Quelques applications de la partie entière d'un nombre réel

Introduction :
On a besoin de rappeler la définition de la partie entière d'un nombre réel et le théorème « division euclidienne dans $\mathbb{N}$ » ainsi que sa démonstration .
On ne mentionne pas, volontairement, les propriétés de $\mathbb{N}$ qui justifient la définition suivante :
Définition :
Pour $x \in IR$ l'unique entier n tel que $x \in [n ; n+1[$ s'appelle la partie entière de x
On note : n = E(x)
Pour les calculatrices TEXAS c'est n = int(x) si elles sont en anglais.

Théorème :
Pour $a \in \mathbb{N}$ et $b \in \mathbb{N}^*$ il existe un unique couple (q ; r) d'entiers naturels tel que :
$a = bq + r$, avec $r \in [0 ; b[$
Preuve
On obtient l'existence et l'unicité par condition nécessaire et suffisante :

$a = bq + r \Leftrightarrow \frac{a}{b} = q + \frac{r}{b} \Rightarrow q \leq \frac{a}{b} < q + 1$

On a donc : $q = E(\frac{a}{b})$ et $r = a - bq$.

Ce sont les deux résultats qui nous intéressent ; beaucoup de machines très diffusées ne sont pas munies des fonctions calculant q et r, c'est le cas des TI 82 et 83 et des CASIO qui ne sont pas à calcul formel, mais c'est a vérifier pour les calculatrices CASIO.

Ou bien on retient les deux formules ou bien on fait le programme suivant sur TI.82-83

: prompt a, b	(on saisit les valeurs de a et b)
: int($\frac{a}{b}$) → q	(q = E($\frac{a}{b}$))
: a-bq → r	(r = a-bq)
: Disp « r = » Disp r	(envoyer la valeur de r)
: Disp « q = » Disp q	(envoyer la valeur de q)

~~~~~~~~~~

Applications

1 Pourquoi la procédure usuelle 1 + int(n×rand) permet elle de simuler une expérience aléatoire ayant un nombre fini n d'éventualités x_i avec une loi de probabilité uniforme $p(x_i) = p_i = \frac{1}{n}$, i = 1 ; 2 ;… ; n.

Principe : Un générateur de nombres aléatoires de [0 ; 1[génère une suite $(u_n) = (u_0 , u_1 , u_2 , u_3 , ... , u_n , u_{n+1} , ...)$, où u_i = rand quel que soit i, qui vérifie la propriété suivante :
pour tout [a; b] ⊂ [0 ; 1[, la probabilité de l'événement « $u_n \in [a; b]$ » est égale à b-a.

La procédure 1 + int(n×rand) renvoie alors un entier k, $k \in \{1 ; 2 ;… ; n\}$, avec la probabilité $p_k = \frac{1}{n}$ car

u_n = rand $\in [0 ; 1[$ et, en partageant [0 ; 1[en n intervalles de même longueur $\frac{1}{n}$,

u_n = rand $\in [\frac{k}{n} ; \frac{k+1}{n}[$, avec $k \in \{0; 2;…; n-1\}$, si et seulement si $n \times u_n = n \times$ rand $\in [k; k+1[$.
~~~~~~~~~~

Ce qui prouve que la partie entière de $n \times rand$ est k et par conséquent $1 + int(n \times rand) = k+1$ sort avec la probabilité $\frac{k+1}{n} - \frac{k}{n} = \frac{1}{n}$ en vertu du principe énoncé plus haut appliqué à $u_n = rand \in [\frac{k}{n} ; \frac{k+1}{n}[$.

Exemple : 1 + int (6×rand) simule le lancé d'un dé bien équilibré dont les 6 faces sont numérotées 1 2 3 4 5 6.

On peut se poser la question : comment simuler l'expérience aléatoire quand **la loi de probabilité n'est pas uniforme** ?

Pour cela on peut traiter un exemple, pour la curiosité, car je ne vois pas là d'application de la partie entière d'un réel :

Un dé tétraédrique est truqué de sorte que : $p(1) = p(2) = \frac{1}{3}$, $p(3) = \frac{1}{4}$. et $p(4) = \frac{1}{12}$

Toujours selon le principe décrit au (a) un programme qui va traduire le processus suivant va simuler le jet du dé précédent :

Si $0 \leq rand < \frac{1}{3}$ envoyer 1

Si $\frac{1}{3} \leq rand < \frac{2}{3}$ envoyer 2

Si $\frac{2}{3} \leq rand < \frac{11}{12}$ envoyer 3

Si $\frac{11}{12} \leq rand < 1$ envoyer 4

Le programme suivant sur TI.83 traduit ce processus:

:Lbl 1	(Ordre numéro 1)
:rand → X	(X = rand)
:If $0 \leq X$ and $X < \frac{1}{3}$	(Si $X \in [0; \frac{1}{3}[$)
:Then	(alors)
:Disp 1	(envoyer 1)
:End	(fin d'ordre If)
:If $\frac{1}{3} \leq X$ and $X < \frac{2}{3}$	(Si $X \in [\frac{1}{3}; \frac{2}{3}[$)
:Then	(alors)
:Disp 2	(envoyer 2)
:End	(fin d'ordre If)
:If $\frac{2}{3} \leq X$ and $X < \frac{11}{12}$	(Si $X \in [\frac{2}{3}; \frac{11}{12}[$)
:Then	(alors)
:Disp 3	(envoyer 3)
:End	(fin d'ordre If)
:If $\frac{11}{12} \leq X$ and $X<1$	(Si $X \in [\frac{11}{12}; 1[$)
:Then	(alors)
:Disp 4	(envoyer 4)
:End	(fin d'ordre If)
:Pause	(On arrête, avant nouvel ordre, le programme)
:Goto 1	(On relance le programme en retournant à l'ordre numéro 1)

On peut améliorer ce programme en lui demandant non pas de renvoyer successivement, à la demande, un chiffre 1 ; 2 ; 3 ou 4 mais de renvoyer une liste de dimension n, n arbitraire, constituée de ces chiffres

pour pouvoir ensuite calculer la fréquence de chaque chiffre dans la liste et vérifier que ces fréquences approchent les probabilités théoriques : $p(1) = p(2) = \frac{1}{3}$, $p(3) = \frac{1}{4}$. et $p(4) = \frac{1}{12}$

Le programme suivant sur TI.83 fait ce travail :

:ClrList L1	(On vide la liste L1)
:Prompt N	(on saisit la valeur de N qui est la dimension de la liste)
:For (P , 1 , N)	(Pour P variant de 1 à N)
:rand $\rightarrow$ X	(X = rand)
:If $0 \leq X$ and $X < \frac{1}{3}$	(Si $X \in [0; \frac{1}{3}[$)
:Then	(alors)
:1 $\rightarrow$ L1 (P)	(1 devient le terme de rang p de la liste L1)
:End	(fin d'ordre If)
:If $\frac{1}{3} \leq X$ and $X < \frac{2}{3}$	(Si $X \in [\frac{1}{3}; \frac{2}{3}[$)
:Then	(alors)
: 2 $\rightarrow$ L1 (P)	(2 devient le terme de rang p de la liste L1)
:End	(fin d'ordre If)
:If $\frac{2}{3} \leq X$ and $X < \frac{11}{12}$	(Si $X \in [\frac{2}{3}; \frac{11}{12}[$)
:Then	(alors)
: 3 $\rightarrow$ L1 (P)	(3 devient le terme de rang p de la liste L1)
:End	(fin d'ordre If)
:If $\frac{11}{12} \leq X$ and $X < 1$	(Si $X \in [\frac{11}{12}; 1[$)
:Then	(alors)
: 4 $\rightarrow$ L1 (P)	(4 devient le terme de rang p de la liste L1)
:End	(fin d'ordre If)
:End	(fin d'ordre For)
:Disp L1	(envoyer la liste L1)

Exemple : pour N = 950 le programme précédent renvoie une liste L_1 de dimension 950 dont les premiers termes sont : L_1 = {2 ;3 ;2 ;2 ;3 ;3 ;1 ;1 ;1 ;1 ;4 ;1 ;2 ;3 ;……..} et la comparaison entre les probabilités théoriques et les fréquences observées est visualisée par le tableau suivant :

Face i	1	2	3	4
Probabilité théorique	1 / 3 = 0.333...	1 /3 = 0.333…	1 / 4 = 0.25	1 / 12 = 0.0833…
Fréquence observée	0.330…	0.325…	0.26	0.084…

2 Comment, à partir d'un nombre à n chiffres, obtenir une liste dont les termes sont les n chiffres de ce nombre ?

Cet exercice a un contexte qu'on décrira après l'avoir résolu.
Principe :
Considérons l'écriture en base 10 de l'entier naturel X non nul :
$X = 10^n a_n + 10^{n-1} a_{n-1} + \ldots + 10\, a_1 + a_0$, $a_i \in \{0 ;1 ;2\ldots\ldots ;9\}$, $i \in \{0;1;2\ldots\ldots;.n\}$ et $a_n \neq 0$.
$10^{-n} X = a_n + 10^{-1} a_{n-1} + \ldots\ldots + 10^{-n+1} a_1 + 10^{-n} a_0$
$0 \leq 10^{-1} a_{n-1} + \ldots\ldots + 10^{-n+1} a_1 + 10^{-n} a_0 \leq 9 . 10^{-1} + 9 . 10^{-2} + \ldots\ldots + 9 . 10^{-n+1} + 9 .10^{-n} =$
$9 . 10^{-1} . (1 + 10^{-1} + 10^{-2} + \ldots. + 10^{-n+1}) = 0.9 \times \frac{1 - 10^{-n}}{0.9} = 1 - 10^{-n} < 1$
On en déduit : $a_n = E(10^{-n} . X)$

Ensuite :
X devient $X - 10^n \times E(10^{-n}. X)$ et on recommence autant de fois qu'il y a de chiffres dans l'écriture de X, c'est à dire n + 1.

Exemple : X = 42378

$4 = E(\frac{X}{10000})$

$X - 10000 \times E(\frac{X}{10000}) \rightarrow X = 2378$

$2 = E(\frac{X}{1000})$

$X - 1000 \times E(\frac{X}{1000}) \rightarrow X = 378$

$3 = E(\frac{X}{100})$

$X - 100 \times E(\frac{X}{100}) \rightarrow X = 78$

$7 = E(\frac{X}{10})$

$X - 10 \times E(\frac{X}{10}) \rightarrow X = 8$

A partir du nombre entier X= 42378 on obtient la liste L1 = { 4 ; 2 ; 3 ; 7 ; 8 }

Ce principe donne le programme suivant sur TI.83 :

:ClrList L1	(On vide la liste L1)
:Prompt X	(on saisit la valeur de X)
:Prompt P	(on saisit la valeur de P qui est égale au nombre de chiffres de X)
:For(K ,1 , P)	(pour K variant de 1 à P)
$:Int(\frac{X}{10^{p-k}}) \rightarrow J$	$(J = E(\frac{X}{10^{p-k}}))$
$:J \rightarrow L1(K)$	(J devient le terme de rang K de la liste L1)
$:X - J \times 10^{p-k} \rightarrow X$	(X devient :$X - J \times 10^{p-k}$)
:End	(fin d'ordre For)
:Disp L1	(envoyer la liste L1)

On peut améliorer ce programme en ne saisissant plus le nombre p de chiffres de X mais en laissant la machine calculer p.

Pour ce faire on a besoin du résultat suivant :

Propriété :
Le nombre p de chiffres d'un entier naturel X non nul écrit en base 10 est :
p = E(log(X)) + 1 où log(X) désigne le logarithme décimal de X : log(X) = $\frac{\ln(X)}{\ln(10)}$

Preuve
Considérons l'écriture en base 10 de l'entier naturel X :
$X = 10^n a_n + 10^{n-1} a_{n-1} + \ldots + 10 a_1 + a_0$, $a_i \in \{0 ; 1 ; 2 \ldots\ldots ; 9\}$, $i \in \{0;1;2\ldots\ldots;.n\}$ et $a_n \neq 0$.
p = n + 1 est alors le nombre de chiffres de X.
En prenant $a_i = 0$ et $a_n = 1$ pour $i \in \{0;1;2\ldots\ldots;n-1\}$ on obtient : $10^n \leq X$ d'une part et en prenant $a_i = 9$ pour $i \in \{0;1;2\ldots\ldots;.n\}$ on obtient :

$X \leq 9 . 10^n + 9 . 10^{n-1} + \ldots + 9 . 10 + 9 = 9 \times \frac{10^{n+1} - 1}{10 - 1} = 10^{n+1} - 1 < 10^{n+1}$ d'autre part.
On a donc : $10^n \leq X < 10^{n+1} \Rightarrow n \leq \log(X) < n+1 \Rightarrow p = n + 1 = E(\log(X)) + 1$.
(Ce qui constitue une autre application de la partie entière).

Le programme précédent est alors modifié comme ceci :

:ClrList L1	(On vide la liste L1)
:Prompt X	(on saisit la valeur de X)
:int(log(X)) + 1 → P	(on calcule la valeur de P qui est égale au nombre de chiffres de X)
:For(K ,1 , P)	(pour K variant de 1 à P)
:Int($\frac{X}{10^{p-k}}$) → J	(J = E($\frac{X}{10^{p-k}}$))
:J → L1(K)	(J devient le terme de rang K de la liste L1)
:X - J× 10^{p-k} → X	(X devient :X - J× 10^{p-k})
:End	(fin d'ordre For)
:Disp L1	(envoyer la liste L1)

Un contexte où intervient le problème précédent est le suivant :
Une suite de chiffres, en base 10, peut-elle être considérée comme une suite de chiffres aléatoires? Outre que la fréquence d'apparition de chacun des chiffres 0 ;1 ;2 ;3 ;4 ;5 ;6 ;7 ;8 ;9 dans la suite doit être de $\frac{1}{10}$ d'autres propriétés doivent être vérifiées par cette suite.
Citons Arthur ENGEL dans « Les certitudes du hasard » page 147 §11.2 :
"Aucun procédé de fabrication de chiffres aléatoires n'est entièrement fiable, Il est donc nécessaire de « mesurer » leur caractère aléatoire. En particulier il ne suffit pas de mesurer la fréquence de chaque chiffre il faut aussi vérifier la fréquence de différents blocs.
Un des tests les plus sûrs est le test dit du poker. Les chiffres sont regroupés par blocs de 5.
La probabilité d'un tel bloc est 10^{-5} . Ces blocs sont regroupés en 7 catégories dont les probabilités sont calculées. On compare alors ces probabilités avec les fréquences observées ".

Le tableau suivant donne la description des 7 catégories, lointainement inspirées du poker (je cite) :

n°	description	type	exemple	probabilité
1	chiffres différents	abcde	34961	0,3024
2	une paire	aabcd	29512	0,5040
3	deux paires	aabbc	44533	0,1080
4	un triplet	aaabc	60366	0,0720
5	paire triplet	aaabb	23223	0,0090
6	quadruplet	aaaab	29222	0,0045
7	quintuplet	aaaaa	55555	0,0001

Le procédé de fabrication de la suite de chiffres dont on veut tester le caractère aléatoire et de leur regroupement en blocs de cinq est ici le suivant :
On génère des nombres aléatoires à l'aide des fonctions rand ou ALEA() et on prend les cinq premières décimales de chacun de ces nombres en faisant : int (10^5·rand) (soit E(10^5·rand)) . Les blocs de cinq chiffres apparaissent donc sous la forme d'un entier naturel à cinq chiffres. Si l'ambition est de faire un programme qui calcule la fréquence de chaque type décrit dans le tableau précédent dans la suite (qui est finie) il est nécessaire, à mon avis, de faire de l'entier naturel à cinq chiffres une liste à cinq éléments, ce que fait le programme donné plus haut.

Quant au programme qui calcule la fréquence de chaque type décrit dans le tableau précédent il n'est pas simple mais faisable puisque déjà fait.

3 Utilisation de la partie entière pour programmer le positionnement régulier d'images numérisées sur une feuille.

Cette application est due à mon collègue Georges LAURENCIN professeur de mathématiques au lycée de Sada à Mayotte.

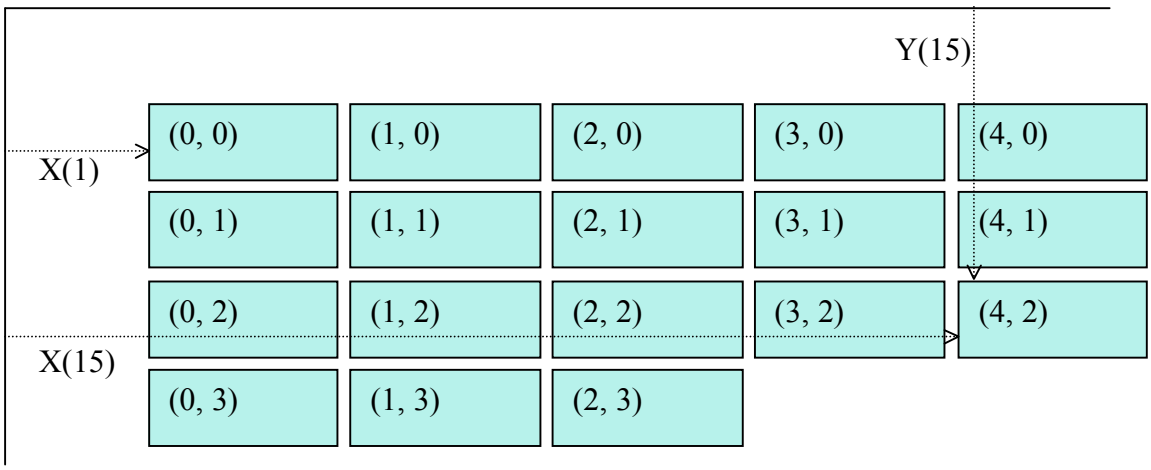
problème : une image virtuelle est positionnée sur la feuille par ses coordonnées X et Y qui sont respectivement la distance du bord gauche de l'image au bord gauche de la feuille et la distance du bord supérieur de l'image au bord supérieur de la feuille;
on dispose de n images (par exemple n photos) à placer par programmation sur une feuille de dimensions données (par exemple au format A4);
dans un contexte de programmation, les images numériques (stockés dans un dossier du disque dur ou créés) sont numérotées de 1 à n, par des instructions comme :

```
For k = 1 to n
    ...
next k
```

Selon les dimensions des photos, les paramètres de mise en page (marges, pas, nombre d'images par ligne) sont déterminés (le nombre d'images par ligne est nécessairement inférieur à un maximum géométriquement possible, de même que le nombre d'images par page, ces valeurs optimales pouvant par ailleurs être calculées automatiquement en fonctions des paramètres géométriques des objets en présence).
Connaissant le nombre p d'images par ligne, et les paramètres de mise en page (marge de gauche e et pas horizontal h, marge du haut m et pas vertical v) l'objectif est de positionner l'image k selon les instructions d'un programme devant fournir les coordonnées X(k) et Y(k).

En d'autres termes, il s'agit de créer, à partir de la suite 1, 2, 3, ... , k, ... , n
les suites X(k) et Y(k) des coordonnées de position de l'image k.

Dans cet exemple, 15 objets sont placés sur la feuille avec un "saut de ligne" tous les multiples de 5 :



X(k), représentant la distance du bord gauche de l'image k au bord gauche de la feuille, est la suite de période p :

X(1) = e , X(2) = e + h , X(3) = e + 2h , ... , X(p) = e + (p-1)h ,
X(p+1) = e , X(p+2) = e + h , X(p+3) = e + 2h , ... , X(2p) = e + (p-1)h ,
X(2p+1) = e , X(2p+2) = e + h , X(2p+3) = e + 2h , ... , X(3p) = e + (p-1)h ,
....

Y(k), représentant la distance du bord supérieur de l'image k au bord supérieur de la feuille est la suite :

Y(1) = m , Y(2) = m , Y(3) = m , ... , Y(p) = m ,
Y(p+1) = m + v , Y(p+2) = m + v , Y(p+3) = m + v , ... , Y(2p) = m + v ,
Y(2p+1) = m + 2v , Y(2p+2) = m + 2v , Y(2p+3) = m + 2v , ... , Y(3p) = m + 2v ,
....

X(k) est constante pour toutes les images d'une même colonne
Y(k) est constante pour toutes les images d'une même ligne

Nonobstant les valeurs des paramètres de mise en page (e, h, m, v), il s'agit en clair de transformer la suite 1, 2, 3, ... , n en suites des coordonnées :

(0; 0) , (1; 0) , (2; 0) , ... , (p-1; 0) , pour la première ligne d'images
(0; 1) , (1; 1) , (2; 1) , ... , (p-1; 1) , pour la deuxième ligne d'images
(0; 2) , (1; 2) , (2; 2) , ... , (p-1; 2) , pour la troisième ligne d'images
(0; 3) , (1; 3) , (2; 3) , ... , (p-1; 3) , etc
...

Application :
Pour k ∈ ℕ*, les fonctions (dont l'écriture doit être adaptée au langage de programmation) :
X(k, p) = (k - 1) Mod p , reste de la division de (k-1) par p
et
Y(k, p) = Int((k - 1) / p) , partie entière du quotient (k-1)/p
donnent les résultats attendus :

k	1	2	3	...	p	p+1	p+2	p+3	...	2p	2p+1	2p+2	2p+3	...	3p	3p+1	3p+2	....
X(k) :	0	1	2	...	p-1	0	1	2	...	p-1	0	1	2	...	p-1	0	1	
Y(k) :	0	0	0	...	0	1	1	1	...	1	2	2	2	...	2	3	3	

Cette application permet donc de gérer les "sauts de lignes" après tous les multiples de p images par ligne.
Remarquons que se pose le problème du "sauts de pages" après tous les multiples de r images par page qu'il est possible de gérer avec des translations de l'index k et des fonctions du même type :

Xs(k) = X(k, r) Mod p
Ys(k) = Int(X(k, r) / p) + Y(k, r) × Ypage, ou Ypage est la dimension Y d'une page

Voir le fichier Excel (Application de Ent().xls) donné en exemple
La dérive de la marge du haut éventuellement observée sur un document Excel est due au fait les sauts de pages ne sont pas positionnés selon la hauteur de la page mais sont un multiple de la hauteur des cellules de Excel.