TRAVAUX PRATIQUES MAPLE NO. 2

MMI 1

Exercices d'arithmétique

Exercice 1 : Écrire une procédure nommée - `nombredededeux` - qui prend en argument un entier n et qui renvoie comme valeur le nombre de fois où l'entier 2 divise n. Puis vérifier les résultats obtenus en utilisant la commande `ifactor`. Ainsi, par ex., `nombredededeux`$(155) = 0$ et `ifactor`$(155) = (5)(31)$, `nombredededeux`$(4640) = 5$ et `ifactor`$(4640) = (2)^5\ (5)\ (29)$.

Exercice 2 : On souhaite étudier la densité des nombres premiers parmi l'ensemble des entiers naturels $\mathbb{N}$.

(1) Écrire une procédure `densitePremiers` qui prend comme argument un entier naturel n et qui renvoie la densité d des nombres premiers contenus dans l'ensemble $\{1, \ldots, n\}$, la densité d étant définie comme le nombre d'entiers premiers appartenant à l'ensemble $\{1, \ldots, n\}$, divisé par n (nombre total des entiers de $\{1, \ldots, n\}$). Par ex., `densitePremiers`$(6) = 0.5$ car il y a trois entiers premiers : $2, 3, 5$ dans l'ensemble $\{1, 2, 3, 4, 5, 6\}$ d'où $d = 3/6 = 0,5$;

(2) Calculer les densités d pour $n = 10,\ 100,\ 1\,000,\ 10\,000,\ 100\,000,\ 1\,000\,000$;

(3) Quelle conclusion en tirez-vous ?

(4) Écrire une procédure `densiteJumaux` qui prend comme argument un entier naturel n et qui renvoie la densité d' des nombres premiers jumaux dans l'ensemble $\{1, \ldots, n\}$. On appelle nombres premiers jumaux tout couple $(p, p+2)$ où p et $p+2$ sont premiers. Par ex., `densiteJumaux`$(6) = 0.1666666$ car 3 et 5 sont premiers jumaux (on a donc $d' = 1/6$) ;

(5) Calculer les densités d' pour $n = 10,\ 100,\ 1\,000,\ 10\,000,\ 100\,000,\ 1\,000\,000$.

Date: 19-01-2009 14:40.

Exercice 3 :

(1) Écrire une procédure `image` qui prend comme argument une fonction Maple f et deux entiers m et n. La procédure renvoie la liste des images par la fonction f de tous les entiers de m à n ;

(2) Appliquer la procédure à l'application polynômiale $\texttt{p} := \texttt{x} \mapsto \texttt{x}^2 + \texttt{x} + 41$ avec les entiers $m = -40$ et $n = 40$;

(3) Écrire une procédure `premiers` qui prend comme argument une liste L et qui renvoie la liste M des nombres premiers contenus dans L ;

(4) Appliquer la procédure à la liste obtenue précédemment. Que constate-t-on ?

Exercice 4 : Les nombres de Fermat sont les entiers $F_k := 2^{(2^k)} + 1$ pour $k \in \mathbb{N}$. On se demande lesquels sont premiers.

Écrire une procédure `fermat` qui ne prend pas d'argument et qui calcule dans l'ordre les nombres de Fermat, tant qu'ils sont premiers. Pour chaque valeur de `k`, la procédure affiche sur une ligne « `Fk` = » suivi de sa valeur calculée, grâce à la commande d'affichage `printf`. Puis, dès que le nombre F_{k_0} n'est pas premier, la procédure affiche le texte « Le $\texttt{k}_0$ème nombre de Fermat n'est pas premier. », suivi de la décomposition en facteurs premiers de F_{k_0}.

Exercice 5 :

(1) Écrire une procédure `somme` qui prend comme argument un entier naturel n et qui calcule la somme des chiffres composant n. Par ex., $\texttt{somme}(28) = 2 + 8 = 10$ car le nombre 28 est composé des chiffres « 2 » et « 8 » ;

(2) Tester la procédure sur les entiers $n = 0$, puis 1, 9, 123456789 et 100! ;

(3) Écrire une procédure `itereSomme` qui prend comme argument un entier naturel n et qui itère la procédure précédente, tant que le nombre obtenu comprend plusieurs chiffres. Par ex., pour $n = 28$, on calcule tout d'abord $\texttt{somme}(28) = 10$ puis on recommence (puisque 10 a deux chiffres), $\texttt{somme}(10) = 1 + 0 = 1$. On arrête étant donné que le résultat obtenu s'écrit sur un seul chiffre ;

(4) Tester cette procédure avec les mêmes entiers que la question 2.

Exercice 6 :

(1) Écrire une procédure `facteurs` qui prend comme argument un entier naturel non nul n et qui renvoie la liste de ses facteurs premiers : par ex., `facteurs`(50) va renvoyer la liste $[2, 5, 5]$ puisque $50 = 2 \times 5^2$. Pour coder cette procédure vous n'utiliserez pas la commande `ifactor` ;

(2) Tester la procédure sur les exemples suivants :
- `facteurs`(360) doit renvoyer $[2, 2, 2, 3, 3, 5]$;
- `facteurs`(81) doit renvoyer $[3, 3, 3, 3]$;
- `facteurs`(13) doit renvoyer $[13]$;
- `facteurs`(1) doit renvoyer $[]$;

Exerice 7 : On se propose de coder l'algorithme d'Euclide qui permet de déterminer le `PGCD` d des entiers a et b.
L'algorithme d'Euclide est défini par la suite des divisions euclidiennes :

$$\begin{array}{lcl} a & = & bq_1 + r_1 \\ b & = & r_1 q_2 + r_2 \\ r_1 & = & r_2 q_3 + r_3 \\ r_2 & = & r_3 q_4 + r_4 \\ & \dots & \\ r_n & = & r_{n+1} q_{n+2} + r_{n+2} \\ r_{n+1} & = & r_{n+2} q_{n+3} + 0 \ . \end{array}$$

Alors $d = \texttt{pgcd}(a, b) = r_{n+2}$.

(1) Écrire une procédure `pgcd` (sans utiliser la commande `igcd`) qui prend comme arguments deux entiers a et b et qui renvoie leur `PGCD` calculé selon l'algorithme d'Euclide ;

(2) Tester la procédure en calculant `pgcd`$(16, 15)$, `pgcd`$(48, 4)$, `pgcd`$(4, 48)$, `pgcd`$(123456789, 987654321)$ et

`pgcd`$(123456789987654321, 159753258456951753852654)$

et en vérifiant chaque résultat obtenu à l'aide de la commande `ifactor`.