第11回　A科　コンピュータ基礎および演習Ⅰ

平成11年6月28日
3時限～4時限

1．前回アンケート調査結果について

2．配列（1次元）

(1) 配列の概念
(2) 配列の宣言
(3) リストボックスの利用

小休止

(4) 繰り返し（for文）
(5) 繰り返しの効果

小休止

3．配列を使ったデータ処理

(1) 合計の計算
(2) 合計を求める仕組み
(3) 平均の計算

4．文の書き換え

(1) while文を使った書き換え
(2) repeat文を使った書き換え

5．宿題

6．アンケート調査

1．　配列（1次元）

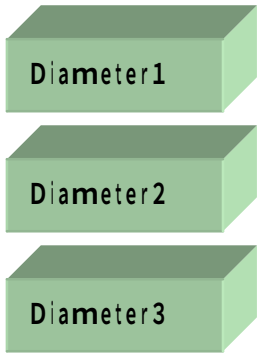
(1) 配列の概念

前回の授業では、ボタンをクリックすると、りんごの直径を1つ生成するプログラム（apple3）を作成しました。それでは、ボタンをクリックすると3個の直径を生成するにはどうすればよいでしょうか？

一つの方法として、整数型の変数を3個宣言します。

```
var
  Diameter1 : integer;
  Diameter2 : integer;
  Diameter3 : integer;
```

このように変数（箱）を3つ用意する



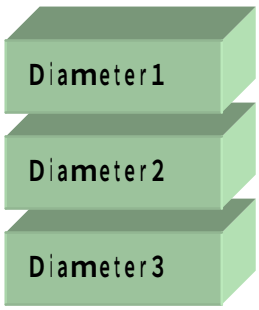
個々の変数に乱数を代入するイベントハンドラを記述します。

```
Diameter1 := Random(10);
Diameter2 := Random(10);
Diameter3 := Random(10);
```

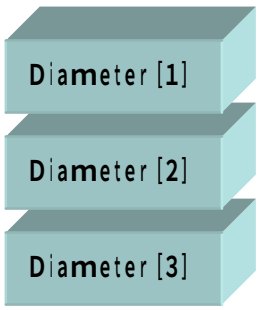
データの数が3個程度であれば、このような方法も可能ですが、１００個や１０００個になると、変数の宣言だけでプログラムがあふれてしまします。このように、**同じ型の変数をたくさん使う必要があるときには、配列を使います。**

配列は、以下のような構造をしています。**配列の要素**は、先頭から順に何番目であるかを指定します。この数字のことを、**添字**と言います。

変数



配列



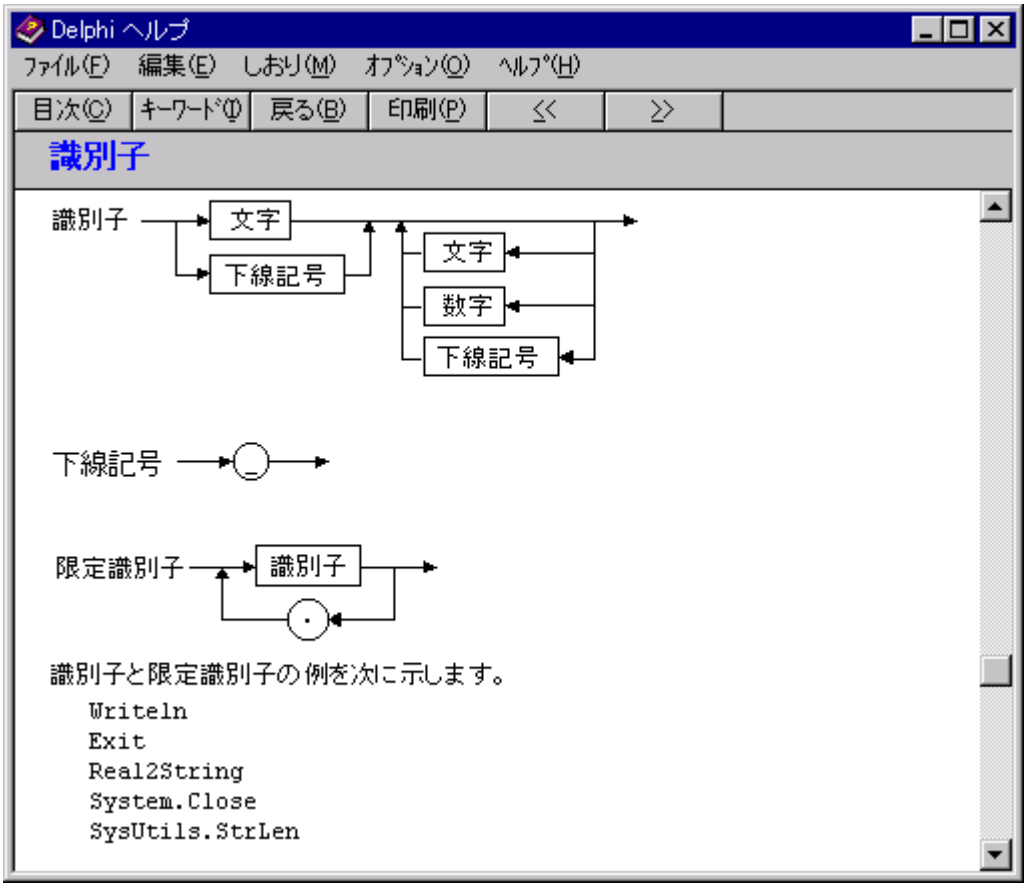
配列の要素は、先頭から何番目かを、かぎ括弧の数字で指定する（添字）。

(2) 配列の宣言

先の図で見る限り、変数を3つ宣言する場合と、大差は無いように見えます。しかし、3個の要素をもつ配列の宣言は、次のように簡潔に記述できます。要素の数が100や1000に増えても、かぎ括弧の中の数字を変更するだけで済みます。

```
var
  Diameter : array [1..3] of integer;
```

名前のつけ方は、以下のヘルプ画面で確認できます。

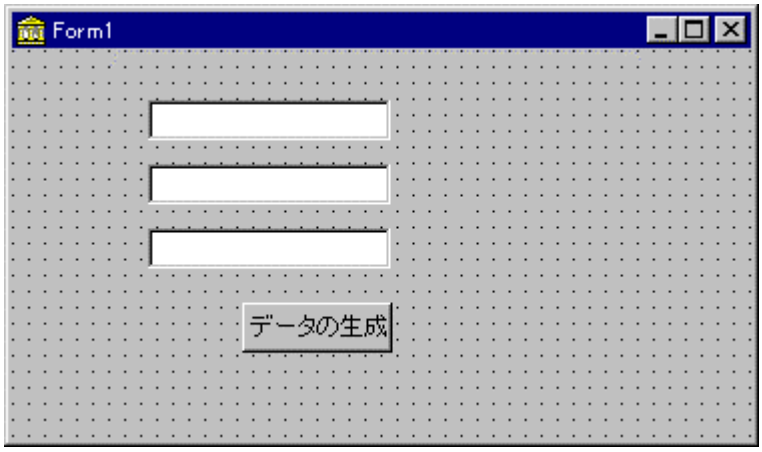


配列を使うと、イベントハンドラは以下のように書けます。かぎ括弧の中に、添字を指定します。

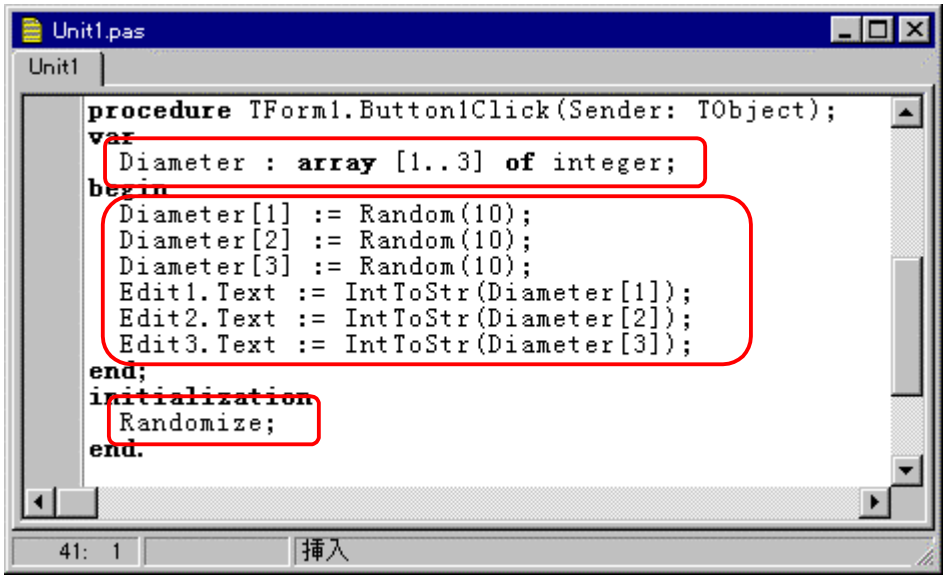
```
Diameter[1] := Random(10);
Diameter[2] := Random(10);
Diameter[3] := Random(10);
```

それでは、フォームにエディットを3つ、ボタンを1つ配置し、プログラムを作成してみ

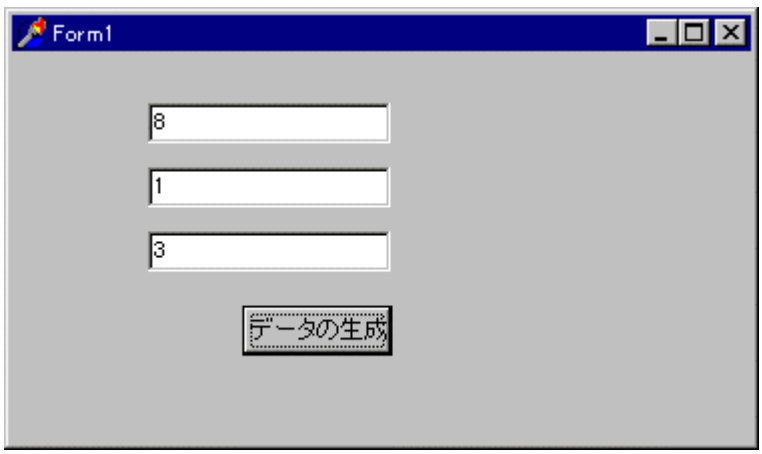
ましょう。ボタンのキャプションは、データの生成にしておきます。



ボタンをダブルクリックして、イベントハンドラを記述します。



Dドライブにフォルダarray1を作成し、ここに保存しておきます。プログラムを実行し、データの生成ボタンをクリックする毎に、乱数が3つ生成されることを確認しましょう。

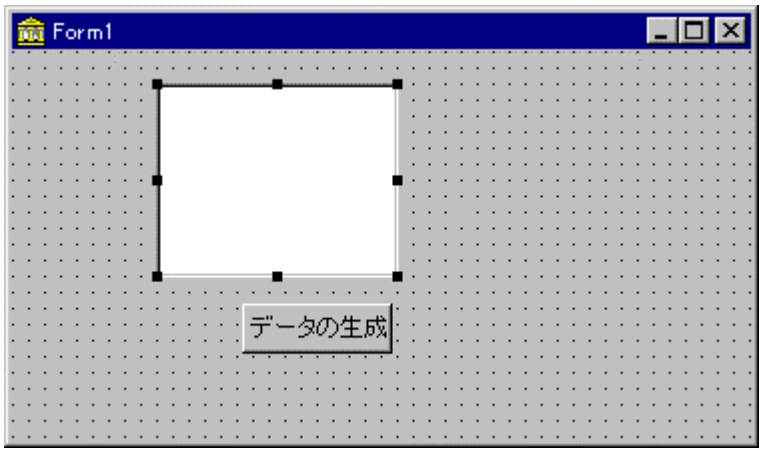


(3) リストボックスの利用

配列の要素の数だけフォームにエディットを配置し、生成したデータを表示する方法には、限界があります。3つ程度のデータであれば、フォームの上にエディットを配置できますが、１００個や１０００個などのデータになると、フォームにエディットを配置することは困難です。

ひとつの解決策として、リストボックスを利用する方法があります。先のプログラムで3つのエディットを削除し、代わりにリストボックスを1つ配置します。

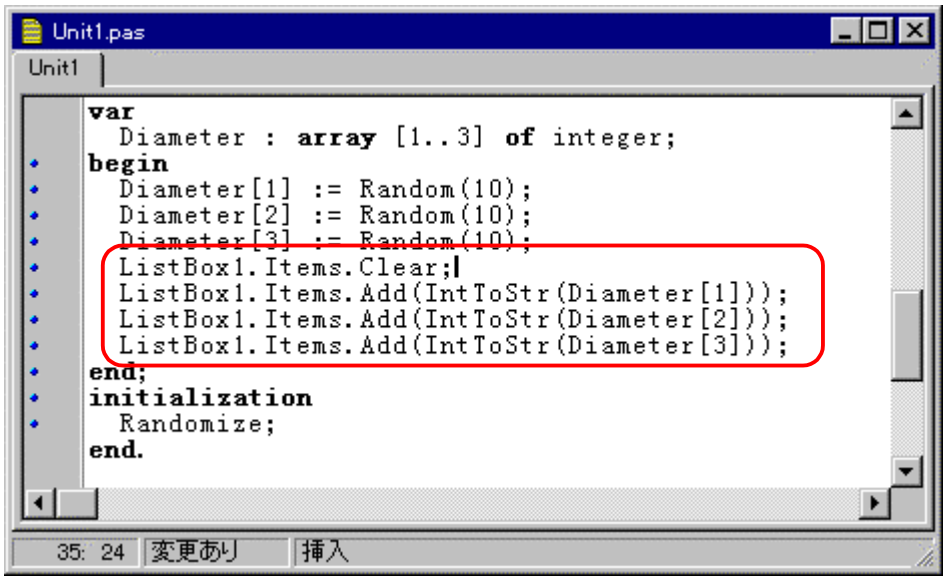
リストボックスは、配列を表示する目的のためだけに使うものではありませんが、この例ではうまく活用できます。



エディットへ文字列を表示するときは、代入によって行いましたが、リストボックスへ文字列を表示するときは、次のような指定を行います。

ListBox1.Items.Add(ここに文字列を入れる)

それでは、ボタンをダブルクリックし、イベントハンドラを以下のように書き換えてみましょう。



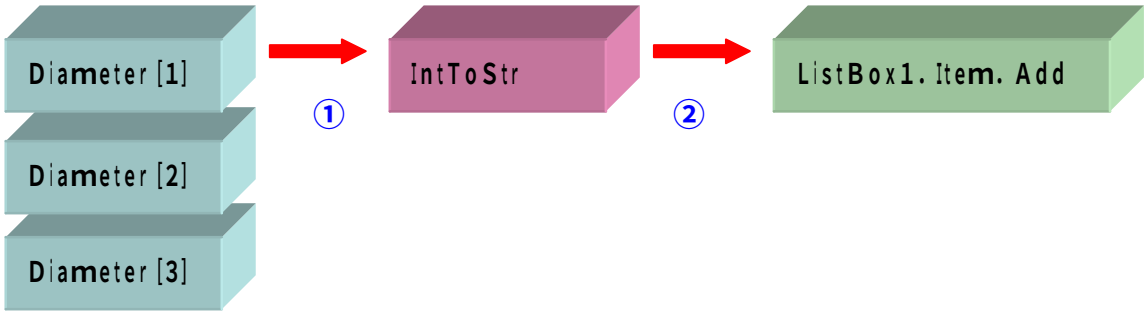
ListBox1.Items.Clearは、リストボックスの内容を消します。これを入れておかないと、プログラム実行時にボタンをクリックする毎に、前の結果に追加してしまいます。

```
ListBox1.Items.Clear;
```

リストボックスへの表示は、複雑に見えますが、以下のように分割して考えると分かりやすくなります。

① 配列Diameterの一番目の要素に入っている数字を文字列に変換します。
② その結果をListBox1.Items.Addに渡すことによって、表示を行います。

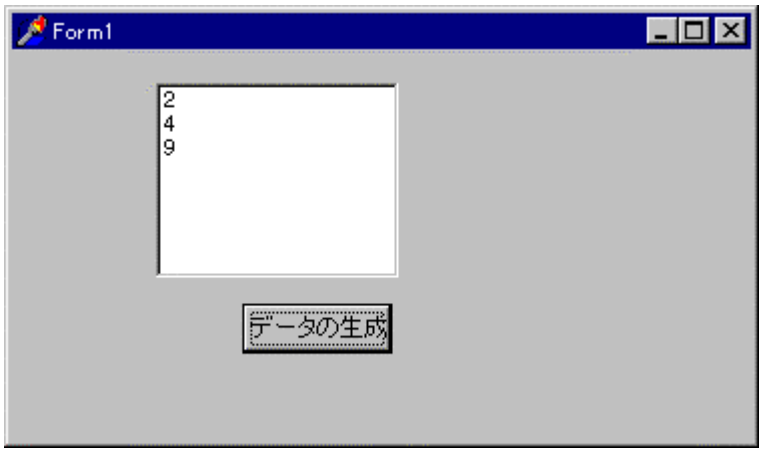
```
ListBox1.Items.Add(  IntToStr(  Diameter[1]  )  );
```



以下同様に、Diameterの残りの要素についても表示を行います。

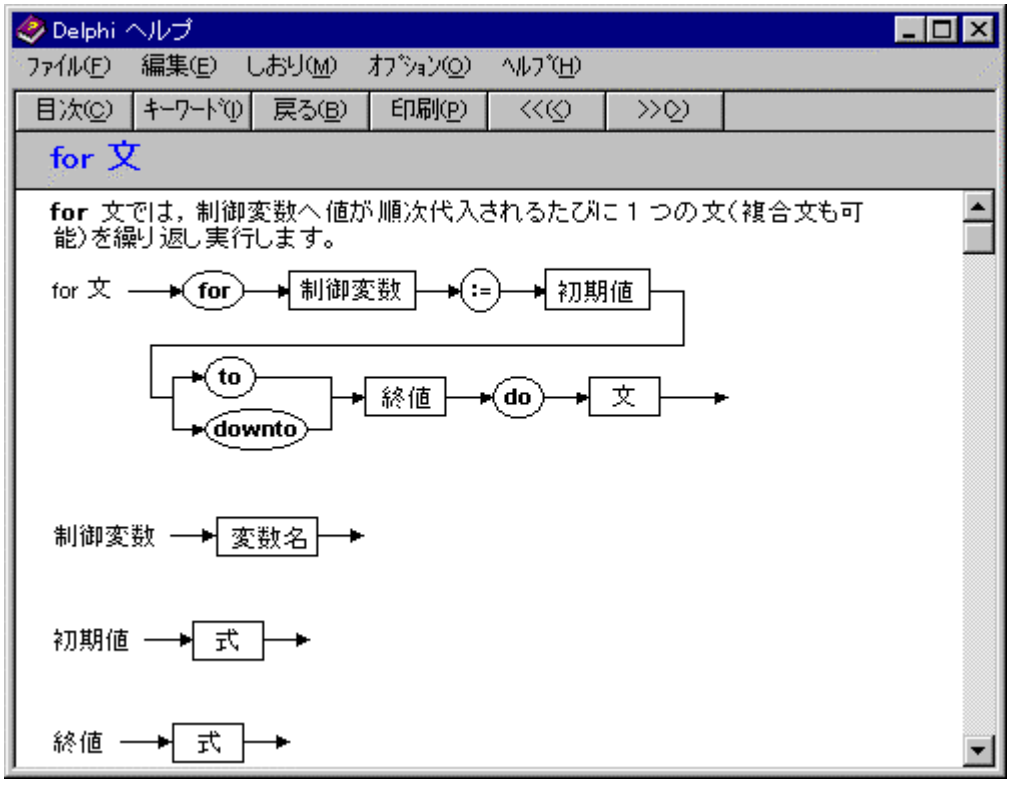
```
ListBox1.Items.Add(  IntToStr(  Diameter[2]  )  );
ListBox1.Items.Add(  IntToStr(  Diameter[3]  )  );
```

先に作ったプログラムを流用したので、保存するときは[ファイル][プロジェクトに名前を付けて保存]を行い，フォルダarray2に保存します。さらに，[ファイル][名前を付けて保存]を行い，フォルダarray2に保存します。
プログラムを実行します。ボタンを押す毎に、乱数が3つ表示されます。



(4) 繰り返し (for文)

このプログラムでは、やはり１００個や１０００個のデータを扱うと、イベントハンドラの記述でプログラムがあふれてしまいます。同じようなことを何度も行うときは、繰り返しを使います。繰り返す回数があらかじめ分かっている場合は、for文を使います。



for文を使うと、乱数を生成する部分は、以下のようになります。

```
Diameter[1] := Random(10);
Diameter[2] := Random(10);
Diameter[3] := Random(10);
```

⇩

```
for I :=1 to 3 do
  Diameter[I] := Random(10);     (繰り返しの対象となる文)
```

for文は、doの次の文を繰り返します。繰り返しの回数は、制御変数 I が数えています。

これ以外に繰り返しを行う文には、while文とrepeat文があります。これらの文は、不定回の繰り返し (繰り返す回数が分からない) を実現できます。後述するように、for文はwhile文やrepeat文で書き換えることができます。

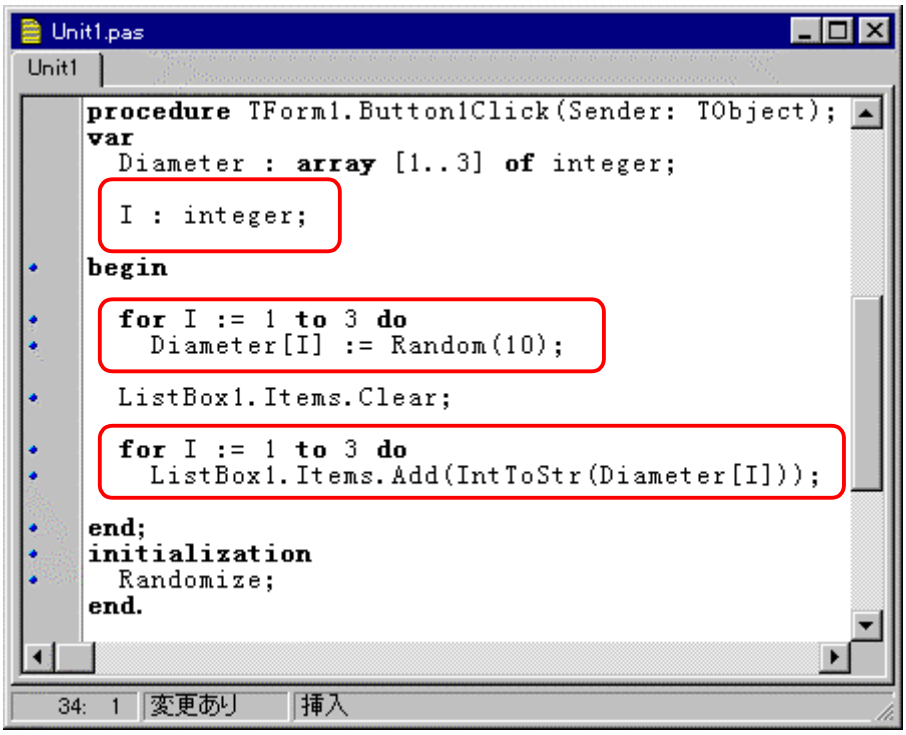
同様に、生成した乱数を表示する部分は、以下のようになります。

```
ListBox1.Items.Add( IntToStr( Diameter[ 1 ] ) );
ListBox1.Items.Add( IntToStr( Diameter[ 2 ] ) );
ListBox1.Items.Add( IntToStr( Diameter[ 3 ] ) );
```

⇩

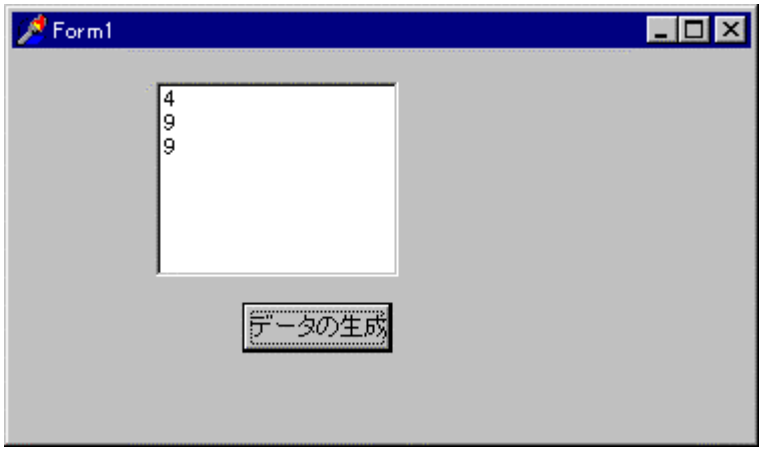
```
for I :=1 to 3 do
  ListBox1.Items.Add(  IntToStr(  Diameter[ I ]  )  );
```

以下のようにイベントハンドラを記述しましょう。



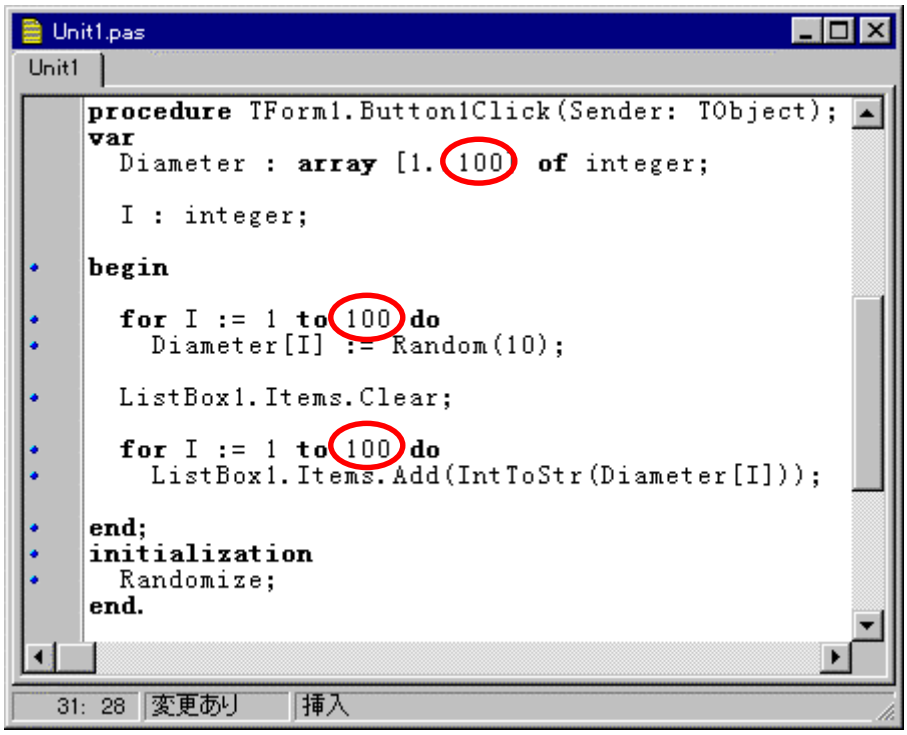
先に作ったプログラムを流用したので、保存するときは［ファイル］［プロジェクトに名前を付けて保存］を行い，フォルダfor1に保存します。さらに，［ファイル］［名前を付けて保存］を行い，フォルダfor1に保存します。

プログラムを実行してみましょう。以下のように、乱数を生成できます。

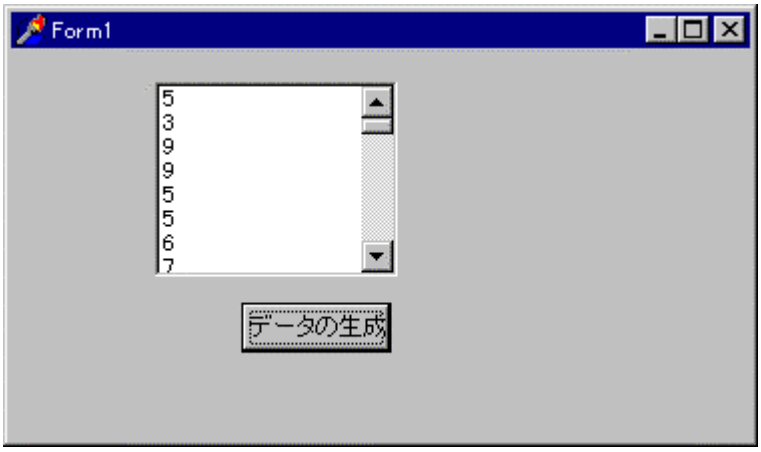


(5) 繰り返しの効果

それでは、繰り返しの効果を見てみましょう。１００個の乱数を生成し、これを表示してみます。以下の数字を１００に変更し、プログラムを実行してみましょう。



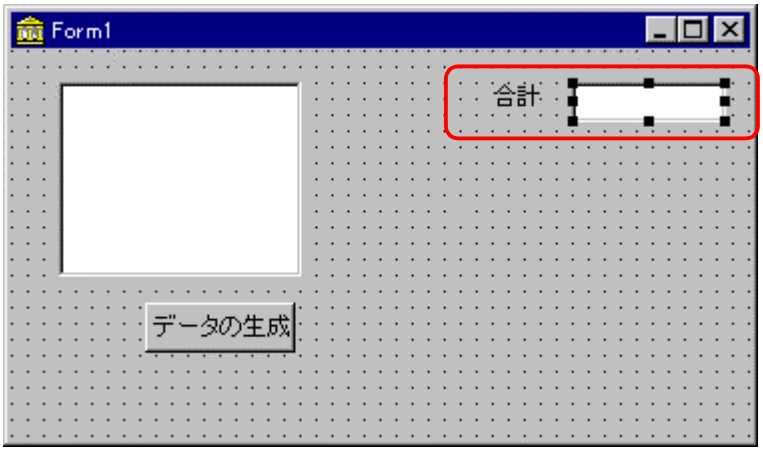
リストボックスに入りきらなくなると、スクロールバーが表示されます。１００個の乱数が生成されています。同様にして、１０００個の場合も試しましょう。



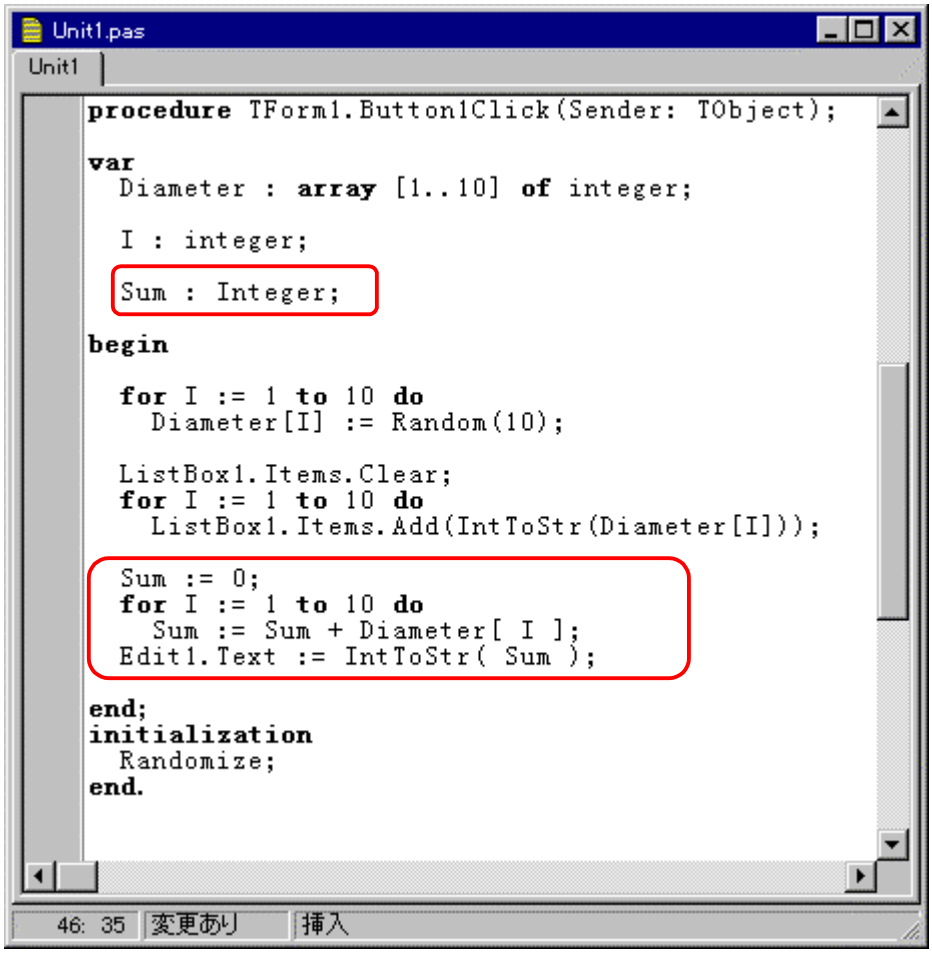
２．　配列（1次元）を使ったデータ処理

(1) 合計の計算

生成したデータの合計を求めてみましょう。フォームに、ラベルとエディットを配置します。ラベルのCaptionは合計、エディットのTextは、ヌルにしておきます。

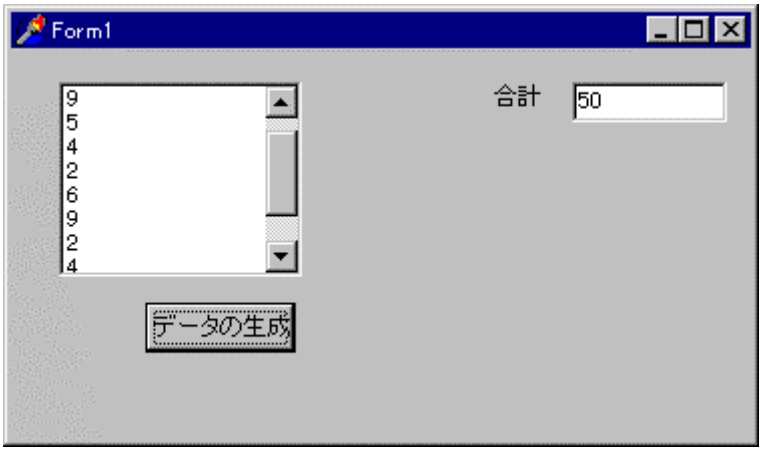


合計を求める変数Sumを宣言します。イベントハンドラは、次のようになります。



先に作ったプログラムを流用したので、保存するときは［ファイル］［プロジェクトに名前を付けて保存］を行い，フォルダfor2に保存します。さらに，［ファイル］［名前を付けて保存］を行い，フォルダfor2に保存します。

プログラムを実行してみましょう。生成したデータの合計が表示されます。

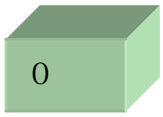


(2) 合計を求める仕組み

合計を求める仕組みは、以下の通りです。

まず、合計を求める変数Sumを0にしておきます(初期値)。

```
Sum := 0;
```

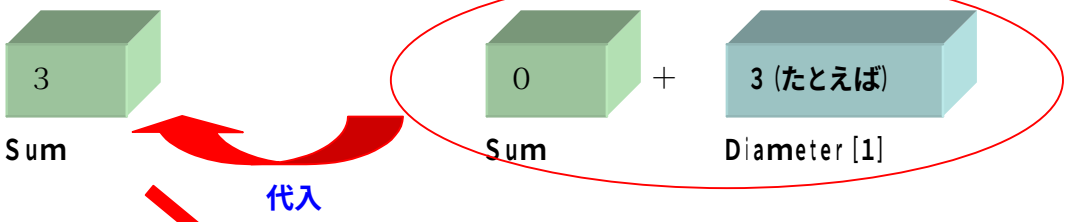


Sumという箱の中を0にする

Sum

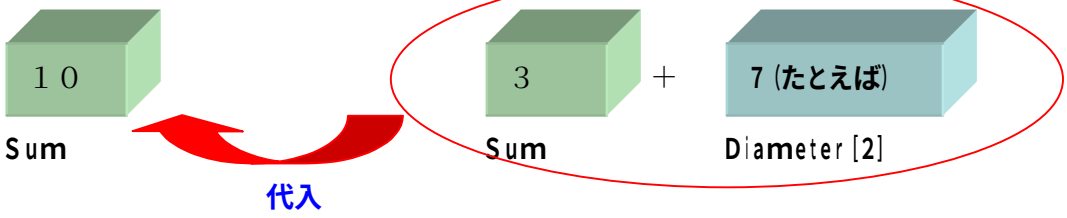
次に、配列の1番目の要素をSumに加え、その結果をSumに代入します。

```
Sum := Sum + Diameter[1];
```



次に、配列の2番目の要素　　をSumに加え、その結果をSumに代入します。

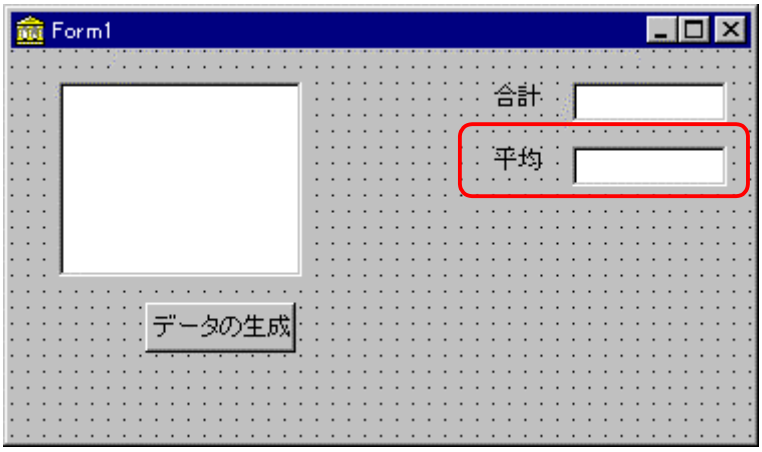
```
Sum := Sum + Diameter[2];
```



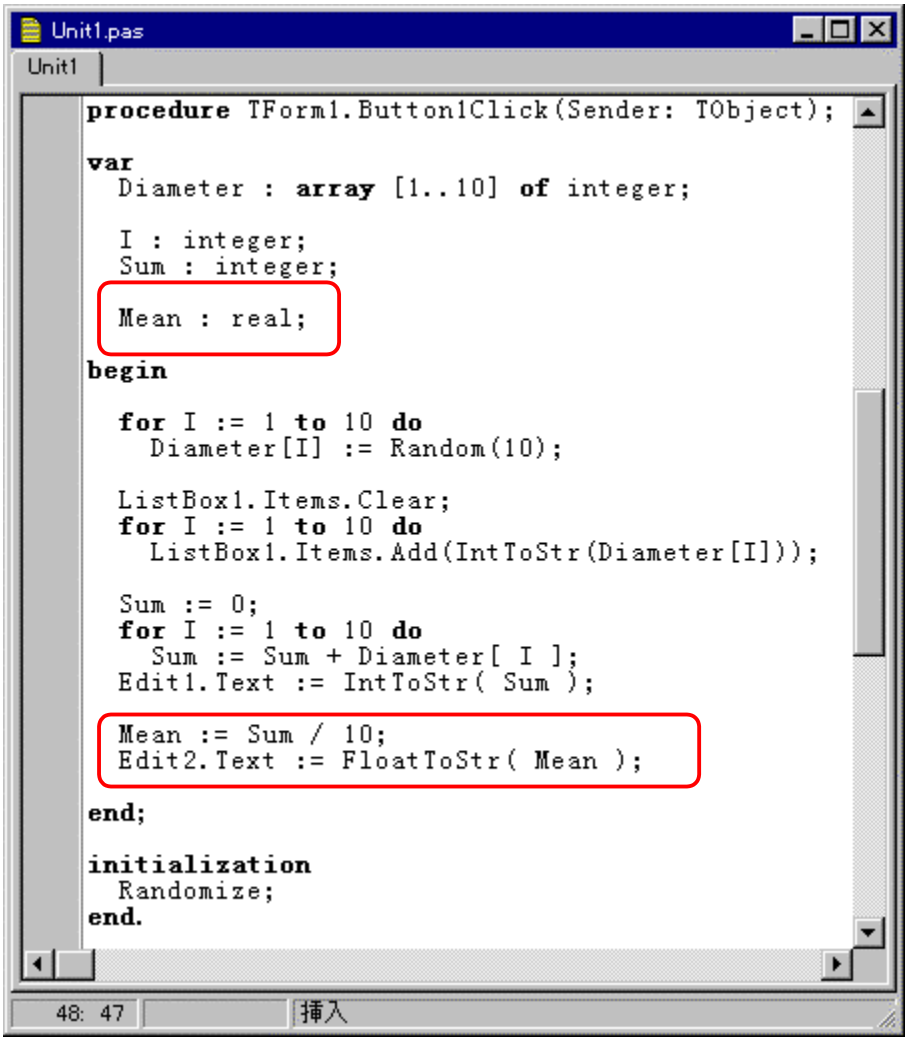
この動作をfor文によって繰り返すことにより、合計が求まります。

(3) 平均の計算

生成したデータの平均を求めてみましょう。フォームに、ラベルとエディットを配置します。ラベルのCaptionは平均、エディットのTextは、ヌルにしておきます。



平均を求める変数、Meanを実数型で宣言します。合計をデータの個数で割り、平均を求め、FloatToStrで文字列に変換した後、エディットに代入します。



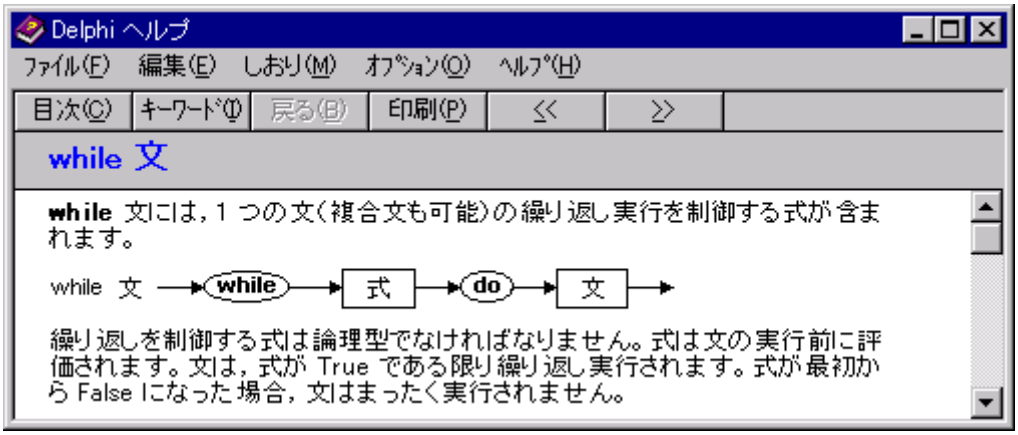
先に作ったプログラムを流用したので、保存するときは［ファイル］［プロジェクトに名前を付けて保存］を行い，フォルダfor3に保存します。さらに，［ファイル］［名前を付けて保存］を行い，フォルダfor3に保存します。

プログラムを実行すると、平均が求まります。

3．　for文の書き換え

繰り返しを行う文には、while文、repeat文があります。これらは、繰り返す回数が不明な場合に使います。しかし、for文のように繰り返す回数が分かっているものについても、while文、repaet文で書き換えることができます。

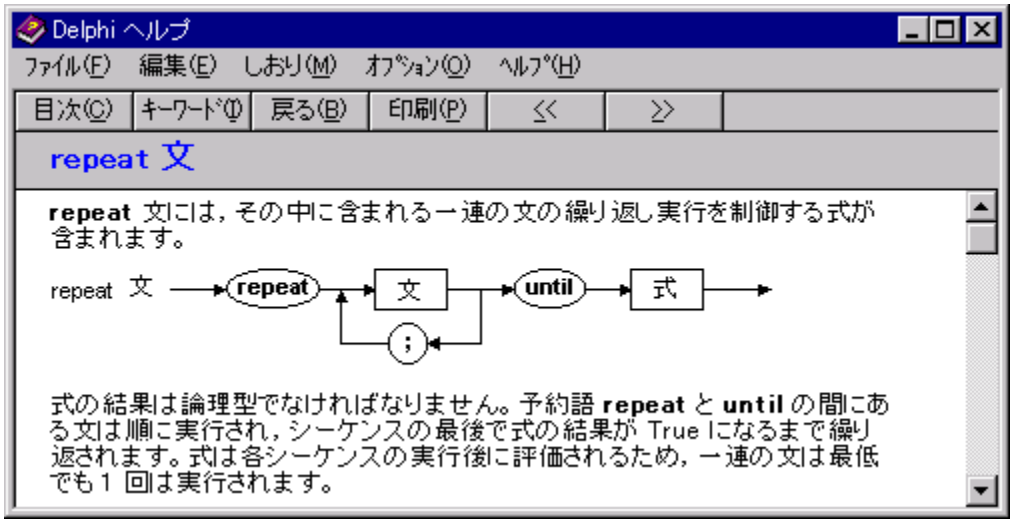
(1) while文を使った書き換え



```
for  I :=1  to 3 do
  Diameter[I] := Random(10);
```

```
I :=1;
while  I <= 3  do
begin
  Diameter[I] := Random(10);
  I := I + 1;
end;
```

(2) repeat文を使った書き換え




```
for  I :=1  to 3 do
  Diameter[I] := Random(10);
```

```
I :=1;
repeat
  Diameter[I] := Random(10);
  I := I + 1;
until I > 3;
```

repeat文の特徴は、繰り返しの対象となる文が、必ず1回実行されることです。

4.　宿題

(1) while文を使った書き換え (必修問題)

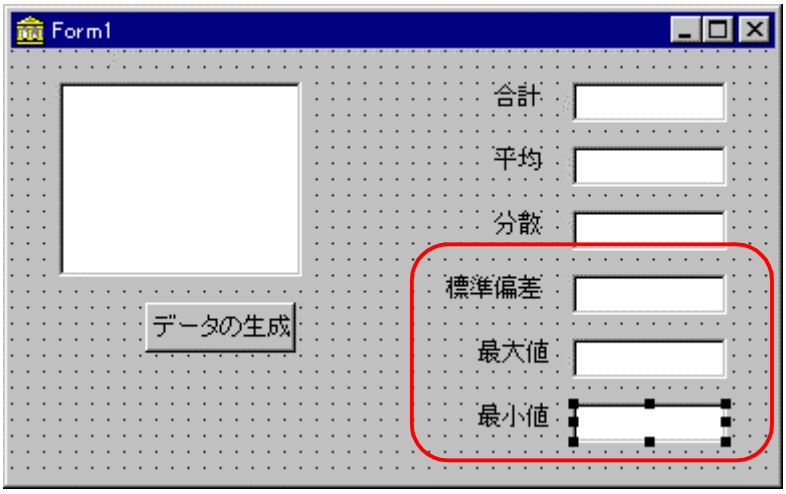
本日の実習で作成したプログラムfor3を、while文で書き換えなさい。完成したプログラムは、レポートシステムへ提出しなさい。

(2) repeat文を使った書き換え (必修問題)

本日の実習で作成したプログラムfor3を、repeat文で書き換えなさい。完成したプログラムは、レポートシステムへ提出しなさい。

(3) 余力のある人 (選択問題)

0～99までの乱数100個を生成し、その合計、平均、分散、標準偏差、最大値、最小値を求めるプログラムを作成しなさい。



実行画面

締切り　7月3日 (土)　　16：40