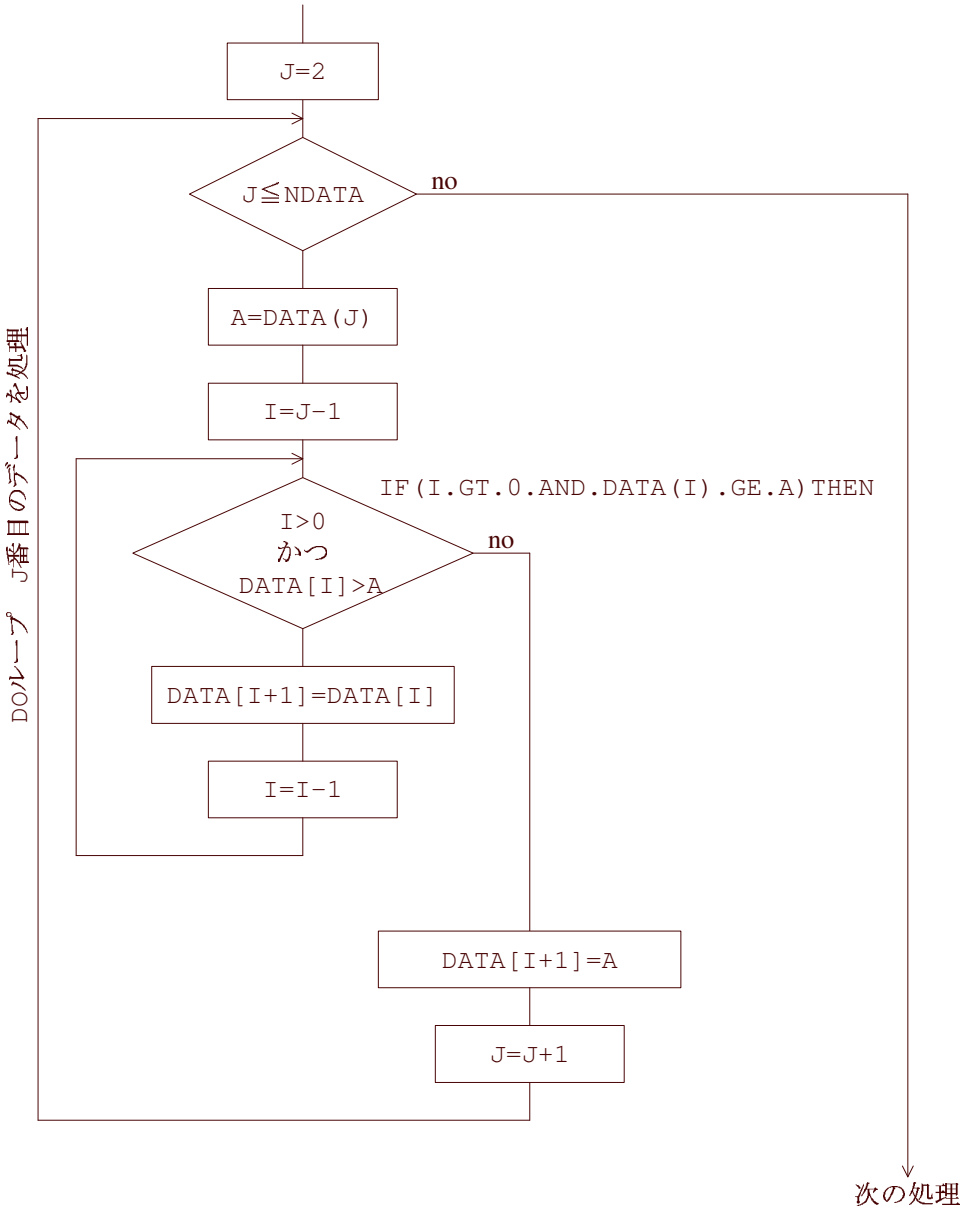

図 1: 単純挿入法のフローチャート

1.3.2 shell ソート

Shell ソート[4]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ます。単純挿入法は、隣同士を比較しましたが、Shell ソートでは、大きな H 飛ばしで比較します。ソートに時間のかかる大ききな数や小さな数は、一気に右や左に行きます。H と飛ばしで比較すると、

$$
\begin{aligned}
&A(1) \leqq A(H+1) \leqq A(2*H+1) \leqq A(3*H+1) \leqq A(4*H+1) \leqq A(5*H+1)\cdots \\
&A(2) \leqq A(H+2) \leqq A(2*H+2) \leqq A(3*H+2) \leqq A(4*H+2) \leqq A(5*H+2)\cdots \\
&A(3) \leqq A(H+3) \leqq A(2*H+3) \leqq A(3*H+3) \leqq A(4*H+3) \leqq A(5*H+3)\cdots \\
&\qquad\qquad \vdots \\
&A(H) \leqq A(H+H) \leqq A(2*H+H) \leqq A(3*H+H) \leqq A(4*H+H) \leqq A(5*H+H)\cdots
\end{aligned}
$$

と並び替えます。この並び替えには単純挿入法を使います。

そうして、とび幅 H をどんどん小さく、最後は $H=1$ にすると並び替えは完了です。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 3 の倍数+1 とするのが良いそうです。良いというのは早いと言うことです。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します。

Shell ソートの手順は、次の通りです。

1. 最初の飛び幅 H を決める。H である 3 倍して 1 を加えた値が、データ数以下になるものを探す。データこ個数の半分以下で最大の H 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=I+1 にして、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。

単純挿入法のプログラムが出来た人は、このプログラムを書いて、実行させなさい。

1.3.3 その他のソーティング

実際のソーティングでは、ヒープソートあるいはクイックソートが使われます。これらの説明は、この講義のレベルを超えますので、興味のある人は自分で調べてください。

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