Algorithmes et Pensée Computationnelle

Programmation de base

Le but de cette séance est d'aborder des notions de base en programmation. Au terme de cette séance, l'étudiant sera capable de :

- Comprendre les différentes représentations de nombres.
- Définir des variables et comprendre les types de variables existants.
- Utiliser des notions d'algèbre booléenne.
- Ecrire des branchements conditionnels.

Les langages qui seront utilisés pour cette séance sont Java et Python. Assurez-vous d'avoir bien installé Intellij. Si vous rencontrez des difficultés, n'hésitez pas à vous référer au guide suivant : tutoriel d'installation des outils et prise en main de l'environnement de travail.

1 Représentation de nombres entiers

Question 1: (*5 minutes*) **Entiers non signés**
Sur 8 bits, convertir $113_{(10)}$ en base binaire.

Conseil

Faire un tableau comme présenté dans la diapositive 10 du cours (programmation de base). Essayer de décomposer le nombre en une somme de puissances de 2.

Solution

Méthode 1 : Utiliser la division par 2 comme vu dans la première séance d'exercices

Méthode 2 : Utiliser les puissances de 2 pour décomposer le nombre (vu également dans la première séance d'exercices).
$113 = 64 + 32 + 16 + 1 = 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^0$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	1	0	0	0	1

On obtient donc $01110001_{(2)}$

Question 2: (*5 minutes*) **Entiers signés négatifs**
En utilisant le résultat de la question précédente, convertir sur 8 bits $-113_{(10)}$ en base binaire.

Conseil

- Il faut prendre la représentation sur 7 bits d'un nombre entier non signé.
- On rajoute un 8ème bit qui sera le signe.
- Pour cet exercice, reprendre l'expression non signée de la question précédente (question 1 - Entiers non signés) et changer le premier bit en conséquence.
- Le premier bit vaut 0 pour un nombre positif et 1 pour un nombre négatif.

Solution

$113_{(10)} = 01110001_{(2)}$

En changeant le premier bit à 1 pour faire passer le nombre en nombre négatif on obtient :

$-113_{(10)} = 11110001_{(2)}$

Question 3: (*5 minutes*) **Complément à 1**
Écrire le complément à 1 de $-113_{(10)}$.

Quelle est la différence entre cette méthode et la précédente ?

Conseil

- L'opposé de 0 en binaire est 1 et inversément.
- Pour étudier la différence, il faut regarder les différentes manières d'exprimer -0 en binaire (se référer à la diapositive 11 du cours de la semaine 3).

Solution

$113_{(10)} = 01110001_{(2)}$

0	1	1	1	0	0	0	1
not	not	not	not	not	not	not	not
1	0	0	0	1	1	1	0

avec cette méthode, $-113_{(10)} = 10001110_{(2)}$

L'intervalle de cette méthode ne change pas par rapport à la précédente. Par contre, l'expression de -0 sera différente.

Signé : $-0_{(10)} = 10000000_{(2)}$

Complément à 1 : $-0_{(10)} = 11111111_{(2)}$

Les deux ont le même intervalle : $[-127_{(10)},+127_{(10)}]$

Question 4: (*5 minutes*) **Complément à 2**
Quel est le complément à 2 de $-113_{(10)}$ et quelle est l'utilité de cette représentation ?

Conseil

Exemple du cours avec $87_{(10)}$

$87_{(10)} = 01010111_{(2)}$

a	0	1	0	1	0	1	1	1
b	1	0	1	0	1	0	0	0
c	1	0	1	0	1	0	0	1

a : convertir le nombre en binaire

b : inverser tous les bits

c : rajouter 1 au nombre pour obtenir le complément à 2

On obtient donc $-87_{(10)} = 10101001_{(2)}$

Solution

Ici, il suffit donc d'ajouter 1 au complément à 1 de $-113_{(10)}$

On obtient donc $10001111_{(2)}$

Concernant l'intervalle, il est changé, étant donné qu'il n'existe plus qu'une seule représentation possible pour $-0_{(10)}$.

Le nouvel intervalle est donc $[-128_{(10)}, +127_{(10)}]$

Question 5: (*10 minutes*) **Floating point**

Voici la représentation en binaire d'un nombre à virgule flottante :

signe	exposant	mantisse
0	10110101	01000001000000000000001

Que vaut cette représentation en base 10 ? Utiliser la représentation des floating points (avec un biais de 127). Arrondir les résultats intermédiaires et la valeur finale au 3ème chiffre significatif après la virgule.

Conseil

- Se référer à la diapositive 15 du cours (programmation de base) pour plus de détails.
- Poser chaque calcul, et ensuite tout fusionner avec la formule.
- L'exposant et la mantisse sont des entiers positifs, donc non signés.
- Pour vérifier votre résultat, vous pouvez utiliser l'outil suivant : `https://www.h-schmidt.net/FloatConverter/IEEE754.html`

Solution

- Signe du nombre : $(-1)^{signe} = (-1)^0 = 1$
- Exposant en base 10 : $10110101_{(2)} = 181_{(10)}$
- Pour obtenir l'exposant, il faut encore lui appliquer le biais, il faut soustraire 127 à notre résultat. Ici on aura $2^{(181-127)} = 2^{54}$
- Mantisse : $01000001000000000000001_{(2)} = 1 + 1*2^{-2} + 1*2^{-8} + 1*2^{-23} = 1.254_{(10)}$
- Valeur = Signe * Mantisse * $2^{exposant-127} = 1 * 1.254 * 2^{54} = 2.259 * 10^{16}$