13　あみだくじ 3

前回，挿入整列法と泡整列法に従ったあみだくじ完成法で，比較回数（橋桁の数）を減らして効率化を図りました。交換回数（橋の数）は，上段のデータで順序が逆転しているペアの数なので，減らすことはできません[1]

あみだくじの橋は，両端のデータを交換するためのものです。橋は 1 本ですが，データの交換には 3 ステップかかります（データの移動を 3 回します）。

今回は，移動回数を減らす工夫をします。

13.1　移動状況表示

橋を架けるとき，データを交換するために一方のデータを一時的に退避する場所がありますが，これを 0 本目の道ということにして，3 回の移動を**一方通行の橋**で表すことにします。

13.1.1　データ構造変更

各番号が 1 から始まっていたのを 0 からにします。

```
type
  TMitiNo   = 0..TateSuu;                          // 道（縦線）の番号
  TKawaNo   = 0..TateSuu-1;                        // 川（縦線の間）の番号
  TYokoNo   = 0..YokoSuu;                          // 橋桁（横線）の番号
```

13.1.2　移動メソッド

```
procedure TAmida.IdouSuru(M1,M2 : TMitiNo; Y : TYokoNo; PM : Integer);
          (* M1 から M2 に一方通行の橋を架ける       *)
          (* 符号 PM によって上下に少しずらして描く *)
begin
  with Canvas do
    begin
      Pen.Width := 1;
      if PM = 0
        then Pen.Color := clred                    // 本来の橋
        else Pen.Color := clYellow;                // 退避，復元用の橋
      MoveTo(M1*YokoHaba,(Y+5)*TateHaba+PM*3);
      LineTo(M2*YokoHaba,(Y+5)*TateHaba+PM*3);
      Genzai[M2] := Genzai[M1];                    // データを移動
    end;
end; (* IdouSuru *)
```

[1] 立体交差する橋（たとえば，2,3,4 番の道を跨いで 1 番と 5 番の道を直接つなぐ橋）を許せば減らすことができます。そのような改良も，後で紹介します。

13.1.3　改良挿入法の一方通行版

メソッド InsetSort2 をコピーして変更します。

InsertSort2 は比較する時に橋桁をかくため，繰り返し（repeat 文）の中で比較（if 文）をしていましたが，こんどは橋桁をかかなくてよいので，比較を繰り返し条件に含めることができます。これが本来の姿ですっきりしたプログラムになります。

```
procedure TAmida.InsertSort3;
          (* 挿入法で整列する 改良版 *)
var
  Kawa : TKawaNo;
  Yoko : 1..YokoSuu+1;
  KawaStart : TKawaNo;
begin
  Yoko := 1;
  for KawaStart := 1 to TateSuu-1 do
    begin
      Kawa := KawaStart;
      while (Kawa > 0) and (Genzai[Kawa] < Genzai[Kawa+1]) do // 左が小さい間
        begin
          IdouSuru(Kawa+1,0,Yoko,-1);                        // 交換する
          IdouSuru(Kawa,Kawa+1,Yoko,0);
          IdouSuru(0,Kawa,Yoko,+1);
          Inc(Yoko);
          Dec(Kawa);                                         // 左の川へ
        end;
      JunretuWoKaku(Genzai,(YokoSuu+7)*TateHaba);
    end;
end; (* InsertSort3 *)
```

13.1.4　実行のために

ButtonInsert2 のコピーを作つて変更します。

Name	ButtonInsert3
Tag	5

その他，必要なことを追加してください。

13.1.5　実行

見にくいかもしれませんが，よく見ると無駄な橋（0 番から来てまた 0 番に行く 2 本の橋の組み合わせ）がいっぱいあります。この無駄をなくしましょう。

13.1.6　改良

メソッド InsetSort3 をコピーして変更します。

```
procedure TAmida.InsertSort4;
          (* 挿入法で整列する 再改良版 *)
var
  Kawa : TKawaNo;
  Yoko : 1..YokoSuu+1;
  KawaStart : TKawaNo;
begin
  Yoko := 1;
  for KawaStart := 1 to TateSuu-1 do
    begin
      Kawa := KawaStart;
      IdouSuru(Kawa+1,0,Yoko,-1);                          // 0 番に退避
      while (Kawa > 0) and (Genzai[Kawa] < Genzai[0]) do   // 小さい間
        begin
          IdouSuru(Kawa,Kawa+1,Yoko,0);                    // 右に移動
          Inc(Yoko);
          Dec(Kawa);                                       // 左の川へ
        end;
      IdouSuru(0,Kawa+1,Yoko-1,+1);                        // 0 番を挿入
      JunretuWoKaku(Genzai,(YokoSuu+7)*TateHaba);
    end;
end; (* InsertSort4 *)
```

13.1.7　実行のために

ボタンをコピーするなど必要なことを追加してください。

13.1.8　実行

無駄な橋がなくなってすっきりしました。

13.2　問題

泡整列法について同様のことをしなさい。

13.3　整列法

大量のデータ（たとえば学生の情報）から目的とするデータを素早く取り出すためには，データをある項目（たとえば氏名）の順序（たとえば五十音順）について整列しておく必要があります。

代表的な整列法に次のものがあります。

単純整列法	選択整列法	Select Sort			
	挿入整列法	Insert Sort	⇒	シェル整列法	Shell Sort
	泡整列法	Bubble Sort	⇒	櫛整列法	Comb Sort
高速整列法	併合整列法	Merge Sort			
	速整列法	Quick Sort			
	堆積整列法	Heap Sort			

13.4　シェル法

データが大量にあるとき，最初は粗く，だんだん細かく整列していくと早く整列できます。

あみだくじではデータが少ないので，ありがたみがよくわからないかもしれませんが，原理を理解するために紹介します。これは，立体交差橋を使います。

```
procedure TAmida.ShellSort;
          (* シェルソート，挿入整列法の高速版           *)
          (* InsertSort2 を初めは粗く、段々細かく行なう *)
var
  KawaStart  : TKawaNo;
  Kawa,Arasa : Integer;
  Yoko       : TYokoNo;
  Bairitu    : Real;
begin
  Bairitu := 0.7;
  Yoko := 1;
  Arasa := TateSuu;
  repeat
    Arasa := Trunc(Arasa*Bairitu);
    if Arasa < 2
      then Arasa := 1;                          // 最後は InsertSort2 になる
    for KawaStart := 1 to TateSuu-Arasa do
      begin
        Kawa := KawaStart;
        while (Kawa > 0) and (Genzai[Kawa] < Genzai[Kawa+Arasa]) do
          begin
            Genzai[0] := Genzai[Kawa+Arasa];   //IdouSuru(Kawa+Arasa,0,Yoko,-1);
            IdouSuru(Kawa,Kawa+Arasa,Yoko,0);
            Genzai[Kawa] := Genzai[0];         //IdouSuru(0,Kawa,Yoko,+1);
            Inc(Yoko);
            Dec(Kawa,Arasa);
          end;
      end;
    JunretuWoKaku(Genzai,(YokoSuu+7)*TateHaba);
  until Arasa = 1;
end; (* ShellSort *)
```