Mapleでのプログラミングの基礎

西谷@関学・理工・情報科学

平成 17 年 8 月 23 日

1 Mapleプログラミングの特徴

Maple は，数式処理，視覚化，プログラミング言語など非常に広い領域をカバーしている．ここではプログラミング言語の一種として Maple スクリプトの文法を解説する．Maple スクリプトの特徴は

手続き型言語 C 言語と非常に似ており，習得すれば多くのプログラミング言語に応用できる．

インタープリター コンパイル作業がいらず，結果を見ながらプログラムを修正することが容易．

である．

いかなるプログラミングでも，核となるコードの構成要素は以下の 5 項目である．最低限これだけを知っていれば，ほとんどの問題を解くことが可能となる．

表 1: コードの構成要素.

変数	箱．値を格納する．
if 文	交通整理
for-loop	何度もぐるぐる
配列	たくさんの同じ箱．指数 (index) を受け取って，値を返す
関数	複雑な手続きは proc で

本文書は，Maple スクリプトのうちで，この基本項目だけを C 言語と対比させながら習得することを目的とする．

2 printf

プログラミングとは直接関係ないが，出力を整えるのに便利な printf 文を最初に示しておく．C 言語では以下のようなコードをエディターで打ち込んで，コンパイル，実行する必要がある．

```
#include <stdio.h>

int main(void){
    printf("Hello world!!\n");
}
```

修正するのにもこの一連の作業を繰り返さねばならない．これに対して Maple では

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

と打ち込んで enter を押せば，出力が即座に表示される．

- 一般的な出力は，

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

  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．

- 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．

3　変数：箱．値を格納

変数 a,b にそれぞれ 10,3 を代入し，a+b の結果を x に代入するというプログラムは以下の通り．Maple では変数の型を指定する必要がない．

C 言語

```
#include <stdio.h>

int main(void){
    double a,b,x;
    a=10;
    b=3;
    x=a+b;
    printf("%10.5f\n",x);
}
```

Maple スクリプト

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

課題 上記にしたがって，a と b の他の四則演算 (-,*,/) をおこなえ．

課題 evalf(a/b); によって浮動小数点の結果を求めよ

課題 iquo(a,b); irem(a,b); をおこない，商，余の意味を確認せよ．

課題 a^b; はどうなるか？

課題 b!; を求めよ．

課題 a:=73;b:=7; とすると上記のそれぞれの結果はどうなるか．

課題 以下の数式を Maple が正確に認識できるように形式を整えて打ち込め

$$\frac{a}{2B}, \sin(2\pi a), \mathrm{e}^{-\frac{E}{kT}}, \frac{a}{2b+\frac{c}{d}}, \log_2 x, \sin^2 x + \cos^2 x, \sqrt{1+x}, \sqrt[3]{x} \tag{1}$$

4　if 文：交通整理

もっとも簡単な if 文の例

Maple スクリプト

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

Maple 文法

```
if 条件文 then
  動作 1;
end if;
```

C 言語

```
x=-4;
y=0;
if (x<0){
 y=-x;
}
```

課題 `x:=-6;` の場合の結果を示せ．`x:=3;` ではどうか？

例外付きの if 文の例

Maple スクリプト

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

Maple 文法

```
if 条件文 then
  動作 1;
else
  動作 2;
end if;
```

C 言語

```
x=-4;
if (x<0) {
  y=-x;
} else {
  y=x;
}
```

2 個条件がある例

Maple スクリプト

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

Maple 文法

```
if 条件文 1 then
  動作 1;
elif 条件文 2 then
  動作 2;
else
  動作 2;
end if;
```

C 言語

```
if (x<0) {
  y=-x;
} else if (x<5) {
  y=x;
} else {
  y=2*x;
}
```

課題 x が-3，3，6 の場合，y はどうなるか？

条件文に使える式と意味

関係演算子		C 言語
(x=y)	x と y の値が一致	x==y
(x<>y)	x と y の値が一致しない場合	x!=y
(x>y), (x>=y), (x<y), (x<=y)		同じ
論理演算子	条件文を複数つなぐ他	
((x>0) and (x<4))		(x>0) && (x<4)
((x<0) or (x>4))		(x<0) || (x>4)
not(x=0)		!(x==0)

5 for-loop：何度もぐるぐる

簡単な for-loop

Maple スクリプト

```
> for i from 1 to 10 do
 i;
end do;
```

C 言語

```
for (i=1;i<=10;i++){
  printf("%3d\n",i)
}
```

課題 n に 10 を代入して，1 から n までの和を求めるプログラムを作れ．和は `total:=0;` として for-loop の前に初期化しておく．

課題 n の階乗 $n! = 1 \times 2 \times 3 \times ... \times (n-1) \times n$ を求めるプログラムを作れ．`n:=4;` として結果を吟味せよ．total の初期値は 1 から始めるように．

課題 n を入れて，次の級数を計算するプログラムを作れ．

$$1 + 1/1! + 1/2! + 1/3! + 1/4! + ... + 1/n! \tag{2}$$

- 第 i 項までの部分和 (sum) を S とし，それに 1/i!を加えて第 i+1 項までの部分和にするという手続きを繰り返す．それに加えていく `1/i!` も同様に階乗 (factorial) を F とし前項までを利用して求められる．
- この級数は `n->infinity` で自然対数の底 e に収束する．n を変えて `evalf(S);evalf(exp(1));` で比較せよ．

初期値や増減を調整した for-loop

Maple スクリプト

```
> for i from 10 by -1 to 1 do
 i;
end do;
```

C 言語

```
for (i=10;i<=1;i--){
  printf("%3d\n",i)
}
```

課題 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．割り算の余りは irem で求めることができる．例えば irem(9,2); としてその動作を確かめよ．

ヒント 番兵 (warden) を置いておき (warden:=0;)，2 から n-1 までの数で n をつぎつぎと割っていき，一度でも割り切れれば番兵にマークをつける (warden:=1;)．ループが終わった後で番兵のマークを見て，素数かどうかを表示する．

```
if␣(warden=0)␣then
```

```
␣␣printf("%d␣is␣a␣prime␣number.\n",n);
else
␣␣printf("%d␣is␣not␣a␣prime␣number.\n",n);
end␣if;
```

二重ループ

Maple スクリプト

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

C 言語

```
for ( i=1 ; i<=3; i++){
  for ( j=1 ; j<=3; j++){
    printf("%3d %3d\n",i,j);
  }
}
```

課題 j-loop を i から始まるようにせよ．結果はどうなるか？

課題 `printf("%3d",???);` と `printf("\n");` を使って下のようなかけ算の九九表を作れ．

```
 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
```

6　配列：たくさんの同じ箱 [ ]

配列とは指数 (index) を受け取って，値を返す．Maple で配列を使うには，などとすればよい．

Maple スクリプト

```
> A:=Array(1..5);
```

C 言語

```
double A[5]={0};
```

これは指数 (index)=1..5 で中身に数値 0 を入れた配列 A を用意しなさいという意味．A[1] として一番目の要素にアクセスできる．

課題 `A:=Array(1..4,[45,78,83,65]);` として初期化した配列を用意する．この平均を求めよ．

課題 前問で用意した配列を `B:=Array(1..4)` にコピー (代入) するプログラムを for-loop をもちいて作れ．

課題 前問で用意した配列を `B:=Array(1..4)` に逆順，つまり `B[4]<-A[1],B[3]<-A[2],..B[1]<-A[4]` とコピーするプログラムを for-loop を用いて作れ．

課題 `A:=Array(1..9,1..9);` として 9x9 の配列を用意し，ここにかけ算の九九の結果を入れるプログラムを作れ．i 行 j 列の要素へのアクセスは `A[i,j]` とする．

7 関数 ( )

複雑な手続きや，何度も繰り返す操作は proc で作る．何を受け取って (引数)，何をして (動作)，何を返す (戻り値) のかを考える．

引数

ユーザ関数は によって定義する．ユーザ定義関数の呼び出しは

Maple 文法

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

C 言語

```
void ユーザ関数名 ( 仮引数 ){
  動作
}
```

```
ユーザ関数名 ( 実引数 );
```

で行う．引数としてはどんな型 (変数や配列) でも，複数でも指定することが可能．C 言語では変数の渡し方に気を使う必要があるが，Maple ではその必要はない．引数の値を関数のなかで変えると怒られる．global で取り込むか，local 変数にコピーして使う．

戻り値

Maple の proc の戻り値は return で指定される．return 文がないときは，最後の動作の結果が戻

Maple 文法

```
関数名 := proc(引数)
  動作
return 戻り値
end proc;
```

C 言語

```
double ユーザ関数名 ( 仮引数 ){
  動作;
  return 変数;
}
```

り値となる．

グローバル (大域) 変数とローカル (局所) 変数

C 言語と同じように変数のスコープがある．

```
関数名 := proc(引数)
local 変数名;
global 変数名;
```

動作
return 戻り値
end proc;

課題 次の Maple スクリプトを指示に従って書き換えよ．

```
> a:=35;b:=29;
c:=a+b;
printf("%3d + %3d=\n",a,b);
printf("Answer=%3d\n",c);
```

1. a1,b1 を引数として，

   ```
   printf("%3d + %3d \n",a1,b1);
   ```

   で表示する関数 sum1 を作成せよ．sum1(a,b); で呼び出せ．
2. a,b を引数として，その和を返す関数 sum2 を作成せよ．戻り値を受け取って

   ```
   printf("Answer=%3d\n",c);
   ```

   として表示せよ．
3. b だけを引数として，グローバル変数 a との和を局所変数 cc にいれて，その二乗を返す関数 sum3 を作成せよ．

課題 前回に作った素数判定プログラムを proc 化せよ．引数は n とする．

8　ハノイの塔 (再帰版)

課題

3 本の杭 (peg) があり, 最初の杭には, おおきな円盤 (ring) から小さな円盤が下から順に積まれて塔のようになっている. 適宜 3 本の杭に円盤を差しかえて, 円盤を最初の杭から最後の杭へすべて移し換えなさい. ただし, 一度に一つの円盤しか動かせない. また, 小さい円盤の上に大きな円盤を置いてはいけない.

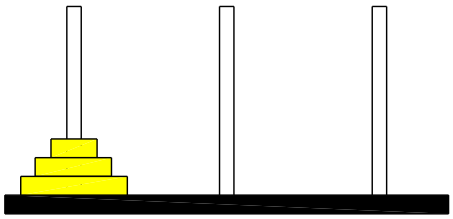

図 1: ハノイの塔の模式図.

ハノイ (バラモン) の塔伝説

> ベナレスにあるバラモン教の大寺院は世界の中心と記されており, そこに, 杭が 3 本立った真鍮製の基盤がある. 神はその一本に 64 枚の純金の円盤をはめた. 僧侶たちは昼夜の別なくそれを別の杭に移し替える作業に専念している. そして移し替えが完了したとき, 寺院もバラモン僧たちも崩壊し, この世が塵に帰るという.

『ハノイの塔』は 1883 年にフランスの数学者 E. リュカ (Edouard Lucas) が考えたゲームです. リュカがこの物語を創作したのか, はたまたこの物語から発案したのかは不明.

課題:MakeHanoi:=proc(n)

円盤の枚数 n を受け取って，3 本の杭 (peg)A,B,C を表わす配列を初期化する関数を作れ.

- global で A,B,C をとり，`A:=Array(0..n);` などとして杭を定義.
- 各杭 A,B,C の第 0 要素には杭に刺さっている円盤の枚数を入れる.
- A,B,C のその他の各要素には，それぞれの高さ (index で指定) に刺さっている円盤のサイズをいれる.
- 初期値は，`A[1]:=n;A[2]:=n-1;...A[A[0]]:=1;` となる.

課題:PrintHanoi:=proc()

3 本の杭 A,B,C に刺さっている円盤のサイズと位置を

```
A:  2  1
B:
C:
-----
```

と表示する関数を作れ．

- A,B,C は global で指定せよ．
- できたら `PrintHanoi()` を `MakeHanoi` 関数の最後に加えておけ．

課題:MovePlate:=proc(Pfrom,Pto)

杭 Pfrom の一番上に載っている円盤一枚を杭 Pto へ動かす操作を記述せよ．

- 動作，表示を

```
> MakeHanoi(2);
MovePlate(A,B);
PrintHanoi();
```

  として確認せよ．

- A の杭の一番上の plate を B の一番上にのせ，A[0],B[0] を調節する．これを `MovePlate` の中で仮引数 Pfrom,Pto を使って記述する．
- `PrintHanoi()` を `MovePlate` 関数の最後に加えておけ．

課題:Hanoi2 関数

n=2 の場合の円盤の動きを制御する `Hanoi2:=proc(Pfrom,Pto,Pwork)` を作れ．ここで，Pwork は作業用の杭で，使い方は以下の表示を参照せよ．

- 杭 A にある円盤を，杭 B を作業用の杭として使って，杭 C に移す場合

```
> MakeHanoi(2);
Hanoi2(A,C,B);
```

  とする．結果は以下のようになる．

```
A:  2  1
B:
C:
-----
```

```
A:  2
B:  1
C:
-----
A:
B:  1
C:  2
-----
A:
B:
C:  2  1
-----
```

課題:Hanoi3 関数

n=3 の場合の動きを予測し，`Hanoi3:=proc(Pfrom,Pto,Pwork)` を作れ．

- Hanoi2 を内部で使え．
- 3 枚の円盤のうち上部のサイズ 1,2 の 2 枚を Hanoi2 を使って杭 A から杭 B へ移す．次にサイズ 3 の円盤を杭 C に移す．最後に，杭 B にある円盤を Hanoi2 を使って杭 C に移せばよい．

課題:Hanoi:=proc(i,Pfrom,Pto,Pwork)

先程の Hanoi3 を参照して，i 枚の場合の Hanoi 関数を作れ．

- 一般化した場合は，i 枚の円盤うち，i-1 枚の円盤を杭 Pfrom から杭 Pto を使って杭 Pwork に移す．次にサイズ i の円盤を杭 Pto に移す．次に杭 Pwork にある i-1 枚の円盤を杭 Pfrom を使って杭 Pto に移す．
- i を引数に加えて，再帰的に Hanoi 関数を使うことによってハノイの塔の移動が自動的に達成される．
- 最後をどうするかはいい加減でいい．例えば，
  - `i=2` まできた時には `Hanoi2(Pfrom, Pto, Pwork)` を使う．
  - あるいは `i>0` ではこの操作を続け，`i=0` では何もしない．
- 次のようにすれば動作するはず．やってみ．

```
> n:=4;
Makeanoi(n);
Hanoi(n,A,C,B);
```

課題:世界の終わりは？

一枚の金貨を別の杭に移すのに 1 秒かかるとすると，バラモン僧たちが 64 枚の金貨を移すのに何年かかるか？

- 移動の回数を記録する global 変数とその操作を MovePlate に加えて，`n:=6;` ぐらいまでとって，金貨の枚数 n と移動回数との関係を予測せよ．
- ちなみに宇宙年齢は 137 億歳 (`1.37*10^(11)`) と計測されています．後何年 ??

9 N-王妃 (or 8-Queen) 問題

NxN マスの盤に N 個の Queen をお互いが縦，横，斜めにあたらないように置く，すべての可能な配置を表示する問題を考える．N=8 の 8-王妃問題が有名．

初期化

Queen の位置を保持する配列を作る．これは

```
> Queen:=Array(1..N);
```

ととる．ここで Queen[1]:=2 とすると，1 行目の Queen が 2 にあることを意味する．

課題:printQ:=proc(Queen)

Queen の配置を表示する関数を考えよ．例えば，

```
> N:=5:
Queen:=Array(1..N):
for i from 1 to N do Queen[i]:=i; end do:
printQ(Queen);
 Q . . . .
 . Q . . .
 . . Q . .
 . . . Q .
 . . . . Q
```

と Queen の配置を表示してくれる．

課題:"あたり"を確認する check 関数

"あたり"を確認する check 関数を作れ．

- この関数の動作は複雑であるが，4，5 行で書ける．i 行目の Queen を j 列目に入れたとして，i,j を引数として受け取る．
- p 行に入れた Queen が q 列目にある (Queen[p]=q) として，縦，前斜め，後ろ斜めにあたってないか調べる．
- これを i-1 行まですべてについて調べる．
- 一つでもあたりがあれば false を返し，あたってない時には true を返すようにする．

課題:再帰関数 try1:=proc(i)

N-王妃の配置を，再帰関数を使って求めよ．

- ある i 行での場所が

```
> Queen[i]:=k
```

  となると仮に決める．

- これ以前の行に置いた Queen と縦・横・斜めにあたってないかを check 関数で確認し，あたってなければ次の”i+1 行”へ行く．
- あたれば次の列に置いてみて check する．
- これを繰り返し，i=N+1 となれば成功．ここまで来た Queen 配置を出力すればよい．
- 再帰関数は，

```
> try1:=proc(i)
global Queen,N;
...
end proc;
```

  として i 行目の配置を作っていく．”...”の部分を考える．