COLLÈGE LIONEL GROULX

MATHS APPLIQUÉES EN INFO

NOTES DE COURS

CHAPITRE 5

ELISE DESGRENIERS, JONATHAN LORTIE ET AUTRES

CHAPITRE 5

NORME IEEE 754

5.1 NOTATION SCIENTIFIQUE

NOTATION SCIENTIFIQUE

Nombre contenant un chiffre entre 1 et 9 à gauche de la virgule et multiplié par une puissance de la base.

Afin de représenter les nombres réels, l'ordinateur a besoin de la notation scientifique.

Un nombre est écrit en **notation scientifique** lorsqu'il possède un seul chiffre non nul à gauche de la virgule, et que ce chiffre est multiplié par une puissance de la base dans laquelle le nombre est écrit (2, 10, 16, etc.).

$$\underbrace{1,2643}_{\text{nombre}} \times \underbrace{10}_{\text{base}}^{\overbrace{8}^{\text{exposant}}}$$

L'exposant correspond au nombre de déplacement(s) de la virgule, à partir du nombre original, afin d'obtenir un seul chiffre non nul à gauche de la virgule lors du passage à la notation scientifique.

- Si la virgule se déplace vers la **droite**, alors l'exposant sera **négatif**.
- Si la virgule se déplace vers la **gauche**, alors l'exposant sera **positif**.

Pas de virgule?

Il ne faut pas oublier que la virgule décimale est présente même quand elle n'est pas explicitement écrite : elle est alors complètement à droite du nombre (par exemple, 9 pourrait s'écrire 9,0).

L'**algorithme pour écrire un nombre en notation scientifique** est le suivant :

1. Déplacer la virgule jusqu'à ce qu'il n'y ait qu'un seul chiffre à gauche de celle-ci.
2. Calculer le nombre de déplacements effectués. Ceci correspond à l'exposant qu'on devra utiliser
3. Ajuster le signe au besoin
    a. Si la virgule se déplace vers la **droite**, alors l'exposant sera **négatif**.
    b. Si la virgule se déplace vers la **gauche**, alors l'exposant sera **positif**.
4. Écrire le nombre obtenu à l'étape 1, puis ajouter le multiplicateur correspondant à la base de travail en y ajoutant l'exposant qu'on vient de calculer.

EXEMPLE 1

Notation scientifique

On veut écrire les nombres suivants en notation scientifique.

a) 123

On déplace la virgule vers la gauche de 2 positions. Donc, $123 = 1,23 \times 10^{2}$.

b) $\left(-0,0011\right)_2$

Puisque la virgule se déplace de 3 positions vers la droite (afin d'obtenir un seul chiffre non nul avant la virgule), on a que $\left(-0,0011\right)_2 = -1,1 \times 2^{-3}$.

Attention! Le multiplicateur est 2 (et non 10) puisque le nombre est écrit dans la base binaire.

c) $-12\,365$

d) $\left(0,0000000101\right)_2$

e) $\left(101,101\right)_2$

Le nombre avant la virgule

Lorsqu'un nombre décimal est écrit en notation scientifique, le premier chiffre avant la virgule sera entre 1 et 9 puisqu'il doit être non nul.

En binaire, le premier chiffre avant la virgule sera **toujours 1**. En effet, on a le choix entre 0 et 1, et il doit être non nul, alors la conclusion s'impose d'elle-même.

5.2 Norme IEEE 754

On a vu au chapitre 3 que pour faire une opération sur deux chaines binaires, l'ordinateur a besoin que les chaines soient de même longueur. C'est pourquoi il est nécessaire de déclarer le type d'une variable lorsque vous en créez une. Par exemple, un *integer*, un *float* et un *double* ne sont pas représentés par des chaînes binaires de la même longueur.

La norme IEEE 754 est une norme pour la représentation des nombres réels en informatique, mise en place par l'*Institute of Electrical and Electronics Engineers*. Cette norme, adoptée en 1985, a permis d'uniformiser le codage des nombres réels et le transfert des informations entre les différents ordinateurs, peu importe le fabricant.

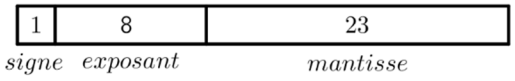
Elle est entre autres utilisée lors de l'utilisation des *float* et des *double*. Un *float* est un format de variable utilisant 32 bits, tandis qu'un *double* en utilise 64.

5.2.1 Norme IEEE 754 pour un float

Les 32 bits utilisés pour représenter un float sont répartis comme suit.

- Le premier bit est réservé pour le signe: 0 pour un nombre positif et 1 pour un nombre négatif.
- Les 8 bits suivants sont utilisés pour l'exposant de la base 2 de la notation scientifique.
- Les 23 bits restants constituent une section appelée la **mantisse**, qui contient la partie fractionnaire du nombre en notation scientifique (les chiffres après la virgule).

La figure suivante illustre de manière graphique la répartition des 32 bits.



Manque-t-il un chiffre?

Un lecteur avisé aura remarqué qu'il n'y a aucun bit réservé, sur les 32 disponibles, pour écrire le chiffre à gauche de la virgule, quand le nombre est écrit en notation scientifique.

Ce n'est pas une erreur. En effet, puisqu'en binaire, ce chiffre est toujours 1, il n'est pas nécessaire d'utiliser de l'espace mémoire pour l'enregistrer. Il faut toutefois se rappeler qu'il existe lorsqu'on passe de la notation IEEE 754 à une écriture normale.

Ce sera aussi le cas quand on voudra représenter un nombre sur 64 bits.

L'**algorithme pour écrire un nombre décimal sur 32 bits** est le suivant :

1. Convertir le nombre décimal en binaire.
2. Réécrire le nombre binaire obtenu en notation scientifique.
3. Effectuer l'opération « exposant de la notation scientifique + 127 ».
4. Convertir le nombre obtenu à l'étape précédente en binaire.
5. Écrire le nombre selon la norme IEEE 754.
    a. Écrire le bit associé au signe du nombre: 0 pour un nombre positif et 1 pour un nombre négatif.
    b. Écrire les 8 bits correspondant à l'exposant. Ajouter des 0 à gauche du nombre au besoin pour utiliser les 8 bits.
    c. Écrire la partie fractionnaire (à droite de la virgule en notation scientifique) dans la mantisse. Compléter avec des 0 pour utiliser les 23 bits disponibles.

EXEMPLE 2

Un *float* selon la norme IEEE 754

On veut écrire les nombres suivants sur 32 bits en respectant la norme IEEE 754.

a) 525

On a que $525 = \left(10\,0000\,1101\right)_2 = 1{,}0000\,0110\,1 \times 2^9$.

Aussi, on a que $9 + 127 = 136 = \left(1000\,1000\right)_2$.

Enfin, puisque 525 est positif, alors le premier bit sera 0.

Donc, $525 = \left(0 \mid 1000\,1000 \mid 0000\,0110\,1000\,0000\,0000\,000\right)$.

À noter l'ajout de 14 zéros à l'extrémité droite de la mantisse après les 9 bits utilisés dans la partie fractionnaire en notation scientifique, pour un total de 23 bits.

b) $-125\,648\,971$

$$\begin{aligned}\left(125\,648\,971\right)_{10} &= \left(1110\,1111\,1010\,1000\,0000\,1001\,011\right)_2 \\ &= 1{,}1101\,1111\,0101\,0000\,0001\,0010\,11 \times 2^{26}\end{aligned}$$

De plus, $26 + 127 = 153$ et $151 = \left(1001\,1001\right)_2$.

Comme on travaille sur un chiffre négatif, le premier bit sera 1.

On a donc que $-125\,648\,971 = \left(1 \mid 1001\,1001 \mid 1101\,1111\,0101\,0000\,0001\,001\right)$.

Lorsqu'on compare la portion fractionnaire de la notation scientifique (26 bits) et la mantisse (23 bits), on remarque que les 3 derniers bits de la notation scientifique ne sont pas pris en charge dans la représentation sur 32 bits en norme IEEE 754. Il y a donc une perte de précision lors de l'encodage sous forme de *float*.

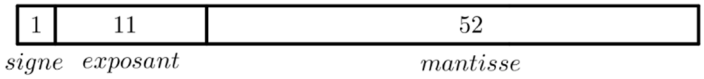
c) 102,5

5.2.2 Norme IEEE 754 pour un double

Si on veut représenter un réel sur 64 bits au lieu de 32, alors on utilise le type de variable *double*.

Le principe est exactement le même que pour un *float*, sauf qu'on a plus de bits, donc on peut représenter des nombres encore plus grands.

La structure d'un *double* est illustrée dans la figure suivante.



Les procédures décrites à la section précédente sont encore valides, à une exception près. Au lieu d'additionner ou soustraire 127 lors de la manipulation de l'exposant en notation scientifique, il faut additionner ou soustraire 1022.

5.3 LIMITES DE LA NORME IEEE 754

5.3.1 MINIMUM ET MAXIMUM

La première limite est liée au maximum et au minimum. En effet, si on additionne des nombres au-delà du maximum, on obtient un négatif et vice versa avec le minimum. Cela est dû à la façon dont fonctionne l'addition et la soustraction en binaire.

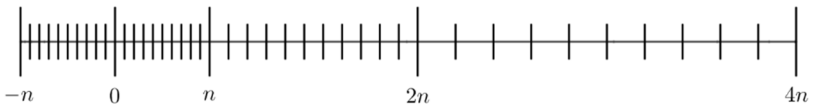
Avec un *float*, il est possible de représenter des nombres énormes ou des valeurs très proches de 0. Le plus gros nombre positif qu'il est possible de représenter sur 32 bits est 340 282 346 638 528 859 811 704 183 484 516 925 440. Le plus petit nombre positif qu'il est possible de présenter sur 32 bits est $1,401\,298\,464\,32 \times 10^{-55}$, soit

0,000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 140 129 846 432.

Les mêmes limitations sont présentes lorsqu'on veut représenter des nombres négatifs, puisque le bit qui gère le signe est indépendant de l'exposant et la mantisse dans les 32 bits de la norme IEEE 754.

Cependant, à l'extérieur de cet intervalle, la norme IEEE 754 pour un *float* est incapable de représenter les nombres. Dans les faits, si le nombre est supérieur au maximum de l'intervalle possible, alors l'ordinateur le traite comme l'infini. À l'inverse, si le nombre est inférieur au minimum de l'intervalle possible, alors l'ordinateur de traite comme un 0. C'est pourquoi il est important de respecter les limites imposées par la norme IEEE 754 pour un *float* sous peine d'erreurs de calcul lors de l'exécution d'un programme.

De plus, plus on s'éloigne du 0, plus l'espace entre deux nombres successifs qu'il est possible de représenter avec exactitude est grand, comme on peut le voir sur l'image suivante :



C'est pourquoi les petits nombres (ex. : entre 1 et 10) peuvent être facilement représentés avec exactitude sur 32 bits, tandis qu'on perd en précision lorsqu'on représente la plupart des gros nombres.

EXEMPLE 3

Étendue de la représentation

Certains sites web, comme le site *IEEE-754 Floating Point Converter*, permettent de convertir et d'afficher les 32 bits d'un *float* correspondant à un nombre décimal ou un nombre binaire.

On peut accéder à ce site avec le code QR ci-contre, l'adresse https://www.h-schmidt.net/FloatConverter/IEEE754.html ou en tapant « convertisseur norme IEEE 754 » dans Google.

On veut utiliser ce site pour répondre aux questions suivantes.

a) Quelle devrait être la mantisse pour obtenir le plus petit nombre possible avec un exposant de 1000 0001? Quel est ce nombre?

b) Trouver le plus grand nombre positif qu'il est possible de représenter en utilisant l'exposant 1000 0001.

c) Quelle est l'étendue des nombres positifs représentés par l'exposant 1000 0001?

d) Refaire les étapes a) à c) en utilisant l'exposant 1000 0111. Que peut-on en déduire?

5.3.2 APPROXIMATION DES NOMBRES

Ensuite, tous les nombres ayant plus de 23 chiffres après la virgule sont des approximations de leur valeur, car le nombre de décimales qu'il est possible de mettre en mémoire est limité à 23. Cela touche, entre autres, tous les nombres périodiques.

Par exemple, dans la norme IEEE 754, le nombre 1,2 est représenté par $\left(0\,|\,0111\,1111\,|\,0011\,0011\,0011\,0011\,0011\,010\right)$.

Or, si on retraduit ce nombre en dans la base décimale à partir de sa représentation sur 32 bits, on obtient la valeur 1,200 000 047 683 715 820 312 5.

On constate que la valeur décimale provenant de la représentation sur 32 bits n'est pas la même que le nombre original ayant été représenté dans la norme IEEE 754. Cela s'explique par le fait que le nombre 1,2 se trouvait entre deux représentants possibles, tel que discuté à la section précédente.

Cette limitation de la norme IEEE 754 cause principalement des problèmes de comparaison entre deux valeurs numériques.

EXEMPLE 4

Des nombres exacts… ou pas?

En utilisant le même convertisseur qu'à l'exemple 3, on veut vérifier si les nombres suivants sont représentés de façon exacte.

a) 1,3

b) 1,4

c) 1,5

d) 1,6

e) 1,7

f) 4,218 62

g) 97 854 613 265 694

h) 97 854 613 880 832

EXEMPLE 5

La fortune de Jeff Bezos

En date du début du mois de septembre 2022, la fortune de Jeff Bezos était estimée à 166 milliards.

a) Est-ce que la limitation du minimum et du maximum s'applique ici?

b) Sachant que 166 milliards = $1,0011\ 0101\ 0011\ 0010\ 1111\ 0111\ 111 \times 2^{37}$, est-il possible de représenter **avec précision** sa fortune en utilisant un *float*? Justifier.

5.3.3 ORDRE DE GRANDEUR

La troisième limitation de la norme IEEE 754 sur un *float* porte sur l'ordre de grandeur des nombres lorsqu'ils sont mis en relation.

Par exemple, si un nombre a a un exposant de 127 et un nombre b a un exposant de 18, alors la différence de position de la virgule entre a et b est de 109 positions. Or, un *float* ne considère que les 23 premières positions après la virgule. Dans un tel scénario, on obtiendrait alors $a + b = a$.

On considère 2 nombres supérieur à 1 en notation scientifique.

- Si les deux nombres ont des exposants inférieurs ou égaux à 23, il n'y a aucun problème de précision lors de l'addition.
- Si au moins un des deux nombres a un exposant supérieur à 23
    - Si l'écart entre les exposants des 2 nombres est supérieur à 23, on va assurément avoir des problèmes de précision en obtenant un résultat du type $a + b = a$.
    - Si l'écart entre les exposants des 2 nombres est inférieur ou égal à 23, on pourrait avoir des problèmes de précision, dépendant de la représentation binaire des nombres impliqués (non traité dans ce document).

Un raisonnement similaire existe pour les nombres inférieurs à 1 (exposant négatif en notation scientifique).

L'exemple suivant illustre différents cas de figure.

EXEMPLE 6

Problèmes de mantisse

Est-ce que les additions suivantes sont précises en norme IEEE 754?

a) $1010\,1101\,0111\,1111 + 1111\,010$

En notation scientifique, on a que $1{,}0101\,1010\,1111\,111 \times 2^{15}$ et $1{,}1110\,1 \times 2^{6}$. Comme les deux exposants sont sous la barre des 23 décimales considérées par la mantisse, le résultat sera précis.

$$\begin{array}{rr} & 1\,0101\,1010\,1111\,111 \\ + & \qquad\qquad 1111\,010 \\ \hline & 1\,\underbrace{0101\,1101\,1111\,001}_{\text{bits considérés dans la mantisse}} \end{array}$$

b) $1010\,1101\,0111\,1111\,0001\,0100\,1010\,000 + 1101$

En notation scientifique, on a que $1{,}0101\,1010\,1111\,1110\,0010\,1001\,0100\,00 \times 2^{30}$ et $1{,}101 \times 2^{3}$.

Puisque l'un des exposants est supérieur à 23, on s'intéresse à l'écart entre les exposants, qui est de $30 - 3 = 27$. Comme l'écart est supérieur à 23, il est garanti que le résultat ne sera pas correct.

$$
\begin{array}{rr}
 & 101\,0110\,1011\,1111\,1000\,1010\,0101\,0000 \\
+ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 1101 \\
\hline
 & 1\,\underbrace{01\,0110\,1011\,1111\,1000\,1010\,0}_{\text{bits considérés dans la mantisse}}\,\underbrace{101\,1101}_{\substack{\text{bits NON}\\ \text{considérés}\\ \text{dans la}\\ \text{mantisse}}}
\end{array}
$$

c) $1010\,1101\,0111\,1111\,0001\,0100\,000 + 1111\,0100\,1100\,1010$

En notation scientifique, on a que $1,0101\,1010\,1111\,1110\,0010\,1 \times 2^{26}$ et $1,1110\,1001\,1001\,01 \times 2^{15}$.

Puisque l'un des exposants est supérieur à 23, on s'intéresse à l'écart entre les exposants, qui est de $26 - 15 = 11$. Comme l'écart est inférieur à 23, le résultat peut être correct.

$$
\begin{array}{rr}
 & 101\,0110\,1011\,1111\,1000\,1010\,0000 \\
+ & \qquad\qquad\qquad\qquad 1111\,0100\,1100\,1010 \\
\hline
 & 1\,\underbrace{01\,0110\,1100\,1110\,1101\,0110\,1}_{\text{bits considérés dans la mantisse}}\,\underbrace{010}_{\substack{\text{bits}\\ \text{NON}\\ \text{considérés}}}
\end{array}
$$

On ne validera pas l'exactitude du résultat d'un tel scénario dans le cours.

EXEMPLE 7

Ordre de grandeur

Reprenons l'exemple sur la fortune de 166 milliards de Jeff Bezos.

Supposons qu'il reçoit un chèque de 500$ du gouvernement et qu'il l'ajoute à sa fortune.

Rappel : 166 milliards $= 1,0011\,0101\,0011\,0010\,1111\,0111\,111 \times 2^{37}$

a) Écrire le nombre 500 sur 32 bits en respectant la norme IEEE 754.

b) Dans la norme IEEE 754, y aura-t-il une différence entre l'ancien et le nouveau solde de son compte de banque? Justifier.

Exercices

Lorsqu'un convertisseur est nécessaire, utiliser celui de l'exemple 3, p. 7.

1. Écrire les nombres suivants en utilisant la norme IEEE 754 sur 32 bits.
   a) -2
   b) 135
   c) 111 111 111
   d) $-289{,}12$
   e) 101,4

2. Soit le nombre $\left(91475623\right)_{10}$, équivalent à $\left(10101110011110011101010011 1\right)_{2}$.
   a) Quelle est la représentation de $\left(91475623\right)_{10}$ en norme IEEE 754?
   b) En utilisant un site web, convertir la réponse obtenue en a) vers la base décimale. Le résultat est-il exact?

3. Grâce à la norme IEEE 754, est-il possible de représenter le même nombre de nombres négatifs que de nombres positifs? Justifier.

4. Est-il possible de représenter le nombre 0 en IEEE 754. Si oui, quelle est cette représentation?

5. En utilisant un convertisseur en ligne, répondre aux questions suivantes.
   a) Trouver le plus petit nombre positif représenté en utilisant l'exposant 0111 1111.
   b) Trouver le plus grand nombre positif représenté en utilisant l'exposant 0111 1111.
   c) Quelle est l'étendue des nombres positifs représentés par l'exposant 0111 1111?

6. On demande à un ordinateur d'utiliser des *float* représentés en norme IEEE 754 pour calculer la moyenne de la richesse entre tous les Canadiens possédant de 100 000$ à 200 000$ et la richesse de Jeff Bezos, évaluée à 166 000 000 000 $. Or, l'ordinateur retourne un nombre qui est BEAUCOUP plus petit que 100 000, ce qui ne fait pas de sens… Expliquer le problème rencontré dans ce calcul.

7. Parmi les comportements suivants, lesquels sont incompatibles avec la norme IEEE 754? Lorsque c'est impossible, spécifier le genre de limitation rencontrée.
   a) Additionner deux nombres qui sont très éloignés l'un de l'autre.
   b) Additionner deux nombres qui sont très près l'un de l'autre.
   c) Additionner un nombre et son opposé.
   d) Additionner deux très très grands nombres.
   e) Représenter le nombre d'atomes dans l'univers.
   f) Représenter un nombre réel.

SOLUTIONS DES EXERCICES

CHAPITRE 5

NORME IEEE 754

1.

   a) $\left(1 \mid 1000\,0000 \mid 0000\,0000\,0000\,0000\,0000\,000\right)$

   b) $\left(0 \mid 1000\,0110 \mid 0000\,1110\,0000\,0000\,0000\,000\right)$

   c) $\left(0 \mid 1001\,1001 \mid 1010\,0111\,1101\,1010\,1111\,000\right)$

   d) $\left(1 \mid 1000\,0111 \mid 0010\,0001\,0001\,1110\,1011\,100\right)$

   e) $\left(0 \mid 1000\,0101 \mid 1001\,0101\,1001\,1001\,1001\,100\right)$

2.

   a) $\left(0 \mid 1001\,1001 \mid 0101\,1100\,1111\,0011\,1010\,101\right)$

   b) Non

3. Oui, il suffit de changer le premier bit en 1.

4. Oui. $\left(0 \mid 0000\,0000 \mid 0000\,0000\,0000\,0000\,0000\,000\right)$

5.

   a) 1
   b) 1,99999988079
   c) 0,99999988079

6. Comme les nombres sont trop éloignés, l'ordinateur retourne constamment le même nombre en *float*, soit 166 000 000 000 $. Ensuite, il divise ce nombre final par la quantité totale de données, ce qui explique le très petit résultat. L'erreur est donc dans les additions successives de *float*.

7.

   a) C'est un problème d'ordre de grandeur.
   b) Situation compatible avec la norme IEEE 754.
   c) Situation compatible avec la norme IEEE 754.
   d) Cela dépend de la grandeur des nombres
   e) Ce nombre dépasse la limite supérieure représentable avec un *float*.
   f) Cela dépend de la précision qu'on a besoin.