

Maple

TP n°3 : Les procédures

Maple intègre une méthode de création de fonctions "simples" via l'opérateur `->`.

Par exemple :

```
f:=x->sin(x)+x+1;
f(0);
f(a);
g:=(x,y)->x*sin(y)+y*cos(x);
g(1,a);
```

Néanmoins, lorsque l'on veut réaliser des procédures plus complexes, on utilise des "sous-programmes" appelés procédures. Une procédure, de type `procedure`, est définie par le mot-clé `proc` et peut être assignée à un nom de variable. Pour définir une procédure intitulée `nom_de_proc`, on utilisera la syntaxe suivante :

```
nom_de_proc:=proc(paramètres_ou_arguments)
  global variables_globales;           (ligne optionnelle)
  local variables_locales;            (ligne optionnelle)
  description chaîne_de_description;  (ligne optionnelle)
  options nom_option;                 (ligne optionnelle)
  . . . instructions . . .            (corps de la procédure)
end;
```

`paramètres_ou_arguments` sont les variables que va utiliser la procédure. Elles sont séparées par une virgule. On peut spécifier le type d'une variable en la post-fixant avec `::` suivi du type. Dans le cas où la variable est typée, en cas d'appel à la procédure, le type des paramètres utilisés est vérifié et doit correspondre au type déclaré.

Les commandes `global` et `local` permettent de définir les variables intermédiaires qui seront nécessaires au fonctionnement de la procédure. Une variable locale ne sera visible qu'à l'intérieur d'une procédure et cessera d'exister en tant que tel dès la fin de celle-ci. Contrairement à une variable globale qui restera accessible en dehors de la procédure.

La commande `description` ne sert pas dans la procédure mais permet d'introduire un commentaire sur la nature de la procédure programmée. C'est très utile en cas d'utilisation ultérieure.

Les options sont de la forme `remember`, `system`, `operator`, `package` ou `copyright`.

De façon générale, le résultat renvoyé par une procédure est celui de la dernière évaluation faite à l'intérieur du corps de la procédure. Néanmoins nous allons voir les instructions `RETURN` et `ERROR` qui permettent respectivement de renvoyer un résultat avant la dernière instruction ou d'arrêter la procédure avec un message d'erreur.

Exemple 1

(Re-)Créons une fonction qui permet de déterminer le maximum de deux entiers.

```
maxi:=proc(u::posint,v::posint)
description "fonction qui determine le maximum de deux entiers
positifs";
if u<v then
    v
else
    u
fi;
end;

maxi(5,9);

maxi(1.2,Pi);
```

La procédure intitulée `maxi` calcule le plus grand de 2 entiers positifs. Elle comporte 2 paramètres `u` et `v`. Leur type peut être omis, mais ici il est précisé. Noter bien la syntaxe `u :: posint, v :: posint` pour signifier que `u` et `v` sont de type `posint`. Elle ne comporte ni variables globales, ni variables locales, mais une chaîne de description de la procédure.

Exemple 2

```
messages:=proc()
    global message1;
    local message2;
    message1:="Je suis une variable globale";
    message2:="Je suis une variable locale";
end;

messages();

message1,message2;
```

La valeur de la variable locale `message2` n'a pas été reconnue, contrairement à celle de la variable globale `message1`.

Remarque

Un paramètre formel passé à une procédure ne peut pas être modifié à l'intérieur de cette procédure

Exemple 3

On cherche à diviser un entier positif `x` par 2 tant qu'il est pair. La méthode qui suit ne fonctionne pas.

```
div:=proc(x::posint)
    while is(x,even) do
        x:=x/2
    od;
end;

div(48);
```

La tentative d'affecter au paramètre formel x sa valeur divisée par 2 provoque une erreur. Une méthode constitue à introduire une variable locale :

```
div:=proc(x::posint)
  local y;
  y:=x;
  while is(y,even) do
    y:=y/2
  od;
end;
```

```
div(48);
```

```
div(45);
```

```
div(1);
```

Remarque

La commande RETURN met fin à la procédure en donnant un résultat.

La commande ERROR permet de gérer les cas particuliers ou les erreurs d'entrée.

Exemple 4

Un codage de la fonction factoriel pourrait être le suivant :

```
factnew := proc(n)
  local i,p;
  if type(n,posint) then
    p:=1;
    for i from 1 to n do
      p:=p*i
    od;
    RETURN(p);
  else ERROR(`factnew nécessite un entier positif`)
  fi;
end;
```

```
factnew(5);
```

```
factnew(3.);
```

Exercice 1

Ecrire une procédure pour un entier p compris entre 1 et 9 affiche la table de multiplication de p .

Exercice 2

Soit p un entier non nul.

1. Donner une procédure qui permette de déterminer la valeur "exacte" du terme d'ordre p de la suite numérique u définie par $u_0 = \sqrt{5}$, $u_1 = \sqrt{3}$ et $u_{n+2} = \frac{2u_{n+1} + 2u_n + 1}{5}$.
2. Que faut-il changer pour avoir une valeur approchée?
3. Calculer u_{1000} avec 50 chiffres significatifs puis avec 100.

Exercice 3

Ecrire une fonction booléenne qui, étant donnés deux ensembles A et B , renvoie `true` si A est un sous-ensemble de B et `false` sinon.

Exercice 4

Le calcul d'un pgcd de deux entiers a et b peut se faire en utilisant la propriété suivante :

$$\begin{aligned}\text{pgcd}(a,b) &= a \text{ si } a = b \\ \text{pgcd}(a,b) &= \text{pgcd}(a - b, b) \text{ si } a > b \\ \text{pgcd}(a,b) &= \text{pgcd}(b - a, a) \text{ si } a < b\end{aligned}$$

Ecrire une procédure qui se sert de cette méthode pour calculer le pgcd de deux entiers donnés (on vérifiera que les paramètres fournis sont bien des entiers).

Exercice 5

Soit p un entier non nul.

1. Donner une procédure qui détermine la valeur "exacte" du terme d'ordre p de la suite numérique u définie par $u_0 = 2$ et $u_{n+1} = \frac{u_n}{3} + 4$.
2. Cette suite est-elle convergente?
Si oui, quelle est sa limite?
3. Déterminer une procédure qui étant donné un réel ϵ , permette d'obtenir le rang jusqu'auquel il faut aller pour avoir une valeur approchée à ϵ près de la limite.

Exercice 6

Un nombre entier positif est dit parfait s'il est la somme de ses diviseurs (différents de lui-même).

Par exemple $6 = 1 + 2 + 3$.

1. Ecrire une procédure qui permette de calculer la somme des diviseurs d'un entier.
2. Ecrire une procédure qui permette de vérifier si un entier est parfait.
3. a. Ecrire une procédure qui affiche tous les entiers parfaits inférieurs à un entier p donné.
b. Tester avec $p = 10000$.

Exercice 7

La multiplication égyptienne est une méthode algorithmique de calcul d'un produit. Elle s'effectue de la façon suivante :

On cherche à calculer $p = a \times b$.

Si b est impair, alors $p = a \times (b - 1) + a$.

Si b est pair, alors $p = a \times 2 \times (b / 2)$.

On s'arrête lorsque $b = 1$.

$$\begin{aligned}\text{Par exemple : } 25 \times 15 &= 25 \times 14 + 25 = 50 \times 7 + 25 = 50 \times 6 + 50 + 25 = 100 \times 3 + 50 + 25 \\ &= 100 \times 2 + 175 = 200 + 175 = 375.\end{aligned}$$

Ecrire une procédure qui ferait une multiplication égyptienne entre deux entiers a et b .

Exercice 8

Ecrire une procédure, qui étant donné un entier n supérieur à 3, représente un polygone régulier à n côtés, de centre $(0,0)$ et de rayon 1.

Exercice 9

On suppose donnés deux réels a et b , une fonction f continue sur $[a,b]$ et un réel e .

Donner une procédure qui, si $f(a)$ et $f(b)$ sont de signes contraires, permette de trouver par dichotomie une racine de f à e près.

Exercice 10

Soit P un polynôme de $\mathbb{R}[X]$. On suppose que $P = \sum_{k=0}^n a_k X^k$.

On considère la fonction polynomiale associée à P définie sur \mathbb{R} par $P(x) = \sum_{k=0}^n a_k x^k$.

1. Soit a un réel.
Combien l'ordinateur doit-il faire d'opérations élémentaires (addition-multiplication) pour calculer "bêtement" $P(a)$?
2. L'algorithme de Hörner est un moyen de diminuer ce nombre de calculs.
La méthode consiste à écrire $P(x)$ sous la forme :
$$P(x) = a_0 + x(a_1 + x(a_2 + x(\dots + x(a_{n-1} + x(a_n))\dots)))$$

Combien l'ordinateur doit-il faire d'opérations élémentaires pour calculer $P(a)$ sous la forme de Hörner?
3. Donner un algorithme qui transforme un polynôme donné sous forme "développé" en la forme de Hörner.

Exercice 11

Ecrire une procédure qui inverse l'ordre des éléments d'une liste .

Exercice 12

Ecrire une procédure qui, à partir d'une liste donnée L , renvoie la liste obtenue par permutation circulaire d'un élément vers la droite des éléments de L .

Exercice 13

1. Ecrire une procédure qui parcourt une liste de nombres avec les règles suivantes
 - arrêt et renvoi la valeur de l'entier positif s'il en existe un dans la liste.
 - arrêt et renvoi de la valeur -1 s'il existe un nombre négatif dans la liste.
 - renvoi de -1 si la liste a été entièrement parcourue.
2. Tester votre programme avec les listes suivantes :
 - a. [12.56,8.9,7,3.14]
 - b. [12.56,-8.9,7,3.14]
 - c. [12.56,8.9,7.1,3.14]

Exercice 14

1. Quelle est la constante d'un carré magique?
2. Etant donné un tableau T à n lignes et p colonnes,
 - a. écrire une procédure qui calcule la somme des éléments de la i ème ligne de T .
 - b. écrire une procédure qui calcule la somme des éléments de la j ème colonne de T .
 - c. écrire une procédure qui calcule la somme des éléments de la première diagonale de T .
 - d. écrire une procédure qui calcule la somme des éléments de la seconde diagonale de T .
3. Utiliser les procédures précédentes pour écrire une fonction booléenne qui, étant donné un tableau T à n lignes et n colonnes, renvoie `true` si celui-ci est un carré magique.