プログラミングの基礎

Mapleプログラミングの特徴

- Mapleは，C言語なんかと非常に似ているので，習得すればほとんどのプログラミング言語に応用できる．
- Mapleは，コンパイル作業がいらないので，結果を見ながらプログラムを修正することが容易．
- いかなるプログラミングでも，核となる要素は以下の項目だけと心得よ．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．

変数と配列

- （１）変数：箱．値を格納する．
- （４）配列：たくさんの同じ箱．指数(index)を受け取って，値を返す

実行の制御

- （２）if文：交通整理
- （３）for-loop：何度もぐるぐる

関数

- （５）複雑な手続きはprocで
- 何を受け取って(引数)，何をして(動作)，何を返す(戻り値)のか？

printf

- プログラム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．
- 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．

初日(変数，if，for-loop)

変数：箱．値を格納

以下の例では，変数a,bにそれぞれ10,3を代入し，a+bの結果をxに代入している．

- 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

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

if文：交通整理

もっとも簡単なif文の例

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

if文の文法

```
if 条件文 then
   動作1;
end if;
```

- 【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

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

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

2個条件がある例

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

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

- 【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

- (x=y) xとyの値が一致
  (x<>y) xとyの値が一致しない場合
- (x>y)，(x>=y)，(x<y)，(x<=y)

▼ 条件文を複数つなぐand，orやnotも使える．

- ((x>0) and (x<4))
- ((x<0) or (x>4))
- not(x=0)

▼ for-loop：何度もぐるぐる

▼ 【課題】以下のfor-loopを回し，結果を表示せよ．

- ```
  for i from 1 to 10 do
   i;
  end do;
  ```
- ```
  for i from 10 by -1 to 1 do
   i;
  end do;
  ```
- ```
  for i from 1 by 2 to 10 do
    i;
  end do;
  ```

- 【課題】nを入れて，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 x 2 x 3 x ... x (n-1) x 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!

- 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．

- ```
  evalf(S);
  evalf(exp(1));
  ```
  で比較できる．

▼ 【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

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

第2日(配列[]，関数())

配列：たくさんの同じ箱[ ]

- 指数(index)を受け取って，値を返す．
- Mapleで配列を使うには，A:=Array(1..5);などとすればよい．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]など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[1]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列へのアクセスはA[i,j]とする．

関数( )

- 複雑な手続きや，何度も繰り返す操作はprocで記述する．
- 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．
- ```
  a:=35;b:=29;
  c:=a+b;
  printf("%3d + %3d=¥n",a,b);
  printf("Answer=%3d¥n",c);
  ```

引数

- 【課題】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．
- Mapleでユーザ関数は
  ```
  ユーザ関数名 := proc( 仮引数 )
    動作
  end proc;
  ```
  によって定義されます．ユーザ定義関数の呼び出しは
  ```
  ユーザ関数名( 実引数 );
  ```
  で行われます．実引数と仮引数の違いは上の例題を参照．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．

戻り値

- 【課題】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．

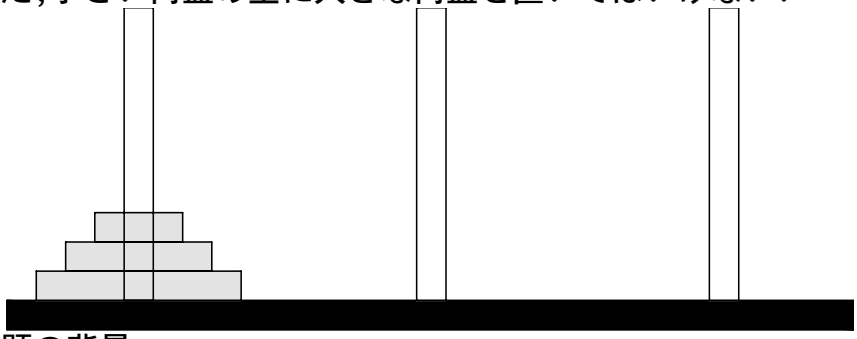
Mapleのprocの戻り値はreturnで指定される．

- 関数名 := proc(引数)
  ```
    動作
  return 戻り値
  end proc;
  ```
  return文がないときは，最後の動作の結果が戻り値とな
  ```

る.

- グローバル(大域)変数とローカル(局所)変数
  - 【課題】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.
    - 関数名 := proc(引数)
      local 変数名;
      global 変数名;
        動作
      return 戻り値
      end proc;
- 変数の操作
  - 【課題】sum3のなかでa,bを一ずつ増やし，その結果を関数の外側で表示せよ．Mapleのproc内でおこなう操作は変数に直接作用する．そうしたくないときにはlocal変数にコピーして使う.
- 【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.

第3日ハノイの塔(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.
  -
  
- 課題の背景
  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.
- make_hanoi:=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にA:=Array(0..n);などとして杭A,B,C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.
    - 初期値は，A[1]:=n;...A[A[0]]:=1;とする.
- print_hanoi:=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表示する関数を作れ. A,B,Cはglobalで指定せよ.
    - A: 2 1
      B:
      C:

    - と表示する.
    - print_hanoi()をmake_hanoi関数の最後に加えておけ.
- move_plate:=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.
    - make_hanoi(2);
      move_plate(A,B);

```
print_hanoi();
```
として動作，表示を確認せよ．

- print_hanoi()をmove_plate関数の最後に加えておけ．

▼ 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(peg)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
```
make_hanoi(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
```

▼ hanoi3関数の作成

- n=3の場合の動きを予測せよ．

▼ n=3の場合に動くhanoi3:=proc(Pfrom,Pto,Pwork)を，hanoi2を内部で使って作れ．

- ヒント：3枚の円盤のうち上部の2枚(i=1,2)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．

▼ hanoi:=proc(i,Pfrom,Pto,Pwork)

▼ 先程のhanoi3を参照して，i枚の場合のhanoi関数を作る．一般化した場合の記述は以下の通り．

- 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完了する．

▼ 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．

-
```
n:=4;
make_hanoi(n);
hanoi(n,A,C,B);
```
とすれば動作するはず．やってみ．

- 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？
  - ヒント:移動の回数を記録するglobal変数とその操作をmove_plate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？？

- N-王妃(or 8-Queen)問題(暇なら以下の問題を考えよ)
  - 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:=Array(1..N);
      ととる．ここでQueen[1]:=2とすると，1行目のQueenが2にあるとする．
  - 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．
  - "あたり"を確認するcheck関数
    - "あたり"を確認するcheck関数を作る．
    - ヒント: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を返すようにすればよい．
  - 再帰関数try1:=proc(i)
    - N-王妃の配置を考えるには，再帰関数を使うと便利である．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関数で確認して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．