

Conversions

Algo & Prog avec R

A. Malapert, B. Martin, M. Pelleau, et J.-P. Roy

4 octobre 2019

Université Côte d'Azur, CNRS, I3S, France
`firstname.lastname@univ-cotedazur.fr`

Épluchage d'un entier chiffre à chiffre

- ▶ Seul le chiffre des unités (le plus à droite) est accessible,
- ▶ ainsi que l'entier obtenu en supprimant le chiffre des unités (décalage à droite).

345		6
%%		%%

```
Eplucher <- function(n) {  
  while(n > 0) {  
    cat(n, "j enleve", n %% 10, "\n")  
    n <- n %/% 10  
  }  
}
```

```
> n <- 3456  
> n %% 10  
[1] 6  
> n %/% 10  
[1] 345
```

```
> Eplucher(3456)  
3456 j enleve 6  
345 j enleve 5  
34 j enleve 4  
3 j enleve 3
```

Épluchage récursif d'un entier

Une fonction récursive s'appelle elle-même.

- ▶ Il est impératif de prévoir une condition d'arrêt à la récursion, sinon le programme ne s'arrête jamais !
- ▶ La récursivité fonctionne car chaque appel de fonction est différent.

```
Eplucher <- function(n) {  
  if(n > 0) {  
    cat(n,"j enleve", n %% 10, "\n")  
    Eplucher(n %/% 10)  
  }  
}
```

Procédure

Une procédure est une routine qui ne retourne pas de valeur.

Épluchage binaire d'un entier

Le principe reste le même !

```
Eplucher <- function(n) {  
  while(n > 0) {  
    print(n %% 2)  
    n <- n %/% 2  
  }  
}
```

Épluchage de la droite vers le gauche.

$$(3456)_{10} = (110110000000)_2$$

```
> Eplucher(3456)  
[1] 0  
[1] 0  
[1] 0  
[1] 0  
[1] 0  
[1] 0  
[1] 0  
[1] 1  
[1] 1  
[1] 0  
[1] 1  
[1] 1
```

Variante : bit de poids fort

Le bit de poids fort, (en anglais **most significant bit**, ou MSB) est le bit, dans une représentation binaire donnée, ayant la plus grande valeur.

Le nombre $(9)_{10}$ s'écrit $(1001)_2$ en binaire

Le MSB (à gauche) contribue pour 8 unités à la valeur totale du nombre.

```
MSB <- function(n) {  
  if (n <= 0) return(0);  
  msb <- 1;  
  while (msb <= n) {  
    msb <- 2 * msb  
  }  
  return(msb / 2)  
}
```

```
> MSB(9)  
[1] 8  
> MSB(16)  
[1] 16  
> MSB(25)  
[1] 16  
> MSB(32)  
[1] 32  
> MSB(33)  
[1] 32
```

Épluchage d'un nombre fractionnaire

- ▶ Seul le chiffre le plus à gauche est accessible,
- ▶ ainsi que le nombre fractionnaire obtenu en supprimant le chiffre des le plus à gauche (décalage à gauche).

3	0.456
<code>floor(10*n)</code>	<code>10*n - floor(10*n)</code>

```
> n <- 0.3456
> m <- 10 * n
> floor(m)
[1] 3
> m - floor(m)
[1] 0.456
```

```
Eplucher <- function(n) {
  while(n > 0) {
    m <- 10 * n
    f <- floor(m)
    cat(n, "j enleve", f, "\n")
    n <- m - f
  }
}
```

```
> Eplucher(1/32)
0.03125 j enleve 0
0.3125 j enleve 3
0.125 j enleve 1
0.25 j enleve 2
0.5 j enleve 5
```

Surprise ! Le calcul fractionnaire n'est pas exact !

```
> Eplucher(0.3456)
0.3456 j enleve 3
0.456 j enleve 4
0.56 j enleve 5
0.6 j enleve 6
4.050094e-13 j enleve 0
4.050094e-12 j enleve 0
4.050094e-11 j enleve 0
4.050094e-10 j enleve 0
4.050094e-09 j enleve 0
4.050094e-08 j enleve 0
4.050094e-07 j enleve 0
...
0.3125 j enleve 3
0.125 j enleve 1
0.25 j enleve 2
0.5 j enleve 5
```

Questions?

Retrouvez ce cours sur le site web

[`www.i3s.unice.fr/~malapert/R`](http://www.i3s.unice.fr/~malapert/R)