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: - définir une variable, de définir son type et sa valeur. - Définir une fonction et comprendre son rôle. - Utiliser des notions d'algèbre booléenne. - Comprendre la notion d'entrée/sortie. Les langages qui seront utilisés pour cette séance sont **Java** et **Python**. Assurez-vous d'avoir bien installé **Pycharm** et **Netbeans**. 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 **.

Exercice 1: Représentation de nombres entier (30 minutes)

Question 1: Unsigned int (5 minutes)

Sur 8 bit écrire 11310 en base binaire.

Hint: Faire un tableau comme présenté dans le cours page 9 de la semaine 3. Essayer de décomposer en une somme de puissances de 2 le nombre.

Solution: 1. Décomposer le nombre en une somme de puissances de 2.

113 = 64 + 32 + 16 + 1 = 1 * 26 + 1 * 25 + 1* 24 + 1 * 20

2. Inclure les coefficients dans un tableau:

27	26	25	24	23	22	21	20
0	1	1	1	0	0	0	1

Donc la représentation en binaire de l'entier non signé 113 est: 01110001

Question 2: Signed integers with signed magnitude (2 minutes)

En utilisant le résultat de la question précédente, écrire sur 8 bit -11310 en base binaire

Hint: Pour cette représentation, on réserve le 7ème bit pour le signe.

Solution:

27	26	25	24	23	22	21	20
1	1	1	1	0	0	0	1

Donc l'expression de -113 en utilisant la méthode de "signed integers" est 11110001

Question 3: Complément à 1 (5 minutes)

Ecrire le complément à 1 de -11310.

Qu'elle est la différence entre cette méthode par rapport à celle précédente ?

Hint:

1. En programmation l'opposé d'une variable est not(la variable). Ici, le même principe s'applique. L'opposé de 0 en binaire est...
2. Comment peut-on exprimer 0 dans cette méthode ?

Solution: 1.

27	26	25	24	23	22	21	20
0	1	1	1	0	0	0	1
not	not	not	not	not	not	not	not
1	0	0	0	1	1	1	0

2. La portée de cette méthode ne change pas par rapport à celle précédente. Par contre, l'expression de -0 sera différente.

Signed magnitude: -010 = 100000002

Complément à 1: -010 = 111111112

Both have the same range: range = [-12710,+12710]

Question 4: Complément à 2 (3 minutes)

Quel est le complément à 2 de -11310 ?

Quelle est l'utilitée de cette représentation ?

Hint: Chaque représentation est-elle unique ?

Solution:

1. Il faut ajouter un au complément à 1 de -11310. C'est à dire faire plus un à la colonne de 20.

27	26	25	24	23	22	21	20
1	0	0	0	1	1	1	1

2. Il n'y a plus qu'une représentation possible pour -010. Il y'a donc un nombre de plus possible par rapport aux dernières représentations. La nouvelle range est donc: range = [-12810,+12710]

Question 5: Floating point (10 minutes)

Que vaut en base 10 le chiffre binaire suivant d'après la représentation floating point ? Arrondir les résultats intermédiaires et la valeur finale au 3ème chiffre

significatif après la virgule.

Sign	Exponent	Mantissa
0	10110101	01000001000000000000001

Hint: Poser chaque calcul, et ensuite tout fusionner avec la formule. Utilisez la page 14 de la semaine 3 comme exemple.

Solution:

sign = 0 –> (-1)sign = (-1)0 = 1

e = exponent = 1 * 27 + 0 * 26 + 1 * 25 + 1 * 24 + 0 * 23 + 1 * 22 + 0 * 21 + 1 * 20 = 181

2e-127 = 2181-127 = 254

mantissa = 01000001000000000000001 = 1 + 0 * 2-1 + 1 * 2-2 + 0 * 2-3 + (. . .) + 1 * 2-8 + 1 * 2-23 = 1.254

Donc Valeur = (+1) * 1.254 * 254 = 2.259 * 1016

Question 6: Conversion d'un nombre binaire au format complément à 2 (5 minutes)

Voici un nombre binaire exprimé sur 8 bit au format complément à 2: 10010011

Convertir ce nombre en base 10.

Hint: Utilisez le même tableau que dans les premières questions

Solution:

Il faut appliquer le processus inverse qu'à la question 4 et 3. Cela permet d'obtenir la valeur positive en binaire du nombre que l'on cherche. Puis il faut convertir cette valeur en base 10 et enfin prendre son inverse.

Opération	27	26	25	24	23	22	21	20
Aucune	1	0	0	1	0	0	1	1
Soustraction	1	0	0	1	0	0	1	0
Opposition	0	1	1	0	1	1	0	1

01101101 = 1 + 4 + 8 + 32 + 64 = 10910

Le nombre est donc -10910

Exercice 2: Conditionnal branching (10 minutes)

Le but de cet exercice est d'entrainer la lecture de code, la compréhension des opérateurs booléens ainsi que le "case switching" à travers le conditionnal branching.

Question 1: Conditionnal branching in java (5 minutes)

Qu'affiche le code suivant ?

Rappel: Le code suivant est en Java

```
int numero_mois = 7

switch(numero_mois) {

    case 1:
              System.out.println("Janvier")
              break;
    case 2:
              System.out.println("Février")
              break;
    case 3:
              System.out.println("Mars")
              break;
    case 4:
              System.out.println("Avril")
              break;
    case 5:
              System.out.println("Mai")
              break;
    case 6:
              System.out.println("Juin")
              break;
    case 7:
              System.out.println("Juillet")
              numero_mois = 9
    case 8:
              System.out.println("Août")
              break;
    case 9:
              if (numero_mois == 8){
                System.out.println("Septembre")
                break;
                }
               else{
                 System.out.println("Décembre")
                 numero_mois = 13
```

```
                    break;
                    }
        case 10:
                    System.out.println("Octobre")
                    break;
        case 11:
                    System.out.println("Novembre")
                    break;
        case 12:
                    System.out.println("Décembre")
                    break;
        default:
                    System.out.println("Ce n'est pas un mois. ")
}
```

Hint:

1. Break indique que l'on sort de l'accolade. Les cas suivants ne seront pas traités.
2. Lorsque l'on pose "case n" où n est un nombre cela est équivalent au test n == numero_mois. Ce test est aussi valable si on cherche a comparer des chaînes de caractères (par exemple si numero_mois = " Juin", à ce moment là n sera aussi une chaîne de caractères).

Solution:

Le code va donc afficher:

Juillet Décembre

Explication: 1. Comme la case 7 ne contient pas de break et modifie numero_mois, la lecture du code va continuer. 2. On rentre dans le case 9, qui contient un break. Le numero_mois sera aussi modifié mais cela ne sera pas important car on sort de l'accolade et les cas succédants ne seront pas traités.

Question 2: Conditionnal branching in python (5 minutes)

Qu'affiche le code suivant ?

Rappel: Ce code est en python.

```
name = "Garbinato"
if name == "Diallo":
    print("Alpha")
elif name == "Michelet":
    print("Gaëtan")
elif name == "Ballèvre":
    print("Nathan")
elif name == "Schwab":
```

```
    print("Johannes")
elif name == "Di Sanza":
    print("Maeva")
elif name == "Kanda Bile":
    print("Richard")
elif name == " Yasser Haddad":
    print("Yann")
elif name == "Garbinato":
    print("Teacher")
else:
    print("Benoît")
print("Algorithmique et pensée computationnelle")
```

Hint: 1. Il faut vérifier la condition de chaque cas de manière linéaire.

2. Une fois une condition vérifiée, toutes celles d'après ne sont pas traitées.

Solution: Le code affiche:

Teacher

Algorithmique et pensée computationnelle

Explication: 1. Les 7 premières conditions ne sont pas vérifiées. Le premier print exécuté sera " print("Teacher") " 2. Le dernier print étant en dehors des if, else et elsif (on peut le voir grâce à l'indentation), il sera forcément exécuté