Programming(I) 変数への代入と出力

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;
```

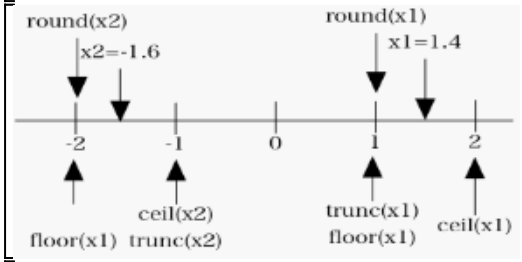
$$a := 10$$
$$b := 3$$
$$c := 13 \qquad \textbf{(1.1.1.1)}$$

整数と浮動小数点数（省略）

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);
```

$$0.7 \qquad \textbf{(1.1.2.1)}$$

整数の割り算は irem(余り) と iquo(商).

```
> irem(7,3);
  iquo(7,3);
```

$$1$$
$$2 \qquad \textbf{(1.1.2.2)}$$

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);
```

$$1 \qquad \textbf{(1.1.3.1)}$$

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．

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, %X	整数を16進数で表示．xは小文字，Xは大文字を使用.
%f	浮動小数点数として表示.
%e, %E	指数形式で表示．eは小文字，Eは大文字を使用.
%s	文字列を出力.

Programming(II) for-loop

解説

do-loop

もっとも単純なfor-loop.

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

```
                                1
                                2
                                3                    (2.1.1.1)
```

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                    (2.1.1.2)
```

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

```
> 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                  (2.1.1.3)
```

while-loop も同じように使える．

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

```
                              i := 0
                              i := 1
                              i := 2
                              i := 3
                              i := 4
                              i := 5                 (2.1.1.4)
```

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×(n-1)×(n-2)×…×3×2×1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);
                              i := 1
1+1=2
```

2

i)

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

1
2
3
4
5 **(2.3.2.1)**

ii)

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

5
4
3
2
1 **(2.3.2.2)**

iii)

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

2
4
6
8
10 **(2.3.2.3)**

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

4

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

sum1 := 0
sum1 := 1
sum1 := 3
sum1 := 6
sum1 := 10
sum1 := 15 **(2.3.4.1)**

5.

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

n := 5
total1 := 1
total1 := 1
total1 := 2
total1 := 6
total1 := 24
total1 := 120 **(2.3.5.1)**

Programming(III) 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];
```

$$list1 := [1, 3, 5, 7] \qquad (3.1.1.1)$$

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];
```

$$3$$

$$7$$

$$[3, 5, 7] \qquad (3.1.1.2)$$

-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[-1];list1[0];
```

$$7$$

```
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;
```

$$[1, 3, x, 7] \qquad (3.1.1.3)$$

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);
```

$$4$$

$$1, 3, x, 7 \qquad (3.1.1.4)$$

for-loopとそれ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);
```

$$aa := [\,]$$

$$[1, 2, 3] \qquad (3.1.2.1.1)$$

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)];
```

$$aa := [1, 2, 3] \qquad (3.1.2.1.2)$$

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):
```

$$6 \qquad (3.1.2.2.1)$$

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を使っているとおくこのようなエラーがでる．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;
```

$$4 \qquad (3.1.2.2.2)$$

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

$$6 \qquad (3.1.2.2.3)$$

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];
```

$$list1 := [1, 3, x, 7, 9] \qquad (3.1.3.1)$$

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;
```

$$[1, 3, 7, x, 9] \qquad (3.1.4.1)$$

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]];
```

$$l2 := [[1, 2, 3, 4], [1, 3, 5, 7]] \qquad (3.1.5.1)$$

要素へのアクセスは以下の通り．

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

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

$$5$$

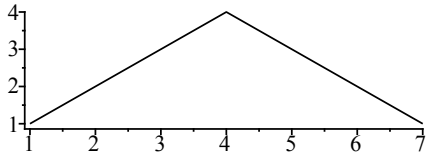
$$5 \qquad (3.1.5.2)$$

listの表示(lisplot)

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);
```

$$la := [1, 2, 3, 4, 3, 2, 1]$$



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 **(3.2.1.1)**

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);
> iquo(7,3);
```

1

2 **(3.2.6.1)**

解答例

1.

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

[27, 96, 17, 90, 34] **(3.3.1.1)**

2.

```
> roll:=rand(1..6):
> A:=[seq(0,i=1..6)];
  for i from 1 to 100 do
    i1:=roll();
    A[i1]:=A[i1]+1;
  end do:
```

$A := [0, 0, 0, 0, 0, 0]$ **(3.3.2.1)**

```
> A;
```

[16, 18, 21, 18, 18, 9] **(3.3.2.2)**

3.

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

3 **(3.3.3.1)**

4.

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

$A := [5, 7, 3, 7, 6]$ **(3.3.4.1)**

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

$sum1 := 0$

57376 **(3.3.4.2)**

5.

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

$n := 8$

$B := 0.19550684$

$A := [\,]$ **(3.3.5.1)**

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

$B := 1.95506840$

[1, 9, 5, 5, 0, 6, 8, 4] **(3.3.5.2)**

6.

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

$B := 246$ **(3.3.6.1)**

```
> 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] **(3.3.6.2)**

Programming(IV) if

解説

if：交通整理

もっとも簡単なif文の例.

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

$x := -4$

$y := 4$ **(4.1.1.1)**

例外付き.

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

$x := 3$

$y := 3$ **(4.1.1.2)**

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;
```

$x := 3$

$y := 6$ **(4.1.1.3)**

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, implies, notがある．その他にもブール値を返す関数として evalb, type などいくつかあり，条件分岐に使える．

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:
```

1
2
4
5 **(4.1.3.1)**

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

1
2 **(4.1.3.2)**

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
> amari:=irem(9,2);
として変数residue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;
```

year := 1890

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;
```

n := 10

2

3

5

7 **(4.3.2.1)**

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 **(4.3.3.1)**

4.

```
> n:=12:
  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
    printf("%d is not prime number.\n",n);
  else
    printf("%d is prime number.\n",n);
  end if;
```

banpei := 0

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.

別解

```
> 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;
```

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.

6.

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

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] **(4.3.6.1)**

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

25 **(4.3.6.2)**

```
> 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

100, 3, 97 **(4.3.6.3)**

Programming(V) proc

解説

複雑な手続きや，何度も繰り返すルーチンはprocで作る．

基本

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;
```

test1 := **proc**(*a*) *print*(*a*) **end proc** (5.1.1.1)

procの呼び出しは，以下のようになる．

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

13 (5.1.1.2)

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の中で変更すると怒られる．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)
    print(a);
    return a+1;
  end proc;
```

test2 := **proc**(*a*) *print*(*a*); **return** *a* + 1 **end proc** (5.1.2.1)

```
> test2(13);
```

13
14 (5.1.2.2)

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．信用できないけど．宣言の仕方は以下の通り．

変数名:=proc(引数)
 local 変数;
 global 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. my_isprime

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my_distance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my_max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, (5.2.4.1)
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 := **proc**(*base*, *height*) 1 / 2 * *base* * *height* **end proc** (5.3.1.1)

```
> area(3,4);
```

6 (5.3.1.2)

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;
```

banpei := 0
1
1
1
19, " is prime number." (5.3.2.1)

```
> my_isprime:=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:
```

```
> my_isprime(104729);
```

$$true \qquad (5.3.2.2)$$

3

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

$$x1 := [0., 0.]$$

$$x2 := [1.0, 1.0] \qquad (5.3.3.1)$$

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

```
> my_distance(x1,x2);
```

$$1.414213562 \qquad (5.3.3.2)$$

```
> 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(my_distance(x[i],x[i+1]),i=1..4);
```

$$x_1 := [0., 0.]$$

$$x_2 := [1.0, 1.0]$$

$$x_3 := [1.0, 0.]$$

$$x_4 := [0., 1.0]$$

$$x_5 := [0., 0.]$$

$$4.828427124 \qquad (5.3.3.3)$$

4.

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

$$A := [93, 45, 96, 6, 98, 59, 44, 100, 38, 69, 27, 96, 17, 90, 34, 18, 52, 56, 43, 83, 25, 90, 93, 60, 93, 14, 50, 47, 8, 46, 44, 9, 77, 59, 16, 1, 70, 77, 39, 92, 71, 67, 78, 51, 53, 12, 19, 63, 40, 90] \qquad (5.3.4.1)$$

```
> my_max:=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;
```

$$my\_max := \mathbf{proc}(A) \qquad (5.3.4.2)$$
$$\quad \mathbf{local}\ imax, i;$$
$$\quad imax := 0;$$
$$\quad \mathbf{for}\ i\ \mathbf{to}\ nops(A)\ \mathbf{do}\ \mathbf{if}\ imax < A[i]\ \mathbf{then}\ imax := A[i]\ \mathbf{end\ if}\ \mathbf{end\ do};$$
$$\quad \mathbf{return}\ imax$$
$$\mathbf{end\ proc}$$

```
> my_max(A);
```

$$100 \qquad (5.3.4.3)$$