Méthodes numériques I : IEEE 754

INFO1 - Semaines 18 & 19

Guillaume CONNAN

Dernière mise à jour : 3 mai 2015 à 11:02

IUT de Nantes - Dpt d'informatique

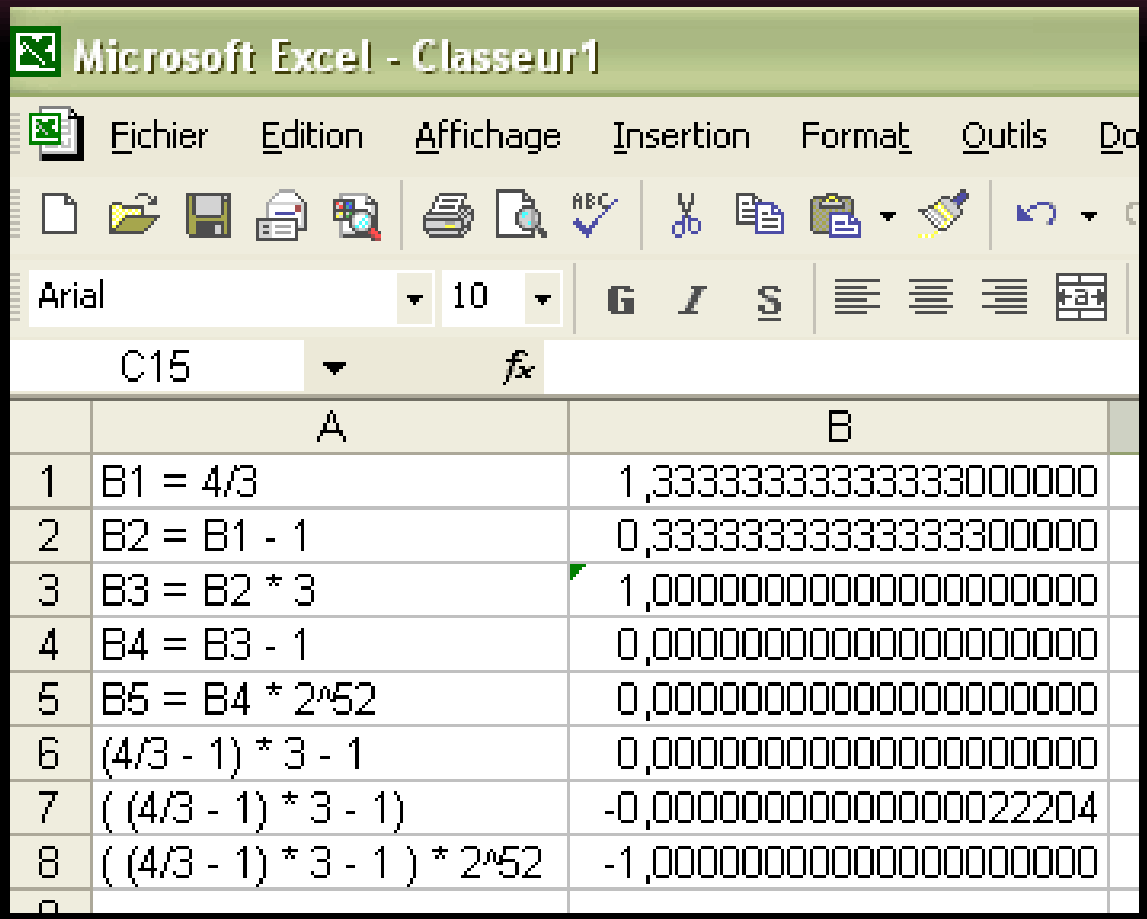
Sommaire

Sommaire

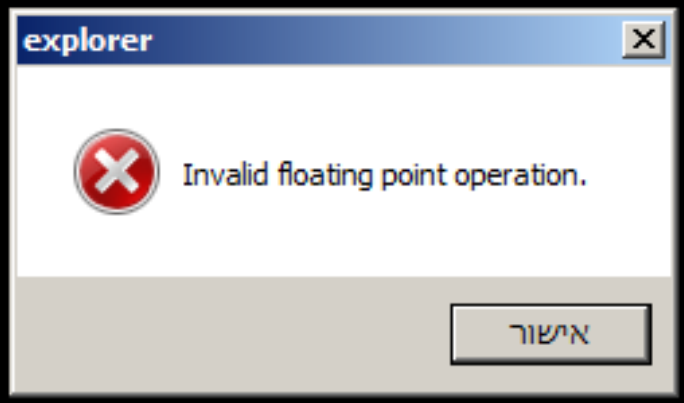

SCUD

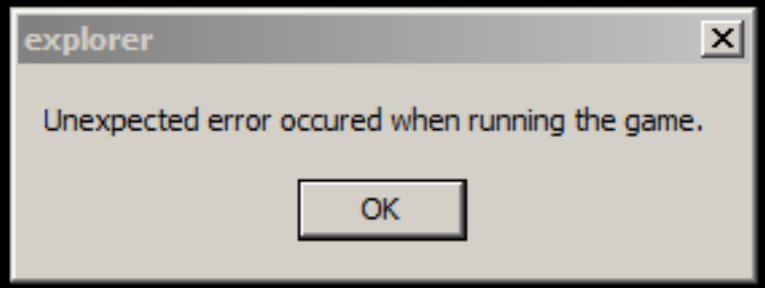

OUPS !

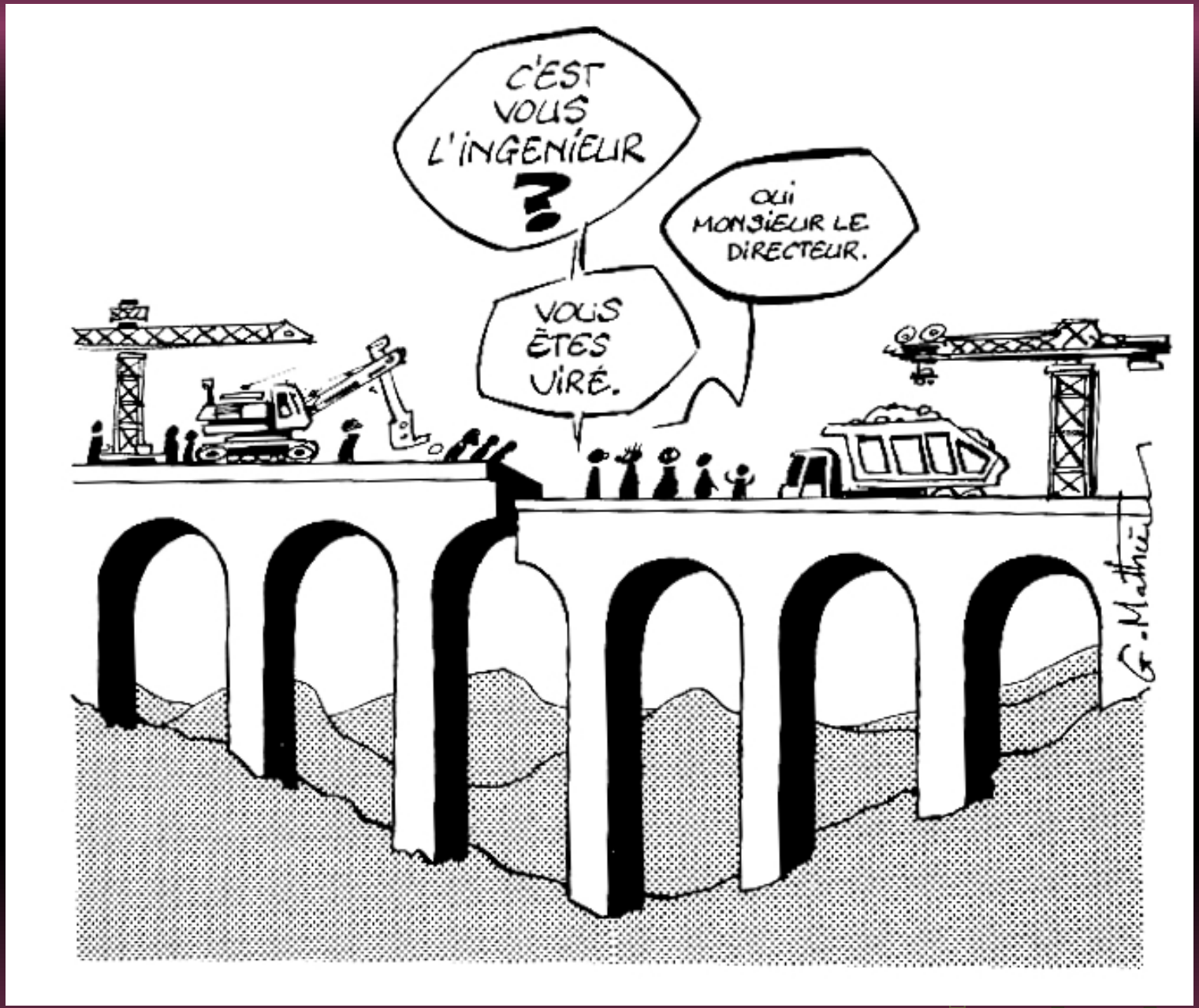

LE 20 JANVIER 1992 LE MATELOT CLOVAIRE LEAHIDU SE COINÇA LE PIED DANS LA CATAPULTE DU PORTE-AVIONS CLÉMENCEAU. ON VIENT D'ABANDONNER LES RECHERCHES.

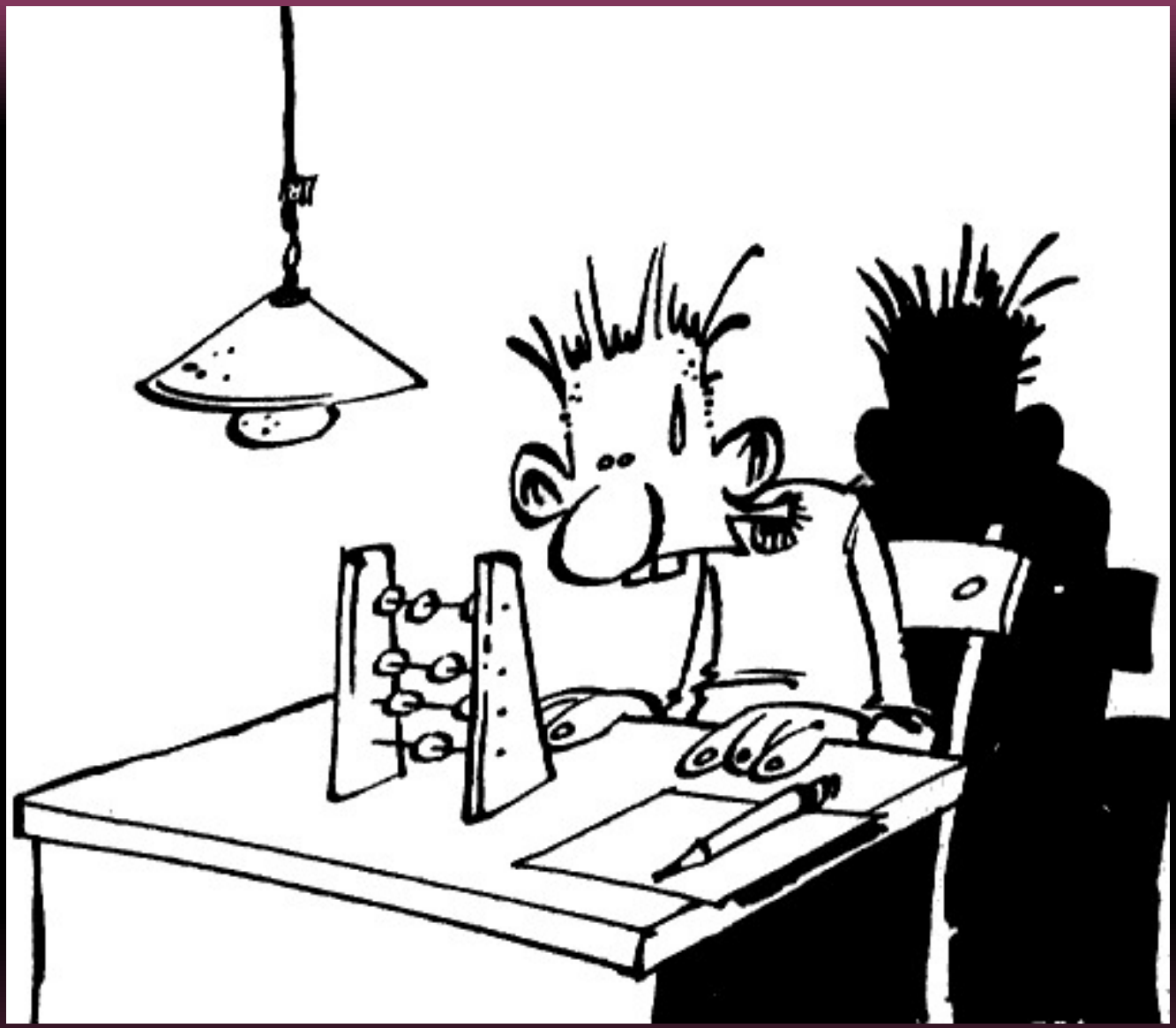


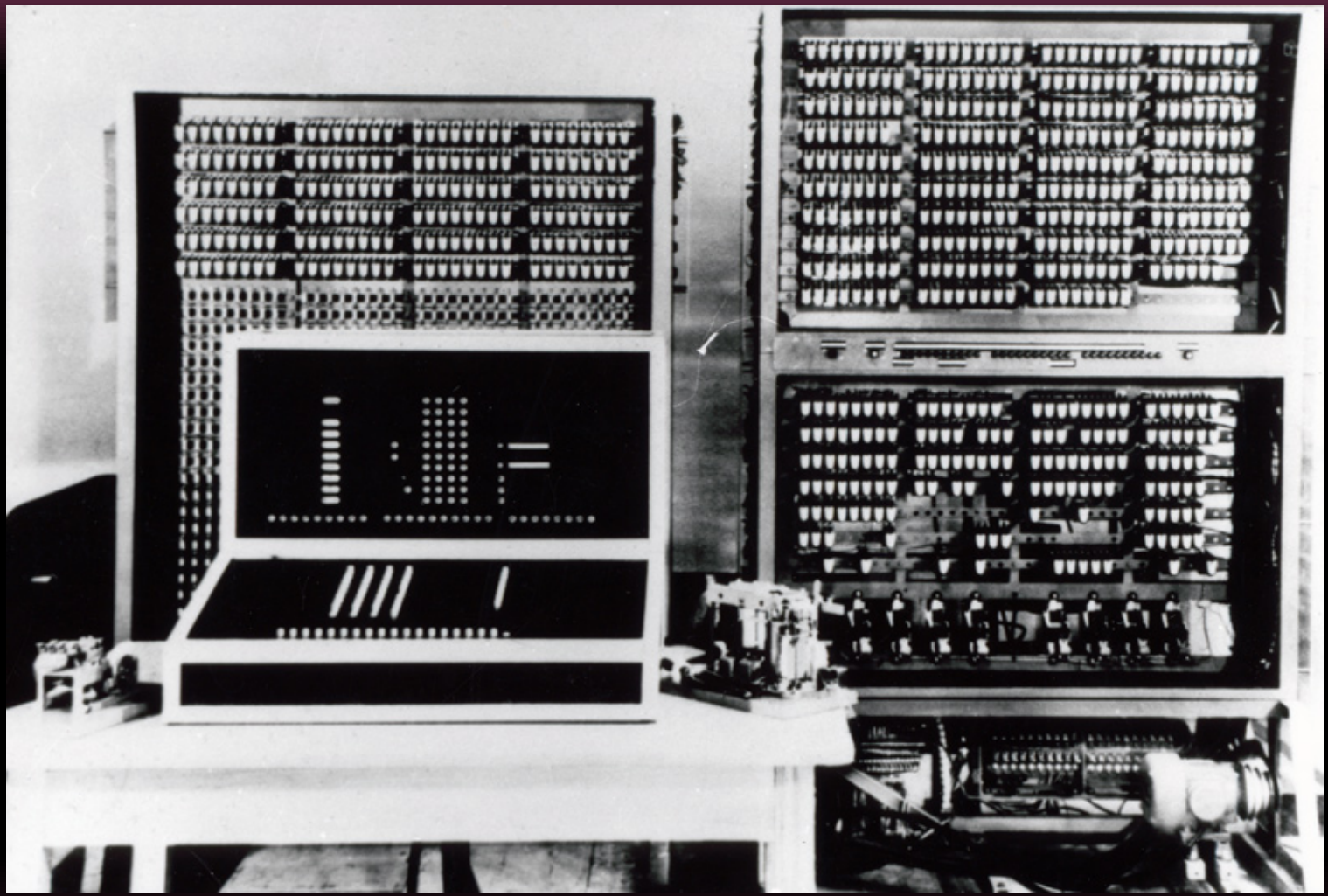
	A	B
1	B1 = 4/3	1,33333333333333000000
2	B2 = B1 - 1	0,33333333333333300000
3	B3 = B2 * 3	1,00000000000000000000
4	B4 = B3 - 1	0,00000000000000000000
5	B5 = B4 * 2^52	0,00000000000000000000
6	(4/3 - 1) * 3 - 1	0,00000000000000000000
7	((4/3 - 1) * 3 - 1)	-0,00000000000000022204
8	((4/3 - 1) * 3 - 1) * 2^52	-1,00000000000000000000


explorer
Invalid floating point operation.
אישור


explorer
Unexpected error occured when running the game.
OK


C'EST VOUS L'INGENIEUR ?
OUI MONSIEUR LE DIRECTEUR.
VOUS ÊTES VIRÉ.
G. Mathieu





- char
- int -2147483647 - 1 — 2147483648

- `char`
- `int` 2147483647 + 1 = −2147483648...

- `char`
- `int` $2147483647 + 1 = -2147483648$...
- `short`

- `char`
- `int` $2147483647 + 1 = -2147483648$...
- `short` `long`

- `char`
- `int` $2147483647 + 1 = -2147483648$...
- `short` `long`

- `char`
- `int` $2147483647 + 1 = -2147483648$...
- `short` `long`
- `float`

- `char`
- `int` $2147483647 + 1 = -2147483648$...
- `short` `long`
- `float`
- `single`
- `double`

- char
- int $2147483647 + 1 = -2147483648$...
- short long
- float
- single
- double
- long double
- 3

- char
- int $2147483647 + 1 = -2147483648$...
- short long
- float
- single
- double
- long double
- 3

- char
- int $2147483647 + 1 = -2147483648$...
- short long
- float
- single
- double
- long double
- 3

- char
- int 2147483647 + 1 = −2147483648...
- short long
- float
- single
- double
- long double
- 3

- char
- int 2147483647 + 1 = −2147483648...
- short long
- float
- single
- double
- long double
- 3 3.0 3.0000 3.0e0 0.3e1

- char
- int $2147483647 + 1 = -2147483648$...
- short long
- float
- single
- double
- long double
- 3 3.0 3.0000 3.0e0 0.3e1

- char
- int $2147483647 + 1 = -2147483648$...
- short long
- float
- single
- double
- long double
- 3 3.0 3.0000 3.0e0 0.3e1

- char
- int $2147483647 + 1 = -2147483648$...
- short long
- float
- single
- double
- long double
- 3 3.0 3.0000 3.0e0 0.3e1

- `char`
- `int` $2147483647 + 1 = -2147483648$...
- `short` `long`
- `float`
- single
- `double`
- `long double`
- 3 3.0 3.0000 3.0e0 0.3e1
- ...

- `char`
- `int` $2147483647 + 1 = -2147483648$...
- `short` `long`
- `float`
- single
- `double`
- `long double`
- 3 3.0 3.0000 3.0e0 0.3e1
- ...

```
*Main> 3 * 0.1
.
.
.
```

```
*Main> 3 * 0.1
0.30000000000000004
.
.
```

```
*Main> 3 * 0.1
0.30000000000000004
*Main> sum $ take 10000000 $ repeat 0.1
.
```

```
*Main> 3 * 0.1
0.30000000000000004
*Main> sum $ take 10000000 $ repeat 0.1
999999.9998389754
```

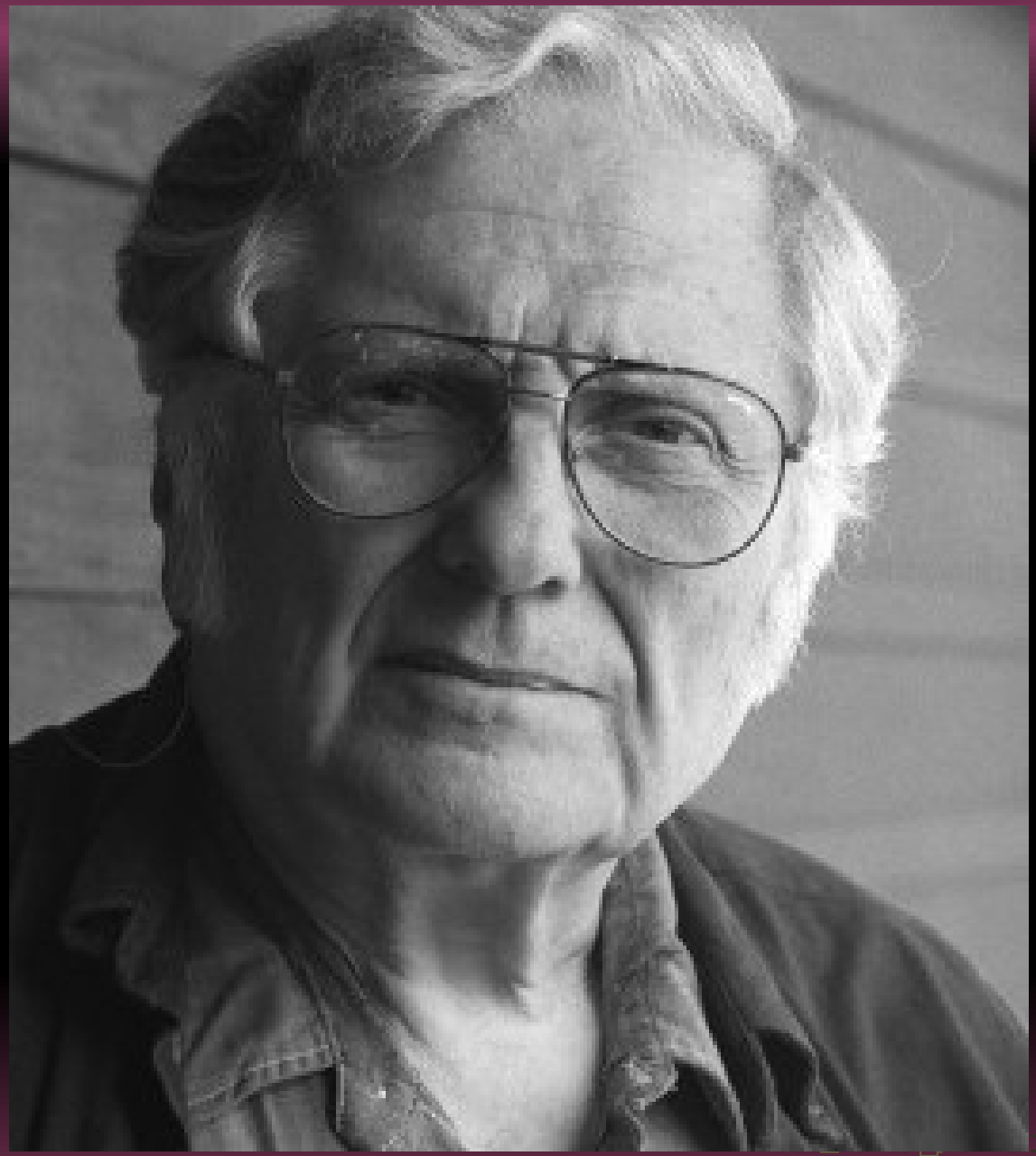
LES NOMBRES RÉELS N'EXISTENT PAS.

TOUT CE QUE VOUS AVEZ VU AU LYCÉE N'EST QU'ILLUSION !

LES NOMBRES RÉELS N'EXISTENT PAS.

TOUT CE QUE VOUS AVEZ VU AU LYCÉE N'EST QU'ILLUSION !

Sommaire



$$v = (-1)^s \times m \times 2^E$$

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.
- *toy8* $(\#E, \#m) = (3, 5)$.

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.
- *toy8* $(\#E, \#m) = (3, 5)$

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.
- *toy8* $(\#E, \#m) = (3, 5)$.

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.
- *toy8* $(\#E, \#m) = (3, 5)$.

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.
- *toy8* $(\#E, \#m) = (3, 5)$.

$$v = (-1)^s \times m \times 2^E$$

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.
- *toy8* $(\#E, \#m) = (3, 5)$.

$$v = 0,010\mathit{01} \quad \times 2^{0}$$
$$v = 0,100\mathit{1} \quad \times 2^{-1}$$
$$v = 1,001 \quad \times 2^{-2}$$

$$1 \leqslant m < 2$$

$$m = 1, f$$

$$v = 0,010\mathit{01} \quad \times 2^{0}$$
$$v = 0,100\mathit{1} \quad \times 2^{-1}$$
$$v = 1,001 \quad \times 2^{-2}$$

$$1 \leqslant m < 2$$

$$m = 1, f$$

$$v = 0,010\mathit{01} \quad \times 2^{0}$$
$$v = 0,100\mathit{1} \quad \times 2^{-1}$$
$$v = 1,001 \quad \times 2^{-2}$$

$$1 \leqslant m < 2$$

$$m = 1, f$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant : il suffit de le translater de $2^{(\#E)-1} - 1$...POURQUOI ?

$$e = E + 2^{(\#E)-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant : il suffit de le translater de $2^{(\#E)-1} - 1$...POURQUOI ?

$$e = E + 2^{(\#E)-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant : il suffit de le translater de $2^{(\#E)-1} - 1$...POURQUOI ?

$$e = E + 2^{(\#E)-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant : il suffit de le translater de $2^{(\#E)-1} - 1$...POURQUOI ?

$$e = E + 2^{(\#E)-1} - 1$$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit 0b00101000 stocké en mémoire sous forme hexadécimale : 0x28

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : 0x28

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en toy8

$0,75_{10}$ en *toy8*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,1000_2 \times 2^{-1}$$

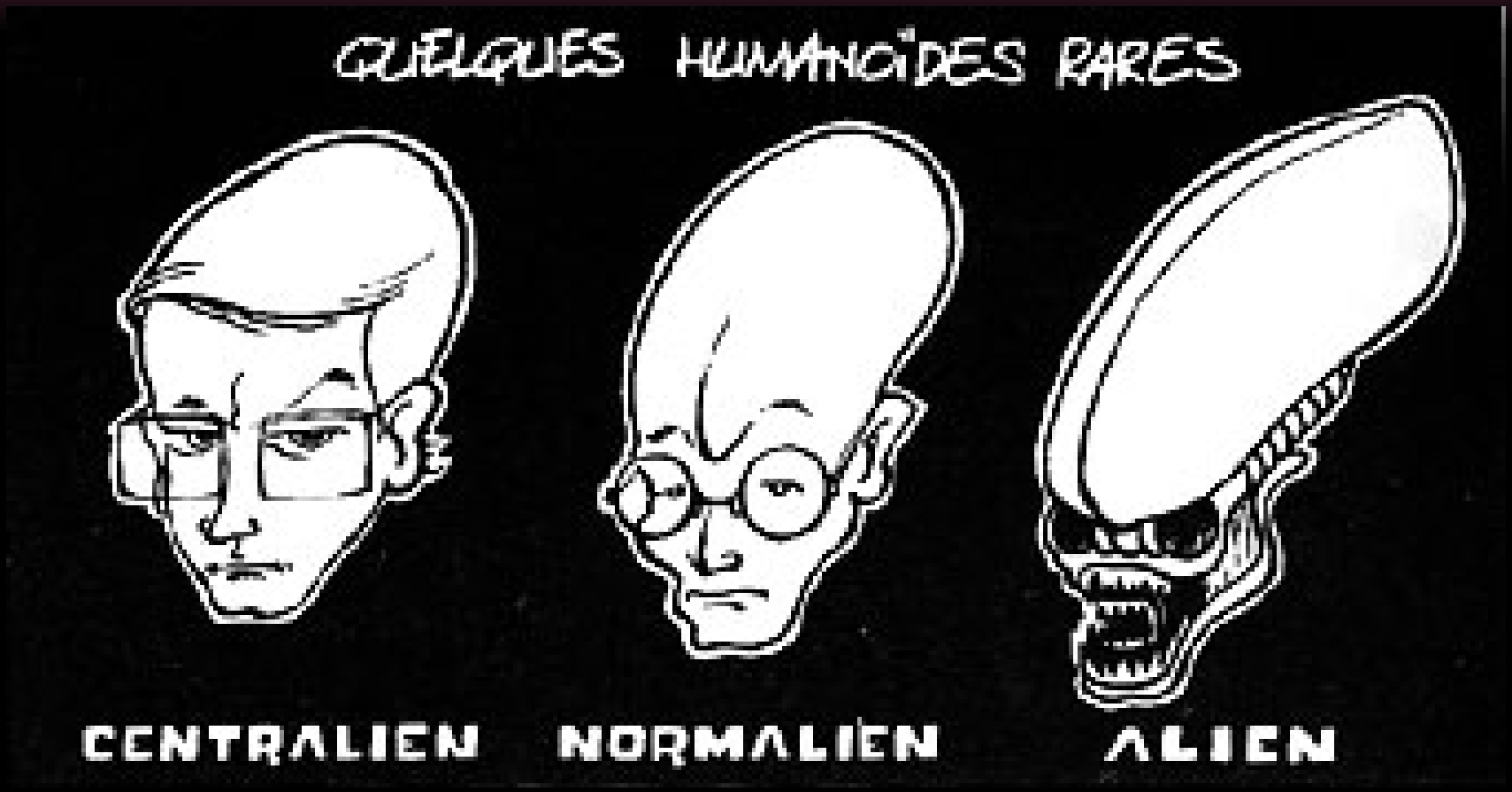
$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (4 bits)			
0	0	1	0	1	0	0	0

Soit $0b00101000$ stocké en mémoire sous forme hexadécimale : $0x28$

Exemple en binary32

Et en *binary32* ?


QUELQUES HUMANOÏDES RARES
CENTRALIEN
NORMALIEN
ALIEN

$$e_{\text{min}} = 0$$

$$e_{\text{max}} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\text{min}} = 000,\ e_{\text{max}} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\text{min}} = \left(e_{\text{min}} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\text{max}} = \left(e_{\text{max}} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$toy7, \#E = 3$, $e_{\min} = 000$, $e_{\max} = 111 = 1000 - 1$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

toy7, $\#E = 3$, $e_{\min} = 000$, $e_{\max} = 111 = 1000 - 1$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

toy7,$\#E = 3$, $e_{\min} = 000$, $e_{\max} = 111 = 1000 - 1$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$\textit{toy7}, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = \left(2^3 - 1\right) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

- zéro sous forme normale ?
- $0.\underbrace{000\ldots00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....en TD

- zéro sous forme normale ?
- $0.\underbrace{000...00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....en TD

- zéro sous forme normale ?
- $0.\underbrace{000...00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....en TD

- zéro sous forme normale ?
- $0.\underbrace{000...00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....en TD

- zéro sous forme normale ?
- $0.\underbrace{000...00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....en TD

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	0	0	0	0	0

s (1 bit)	e (3 bits)			f (3 bits)		
1	0	0	0	0	0	0

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	0	0	0	0	0

s (1 bit)	e (3 bits)			f (3 bits)		
1	0	0	0	0	0	0

```
*Main> let x = 1 / 1e500
.
.
.
.
.
.
.
.
.
.
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
.
.
.
.
.
.
.
.
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
.
.
.
.
.
.
.
.
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
*Main> x == 0
.
.
.
.
.
.
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
*Main> x == 0
True
*Main> let y = -1 / 1e500
.
.
.
.
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
*Main> x == 0
True
*Main> let y = -1 / 1e500
*Main> y
.
.
.
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
*Main> x == 0
True
*Main> let y = -1 / 1e500
*Main> y
-0.0
.
.
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
*Main> x == 0
True
*Main> let y = -1 / 1e500
*Main> y
-0.0
*Main> y == 0
.
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
*Main> x == 0
True
*Main> let y = -1 / 1e500
*Main> y
-0.0
*Main> y == 0
True
.
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
*Main> x == 0
True
*Main> let y = -1 / 1e500
*Main> y
-0.0
*Main> y == 0
True
*Main> x == y
.
```

```
*Main> let x = 1 / 1e500
*Main> x
0.0
*Main> x == 0
True
*Main> let y = -1 / 1e500
*Main> y
-0.0
*Main> y == 0
True
*Main> x == y
True
```

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$$1,111 \times 2^4 \ ?$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$

$$x^2 + y^2 = 1,111 \times 2^4$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$x = 1 \times 2^3$ $y = 1,1 \times 2^3$

toy7

$1,111 \times 2^4$?

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$x^2 = 1,111 \times 2^4$

$y^2 = 1,111 \times 2^4$

$x^2 + y^2 = 1,111 \times 2^4$

$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$1,111 \times 2^4$?

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$

$$x^2 + y^2 = 1,111 \times 2^4$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$1,111 \times 2^4$?

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$
$$x^2 + y^2 = 1,111 \times 2^4$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$$1,111 \times 2^4 \ ?$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$
$$x^2 + y^2 = 1,111 \times 2^4$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$1,111 \times 2^4$?

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$
$$x^2 + y^2 = 1,111 \times 2^5$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$$1,111 \times 2^4 \ ?$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$

$$y^2 = 1,111 \times 2^4$$

$$x^2 + y^2 = 1,111 \times 2^4$$

$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$$1,111 \times 2^4 \ ?$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$
$$x^2 + y^2 = 1,111 \times 2^4$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$$1,111 \times 2^4 \ ?$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$
$$x^2 + y^2 = 1,111 \times 2^4$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$$1,111 \times 2^4 \ ?$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$
$$x^2 + y^2 = 1,111 \times 2^4$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$$N = \sqrt{x^2 + y^2}$$

$$x = 1 \times 2^3 \qquad y = 1,1 \times 2^3$$

toy7

$$1,111 \times 2^4 \ ?$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	1	1	1

$$x^2 = 1,111 \times 2^4$$
$$y^2 = 1,111 \times 2^4$$
$$x^2 + y^2 = 1,111 \times 2^4$$
$$N = \left(1,111 \times 2^4\right)^{1/2} \approx 1,010 \times 2^2$$

$+\infty$:

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	0	0	0

$-\infty$:

s (1 bit)	e (3 bits)			f (3 bits)		
1	1	1	1	0	0	0

$e = e_{\max}$ et $f = 0$

$+\infty$:

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	0	0	0

$-\infty$:

s (1 bit)	e (3 bits)			f (3 bits)		
1	1	1	1	0	0	0

$e = e_{\max}$ et $f = 0$

$+\infty$:

s (1 bit)	e (3 bits)			f (3 bits)		
0	1	1	1	0	0	0

$-\infty$:

s (1 bit)	e (3 bits)			f (3 bits)		
1	1	1	1	0	0	0

$e = e_{\max}$ et $f = 0$

```
*Main> 1 + 1e500
.
.
.
.
.
.
.
.
.
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
.
.
.
.
.
.
.
.
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
.
.
.
.
.
.
.
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
.
.
.
.
.
.
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
.
.
.
.
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
*Main> 1 + x
.
.
.
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
*Main> 1 + x
Infinity
.
.
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
*Main> 1 + x
Infinity
*Main> x^3
.
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
*Main> 1 + x
Infinity
*Main> x^3
Infinity
.
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
*Main> 1 + x
Infinity
*Main> x^3
Infinity
*Main> 1 / x
.
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
*Main> 1 + x
Infinity
*Main> x^3
Infinity
*Main> 1 / x
0.0
.
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
*Main> 1 + x
Infinity
*Main> x^3
Infinity
*Main> 1 / x
0.0
*Main> 4 - x
.
```

```
*Main> 1 + 1e500
Infinity
*Main> let x = 1e500
*Main> x
Infinity
*Main> 1 + x
Infinity
*Main> x^3
Infinity
*Main> 1 / x
0.0
*Main> 4 - x
-Infinity
```

$$\lim_{x \to +\infty} x^2 - x$$

```
*Main> x^2 - x
.
```

$$\lim_{x \to +\infty} x^2 - x$$

```
*Main> x^2 - x
.
```

$$\lim_{x \to +\infty} x^2 - x$$

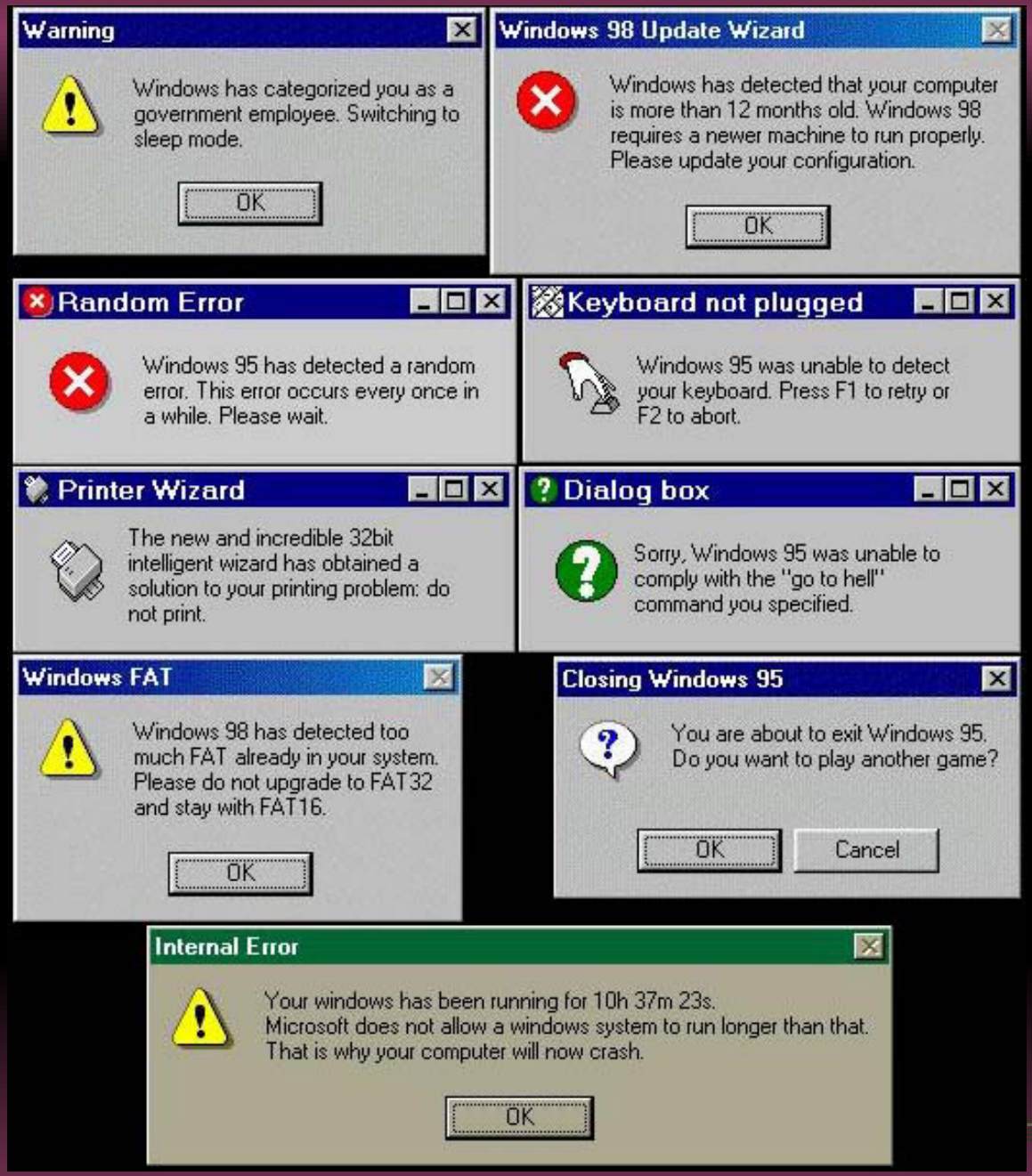
```
*Main> x^2 - x
NaN
```


Aladdin
NAN
PITA BREAD
15 OZ

```
*Main> let x = 1e500
*Main> x - x
NaN
*Main> x / x
NaN
*Main> sqrt(-1)
NaN
*Main> 0 / 0
NaN
```

```
-- Bizarre ?
*Main> x - x
NaN
*Main> x - x == x - x
False
*Main> x^2 - x
NaN
*Main> x * (x - 1)
Infinity
```


48


Warning
Windows has categorized you as a government employee. Switching to sleep mode.
OK
Windows 98 Update Wizard
Windows has detected that your computer is more than 12 months old. Windows 98 requires a newer machine to run properly. Please update your configuration.
OK
Random Error
Windows 95 has detected a random error. This error occurs every once in a while. Please wait.
Keyboard not plugged
Windows 95 was unable to detect your keyboard. Press F1 to retry or F2 to abort.
Printer Wizard
The new and incredible 32bit intelligent wizard has obtained a solution to your printing problem: do not print.
Dialog box
Sorry, Windows 95 was unable to comply with the "go to hell" command you specified.
Windows FAT
Windows 98 has detected too much FAT already in your system. Please do not upgrade to FAT32 and stay with FAT16.
OK
Closing Windows 95
You are about to exit Windows 95. Do you want to play another game?
OK
Cancel
Internal Error
Your windows has been running for 10h 37m 23s.
Microsoft does not allow a windows system to run longer than that.
That is why your computer will now crash.
OK

```
*Main> let x = 1e500
*Main> x^2 - x
NaN
```

```
*Main> x*(x - 1)
Infinity
```

```
*Main> x - x
NaN
*Main> x - x == x - x
False
```

```
*Main> let x = 1e500
*Main> x^2 - x
NaN
```

```
*Main> x*(x - 1)
Infinity
```

```
*Main> x - x
NaN
*Main> x - x == x - x
False
```

```
*Main> let x = 1e500
*Main> x^2 - x
NaN
```

```
*Main> x*(x - 1)
Infinity
```

```
*Main> x - x
NaN
*Main> x - x == x - x
False
```

```
In [53]: x = 1e-500

In [54]: x
Out[54]: 0.0

In [55]: x / x
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call
        ↪  last)
<ipython-input-55-fd52a7f8b5f1> in <module>()
----> 1 x / x

ZeroDivisionError: float division by zero

In [56]: sqrt(-1.0)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call
        ↪  last)
<ipython-input-56-d1c09f21b443> in <module>()
----> 1 sqrt(-1.0)

ValueError: math domain error
```

Air France 447 – 1er juin 2009

s (1 bit)	e (3 bits)			f (3 bits)		
1	1	1	1	0	1	0

$n = \#f$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

$$n = \#f$$

toy7, $n = 3$

binary32, $n = 23$

binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v)+1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

$$n = \#f$$

toy7, $n = 3$

binary32, $n = 23$

binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$succ(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v)+1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

$$n = \#f$$

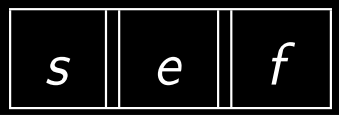
toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

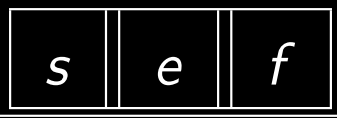
Si v est un VF, entre v et son successeur, il y a...

Arbre de lecture



$e = 0\ldots0$	$f = 0$	$\rightarrow$	$v = (-1)^s \times 0.0$
$e = 0\ldots0$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 0.f \times 2^{1-E_{\max}}$
$e = 1\ldots1$	$f = 0$	$\rightarrow$	$v = (-1)^s \times \infty$
$e = 1\ldots1$	$f \neq 0$	$\rightarrow$	NaN
$0\ldots0 < e < 1\ldots1$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 1.f \times 2^{e-E_{\max}}$

Arbre de lecture



$e = 0...0$	$f = 0$	$\rightarrow$	$v = (-1)^s \times 0.0$
$e = 0...0$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 0.f \times 2^{1-E_{\max}}$
$e = 1...1$	$f = 0$	$\rightarrow$	$v = (-1)^s \times \infty$
$e = 1...1$	$f \neq 0$	$\rightarrow$	NaN
$0...0 < e < 1...1$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 1.f \times 2^{e-E_{\max}}$

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Format	$\#E$	$\#f$	$E_{\min}$	$E_{\max}$	ε_m	λ	μ	Ω
toy7	3	3	-2	3	$2^{-3} = 1/8$	$2^{-2} = 1/4$	$2^{-5} = 1/32$	$1,111 \times 2^{3} = 15$
bin32	8	23	-126	127	2^{-23} $\approx 1,2 \times 10^{-7}$	2^{-126} $\approx 1,2 \times 10^{-38}$	$2^{-126-23}$ $\approx 1,4 \times 10^{-45}$	$(2^{24}-1) \times 2^{127-2}$ $\approx 3,4 \times 10^{38}$
bin64	11	52	-1022	1023	2^{-52} $\approx 2,2 \times 10^{-16}$	2^{-1022} $\approx 2,2 \times 10^{-308}$	2^{-1074} 5×10^{-324}	$(2^{53}-1)2^{1023-5}$ $\approx 1,8 \times 10^{308}$

Sommaire

$$\mathbb{V}_b$$

$$\overline{\mathbb{V}_b}$$

$$\mathbb{V}_b$$

$$\overline{\mathbb{V}_b}$$

Comparaison

Il est très simple et rapide de comparer deux VF : comment la machine procède-t-elle ? Quel est l'avantage de ce stockage des VF ?

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;
2. on ajoute les deux mantisses *complètes* en tenant compte du signe ;
3. on renormalise le nombre obtenu.

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;
2. on ajoute les deux mantisses *complètes* en tenant compte du signe ;
3. on renormalise le nombre obtenu.

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;
2. on ajoute les deux mantisses *complètes* en tenant compte du signe ;
3. on renormalise le nombre obtenu.

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,00111
_____
1,10111

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,00111
_____
1,10111

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,00111
– – – –
1,10111

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––
1,101*11*

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100

0,001*11*

– – ––

1,101*11*

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––
1,101*11*

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––
1,101*11*

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{succ(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - succ(v)| = 0,01\,ulp(x)$ donc $RN(x) = succ(v)$

$$1,1 \oplus 0,00111 = 1,110$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{succ(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - succ(v)| = 0,01\,ulp(x)$ donc $RN(x) = succ(v)$

$$1,1 \oplus 0,00111 = 1,110$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{succ(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - succ(v)| = 0,01\,ulp(x)$ donc $RN(x) = succ(v)$

$$1,1 \oplus 0,00111 = 1,110$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{succ(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - succ(v)| = 0,01\,ulp(x)$ donc $RN(x) = succ(v)$

$$1,1 \oplus 0,00111 = 1,110$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{succ(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - succ(v)| = 0,01\,ulp(x)$ donc $RN(x) = succ(v)$

$$1,1 \oplus 0,00111 = 1,110$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1,00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1,11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10,00}\,ulp(x)}_{\mathsf{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathsf{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathsf{succ}(v)$

$$1,1 \oplus 0,00111 = 1,110$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1,00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1,11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10,00}\,ulp(x)}_{\mathrm{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathrm{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathrm{succ}(v)$

$1,1 \oplus 0,00111 = 1,110$

1,100
0,001*11*
– – ––
1,101*11*

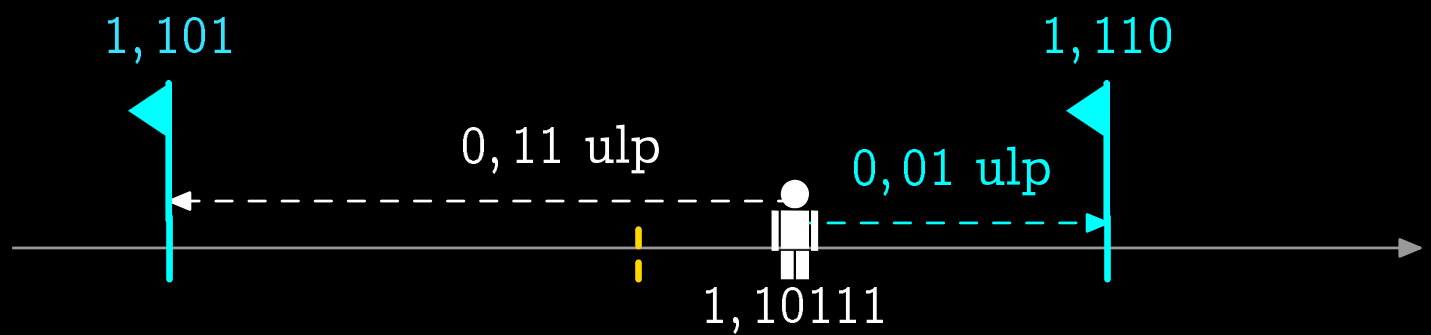
$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1,00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1,11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10,00}\,ulp(x)}_{\mathrm{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathrm{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathrm{succ}(v)$

$$1,1 \oplus 0,00111 = 1,110$$


1, 101
1, 110
0, 11 ulp
0, 01 ulp
1, 10111

1. Est-ce que l'addition des VF est associative ?

```
Prelude> (1 + 1e-16) + 1e-16
1.0
Prelude> 1 + (1e-16 + 1e-16)
1.0000000000000002
```

2. Est-on sûr de l'ordre dans le quel un compilateur calcule $a + b + c + d$?

1. Est-ce que l'addition des VF est associative ?

```
Prelude> (1 + 1e-16) + 1e-16
1.0
Prelude> 1 + (1e-16 + 1e-16)
1.0000000000000002
```

2. Est-on sûr de l'ordre dans le quel un compilateur calcule $a + b + c + d$?

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

- 0 xor 1 donne 1 : le bit de signe est 1 ;

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. 101 + 100 − 11 = 1001 − 11 = 110 : l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 : | 0 | 1 | 0 | 1 | 0 | 1 | 1 | -10,01 : | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011\,ulp(x)$ et $|x - 1,101| = 0,101\,ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011\,ulp(x)$ et $|x - 1,101| = 0,101\,ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011\,ulp(x)$ et $|x - 1,101| = 0,101\,ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

1. Est-ce que la multiplication des VF est associative ?

```
Prelude> (1.00000001 * 1e-15) * 1e15
1.0000000100000002
Prelude>  1.00000001 * (1e-15 * 1e15)
1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

```
Prelude>  1e15 * (1e-16 + 1)
1.0e15
Prelude> (1e15 * 1e-16) + (1e15 * 1)
1.0000000000000001e15
```

1. Est-ce que la multiplication des VF est associative ?

```
Prelude> (1.00000001 * 1e-15) * 1e15
1.0000000100000002
Prelude>  1.00000001 * (1e-15 * 1e15)
1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

```
Prelude>  1e15 * (1e-16 + 1)
1.0e15
Prelude> (1e15 * 1e-16) + (1e15 * 1)
1.0000000000000001e15
```

1. Est-ce que la multiplication des VF est associative ?

```
Prelude> (1.00000001 * 1e-15) * 1e15
1.0000000100000002
Prelude>  1.00000001 * (1e-15 * 1e15)
1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

```
Prelude>  1e15 * (1e-16 + 1)
1.0e15
Prelude> (1e15 * 1e-16) + (1e15 * 1)
1.0000000000000001e15
```

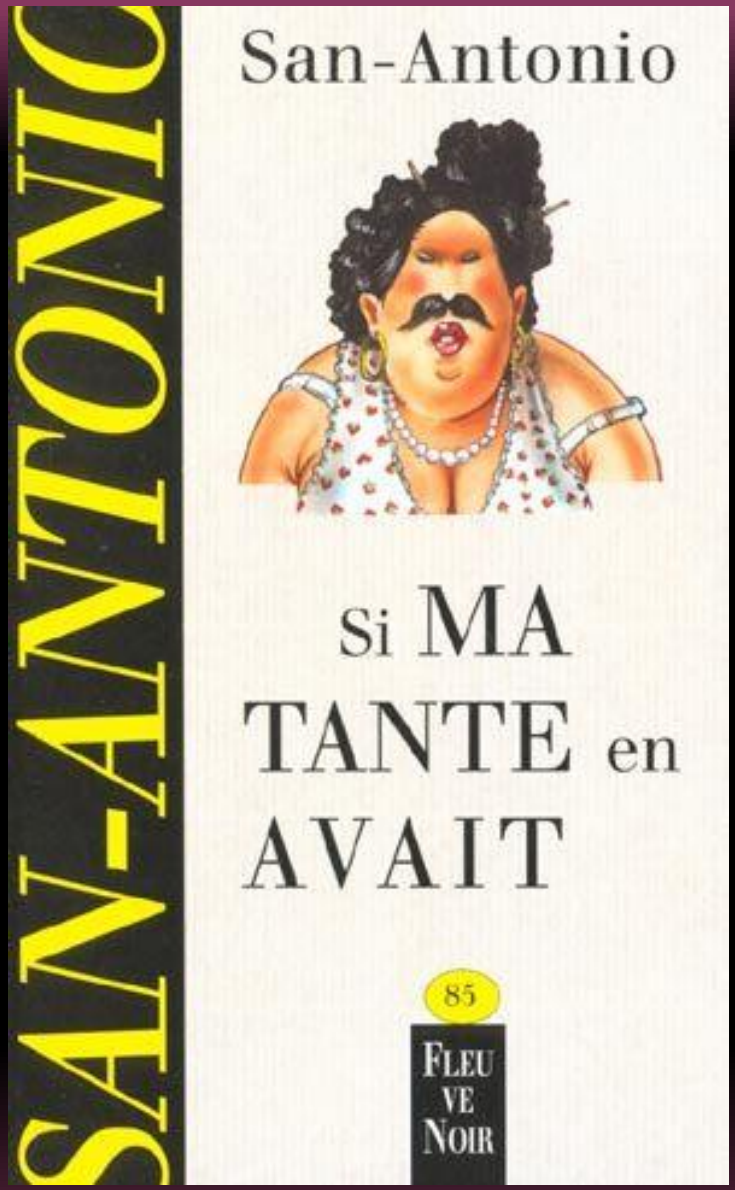
Sommaire

```haskell
f1 a =
  if (a + 1.0) - a /= 1.0 then a
  else f1 (2.0 * a)

f2 a b =
  if (a + b) - a == b then b
  else f2 a (b + 1.0)

b = f2 (f1 1) 1
```

????!!!????!!???

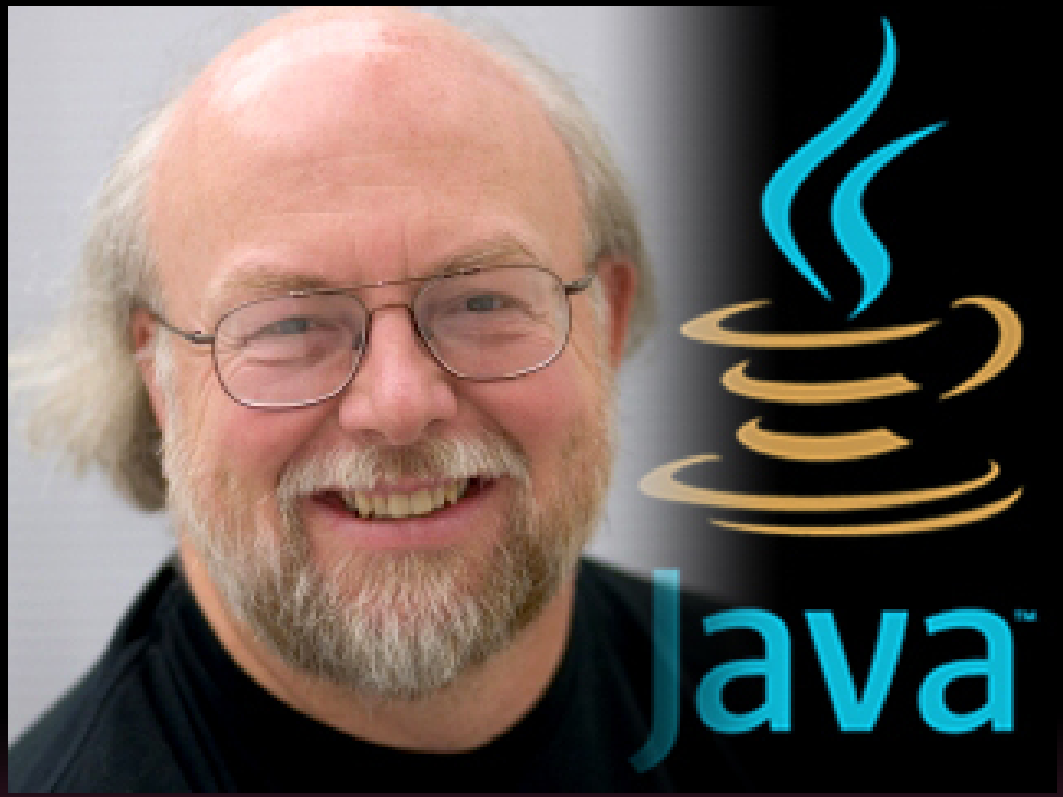
```haskell
f1 a =
  if (a + 1.0) - a /= 1.0 then a
  else f1 (2.0 * a)

f2 a b =
  if (a + b) - a == b then b
  else f2 a (b + 1.0)

b = f2 (f1 1) 1
```

????!!!!????!!???


SAN-ANTONIO
San-Antonio
Si MA TANTE en AVAIT
85
FLEUVE NOIR

95 % of the folks out there are completely clueless about floating-point

James Gosling *(M. Java) - 28 février 1998*




LA COLLECTION
RYAN GOSLING
DRIVE // LES MARCHES DU POUVOIR // CRAZY STUPID LOVE
Blu-ray Disc
LA COLLECTION
RYAN GOSLING
DRIVE // LES MARCHES DU POUVOIR // CRAZY STUPID LOVE

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple 3, 3.0, 3.0000, 3.0e0 n'ont pas la même précision, précision qui dépend en plus du langage.

3,1777777777777 est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

`3.1777777777777` est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple 3, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,17777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,17777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple 3, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,177777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple 3, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,177777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x - \widehat{x}}{x}$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1-\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

$r = f(d)$

Définition 2

- L'erreur *aval* (*forward error*) est la différence entre le résultat mesuré $\widehat{r}$ et le résultat théorique r.
- L'erreur *amont*, ou erreur *inverse* (*backward error*) est le plus petit $|\Delta d|$ tel que $f(d + \Delta d) = \widehat{r}$.

$r = f(d)$

Définition 2

- L'erreur *aval* (*forward error*) est la différence entre le résultat mesuré $\widehat{r}$ et le résultat théorique r.
- L'erreur *amont*, ou erreur *inverse* (*backward error*) est le plus petit $|\Delta d|$ tel que $f(d + \Delta d) = \widehat{r}$.

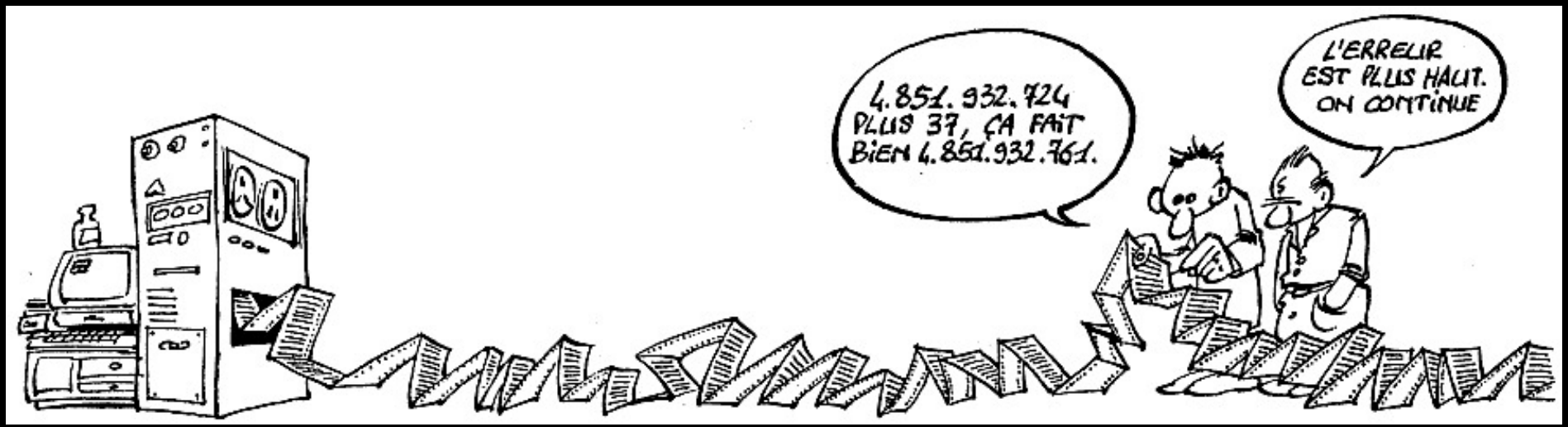
$r = f(d)$

Définition 2

- L'erreur *aval* (*forward error*) est la différence entre le résultat mesuré $\widehat{r}$ et le résultat théorique r.
- L'erreur *amont*, ou erreur *inverse* (*backward error*) est le plus petit $|\Delta d|$ tel que $f(d + \Delta d) = \widehat{r}$.

Feel nervous, but feel in control. It's not dark magic, it's science.

Florent DE DINECHIN



Sommaire

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

- Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

- Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} \leqslant \frac{1}{2}\frac{1}{m}$$

- Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

- Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

- Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

- Si $\widehat{x} \in \mathbb{F}$, alors l'erreur relative est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
- Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\mathbf{min}}-1-n}$

NE FAITES PAS DES TESTS D'ÉGALITÉ MAIS DES TESTS D'APPARTENANCE À DES INTERVALLES DE LARGEUR L'ε MACHINE !

Précision machine

$$\widehat{x} = x(1 + \eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\mathbf{min}}-1-n}$

Mouais...

Précision machine

$$\widehat{x} = x(1 + \eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\mathbf{min}}-1-n}$

Mouais...

Précision machine

$$\widehat{x} = x(1 + \eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\mathbf{min}}-1-n}$

Mouais...

Précision machine

$$\widehat{x} = x(1 + \eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\mathbf{min}}-1-n}$

Mouais...

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a - b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Catastrophe

```haskell
deriv :: (Float -> Float) -> Float -> Float -> Float
deriv     f                  x        h     =
    ( f(x + h) - f(x) ) / h
```

```haskell
derivn :: (Float -> Float) -> Float -> Float -> Int -> Float
derivn    f                  x        _       0   =  f x
derivn    f                  x        h       n   =
    derivn (\ t -> deriv f t h) x h (n - 1)
```

Catastrophe

```haskell
deriv :: (Float -> Float) -> Float -> Float -> Float
deriv     f                  x        h     =
    ( f(x + h) - f(x) ) / h
```

```haskell
derivn :: (Float -> Float) -> Float -> Float -> Int -> Float
derivn    f                  x        _       0   =  f x
derivn    f                  x        h       n   =
    derivn (\ t -> deriv f t h) x h (n - 1)
```

Catastrophe

```
λ> [ [ derivn (\ t -> t**4) 1 (10**(-d)) k | k <- [0..5]] | d <-
      ↪  [1..8]]
```

```
[[1.0,4.641001,14.540005,27.60004,24.001312,-3.6239624e-2],
[1.0,4.0603995,12.24041,24.557114,0.0,1192.0929],
[1.0,4.0061474,12.159347,-238.41856,476837.1,-8.3446483e8],
[1.0,4.001856,0.0,238418.58,-4.7683717e9,9.536743e13],
[1.0,4.005432,0.0,0.0,0.0,0.0],
[1.0,3.8146973,0.0,0.0,0.0,0.0],
[1.0,4.7683716,0.0,0.0,0.0,0.0],
[1.0,0.0,0.0,0.0,0.0,0.0]]
```

Catastrophe

```
λ> [ [ derivn (\ t -> t**4) 1 (10**(-d)) k | k <- [0..5]] | d <-
      ↪  [1..8]]
```

```
[[1.0,4.641001,14.540005,27.60004,24.001312,-3.6239624e-2],
[1.0,4.0603995,12.24041,24.557114,0.0,1192.0929],
[1.0,4.0061474,12.159347,-238.41856,476837.1,-8.3446483e8],
[1.0,4.001856,0.0,238418.58,-4.7683717e9,9.536743e13],
[1.0,4.005432,0.0,0.0,0.0,0.0],
[1.0,3.8146973,0.0,0.0,0.0,0.0],
[1.0,4.7683716,0.0,0.0,0.0,0.0],
[1.0,0.0,0.0,0.0,0.0,0.0]]
```

Catastrophe

```haskell
deriv :: (Double -> Double) -> Double -> Double -> Double
deriv     f                   x        h     =
    ( f(x + h) - f(x) ) / h
```

```haskell
derivn :: (Double -> Double) -> Double -> Double -> Int -> Double
derivn    f                   x        _        0    =  f x
derivn    f                   x        h        n    =
    derivn (\ t -> deriv f t h) x h (n - 1)
```

Catastrophe

```haskell
deriv :: (Double -> Double) -> Double -> Double -> Double
deriv     f                   x        h     =
    ( f(x + h) - f(x) ) / h
```

```haskell
derivn :: (Double -> Double) -> Double -> Double -> Int -> Double
derivn    f                   x        _        0   =  f x
derivn    f                   x        h        n   =
    derivn (\ t -> deriv f t h) x h (n - 1)
```

Catastrophe

```
λ> [ [ derivn (\ t -> t**4) 1 (10**(-d)) k  | k <- [0..5]] | d <-
        ↪  [1..8]]
[
[1.0,4.641000000000004
        ↪  ,14.540000000000042,27.599999999999874,24.000000000002,1.06581
[1.0,4.060401000000002
        ↪  ,12.241399999999292,24.360000000278603,23.99999994295854,8.8817
[1.0,4.006004000999486 ,12.024014000244776,24.0359941303014
        ↪  ,24.001245435556484,-2.442490654175344 4],
[1.0,4.000600039999469 ,12.002400162636206,24.00324383700081
        ↪  ,28.865798640254 07,-66613.38147750939],
[1.0,4.000060004308495,12.000238580611 72 ,24.20286193682841
        ↪  ,-22204.46049250313,2.220446049250313e9],
[1.0,4.000005999760248
        ↪  ,11.999956583963467,0.0,2.220446049250313e8,-6.661338147750 94e1
[1.0,4.000000601855902
        ↪  ,12.012613126444194,-222044.6049250313,2.220446049250313e12,0.0
[1.0,4.000000042303498
        ↪  ,11.102230246251565,0.0,2.220446049250 3128e16,-4.440892098500 62
]
```

Catastrophe

```
λ> [ [ derivn (\ t -> t**4) 1 (2**(-d)) k  | k <- [0..5]] | d <-
        ↪  [1..15]]
[
[1.0,8.125,27.5,42.0,24.0,0.0],
[1.0,5.765625,18.875,33.0,24.0,0.0],
[1.0,4.814453125,15.21875,28.5,24.0,0.0],
[1.0,4.390869140625,13.5546875,26.25,24.0,0.0],
[1.0,4.191436767578125,12.763671875,25.125,24.0,0.0],
[1.0,4.094730377197266,12.37841796875,24.5625,24.0,0.0],
[1.0,4.047119617462158,12.1883544921875,24.28125,24.0,0.0],
[1.0,4.023498594760895,12.093963623046875,24.140625,24.0,0.0]
[1.0,4.011734016239643,12.046928405761719,24.0703125,24.0,0.0],
[1.0,4.005863190628588,12.02345085144043,24.03515625,24.0,0.0],
[1.0,4.002930641290732,12.011722087860107,24.017578125,24.0,0.0],
[1.0,4.001465082183131,12.005860209465027,24.0087890625,24.0,0.0],
[1.0,4.000732481481464,12.002929896116257,24.00439453125,24.0,0.0],
[1.0,4.000366225838661,12.001464903354645,24.001953125,32.0,-262144.0],
[1.0,4.000183109194044,12.000732421875,24.0,256.0,-2.5165824e7]]
]
```

TROP DE PRÉCISION TUE LA PRÉCISION !

DIVISEZ PAR DES PUISSANCES DE 2...PAS DES PUISSANCES DE 10 !

TROP DE PRÉCISION TUE LA PRÉCISION !
DIVISEZ PAR DES PUISSANCES DE 2...PAS DES PUISSANCES DE 10 !

TROP DE PRÉCISION TUE LA PRÉCISION !
DIVISEZ PAR DES PUISSANCES DE 2...PAS DES PUISSANCES DE 10 !

Recherche 1

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*
4. *Et dans le cas de $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$?*
5. *Macalité ?*

Recherche 1

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*
4. *Et dans le cas de $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$?*
5. *Maculité ?*

Recherche 1

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ *avec* $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b *et* c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque* $b^2 \gg |4ac|$ *? En quoi la formule* $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ *peut aider ?*
2. *Que se passe-t-il lorsque* $b^2 \approx 4ac$ *? Peut-on y remédier ? Que peut-on dire de* Δ *par rapport à* b^2 *? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation* $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$ *?*
4. *Et dans le cas de* $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$ *?*
5. *Moralité ?*

Recherche 1

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ *avec* $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b *et* c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque* $b^2 \gg |4ac|$ *? En quoi la formule* $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ *peut aider ?*
2. *Que se passe-t-il lorsque* $b^2 \approx 4ac$ *? Peut-on y remédier ? Que peut-on dire de* Δ *par rapport à* b^2 *? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation* $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$ *?*
4. *Et dans le cas de* $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$ *?*
5. *Moralité ?*

Recherche 1

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ *avec* $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b *et* c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque* $b^2 \gg |4ac|$ *? En quoi la formule* $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ *peut aider ?*
2. *Que se passe-t-il lorsque* $b^2 \approx 4ac$ *? Peut-on y remédier ? Que peut-on dire de* Δ *par rapport à* b^2 *? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation* $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$ *?*
4. *Et dans le cas de* $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$ *?*
5. *Moralité ?*

Recherche 1

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ *avec* $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b *et* c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque* $b^2 \gg |4ac|$ *? En quoi la formule* $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ *peut aider ?*
2. *Que se passe-t-il lorsque* $b^2 \approx 4ac$ *? Peut-on y remédier ? Que peut-on dire de* Δ *par rapport à* b^2 *? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation* $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$ *?*
4. *Et dans le cas de* $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$ *?*
5. *Moralité ?*

Recherche 2 (Max IEEE)

Fonction `max(` x, y : *flottants* `)` : *flottant*

Si $x >= y$ Alors

| max = x

Sinon

| max = y

FinSi

```
λ> max 1 (1 + 1e-16)
1.0
λ> max 0.1 (1e500 / 1e500)
0.1
λ> 0.1 <= 1e500/1e500
False
λ> 0.1 > 1e500/1e500
False
```

JavaScript

JavaScript

```
      1.7          3 2012 05 18

js> Math.pow(2, 53)
9007199254740992
js> Math.pow(2, 53) + 1
9007199254740992
js> Math.pow(2, 53) + 2
9007199254740994
js> Math.pow(2, 53) + 3
9007199254740996
js> Math.pow(2, 53) + 4
9007199254740996
js> Math.pow(2, 53) + 5
9007199254740996
```

JavaScript

```
js> Math.pow(2, 54)
18014398509481984
js> Math.pow(2, 54) + 2
18014398509481984
js> Math.pow(2, 54) + 3
18014398509481988



js> Math.pow(2, 55)
36028797018963970
Math.pow(2, 55) + 4
36028797018963970
```

JavaScript

JavaScript

```
      1.7          3 2012 05 18

js> Math.pow(2, 53)
9007199254740992
js> Math.pow(2, 53) + 1
9007199254740992
js> Math.pow(2, 53) + 2
9007199254740994
js> Math.pow(2, 53) + 3
9007199254740996
js> Math.pow(2, 53) + 4
9007199254740996
js> Math.pow(2, 53) + 5
9007199254740996
```

JavaScript

```
js> Math.pow(2, 54)
18014398509481984
js> Math.pow(2, 54) + 2
18014398509481984
js> Math.pow(2, 54) + 3
18014398509481988



js> Math.pow(2, 55)
36028797018963970
Math.pow(2, 55) + 4
36028797018963970
```

```
1  Algorithme de Malcolm-Gentleman
2  a ← 1.0
3  b ← 1.0
4  TantQue R(R(a + 1.0) − a) == 1.0 Faire
5  |  a ← R(2 × a)
6  FinTantQue
7  TantQue R(R(a + b) − a) ≠ b Faire
8  |  b ← b + 1
9  FinTantQue
10 Retourner b
```

```
1  Algorithme de Malcolm-Gentleman
2  a ← 1.0
3  b ← 1.0
4  TantQue R(R(a + 1.0) − a) == 1.0 Faire
5  |  a ← R(2 × a)
6  FinTantQue
7  TantQue R(R(a + b) − a) ≠ b Faire
8  |  b ← b + 1
9  FinTantQue
10 Retourner b
```

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

$a_{i+1} = 2a_i \quad a_0 = 1.0 \quad a_i = 2^i$ ... Mouais

Dans quel espace tu travailles tu me diras et si ta proposition est vraie je saurai

Mathemator - 44 ap. GC

En fait $a_{i+1} = K(2 \times a_i)$...

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

$a_{i+1} = 2a_i$ $a_0 = 1.0$ $a_i = 2^i$ .... Mouais

Dans quel espace tu travailles tu me diras et si ta proposition est vraie je saurai

Mathemator - 44 ap. GC

En fait $a_{i+1} = K(2 \times a_i)$...

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

$a_{i+1} = 2a_i \quad a_0 = 1.0 \quad a_i = 2^i$ .... Mouais

Dans quel espace tu travailles tu me diras et si ta proposition est vraie je saurai

Mathemator - 44 ap. GC

En fait $a_{i+1} = K(2 \times a_i)$...

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

$a_{i+1} = 2a_i \quad a_0 = 1.0 \quad a_i = 2^i$ .... Mouais

Dans quel espace tu travailles tu me diras et si ta proposition est vraie je saurai

Mathemator - 44 ap. GC

En fait $a_{i+1} = R(2 \times a_i)$...

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

$a_{i+1} = 2a_i \quad a_0 = 1.0 \quad a_i = 2^i$ .... Mouais

Dans quel espace tu travailles tu me diras et si ta proposition est vraie je saurai

Mathemator - 44 ap. GC

En fait $a_{i+1} = R(2 \times a_i)$...

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

$a_{i+1} = 2a_i \quad a_0 = 1.0 \quad a_i = 2^i$ .... Mouais

Dans quel espace tu travailles tu me diras et si ta proposition est vraie je saurai

Mathemator - 44 ap. GC

En fait $a_{i+1} = R(2 \times a_i)$...

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

$a_{i+1} = 2a_i \quad a_0 = 1.0 \quad a_i = 2^i$ .... Mouais

Dans quel espace tu travailles tu me diras et si ta proposition est vraie je saurai

Mathemator - 44 ap. GC

En fait $a_{i+1} = R(2 \times a_i)$...

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$
4. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$ La condition de la première boucle est donc vérifiée tant que $2^i \leqslant \beta^p - 1$.
4. Première itération i telle que $2^i > \beta^p - 1$.
5. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_p telle que $2^{i_p} \geqslant \beta^p$.
5. $a_{i_p} = 2^{i_p}$.
6. $R(a_{i_p} + 1.0) = a_{i_p} + 1.0$: pas d'arrondi.
7. $R(R(a_{i_p} + 1.0) - a_{i_p}) = R(a_{i_p} + 1.0 - a_{i_p}) = 1.0$: la condition de la première boucle est vérifiée.
8. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s - 1})$
6. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s - 1})$ $R(2 \times (2^{i_s - 1}))$ $= 2^{i_s}$
6. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s - 1}) = R(2 \times 2^{i_s - 1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s - 1}) = R(2 \times 2^{i_s - 1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s - 1}) = R(2 \times 2^{i_s - 1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$

6. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p < a_{i_s} < \beta^{p+1}$
7. ...
8. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p \leqslant a_{i_s} < \beta^{p+1}$
7. $succ(a_{i_s}) = a_{i_s} + \beta^{E(a_{i_s} - p + 1)}$
8. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p \leqslant a_{i_s} < \beta^{p+1}$
7. $\mathrm{succ}(a_{i_s}) = a_{i_s} + \beta^{E(a_{i_s}-p+1)}$
8. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p \leqslant a_{i_s} < \beta^{p+1}$
7. $\mathrm{succ}(a_{i_s}) = a_{i_s} + \beta^{E(a_{i_s}-p+1)} = a_{i_s} + \beta^{p-p+1} = a_{i_s} + \beta$
8. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p \leqslant a_{i_s} < \beta^{p+1}$
7. $\text{succ}(a_{i_s}) = a_{i_s} + \beta^{E(a_{i_s}-p+1)} = a_{i_s} + \beta^{p-p+1} = a_{i_s} + \beta$
8. $R(a_{i_s} + 1.0)$ vaut a_{i_s} ou $a_{i_s} + \beta$
9. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p \leqslant a_{i_s} < \beta^{p+1}$
7. $\mathrm{succ}(a_{i_s}) = a_{i_s} + \beta^{E(a_{i_s}-p+1)} = a_{i_s} + \beta^{p-p+1} = a_{i_s} + \beta$
8. $R(a_{i_s} + 1.0)$ vaut a_{i_s} ou $a_{i_s} + \beta$
9. $R(R(a_{i_s} + 1.0) - a_{i_s})$ vaut 0 ou β mais en aucun cas 1.0
10. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p \leqslant a_{i_s} < \beta^{p+1}$
7. $\mathrm{succ}(a_{i_s}) = a_{i_s} + \beta^{E(a_{i_s}-p+1)} = a_{i_s} + \beta^{p-p+1} = a_{i_s} + \beta$
8. $R(a_{i_s} + 1.0)$ vaut a_{i_s} ou $a_{i_s} + \beta$
9. $R(R(a_{i_s} + 1.0) - a_{i_s})$ vaut 0 ou β mais en aucun cas 1.0.
10. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p \leqslant a_{i_s} < \beta^{p+1}$
7. $\mathrm{succ}(a_{i_s}) = a_{i_s} + \beta^{E(a_{i_s}-p+1)} = a_{i_s} + \beta^{p-p+1} = a_{i_s} + \beta$
8. $R(a_{i_s} + 1.0)$ vaut a_{i_s} ou $a_{i_s} + \beta$
9. $R(R(a_{i_s} + 1.0) - a_{i_s})$ vaut 0 ou β mais en aucun cas 1.0.
10. on sort donc de la boucle.

1. Par récurrence : $a_i = R(2^i) = 2^i$ tant que i vérifie $2^i \leqslant \beta^p - 1$.
2. Dans ce cas $R(a_i + 1.0) = a_i + 1.0$: pas d'arrondi.
3. $R(R(a_i + 1.0) - a_i) = R(a_i + 1.0 - a_i) = 1.0$: la condition de la première boucle est donc vérifiée tant que $2^i < \beta^p$.
4. Première itération i_s telle que $2^{i_s} \geqslant \beta^p$.
5. $a_{i_s} = R(2 \times a_{i_s-1}) = R(2 \times 2^{i_s-1}) < R(2 \times \beta^p) \leqslant R(\beta \times \beta^p) = \beta^{p+1}$
6. $\beta^p \leqslant a_{i_s} < \beta^{p+1}$
7. $\mathrm{succ}(a_{i_s}) = a_{i_s} + \beta^{E(a_{i_s}-p+1)} = a_{i_s} + \beta^{p-p+1} = a_{i_s} + \beta$
8. $R(a_{i_s} + 1.0)$ vaut a_{i_s} ou $a_{i_s} + \beta$
9. $R(R(a_{i_s} + 1.0) - a_{i_s})$ vaut 0 ou β mais en aucun cas 1.0.
10. on sort donc de la boucle.

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

Notons `a = boucle1 1`

Son successeur est $a + \beta$.

Tant que $b < \beta$..

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

Notons `a = boucle1 1`

Son successeur est $a + \beta$.

Tant que $b < \beta$..

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

Notons `a = boucle1 1`

Son successeur est $a + \beta$.

Tant que $b < \beta$..

```haskell
base = boucle2 (boucle1 1) 1
  where
    boucle1 a =
      if (a + 1.0) - a /= 1.0 then a
      else boucle1 (2.0 * a)
    boucle2 a b =
      if (a + b) - a == b then b
      else boucle2 a (b + 1.0)
```

Notons `a = boucle1 1`

Son successeur est $a + \beta$.

Tant que $b < \beta$..

1. elle nous montre bien que nous changeons de monde : un test du style `if (a + 1.0)- a /= 1.0` paraîtrait bien hors de propos si nous raisonnions avec des réels ;
2. nous avons vu que l'on pouvait raisonner rigoureusement sur les VF ;
3. nous pouvons malgré tout retenir la trame intuitive de la démonstration : tant que notre nombre entier est représentable avec p chiffres, on reste exact. Les problèmes arrivent lorsqu'on n'a plus assez de place pour stocker tous les chiffres : il y a de la perte d'information...
4. Ce genre de raisonnement se retrouve très souvent pour travailler sur les erreurs commises : il faudra donc en retenir la substantifique moelle...

1. elle nous montre bien que nous changeons de monde : un test du style `if (a + 1.0)- a /= 1.0` paraîtrait bien hors de propos si nous raisonnions avec des réels ;
2. nous avons vu que l'on pouvait raisonner rigoureusement sur les VF ;
3. nous pouvons malgré tout retenir la trame intuitive de la démonstration : tant que notre nombre entier est représentable avec p chiffres, on reste exact. Les problèmes arrivent lorsqu'on n'a plus assez de place pour stocker tous les chiffres : il y a de la perte d'information...
4. Ce genre de raisonnement se retrouve très souvent pour travailler sur les erreurs commises : il faudra donc en retenir la substantifique moelle...

1. elle nous montre bien que nous changeons de monde : un test du style `if (a + 1.0)- a /= 1.0` paraîtrait bien hors de propos si nous raisonnions avec des réels ;
2. nous avons vu que l'on pouvait raisonner rigoureusement sur les VF ;
3. nous pouvons malgré tout retenir la trame intuitive de la démonstration : tant que notre nombre entier est représentable avec p chiffres, on reste exact. Les problèmes arrivent lorsqu'on n'a plus assez de place pour stocker tous les chiffres : il y a de la perte d'information...
4. Ce genre de raisonnement se retrouve très souvent pour travailler sur les erreurs commises : il faudra donc en retenir la substantifique moelle...

1. elle nous montre bien que nous changeons de monde : un test du style `if (a + 1.0)- a /= 1.0` paraîtrait bien hors de propos si nous raisonnions avec des réels ;
2. nous avons vu que l'on pouvait raisonner rigoureusement sur les VF ;
3. nous pouvons malgré tout retenir la trame intuitive de la démonstration : tant que notre nombre entier est représentable avec p chiffres, on reste exact. Les problèmes arrivent lorsqu'on n'a plus assez de place pour stocker tous les chiffres : il y a de la perte d'information...
4. Ce genre de raisonnement se retrouve très souvent pour travailler sur les erreurs commises : il faudra donc en retenir la substantifique moelle...

1. elle nous montre bien que nous changeons de monde : un test du style `if (a + 1.0)- a /= 1.0` paraîtrait bien hors de propos si nous raisonnions avec des réels ;
2. nous avons vu que l'on pouvait raisonner rigoureusement sur les VF ;
3. nous pouvons malgré tout retenir la trame intuitive de la démonstration : tant que notre nombre entier est représentable avec p chiffres, on reste exact. Les problèmes arrivent lorsqu'on n'a plus assez de place pour stocker tous les chiffres : il y a de la perte d'information...
4. Ce genre de raisonnement se retrouve très souvent pour travailler sur les erreurs commises : il faudra donc en retenir la substantifique moelle...

Nouvelles opérations

- $R(x+y)$ sous la forme $x \oplus y$;
- $R(x-y)$ sous la forme $x \ominus y$;
- $R(x \times y)$ sous la forme $x \otimes y$;
- $R(x/y)$ sous la forme $x \oslash y$.

Nouvelles opérations

- $R(x+y)$ sous la forme $x \oplus y$;
- $R(x-y)$ sous la forme $x \ominus y$;
- $R(x \times y)$ sous la forme $x \otimes y$;
- $R(x/y)$ sous la forme $x \oslash y$.

Nouvelles opérations

- $R(x+y)$ sous la forme $x \oplus y$;
- $R(x-y)$ sous la forme $x \ominus y$;
- $R(x \times y)$ sous la forme $x \otimes y$;
- $R(x/y)$ sous la forme $x \oslash y$.

Nouvelles opérations

- $R(x+y)$ sous la forme $x \oplus y$;
- $R(x-y)$ sous la forme $x \ominus y$;
- $R(x \times y)$ sous la forme $x \otimes y$;
- $R(x/y)$ sous la forme $x \oslash y$.

Trouver la précision

M.A. Malcolm 1972 :

```haskell
precision = toPrec 1.0 0
  where
    toPrec a i =
      if (a + 1.0) - a /= 1.0 then i
      else toPrec (base * a) (i + 1)
```

????

Trouver la précision

M.A. Malcolm 1972 :

```haskell
precision = toPrec 1.0 0
  where
    toPrec a i =
      if (a + 1.0) - a /= 1.0 then i
      else toPrec (base * a) (i + 1)
```

????

Attention !

Pour éviter de déduire des lemmes suivants des théorèmes totalement faux, n'oubliez pas que DANS CE QUI SUIT X ET Y SONT DES NOMBRES À VIRGULE FLOTTANTE !

Lemme 3 (Majoration de l'erreur d'une somme)

Posons $x \oplus y = x + y + \mathsf{err}(x \oplus y)$. *Alors, s'il n'y a pas de dépassement de capacité,*

$$|\mathsf{err}(x \oplus y)| \leqslant \min(|x|, |y|)$$

On a bien sûr un résultat analogue pour la différence

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

FAITES UN DESSIN !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

FAITES UN DESSIN !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

FAITES UN DESSIN !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

FAITES UN DESSIN !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

FAITES UN DESSIN !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

FAITES UN DESSIN !

À noter :

De l'importance de disposer avec la IEEE 754 de la meilleure approximation !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

FAITES UN DESSIN !

À noter

De l'importance de disposer avec la IEEE 754 de la meilleure approximation !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

FAITES UN DESSIN !

À noter

De l'importance de disposer avec la IEEE 754 de la **meilleure approximation** !

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathsf{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF
- $\mathsf{err}(x \oplus y) = x \otimes y \ominus (x \oplus y)$

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

⇒ $|\mathrm{err}(x \oplus y)| \leqslant |y|$

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$ donc la mantisse entière de $\mathrm{err}(x \oplus y)$ est une fraction inférieure à p bits

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$ donc la mantisse entière de $\mathrm{err}(x \oplus y)$ a une longueur inférieure à p bits

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$donc la mantisse entière de $\mathrm{err}(x \oplus y)$ a une longueur inférieure à p bits

Corollaire 4 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$donc la mantisse entière de $\mathrm{err}(x \oplus y)$ a une longueur inférieure à p bits

Lemme 5

Supposons que $|x + y| \leqslant \min(|x|, |y|)$*, alors* $x \oplus y = x + y$*.*
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x + y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x + y| \leqslant |y|$ donc la mantisse entière de $x + y$ a une longueur inférieure à p bits

Lemme 5

Supposons que $|x+y| \leqslant \min(|x|,|y|)$*, alors* $x \oplus y = x + y$*.*
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x+y| \leqslant |y|$ donc la mantisse entière de $x+y$ a une longueur inférieure à p bits

Lemme 5

Supposons que $|x+y| \leqslant \min(|x|,|y|)$*, alors* $x \oplus y = x + y$*.*
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x+y| \leqslant |y|$ donc la mantisse entière de $x+y$ a une longueur inférieure à p bits

Lemme 5

Supposons que $|x+y| \leqslant \min(|x|,|y|)$*, alors* $x \oplus y = x + y$*.*
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x+y| \leqslant |y|$ donc la mantisse entière de $x+y$ a une longueur inférieure à p bits

DANGER

Nous avons démontré nos lemmes en considérant des VF x et y.
Est-ce que le résultat suivant contredit notre dernier lemme ?

```
*Main> 1 - 0.9
9.999999999999998e-2
```

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

Si $x < y$: alors $x \leqslant y \leqslant 2x$ donc $0 \leqslant y - x \leqslant x \leqslant y$.

Si $y < x$: ...

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 5 page 346.

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $x \leqslant y$: alors $x \leqslant y \leqslant 2x$ donc $0 \leqslant y - x \leqslant x \leqslant y$;
2. $x \geqslant y$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 5 page 346.

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 5 page 346.

Lemme 6 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 5 page 346.

- Concrètement, à quoi correspond cette condition $\frac{x}{2} \leqslant y \leqslant 2x$?
- Demandez-vous ce que l'on peut dire de l'écart maximum entre les exposants de x et y.

- Concrètement, à quoi correspond cette condition $\frac{x}{2} \leqslant y \leqslant 2x$?
- Demandez-vous ce que l'on peut dire de l'écart maximum entre les exposants de x et y.

Théorème 7 (Fast2Sum - Dekker & Kahan)

On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

```
1  s  ← x ⊕ y
2  y_v ← s ⊖ x
3  d  ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = \mathrm{err}(x \oplus y)$. De plus s et d ne se chevauchent pas.

- si x et y sont de même signe OU si $|y| \leq \frac{|x|}{2}$ alors $\frac{x}{2} \leq s \leq 2x$ et on peut appliquer le lemme ;
- **sinon** : on a x et y de signes opposés ET $y > \frac{|x|}{2}$ alors si y est négatif $x/2 < -y < x$ et sinon $-x/2 < y < -x$. Dans les deux cas, d'après le lemme de Sterbenz, s est calculée exactement et alors $y_v = y$.

Théorème 7 (Fast2Sum - Dekker & Kahan)

On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

```
1  s  ← x ⊕ y
2  y_v ← s ⊖ x
3  d  ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = \mathrm{err}(x \oplus y)$. De plus s et d ne se chevauchent pas.

- **si x et y sont de même signe OU si $|y| \leqslant \frac{|x|}{2}$** : alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme ;
- **sinon** : on a x et y de signes opposés ET $y > \frac{|x|}{2}$ alors si y est négatif $x/2 < -y < x$ et sinon $-x/2 < y < -x$. Dans les deux cas, d'après le lemme de Sterbenz, s est calculée exactement et alors $y_v = y$.

Théorème 7 (Fast2Sum - Dekker & Kahan)

On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

```
1  s  ← x ⊕ y
2  y_v ← s ⊖ x
3  d  ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = \mathrm{err}(x \oplus y)$. De plus s et d ne se chevauchent pas.

- **si x et y sont de même signe OU si $|y| \leqslant \frac{|x|}{2}$** : alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme ;
- **sinon** : on a x et y de signes opposés ET $y > \frac{|x|}{2}$ alors si y est négatif $x/2 < -y < x$ et sinon $-x/2 < y < -x$. Dans les deux cas, d'après le lemme de Sterbenz, s est calculée exactement et alors $y_v = y$

Théorème 7 (Fast2Sum - Dekker & Kahan)

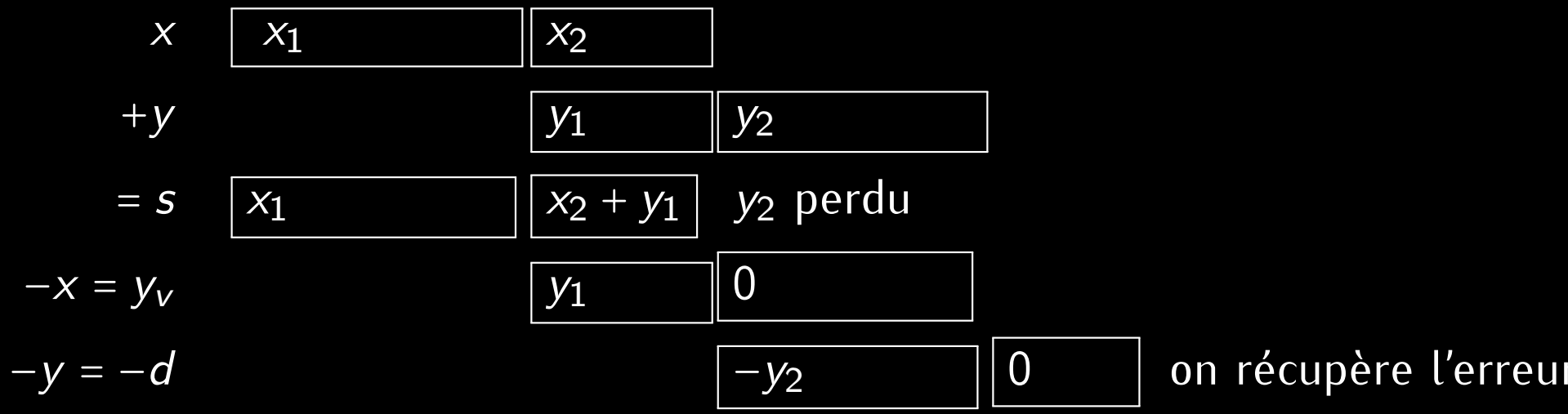
On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

```
1  s  ← x ⊕ y
2  y_v ← s ⊖ x
3  d  ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = \mathrm{err}(x \oplus y)$. De plus s et d ne se chevauchent pas.

- **si x et y sont de même signe OU si $|y| \leqslant \frac{|x|}{2}$** : alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme ;
- **sinon** : on a x et y de signes opposés ET $y > \frac{|x|}{2}$ alors si y est négatif $x/2 < -y < x$ et sinon $-x/2 < y < -x$. Dans les deux cas, d'après le lemme de Sterbenz, s est calculée exactement et alors $y_v = y$.

Somme compensée



```haskell
fastTwoSum x y
    | abs x >= abs y =
      let s  = x + y   in
      let yv = s - x  in
      let d  = y - yv in
      (s, d)
    | otherwise =  fastTwoSum y x
```

Théorème 8 (2Sum - Knuth)

On considère deux VF x et y et l'algorithme suivant :

1 $s \leftarrow x \oplus y$

2 $y_v \leftarrow s \ominus x$

3 $x_v \leftarrow s \ominus y_v$

4 $y_a \leftarrow y \ominus y_v$

5 $x_a \leftarrow x \ominus x_v$

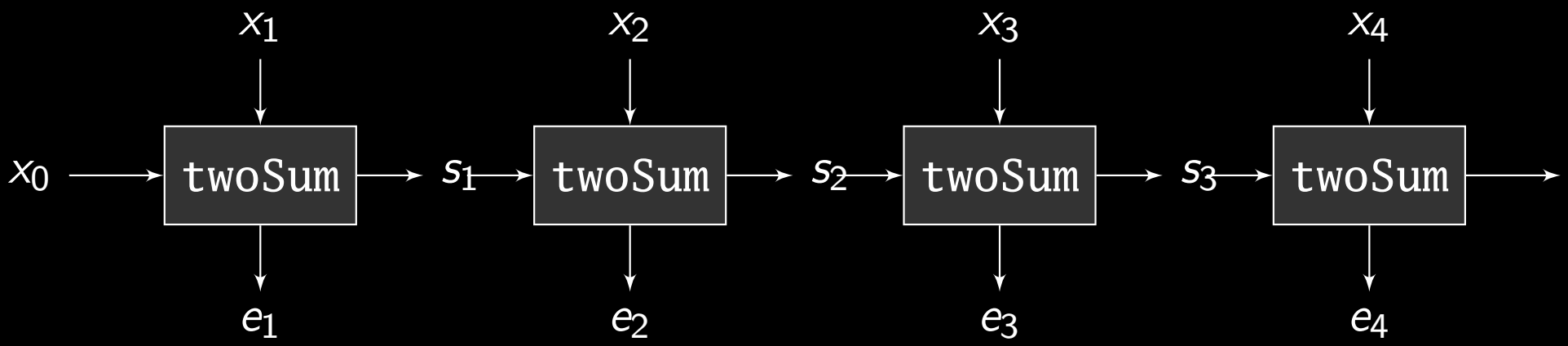
6 $d \leftarrow x_a \oplus y_a$

7 Retourner (s, d)

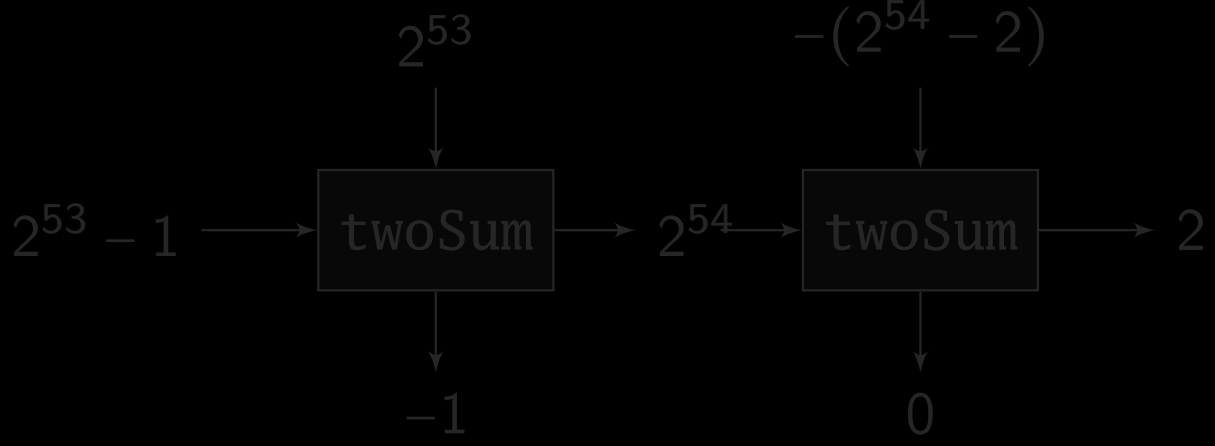
Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = \mathrm{err}(x \oplus y)$. De plus s et d ne se chevauchent pas.

Somme compensée d'un nombre quelconque de flottants

```
sommeKahan liste =
  let (s,e) = foldl (\ (s,c) x -> twoSum s (x + c)) (0,0) liste in s
```


x_1
x_2
x_3
x_4
x_0
twoSum
s_1
twoSum
s_2
twoSum
s_3
twoSum
e_1
e_2
e_3
e_4

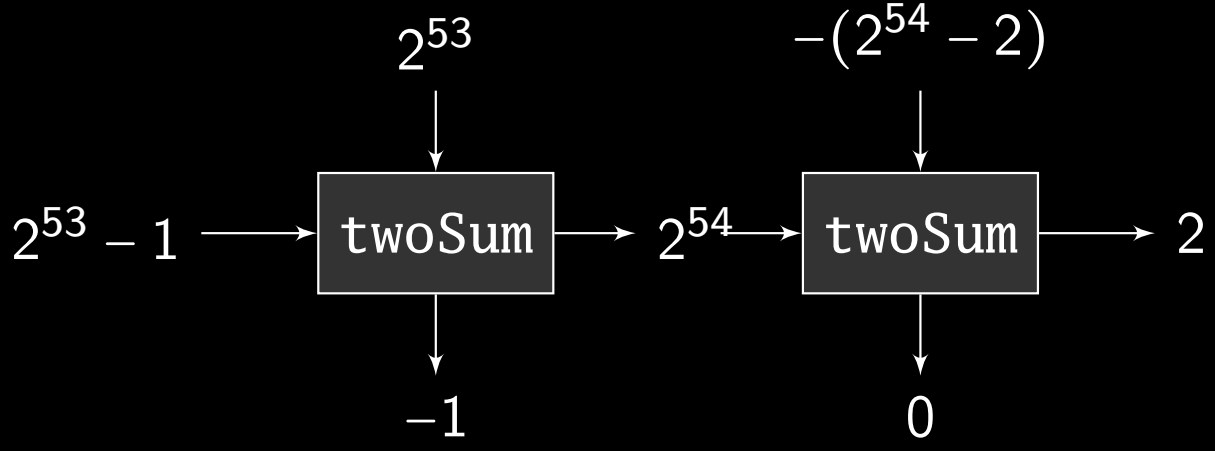
$x_0 = 2^{53} - 1$, $x_1 = 2^{53}$ et $x_2 = -(2^{54} - 2)$



Recherche 1

Expliquez les résultats trouvés. Que va faire l'algorithme de somme compensée ? Et celui de somme en cascade ?

$x_0 = 2^{53} - 1$, $x_1 = 2^{53}$ et $x_2 = -(2^{54} - 2)$



Recherche 1

Expliquez les résultats trouvés. Que va faire l'algorithme de somme compensée ? Et celui de somme en cascade ?

Sommaire

```c
#include <stdio.h>
#include <mpfr.h>

int main (void){
  mpfr_t un , dix , trois, zero_1, zero_3, trois_fois_zero_1, diff ; /*
      ↪ Déclarations simultanées */
  int test;

  mpfr_set_default_prec (53) ; /* Fixer la precision par defaut */
  mpfr_set_emin ( -1073) ; /* Fixer l' exposant emin ( en realite , emin -
      ↪ precision +2) */
  mpfr_set_emax (1024) ; /* Fixer l' exposant emax ( en realite , emax +1) */

  mpfr_inits (zero_1,zero_3, trois_fois_zero_1, (mpfr_ptr) 0); /* Précision par
      ↪ defaut 53 bits */
  mpfr_inits2 (128, un, trois, dix , diff,  (mpfr_ptr) 0) ; /* Précision de 128
      ↪ bits exactement */

  mpfr_set_d (un , 1.0 , MPFR_RNDN ); /* MPFR_RND Division en arrondi au plus
      ↪ près */
  mpfr_set_d (trois , 3.0 , MPFR_RNDN );
  mpfr_set_d (dix , 10. , MPFR_RNDN );

  mpfr_div (zero_1 , un , dix , MPFR_RNDN );
  mpfr_div (zero_3 , trois , dix , MPFR_RNDN );
  mpfr_mul (trois_fois_zero_1, trois, zero_1, MPFR_RNDN  );
```

Banque chaotique

Exemple dû à Jean-Michel Muller.

M. X a récemment été à sa banque (Chaotic Bank Society), pour connaître les nouvelles offres proposées aux meilleurs clients. Son banquier lui propose l'offre suivante : « vous déposez tout d'abord e - 1 euros sur votre compte (où e = 2.7182818... est la base du logarithme népérien). La première année, nous prenons 1 euro sur votre compte de frais de gestion. Par contre, la deuxième année est plus intéressante pour vous, car nous multiplions votre capital restant par 2 et prenons 1 euro de frais de gestion. La troisième année est encore plus intéressante, car nous multiplions votre capital par 3 et prenons 1 euro de frais de gestion. Et ainsi de suite : la n-ième année, nous multiplions votre capital par n et prenons 1 euro de frais de gestion. Intéressant, non ? » Pour prendre sa décision, M. X décide de demander l'aide d'un informaticien et se retourne vers vous.

Banque chaotique

Exemple dû à Jean-Michel Muller.
M. X a récemment été à sa banque (Chaotic Bank Society), pour connaître les nouvelles offres proposées aux meilleurs clients. Son banquier lui propose l'offre suivante : « vous déposez tout d'abord e - 1 euros sur votre compte (où e = 2.7182818... est la base du logarithme népérien). La première année, nous prenons 1 euro sur votre compte de frais de gestion. Par contre, la deuxième année est plus intéressante pour vous, car nous multiplions votre capital restant par 2 et prenons 1 euro de frais de gestion. La troisième année est encore plus intéressante, car nous multiplions votre capital par 3 et prenons 1 euro de frais de gestion. Et ainsi de suite : la n-ième année, nous multiplions votre capital par n et prenons 1 euro de frais de gestion. Intéressant, non ? » Pour prendre sa décision, M. X décide de demander l'aide d'un informaticien et se retourne vers vous.