第1章　Programming

1.1　代入と出力 (Variables, printf)

[[[解説]]]

値の変数への代入 (:=)

Maple は変数の初期設定で型宣言をする必要がない．数式処理の章で示したとおり，変数への代入は:=を使う．変数 a,b にそれぞれ 10,3 を代入し，a+b の結果を c に代入するというプログラムは以下の通り．

```
> a:=10: b:=3: c:=a+b;
```

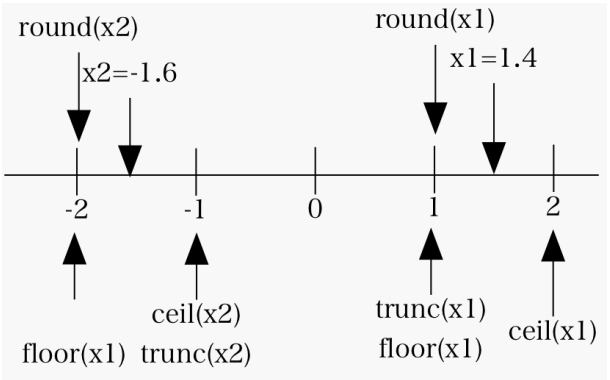
$$c := 13$$

整数と浮動小数点数

浮動小数点数から整数に直すにはいくつかの関数がある．

- trunc：数値から数直線で 0 に向って最も近い整数
- round：数値の四捨五入
- floor：数値より小さな最も大きな整数
- ceil：数値より大きな最も小さな整数

負の値の時に floor と trunc は違った値を返す．



小数点以下を取りだすには frac が用意されている．

```
> frac(1.7);
```

$$0.7$$

整数の割り算は irem(余り) と iquo(商).

```
> irem(7,3); #res: 1
> iquo(7,3); #res: 2
```

出力 (print, printf)

Maple ではデフォルトで結果が出力される．これを抑えるには行末の ”;” を ”:” に変える必要がある．出力を明示的におこなうには print を使う．デバッグの時に便利．

```
> x:=1: print(x); #res: 1
```

さらに，出力を整えるのに便利な printf 関数がある．これは C 言語と同じ構文で，

```
> printf("Hello world!!\n");
```

```
Hello world!!
```

と打ち込んで enter を押せば，出力が即座に表示される．値を表示するときには，

```
> i:=3: printf("%3d\n",i);
```

```
  3
```

となる．これは

「変数 i に入っている値を，3 桁の整数形式で打ち出した後，改行せよ」

と言う意味．%3d が出力の形式，\n が改行を意味する．OS によっては，\は ¥ と画面あるいはキーボードで表示されているかもしれない．実数の出力指定は%10.5f で，全部で 10 桁，小数点以下 5 桁で浮動小数点数を表示．複数の変数の出力は

```
> printf("%3d : %10.5f \n",i,a);
```

などとなる．

表 1.1: printf の書式指定

%指定	意味
%o	整数を 8 進数で表示.
%d	整数を 10 進数で表示.
%x,%f	浮動小数点数として表示.
%e,%s	文字列を出力.

1.2　ループ (Loop)

[[[解説]]]

for-loop

繰り返す操作は loop でおこなう．もっとも単純な for-loop.

```
> for i from 1 to 3 do
    i;
  end do;
```

```
1
2
3
```

初期値や増減を調整した for-loop

```
> for i from 10 by -2 to 0 do
    i;
  end do;
```

```
10
8
6
4
2
0
```

loop 回数が少ないときは，loop の中身も出力される．これを止めるには，end do; の最後のセミコロンをコロンに変える．

二重ループ

i,j という二つの変数を使って 2 重化したループ.

```
> for i from 1 to 3 do
    for j from 1 to 3 do
      print(i,j);
    end do;
  end do;
```

1, 1

1, 2

1, 3

2, 1

2, 2

2, 3

3, 1

3, 2

3, 3

while-loop も同じように使える.

```
> i:=0;
  while i<5 do
    i:=i+1;
  end do;
```

0

1

2

3

4

5

[[[課題]]]

1. printf を使って次のように表示せよ.

   i) Hello world. ii) 1+1=2

2. 次の数を順に表示せよ.

   i) 1 から 5 までの整数. ii) 5 から 1 までの整数 . iii) 1 から 10 にある偶数.

3. 9x9 表を作れ.

4. 1 から 5 までの和を求めよ.

5. n を 5 にして，$n! = n \times (n-1) \times (n-1) \cdots 3 \times 2 \times 1$ を求めよ.

[[[解答例]]]

1. ```
   > printf("Hello world!!\n");

   Hello world!!
   ```

   ```
   > i:=1;
   > printf("%d+%d=%d\n",i,i,i+i);
   ```
   ```

$$1$$

$$1 + 1 = 2$$

2. i)

```
> for i from 1 to 5 do
    i;
  end do;
```

$$1$$

$$2$$

$$3$$

$$4$$

$$5$$

ii)

```
> for i from 5 to 1 by -1 do
    i;
  end do;
```

$$5$$

$$4$$

$$3$$

$$2$$

$$1$$

iii)

```
> for i from 2 to 10 by 2 do
    i;
  end do;
```

$$2$$

$$4$$

$$6$$

$$8$$

$$10$$

3.
```
> for i from 1 to 9 do
    for j from 1 to 9 do
      printf("%4d",i*j);
    end do;
    printf("\n");
  end do;
```

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

```
> sum1:=0; for i from 1 to 5 do
    sum1:=sum1+i;
  end do;
```

0

1

3

6

10

15

4.
```
> n:=5:
  total1:=1:
  for i from 1 to n do
    total1:=total1*i;
  end do;
```

1

2

6

24

120

1.3 配列 (List)

[[[解説]]]

配列は変数を入れる箱が沢山用意されていると考えればよい．配列を使うときは，箱を指す数 (示数，index) をいじっているのか，箱の中身 (要素) をいじっているのかを区別すれば，動作を理解しやすい．Maple にはいくつかの配列構造が用意されている．もっとも，頻繁に使う list を示す．

基本

リスト構造は，中に入れる要素を [] でくくる．

```
> restart; list1:=[1,3,5,7];
```

$$list1 := [1, 3, 5, 7]$$

要素にアクセスするには，以下のようにインデックスを指定する．

```
> list1[2]; list1[-1]; list1[2..4];
```

$$3$$
$$7$$
$$[3, 5, 7]$$

-1,-2 等は後ろから 1 番目，2 番目を指す．C 言語と違い 0 番目はない．

```
> list1[0];
```

```
Error, invalid subscript selector
```

ひとつの要素を書き換えるには，以下のようにする．

```
> list1[3]:=x: list1;
```

$$[1, 3, x, 7]$$

要素の数，および要素の中身を取り出すには以下のようにする．

```
> nops(list1);
> op(list1);
```

$$4$$
$$1, 3, x, 7$$

for-loop の省略形

for-loop を省略するのによく使う手を二つ．(#より後ろはコメント文です)

配列の生成 (seq)

```
> aa:=[]; #空で初期化
  for i from 1 to 3 do
    aa:=[op(aa),i]; #付け足していく
  end do:
  print(aa);
```

$$aa := []$$

$$[1, 2, 3]$$

同じことを seq を使って短く書くことができる.

```
> aa :=[seq(i,i=1..3)];
```

```
{\it aa}\, := \,[1,2,3]
```

配列の和 (sum)

```
> n:=nops(aa):
  total:=0:
  for i from 1 to n do
    total:=total+aa[i];
  end do:
  print(total):
```

$$6$$

同じことを sum を使って短く書くことができる.

```
> sum(aa[i],i=1..nops(aa));
```

```
Error, invalid subscript selector
```

sum や seq を使っていると，このようなエラーがよくでる．これは，for-loop をまわすときに i に値が代入されているため引っかかる．変数を換えるか，i を初期化すればよい.

```
> i;
```

$$4$$

```
> sum(aa[j],j=1..nops(aa));
```

$$6$$

リストへの付け足し (append, prepend)

op を用いると，リストに新たな要素を前後，あるいは途中に付け足すことができる.

```
> list1:=[op(list1),9];
```

$$list1 := [1, 3, x, 7, 9]$$

2 つの要素の入れ替え

要素の 3， 4 番目の入れ替えは以下の通り.

```
> tmp:=list1[3]:
  list1[3]:=list1[4]:
  list1[4]:=tmp:
  list1;
```

$$[1, 3, 7, x, 9]$$

2 次元配列 (listlist)

[ ] を二重化することで 2 次元の配列を作ることも可能で, リストのリスト (listlist) と呼ばれる.

```
> l2:=[[1,2,3,4],[1,3,5,7]];
```

$$l2 := [[1, 2, 3, 4], [1, 3, 5, 7]]$$

要素へのアクセスは以下の通り.

```
> l2[2]; l2[2,3]; l2[2][3];
```

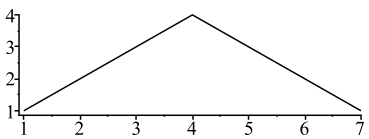
$$[1, 3, 5, 7]$$
$$5$$
$$5$$

list の表示 (listplot)

list に入っている数値を視覚化するのには listplot が便利.

```
> la:=[1,2,3,4,3,2,1];
  with(plots):
  listplot(la);
```

$$[1, 2, 3, 4, 3, 2, 1]$$



[[[課題]]]

1. 1 から 100 までの整数のうち 5 個をランダムに含んだ配列を生成せよ.

   1 から 6 までのランダムな数を生成する関数は,

   ```
   > roll:=rand(1..6):
   ```

として作ることができる．実行は次の通り．

```
> seq(roll(),i=1..10);
```

$$5, 2, 5, 6, 2, 3, 4, 4, 6, 5$$

2. さいころを１００回振って，出た目１から６が何回出たかを表示せよ．
3. コイン６枚を一度に投げて，表向きの枚数を数えるプログラムを書け．
4. 0 から 9 までの整数 5 個から 5 桁の整数を作れ．（1 桁目が 0 になっても気にするな）
5. 小数点以下 8 桁のそれぞれの桁の数を配列に格納せよ．8 桁の少数は以下のようにして作られる．
6. 255 以下の 10 進数をランダムに生成して，8 桁の 2 進数へ変換せよ．
   整数の割り算には irem(余り) と iquo(商) がある．使用法は以下の通り．

```
> irem(7,3); #res: 1
> iquo(7,3); #res: 2
```

[[[解答例]]]

1.
```
> roll:=rand(1..100):
  [seq(roll(),i=1..5)];
```

$$[27, 96, 17, 90, 34]$$

2.
```
> roll:=rand(1..6):
> A:=[seq(0,i=1..6)];
```

$$[0, 0, 0, 0, 0, 0]$$

```
> for i from 1 to 100 do
    i1:=roll();
    A[i1]:=A[i1]+1;
  end do:
  A;
```

$$[16, 18, 21, 18, 18, 9]$$

3.
```
> toss:=rand(0..1):
  n:=6:
  up:=0:
  for i from 1 to n do
    up:=up+toss();
  end do:
  up;
```

4.
```
> roll:=rand(0..9):
  n:=5:
  A:=[seq(roll(),i=1..n)];
```

$$[5, 7, 3, 7, 6]$$

```
> sum1:=0:
  for i from 1 to n do
    sum1:=sum1*10+A[i];
  end do:
  sum1;
```

$$57376$$

5.
```
> restart;
  n:=8:
  roll:=rand(10^(n-1)..10^n):
  B:=evalf(roll()/10^n,8);
  A:=[]:
```

$$0.19550684$$

```
> B:=10*B;
  for i from 1 to n do
    A:=[op(A),floor(B)];
    B:=(B-A[i])*10;
  end do:
  A;
```

$$1.95506840$$

$$[1, 9, 5, 5, 0, 6, 8, 4]$$

6.
```
> n:=8:
  roll:=rand(0..2^n-1):
  B:=roll();
```

$$246$$

```
> A:=[seq(0,j=1..n)]:
  for i from 1 to n do
    A[n-i+1]:=irem(B,2);
    B:=iquo(B,2);
  end do:
  A;
```

$$[1, 1, 1, 1, 0, 1, 1, 0]$$

1.4　交通整理 (If)

[[[解説]]]

if

もっとも簡単な if 文の例.

```
> x:=-4:
  if (x<0) then
     y:=-x;
  end if;
```

4

例外付き.

```
> x:=3:
  if (x<0) then
    y:=-x;
  else
    y:=x;
  end if;
```

3

2 個の条件がある例

```
> x:=3:
  if (x<0) then
    y:=-x;
  elif (x>5) then
    y:=x;
  else
    y:=2*x;
  end if;
```

6

条件文に使える式と意味　関係演算子は<，<=，>，>=，=，<>で表記される．論理演算子には and, or, xor, not がある．その他にもブール値を返す関数として implies, evalb, type などいくつかあり，条件分岐に使える．

表 1.2: 条件分岐のいくつかの例

x と y の値が一致	(x=y)
x と y の値が一致しない	(x<>y)
条件文を複数つなぐ	((x>0) and (x<4))
	((x<0) or (x>4))
	not (x=0)

next と break

do-loop の途中で流れを変更するための命令．next は do-loop を一回スキップ．break はそこで do-loop を一つ抜ける．以下のコードの出力結果を参照．

```
> for i from 1 to 5 do
    if (i=3) then
  next;
    end if;
    print(i);
  end do:

#res: 1 2 4 5

> for i from 1 to 5 do
    if (i=3) then
      break;
    end if;
    print(i);
  end do:

#res: 1 2
```

[[[課題]]]

1. 西暦を代入したら，明治，大正，昭和，平成で答えてくれるプログラムを作成せよ．西暦 1868, 1912,1926,1989 年をそれぞれの元年とする．
2. 整数を代入したら，それ以下の素数をすべて表示するプログラムを作れ．素数かどうかの判定は Maple コマンドの isprime を用いよ．
3. p が素数で p+2 も素数のとき，これらは双子の素数と呼ばれる．10 以上，100 以下の双子の素数を全部見つけて出力せよ．
4. 素数判定を原理から実現せよ．ある数 n が素数かどうか (自分自身の数 n と 1 以外の数で割りきれないかどうか) を判定せよ．割り算の余り (剰余) は irem で求まる．例えば

   ```
   > residue:=irem(9,2);
   ```

   として変数 residue(余りの英語) を printf してみよ．番兵を置いておいて，n-1 から 2 までの数で n を次々と割っていき，一度でも割り切れれば番兵にマークをつける．ループが終わった後に番兵のマークを見て素数 (prime number) かどうかを判定すればよい．
5. うるう年かどうかを表示するプログラムをかけ．
   うるう年は 4 で割り切れる数の年．ただし，100 で割り切れる年はうるう年でなく，400 で割り切れる年はうるう年．
6. ゴールドバッハの予想
   「6 以上の偶数は二つの素数の和として表わされる」という予想を 100 以下の偶数について検証せよ．あらかじめ 100 までの素数をリストアップしておいてそのなかから組み合わせを探すと便利．

[[[解答例]]]

1. ```
   > year:=1890;
   if year<1868 then printf("明治より前です. \n");
   elif year<1912 then printf("明治%d 年です. \n",year-1868+1);
   elif year<1926 then printf("大正%d 年です. \n",year-1912+1);
   elif year<1989 then printf("昭和%d 年です. \n",year-1926+1);
   elif year<2011 then printf("平成%d 年です. \n",year-1989+1);
   else printf("今年より後です. \n");
   end;
   ```

   明治 23 年です.

2. ```
   > n:=10:
   for i from 1 to n do
     if (isprime(i)) then
       print(i);
     end if;
   end do;
   ```

   ```
   #res: 2 3 5 7
   ```

3. ```
   > for i from 10 to 100-2 do
       if (isprime(i) and isprime(i+2)) then
         print(i,i+2);
       end if;
     end do;
   ```

   ```
                                     11, 13
                                     17, 19
                                     29, 31
                                     41, 43
                                     59, 61
                                     71, 73
   ```

4. ```
   > n:=12:
   banpei:=0:
   for i from 2 to n-1 do
     residue:=irem(n,i);
     # print(n,residue):
     if residue=0 then
       banpei:=1;
       break;
     end if;
   end do:
   if banpei=1 then
     printf("%d is not prime number.\n",n);
   else
     printf("%d is prime number.\n",n);
   end if;
   ```

```
12 is not prime number.
```

5.
```
> year:=[2010,1984,2004,1800,1900,1600,2000]:
  for i from 1 to nops(year) do
    if (irem(year[i],400)=0) then
      printf("%d is a leap year.\n",year[i]);
    elif (irem(year[i],4)=0) and (irem(year[i],100)<>0) then
      printf("%d is a leap year.\n",year[i]);
    else printf("%d is not a leap year.\n",year[i]);
    end if;
  end do;
```

```
2010 is not a leap year.
1984 is a leap year.
2004 is a leap year.
1800 is not a leap year.
1900 is not a leap year.
1600 is a leap year.
2000 is a leap year.
```

别解

```
> for i from 1 to nops(year) do
    if (irem(year[i],4)=0) and ((irem(year[i],100)<>0) or (irem(year[i],400)=0)) then
      printf("%d is a leap year.\n",year[i]);
    else
      printf("%d is not a leap year.\n",year[i]);
    end if;
  end do;
```

略

6.
```
> prime1:=[];
  for i from 1 to 100 do
    if isprime(i) then
      prime1:=[op(prime1),i];
    end if;
  end do;
  prime1;
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97]
```

```
> nops(prime1);
```

```
> for i from 6 to 100 by 2 do
    for j1 from 1 to nops(prime1) do
      for j2 from 1 to nops(prime1) do
        if i=(prime1[j1]+prime1[j2]) then
          print(i,prime1[j1],prime1[j2]);
          break;
        end if
      end do;
      if j2<=nops(prime1) then
        break;
      end if;
    end do;
  end do;
```

```
6, 3, 3
8, 3, 5
10, 3, 7
中略
98, 19, 79
100, 3, 97
```

1.5　手続き関数 (Procedure)

[[[解説]]]

基本

複雑な手続きや，何度も繰り返すルーチンは proc で作る．proc は以下のようにして作る．

```
ユーザ関数名:=proc(仮引数)
  動作
end proc;
```

```
> test1:=proc(a)
    print(a);
  end proc:
```

proc の呼び出しは，以下のようになる．

```
> test1(13);
```

13

仮引数としてはどんな型 (変数や配列) でもよい．複数指定するときにはコンマで区切る．仮引数を proc の中で変更することはできない．下で示す global で取り込むか，local 変数にコピーして使う．

戻り値

proc の戻り値は return で指定される．return 文がないときは，最後の動作結果が戻り値となる．

```
> test2:=proc(a)
    return a+1;
  end proc:

> test2(13);
```

14

グローバル (大域)，ローカル (局所) 変数

proc の内部だけで使われるのが local，外部を参照するのが global．global,local を省略しても Maple が適当に判断してくれるが，あまり信用せず，明示的に宣言した方が良い．宣言の仕方は以下の通り．

```
変数名:=proc(引数)
  local 変数, 変数...;
  global 変数, 変数...;
  動作の記述
end proc;
```

[[[課題]]]

1. 三角形の面積底辺と高さを引数として，面積を返す関数 area を作れ．

2. MyIsprime 前章の課題４で求めた素数判定プログラムを proc にせよ．

3. ルートの距離

   二つの位置座標 x1:=[0.0, 0.0] x2:=[1.0, 1.0] から距離を求める MyDistance 関数を作れ．

   次に，４つの位置座標 x[1]=[0.0, 0.0] x[2]=[1.0, 1.0] x[3]=[1.0, 0.0] x[4]=[0.0, 1.0] を読み込んで，座標順に [1,2,3,4,1] と巡る距離を求めよ．

4. 最大数

   ランダムな整数が格納されたリストを受け取り，そのリスト中の最大数を返す関数 MyMax を作れ．1 から 100 までのランダムな整数のリストは次のようにして作れる．

```
> roll:=rand(1..100):
  n:=50:
  A:=[seq(roll(),i=1..n)];
```

$$A := [45, 96, 6, 98, 59, 44, 100, 38, 69, 27, 96, 17, 90, 34, 18, 52, 56, 43, 83, 25, 90, 93, 60, 93, 14, 50, 47, 8, 46, 44, 9, 77, 59, 16, 1, 70, 77, 39, 92, 71, 67, 78, 51, 53, 12, 19, 63, 40, 90, 3]$$

[[[解答例]]]

1.
```
> area:=proc(base,height)
    base*height/2;
  end proc:

> area(3,4); #res: 6
```

2.
```
> restart;
  n:=19:
  banpei:=0;
  for i from 2 to n-1 do
    amari:=irem(n,i);
    print(amari):
    if amari=0 then
      banpei:=1;
      break;
    end if;
  end do:
  if banpei=1 then
    print(n," is not prime number.");
  else
    print(n," is prime number.");
  end if;
```

$$0$$

$$1$$

$$1$$

$$1$$

$$19, \text{" is prime number."}$$

```
> MyIsprime:=proc(n)
    local i,amari;
    for i from 2 to evalf(sqrt(n)) do
      amari:=irem(n,i);
      if amari=0 then
        return false;
      end if;
    end do:
    return true;
  end proc:

> MyIsprime(104729);
```

true

3.
```
> restart; x1:=[0.0, 0.0]: x2:=[1.0, 1.0]:

> MyDistance:=proc(x1,x2)
    local dx,dy;
    dx:=(x1[1]-x2[1]);
    dy:=(x1[2]-x2[2]);
    sqrt(dx^2+dy^2);
  end proc:

> MyDistance(x1,x2);
```

1.414213562

```
> x[1]:=[0.0, 0.0]: x[2]:=[1.0, 1.0]: x[3]:=[1.0, 0.0]: x[4]:=[0.0, 1.0]: x[5]:=x[1]:
  sum(MyDistance(x[i],x[i+1]),i=1..4);
```

4.828427124

4.
```
> MyMax:=proc(A)
    local imax,i;
    imax:=0;
    for i from 1 to nops(A) do
      if A[i]>imax then
          imax:=A[i];
      end if
    end do;
    return imax;
  end proc:

> MyMax(A);
```

100