8　再帰（数学的帰納法）

8.1　数列 a_n，b_b，c_n

次の “漸化式で”（あるいは，“数学的帰納法によって”，“帰納的に”，“再帰的に”）定義された数列の初めの 4 項と第 9 項を書きなさい。ただし，演算 $x+y$ は足し算ではなく，x の後ろに y をつなげて書いた数とします。たとえば，$123+45=12345$ です。すなわち，数値としてではなく文字列として考えています。

(1)

$$\begin{cases} a_n = 1 & (n=1) \\ a_n = a_{n-1} + n & (n>1) \end{cases}$$

$$\begin{aligned} a_1 &= 1 \\ a_2 &= a_1 + 2 = 12 \\ a_3 &= a_2 + 3 = 123 \\ a_4 &= a_3 + 4 = 1234 \\ &\cdots \\ a_9 &= 123456789 \end{aligned}$$

(2)

$$\begin{cases} b_n = 1 & (n=1) \\ b_n = n + b_{n-1} & (n>1) \end{cases}$$

$$\begin{aligned} b_1 &= \\ b_2 &= \\ b_3 &= \\ b_4 &= \\ &\cdots \\ b_9 &= \end{aligned}$$

(3)

$$\begin{cases} c_n = 1 & (n=1) \\ c_n = c_{n-1} + n + c_{n-1} & (n>1) \end{cases}$$

$$\begin{aligned} c_1 &= \\ c_2 &= \\ c_3 &= \\ c_4 &= \\ &\cdots \\ c_9 &= \text{(何桁になるか，桁数を答えなさい)} \end{aligned}$$

8.2　再帰関数

次のプログラムは，数列 $\{a_n\}$ を求めるコンソールアプリケーションです。

SuuretsuA は自分自身 SuuretsuA を呼び出しています。このように自分自身を呼ぶ関数を**再帰関数 (recursive function)** と言います。

```
program SaikiCA;

$APPTYPE CONSOLE
uses SysUtils;

function SuuretsuA(N : Byte) : String;
         (* A(n) = 1         (n=1)  *)
         (*      = A(n-1)+n  (n>1)  *)
         (* ただし，+ は文字列の足し算 *)
begin
  if N <= 1
    then SuuretsuA := IntToStr(N)
    else SuuretsuA := SuuretsuA(N-1)+IntToStr(N);
end; (* SuuretsuA *)

begin
  WriteLn(SuuretsuA(4));
  ReadLn;
end.
```

これを実行しなさい。

SuuretsuA(n) がどういう順番で実行されるかを見るために，入ったときと出るときにメッセージを書くようにしました。

```
function SuuretsuA(N : Byte) : String;
         (* A(n) = 1         (n=1)  *)
         (*      = A(n-1)+n  (n>1)  *)
         (* ただし，+ は文字列の足し算 *)
begin
  WriteLn('SuuretsuA(', N, ') begin');
  if N <= 1
    then SuuretsuA := IntToStr(N)
    else SuuretsuA := SuuretsuA(N-1)+IntToStr(N);
  WriteLn('SuuretsuA(', N, ') end');
end; (* SuuretsuA *)
```

実行しなさい。

8.3　再帰手続き

関数 SuuretsuA で文字列 a_n を求めて，メインルーチンでそれを書くのではなく，$+n$ をするときに，n を直接コンソール画面に書く手続き KakuA に変更しました。

KakuA も自分自身を呼び出しています。このようなプロシージャを**再帰手続き (recursive procedure)** と言います。

```
procedure KakuA(N : Byte);
         (* A(n) = 1          (n=1)  *)
         (*      = A(n-1)+n   (n>1)  *)
         (* ただし，+ は文字列の足し算 *)
begin
  if N <= 1
    then Write(N)
    else begin
           KakuA(N-1);
           Write(N);
         end;
end; (* KakuA *)

begin
  //WriteLn(SuuretsuA(4));
  KakuA(4);
  ReadLn;
end.
```

実行しなさい。

やはり，プロシージャの開始と終了がわかるようにメッセージを出してみましょう。メッセージを書いた後と，n を書いた後で， Enter キーを押すまで停止しているので，次に何が書かれるか考えてから Enter キーを押して次に進んでください。

```
procedure KakuA(N : Byte);
         (* A(n) = 1          (n=1)  *)
         (*      = A(n-1)+n   (n>1)  *)
         (* ただし，+ は文字列の足し算 *)
begin
  Write('KakuA(', N, ') begin');
  ReadLn;
  if N <= 1
    then begin
           Write(N) ;
           ReadLn;
         end
    else begin
           KakuA(N-1);
           Write(N);
           ReadLn;
         end;
  Write('KakuA(', N, ') end');
  ReadLn;
end; (* KakuA *)
```

8.4　数列 b_n，c_n の場合

数列 b_n，c_n についても同様の関数と手続きを書いて実行してみましょう。特に c_n は複雑ですから，動作をよく理解してください。

再帰手続きの動作をよく理解したら，次の問題に進んでください。

8.5 問題

次のプログラムを実行して，“a,b ? ” に対し “13 8” と入力したときの出力を書きなさい。出力は，プロシージャに入った時と出る時にしています。途中の計算は出力ではないので，解答とは別のところに書くようにしなさい。

```
program GcdCA;
$AppType Console
uses SysUtils;

procedure GcdWithXY(M,N : integer; var D,X,Y : integer);
  var
    MdivN,MmodN,D1,X1,Y1 : integer;
  begin
    WriteLn('m=',M:4, '  n=',N:4, '  d=','?':4, '  x=','?':4, '  y=','?':4);
    if N = 0
      then begin
             D := M;
             X := 1;
             Y := 0;
           end
      else begin
             MdivN := M div N;
             MmodN := M mod N;
             GcdWithXY(N,MmodN,D1,X1,Y1);
             D := D1;
             X := Y1;
             Y := X1 - Y1 * MdivN;
           end;
    WriteLn('m=',M:4, '  n=',N:4, '  d=',D:4, '  x=',X:4, '  y=',Y:4);
  end; {GcdWithXY}

var
  A,B,GCD,P,Q : integer;
begin Main
  Write('a b ? ');
  ReadLn(A,B);
  WriteLn;
  GcdWithXY(A,B,GCD,P,Q);
  WriteLn;
  WriteLn(A, ' * ', P, ' + ', B, ' * ', Q, ' = ', GCD);
  Write('Push return key':79);
  ReadLn;
end.
```