数学実習 2011 年度第 2 回
担当：中口悦史

今週の講義メモ：Pascal プログラムの配列の利用，文の種類，実行順序の制御

配列 array

- 配列 ≒ 変数を並べて，順に番号を振って，表のようにひとまとめにしたもの
  2 次元，3 次元，…の配列も使える。
- 配列の宣言：　var で始まる変数宣言部で，配列の名前，キーワード array，添字 (番号) の範囲を四角カッコ [ ] で括ったもの，データ型，を並べて書く。
  例：　
  ```
  var a: array[1..5] of integer;
      b: array[1..3,1..4] of real;
  ```
  注：　添字の範囲を，実際に使用する範囲より広めに宣言しても，問題はない。むしろ推奨。
- 配列の名前付けのルールも，変数と同じ
- 要素：　配列を構成する，個々の変数に相当するもの。添字を使って指定して，変数と同じように扱える。
- 添字：四角カッコの [ ] 中に書く番号指定。整数型でないといけない。整数型なら変数も使える。
- 要素の指定：　1 次元配列のときは a[2] , a[i] など，2 次元配列のときは b[2,1] , b[i,j] など。

文の種類

- 単純文
  - 代入文：　変数 := 式 の形。基本形。
  - 手続き呼出し：　writeln(..) など。
- 構造文
  - 複合文：　begin と end の間に，複数の文をセミコロン ; で区切って，並べたもの。
    プログラムの実行文部全体もひとつの複合文の形をしている。
  - 条件文：　条件によって実行する文を選択する。if 文，case 文。
  - 繰返し文：　何度か繰り返し実行する。while 文，repeat 文，for 文。

以降，単に「文」と書いてあるところには，上のどれかの種類の文を書かなければならない。例えば,「文」と書いてあるところに複数の文を書く必要がある場合は，複合文の形 begin 〜 end で書かなければならない。

文の区切りについて

「文」の区切りにはセミコロンが必要だが，セミコロンと改行をペアで使う必要は無く,「行」の最後にセミコロンが必要とは限らない。セミコロンを不用意に書くとコンパイラが誤解する。特に if 文は要注意。改行は空白と同じで，記号の区切りを示す役割しかない。

字下げ（インデント）について

文が複数行に渡るときは，文のまとまりが一目で分かるように，字下げをして書くことが多い。特に，begin と end，if〜then と else，while と do，repeat と until といった，対になっている語句を目立たせるために，その間の行を字下げするとよい。

if 文 ：　条件分岐の基本

基本形：

```
if 論理式 then 文 A else 文 B
```

論理式の値が true なら文 A を，false なら文 B を実行する。

else 以下は省略可能：

```
if 論理式 then 文
```

論理式の値が true なら文を実行する。false のときは何もしない。

多重の分岐が必要なときは，if 文を重ねて書く：

if 論理式 A then 文 A else if 論理式 B then 文 B else 文 C

論理式 A の値が true なら文 A を，論理式 A の値が false で論理式 B の値が true なら文 B を，論理式 A も論理式 B も値が false なら文 C を実行する。

注意：else の直前の文の終わりにセミコロンを書くとエラーになる：else は if 文の途中に現れる記号。

論理式

- Boolean 型の定数：　真 true，偽 false
- Boolean 型の変数
- Boolean 型の値を返す関数
- 比較演算式：　二つの文字型あるいは数値型データを比較演算子でつないだもの
  比較演算子：　等しい　=,　等しくない　<>,　不等号　<, >, <=, >=
- 論理式を論理演算子でつないだもの
  論理演算子　　not, and, or （優先順位の高い順）

例：　y <> 100　　(a >= 0) and (a < 10)　　((x < 0) or (x > 1)) and (z <> 0)

注意 1：論理演算と比較演算が混在するときは，比較演算を丸カッコで括っておかないといけない。

注意 2：$0 \leq a < 10$ など 3 項の比較の場合は，上の例のように and を使って 2 項の比較に分解して書かないといけない。

case 文

case 式 of　値 A1, ..., 値 Am：文 A;　⋯：　値 Z1, ..., 値 Zn：文 Z　end

式 A の値が　値 A1, ..., 値 Am のときは文 A を，⋯，　値 Z1, ..., 値 Zn のときは文 Z を，実行する。

式 A は整数型，文字型，論理型のいずれかのデータ型の値を持つものであればよい。

while 文

条件が成立している間処理を繰り返す：

while 論理式 do 文

論理式の値が true なら文を実行する，を繰り返す。文に複数の命令文を書く場合は複合文 begin〜end で書く。

文に複数の命令を書く場合は複合文 begin〜end で書く：

while 論理式 do begin 文 1; 文 2; ⋯; 文 n end

repeat 文

条件が成立するまで処理を繰り返す：

repeat 文 until 論理式

文を実行して，論理式の値が false なら repeat の次の行へ戻る，を繰り返す。文には複合文 begin〜end も書ける。

for 文

回数を指定して処理を繰り返す：

for 制御変数 := 初期値 to 最終値 do 文

初めに制御変数に初期値を代入する。制御変数の値が最終値を超えなければ，次の処理を繰り返す：

文を実行して，制御変数の値を 1 増やす。

文に複数の命令を書く場合は複合文 begin〜end で書く：

for 制御変数 := 初期値 to 最終値 do begin 文 1; 文 2; ⋯; 文 n end

これと次と同じ：

制御変数 := 初期値 ;
while (制御変数 <= 最終値) do begin 文 1; 文 2; ⋯; 文 n; 制御変数 := 制御変数 +1 end

あるいはこれも同じ：

制御変数 := 初期値 ; if (制御変数 <= 最終値) then
repeat begin 文 1; 文 2; ⋯; 文 n; 制御変数 := 制御変数 +1 end until (制御変数 > 最終値)

今週のサンプルプログラム

(2a) [Renshu2a.pas] 配列への代入と読み出し

```
program Renshu2a( input, output );
var a: array [1..4] of real;
    i: integer;
begin
  a[1] := 3; a[2] := 5; a[3] := 7; a[4] := 1; { 配列の各要素へ値を代入 }
  writeln( '配列 a の 1 番目は', a[1] );
  i := 3;
  writeln( '配列 a の', i, '番目は', a[i] );
  writeln( '配列 a の全体：', a[1], ' ', a[2], ' ', a[3], ' ', a[4] );
  writeln( '実数を入力してください。' );
  read( a[i] );
  writeln( '配列 a の全体：', a[1], ' ', a[2], ' ', a[3], ' ', a[4] )
end.
```

(2b) [Renshu2b.pas] 数値データの配列 2 個の間の演算

```
program Renshu2b( input, output );
var a, b, c, d: array [1..4] of real;
begin
  a[1] := 3; a[2] := 5; a[3] := 7; a[4] := 1;
  b[1] := 2; b[2] := 4; b[3] := 6; b[4] := 8;
  { c := a + b }
  c[1] := a[1] + b[1]; c[2] := a[2] + b[2];
  c[3] := a[3] + b[3]; c[4] := a[4] + b[4];
  { d := sum of a }
  d[1] :=        a[1]; d[2] := d[1] + a[2];
  d[3] := d[2] + a[3]; d[4] := d[3] + a[4];
  writeln( '配列 a, b, c, d: ' );
  writeln( a[1], ' ', b[1], ' ', c[1], ' ', d[1] );
  writeln( a[2], ' ', b[2], ' ', c[2], ' ', d[2] );
  writeln( a[3], ' ', b[3], ' ', c[3], ' ', d[3] );
  writeln( a[4], ' ', b[4], ' ', c[4], ' ', d[4] )
end.
```

(2c) [Renshu2c.pas] 数値の比較

```
program Renshu2c( input, output );
var n: integer;
begin
  writeln( '整数を入力してください。' );
  read( n );

  { とりあえず 2 分岐の例 }
  if (n > 0) then
    writeln( 'それは正の数です。' )
```

```
  else   { n>0 でない場合 }
    writeln( ’それは正の数ではありません。’ );

  { 3 分岐の作り方 }
  if (n >= 0) and (n <= 9) then
    writeln( ’それは 0 以上 9 以下の整数です。’ )
  else if (n >= 10) then
    writeln( ’それは 10 以上の整数です。’ )
  else   { n<0 の場合しか残らないはず }
    writeln( ’それは負の数です。’ );

  { 次のように書いても同じこと }
  if (n >= 10) then
    writeln( ’それは 10 以上の整数です。’ )
  else if (n >= 0) then   { n<10 が確定している }
    writeln( ’それは 0 以上 9 以下の整数です。’ )
  else   { n<0 が確定している }
    writeln( ’それは負の数です。’ );

  { 失敗例 }
  if (n >= 0) then
    writeln( ’それは 0 以上の整数です。’ )
  else if (n >= 10) then   { n<0 が確定しているので，この分岐は使われない }
    writeln( ’それは 10 以上の整数です。’ )
  else   { n<0 が確定している }
    writeln( ’それは負の数です。’ );
end.
```

(2d) [Renshu2d.pas] BMI で肥満判定

```
program Renshu2d( input, output );
var height, weight, bmi: real;
begin
  writeln( ’身長 [m] を入力してください。’ );
  read( height );
  writeln( ’体重 [kg] を入力してください。’ );
  read( weight );

  bmi := weight / sqr(height);

  if (bmi >= 25.0) then
    writeln( ’太り気味ですね。’ )
  else if (bmi >= 18.5) then   { 18.5 <= bmi < 25.0 が普通体重 }
    writeln( ’標準的な体型ですね。’ )
  else   { n<0 が確定している }
    writeln( ’やせ気味ですね。’ )
end.
```

(2e) [Renshu2e.pas] 1 次方程式 $ax + b = 0$ の解

```
program Renshu2e( input, output );
var a, b, x: real;
begin
  write( 'a を入力：' ); read( a );
  write( 'b を入力：' ); read( b );
  writeln( '方程式 ax+b=0 の解は ' );
  if (a = 0) then begin   { 0 x + b = 0 の場合，割り算ができない }
    writeln( 'b=0 の解なので，' );
    if (b = 0) then writeln( '任意の実数！' )
    else            writeln( 'ありません！' )
  end
  else begin   { a<>0 だから割り算ができる }
    x := -b/a;
    writeln( x )
  end
end.
```

(2f) [Renshu2f.pas] サイコロの目は？

```
program Renshu2f( input, output );
var k: integer;
begin
  writeln( 'サイコロの目はいくつ？ (1 から 6 まで):' );
  read( k );
  case k of
    1, 3, 5: writeln( '残念..' );
    2, 4: writeln( 'ラッキー！' );
    6: writeln( 'おめでとう !!' )
  end   { case 文の終わり }
end.
```

(2g) [Renshu2g.pas] 繰り返し表示：while の場合

```
program Renshu2g( input, output );
var i: integer;
begin
  i := 1;
  while ( i <= 10 ) do
  begin
    writeln( 'No. ', i );
    i := i + 1
  end
end.
```

(2h) [Renshu2h.pas] 繰り返し表示：repeat の場合

```
program Renshu2h( input, output );
var i: integer;
begin
  i := 1;
```

```
  repeat
  begin
    writeln( 'No. ', i );
    i := i + 1
  end
  until ( i > 10 )
end.
```

(2i) [Renshu2i.pas] 繰り返し表示：for の場合

```
program Renshu2i( input, output );
var i: integer;
begin
  for i := 1 to 10 do
    writeln( 'No. ', i )
end.
```

(2j) [Renshu2j.pas] 2 で何回割り切れる？(配列版)

```
program Renshu2j( input, output );
var w: array[0..100] of integer;
    x, n: integer;
begin
  x := 200;
  n := 0;
  w[0] := x;
  while ( w[n] mod 2 = 0 ) do
  begin
    w[n+1] := w[n] div 2;
    n := n + 1
  end;
  writeln( x, 'は2の', n, '乗で割り切れる。' )
end.
```

(2k) [Renshu2k.pas] 2 で何回割り切れる？(変数使い回し版)

```
program Renshu2k( input, output );
var x, w, n: integer;
begin
  x := 200;
  n := 0;
  w := x;
  while ( w mod 2 = 0 ) do
  begin
    w := w div 2;
    n := n + 1
  end;
  writeln( x, 'は2の', n, '乗で割り切れる。' )
end.
```

(2l) [Renshu2l.pas] 平方根を計算する（配列版）

```
program Renshu2l( input, output );
var x: array[0..100] of real;
    a, eps: real;
    n: integer;
begin
  a := 5.0; { calculate square root of 5.0 }
  n := 0;
  x[0] := 1.0; { initialize }
  eps := 1.0e-7; { precision }
  repeat
  begin
    x[n+1] := ( a / x[n] + x[n] ) / 2;
    n := n + 1
  end
  until ( abs( x[n] - x[n-1] ) < eps );
  writeln( a, ' の平方根は', x[n] )
end.
```

(2m) [Renshu2m.pas] 平方根を計算する（変数使い回し版）

```
program Renshu2m( input, output );
var a, x, xold, eps: real;
begin
  a := 5.0; { calculate square root of 5.0 }
  x := 1.0; { initialize }
  eps := 1.0e-7; { precision }
  repeat
  begin
    xold := x;
    x := ( a / xold + xold ) / 2
  end
  until ( abs( x - xold ) < eps );
  writeln( a, ' の平方根は', x )
end.
```

(2n) [Renshu2n.pas] 1 から 100 までの和と二乗和を，漸化式で計算（配列版）

```
program Renshu2n( input, output );
var sum1, sum2: array[0..100] of integer;
    n, i: integer;
begin
  n := 100;
  sum1[0] := 0; sum2[0] := 0;
  for i := 1 to n do
  begin
    sum1[i] := sum1[i-1] + i;
    sum2[i] := sum2[i-1] + sqr( i )
  end;
  writeln( '1 から', n, ' までの和 = ', sum1[n] );
```

```
  writeln( '1から', n, 'までの平方和 = ', sum2[n] )
end.
```

(2o) [Renshu2o.pas] 1 から 100 までの和と二乗和を，漸化式で計算（変数使い回し版）

```
program Renshu2o( input, output );
var n, sum1, sum2, i: integer;
begin
  n := 100;
  sum1 := 0; sum2 := 0;
  for i := 1 to n do
  begin
    sum1 := sum1 + i;
    sum2 := sum2 + sqr( i )
  end;
  writeln( '1から', n, 'までの和 = ', sum1 );
  writeln( '1から', n, 'までの平方和 = ', sum2 )
end.
```

(2p) [Renshu2p.pas] ベキ乗を漸化式で計算

```
program Renshu2p( input, output );
var a, p: real;
    n, i: integer;
begin
  a := 5;
  n := 8;
  p := 1;
  for i := 1 to n do { 1にaをn回掛けるとaのn乗になる }
    p := p * a;
  writeln( a, 'の', n, '乗は', p )
end.
```

(2q) [Renshu2q.pas] ベキ乗を計算：負ベキも考慮

```
program Renshu2q( input, output );
var a, p, b: real;
    n, i, m: integer;
begin
  write( 'a = ? ' ); read( a ); { aを入力 }
  write( 'n = ? ' ); read( n ); { nを入力 }
  if ( n < 0 ) then             { nが負なら，逆数の正ベキを計算 }
    begin m := -n; b := 1/a end
  else
    begin m := n; b := a end;
  p := 1;
  for i := 1 to m do { 1にbをm回掛けるとbのm乗つまりaのn乗になる }
    p := p * b;
  writeln( a, 'の', n, '乗は', p )
end.
```

(2r) [Renshu2r.pas] 数値データをキーボードから読み込んで，配列に保存して，和を計算する

```
program Renshu2r( input, output );
var data: array [1..100] of real;
    sum: real;  n, i: integer;
begin
  writeln( ’データの個数を入力してください (100 まで)’ );
  read( n );
  writeln( n, ’個のデータを入力してください。’ );
  for i := 1 to n do
    read( data[i] );
  sum := 0;
  for i := 1 to n do
    sum := sum + data[i];
  writeln( ’入力データの和は ’, sum )
end.
```

(2s) [Renshu2s.pas] 数値データの配列 2 個の間の演算：for を使って

```
program Renshu2s( input, output );
var a, b, c, d: array [1..4] of real;
    i: integer;
begin
  a[1] := 3; a[2] := 5; a[3] := 7; a[4] := 1;
  for i := 1 to 4 do begin
    b[i] := 2*i;
    c[i] := a[i] + b[i]
  end;
  d[1] := a[1];
  for i := 2 to 4 do
    d[i] := d[i-1] + a[i];
  writeln( ’配列 a, b, c, d: ’ );
  for i := 1 to 4 do
    writeln( a[i], ’ ’, b[i], ’ ’, c[i], ’ ’, d[i] )
end.
```

(2t) [Renshu2t.pas] 数値データをキーボードから読み込んで，配列に保存して，逆順に出力

```
program Renshu2t( input, output );
var data: array [1..100] of real;
    n,i: integer;
begin
  n := 10;
  writeln( n, ’個の数値を入力してください。’ );
  for i := 1 to n do
    read( data[i] );
  writeln( ’逆順に出力すると：’ );
  for i := n downto 1 do
    write( data[i], ’ ’ );
  writeln()
end.
```

(2u) [Renshu2u.pas] 行列の和，2 次元配列，2 重の for ループ

```
program Renshu2u( input, output );
var a, b, c: array[1..5,1..5] of real;
    i, j: integer;
begin
  a[1,1]:=1; a[1,2]:=2; a[2,1]:=3; a[2,2]:=4;
  b[1,1]:=5; b[1,2]:=8; b[2,1]:=6; b[2,2]:=7;

  writeln( '行列 a:' );
  for i := 1 to 2 do begin
    for j := 1 to 2 do write( ' ', a[i,j] );
    writeln
  end;
  writeln( '行列 b:' );
  for i := 1 to 2 do begin
    for j := 1 to 2 do write( ' ', b[i,j] );
    writeln
  end;

  { sum c := a + b }
  for i := 1 to 2 do
    for j := 1 to 2 do
      c[i,j] := a[i,j] + b[i,j];

  writeln( '行列 c := a + b:' );
  for i := 1 to 2 do begin
    for j := 1 to 2 do write( ' ', c[i,j] );
    writeln
  end
end.
```

今週の練習問題　課題提出は不要です。

2-1. サンプルプログラムそれぞれの計算の流れを**手計算で**チェックしてみましょう。

2-2. サンプルプログラム (2c) の数値や条件文をいろいろ変えて，`if` 文の動きを確認しましょう。

2-3. サンプルプログラム (2b) と (2s) を比較して，`for` 文の動きを確認しましょう。

2-4. サンプルプログラム (2g), (2h), (2i) を比較して，繰返し文の性質の違いについて考察しましょう。

2-5. サンプルプログラム (2j), (2k) を `repeat` 文に書き換えてみましょう。

2-6. サンプルプログラム (2l), (2m) を `while` 文に書き換えてみましょう。

2-7. サンプルプログラム (2n), (2o), (2p), (2q) を `while` 文や `repeat` 文に書き換えてみましょう。

2-8. サンプルプログラム (2j) と (2k), (2l) と (2m), (2n) と (2o), (2p) と (2q) をそれぞれ比較して，配列の利用について考察しましょう。