TD11 - Ecriture d'une entier dans une base Fichier hexadécimal

MPSI-PCSI - Lycée Thiers

Ecriture d'un nombre dans une base

Rappel : Division euclidienne. $\forall\, n \in \mathbb{N},\ \forall\, m \in \mathbb{N}^*$:

$$\exists!\,(q,r) \in \mathbb{N}^2, \quad \text{tel que} \begin{cases} n = q \times m + r \\ 0 \leqslant r < m \end{cases}$$

Corollaire. *Pour tout entier $m \in \mathbb{N} \setminus \{0,1\}$, pour tout entier $n \in \mathbb{N}$, il existe une unique suite finie $(u_n)_{n\in[[0,N]]}$ d'entiers compris entre 0 et $m-1$, tels que :*

$$n = \sum_{i=0}^{N} u_i \times m^i$$

et si $N > 0$, $u_N \neq 0$. L'écriture en base m de n est $u_N\, u_{N-1} \cdots u_1\, u_0$.

Ecriture d'un nombre en base $m > 1$

Preuve. Existence : par division euclidienne $\exists\,!(q_0, u_0) \in \mathbb{N}^2$ avec $n = q_0 \times m + u_0$ et $0 \leqslant u_0 < m$. Si $q_0 = 0$ la suite u_0 convient puisque $n = u_0 \times m^0$. Si $q_0 \neq 0$ alors par division euclidienne $\exists\,!(q_1, u_1) \in \mathbb{N}^2$ avec $q_0 = q_1 \times m + u_1$ et $0 \leqslant u_1 < m$. Remarquer que $n > q_0 > q_1$ puisque $m > 1$. Ainsi en poursuivant ce procédé on construit une suite finie $(u_0, u_1, \ldots, u_N)$ avec $0 \leqslant u_i < m$, $q_N = 0$ et :

$$n = q_0 \times m + u_0 = (q_1 \times m + u_1) \times m + u_0$$

$$= ((\cdots((0 \times m + u_N) \times m + u_{N-1})\cdots) \times m + u_1) \times m + u_0 = \sum_{i=0}^{N} u_i \times m^i.$$

Unicité. Elle provient de l'unicité du quotient et du reste dans la division euclidienne : si $n = \sum_{i=0}^{N} u_i \times m^i$ alors le procédé précédent produit la suite $(u_0, u_1, \ldots, u_N)$.

Exemple : Ecriture de 72 en base 2 :

$72 = 2 \times 36 + 0 \quad u_0 = 0$
$36 = 2 \times 18 + 0 \quad u_1 = 0$
$18 = 2 \times 9 + 0 \quad u_2 = 0$
$9 = 2 \times 4 + 1 \quad u_3 = 1$
$4 = 2 \times 2 + 0 \quad u_4 = 0$
$2 = 2 \times 1 + 0 \quad u_5 = 0$
$1 = 2 \times 0 + 1 \quad u_6 = 1$

72 s'écrit 1001000_2 en base 2 (ou binaire).

En python :

```
>>> a = 72
>>> while a>0:
...     print(a%2)
...     a = a//2
...
0
0
0
1
0
0
1
```

La "boucle" `while` s'exécute tant que la condition a>0 est satisfaite.

Exemple : Ecriture de 72 en base 16 :

$72 = 16 \times 4 + 8 \quad u_0 = 8$

$4 = 16 \times 0 + 4 \quad u_1 = 4$

72 s'écrit 48_{16} ou 0x48 en base 16 (ou hexadécimal).

```
>>> a = 72
>>> while a>0:
...     print(a%16)
...     a = a//16
...
8
4
```

Pour écrire un nombre en base 16 on utilise les "chiffres" de 0 à 9 ainsi que :

A=10, B=11, C=12, D=13, E=14, F=15.

En 2 chiffres on écrit tous les nombres de 0 à $FF = 15 + 15 * 16 = 16^2 - 1 = 255$.
Autant qu'en binaire avec 8 'bits' (ou chiffre 0,1) car $11111111 = 2^8 - 1 = 255$.
Puisque $2^4 = 16$ on a la conversion binaire/hexadecimal :

$0000_2 = 0$, $0001_2 = 1$, $0010_2 = 2$, $0011_2 = 3$, $0100_2 = 4$, $0101_2 = 5$,
$0110_2 = 6$, $0111_2 = 7$, $1000_2 = 8$, $1001_2 = 9$, $1010_2 = A$, $1011_2 = B$,
$1100_2 = C$, $1101_2 = D$, $1110_2 = E$, $1111_2 = F$.

Ex : $72_{10} = 0100\,1000_2 = 4\,8_{16}$.

Voilà pourquoi un ordinateur fonctionne en 8, 16, 32 ou 64 bits
(multiples de 4... par une puissance de 2)...

Exercice 1. Enoncé

Exercice 1. Ecriture d'un entier naturel dans une base.

1. Définir une variable globale `chiffres = '0123456789ABCDEF'`.
2. Ecrire une fonction `base(n,a)` qui prend en paramètre un nombre entier positif `n` et un entier `a` entre 2 et 16, et qui retourne une chaine de caractères contenant l'écriture de l'entier `n` en base `a`.
3. Vérifier son fonctionnement en obtenant l'écriture du nombre 255 en bases 2, 10 et 16.
4. Ecrire la fonction `index` prenant en paramètre un caractère apparaissant dans la chaine `chiffres` et qui renvoie l'indice ou il apparait. A quoi correspond cet indice ?
5. Ecrire la fonction inverse `valeur(ch,a)` prenant en paramètre un entier `a` entre 2 et 16 et une chaine de caractères `ch` représentant l'écriture en base `a` d'un nombre entier `n`, et qui renvoie l'entier `n`.

Exercice 1. Corrigé

1)

```
chiffres = "0123456789ABCDEF"
```

2)

```
def base(n,a):
    if n==0:
        return '0'
    ch = ''
    while n>0:
        r = n%a
        ch = chiffres[r] + ch
        n = n//a
    return ch
```

3)

```
In [1]: base(255,2), base(255,10), base(255,16)
Out[1]: ('11111111', '255', 'FF')
```

Exercice 1. Corrigé

4)

```
def index(c):
    for k in range(len(chiffres)):
        if c == chiffres[k]:
            return k
```

5)

```
def valeur(ch,a):
    n = 0
    for x in ch:
        n *= a
        n += index(x)
    return n
```

Exercice 2. Enoncé

Exercice 2. Fichier hexadécimal

1. Ouvrir le fichier `texteHexa.txt` et l'afficher. Ecrire le code permettant de récupérer son contenu dans une variable `contenu` de type chaine de caractère.
2. Le fichier contient un texte encodé de la façon suivante :
   - chaque caractère du texte original a été remplacé par son code dans la table ASCII, soit un nombre entre 0 et 255.
   - ce code a été écrit en hexadécimal sur deux caractères. Par exemple le caractère `z` a pour code ASCII 122 qui s'écrit en base 16 : 7A. Plutôt que le caractère `z`, on a écrit dans le fichier les 2 caractères `7A`.

   Ecrire le code permettant de décoder, d'afficher, puis de sauvegarder dans un autre fichier texte, tout le texte contenu dans le fichier `texteHexa.txt`.

Exercice 2. Corrigé

```
f = open('texteHexa.txt','r')
contenu = f.read()
f.close()
```

```
texte = ""
for k in range(0,len(contenu),2):
    hexa = contenu[k:k+2]
    texte += chr(valeur(hexa,16))
print(texte)

f = open('texteClair.txt','w')
f.write(texte)
f.close()
```

Exercice 2. Corrigé

Théorème : Un réel x a une écriture décimale finie si et seulement si :
- x est rationnel, x= p/q avec p et q premiers entre eux
- Un diviseur premier de q ne peut être que 2 ou 5.

Preuve : Si x a une écriture décimale finie, disons m chiffres après la virgule, alors il existe m dans N tel que x.10m̂ soit un entier n. Ainsi : x = n/10^m = n/(2^m.5^m) et x est rationnel.
Si x s écrit sous forme irréductible p/q, alors n.q = p.(2^m.5^m) avec p et q premiers entre eux.
D après le Lemme de Gauss : q est un diviseur de (2^m.5^m) ; les diviseurs premiers de q sont alors aussi des diviseurs premiers de (2^m.5^m) et ne peuvent donc être que 2 ou 5.

Réciproque. Si x=p/q avec q=2^n1.5^n2.
Si n1>=n2 alors x = p.5^(n1-n2)/10^n1.
Si n1< n2 alors x = p.2^(n2-n1)/10^n2. Dans tous les cas x est le quotient d un entier par 10^m (où m=max(n1,n2)), et donc x a une écriture décimale finie (ayant au plus m chiffres après la virgule). CQFD

Complément. Enoncé

Exercice 3. Complément hors-programme : Interface graphique pour l'obtention de l'écriture binaire d'un nombre.

Pour créer une interface graphique :
– Dans l'onglet Pyzo, 'Préférences', choisir pour Gui : Tk - Tk Widget Toolkit.
– Relancer Pyzo. On pourra désormais utiliser le module `tkinter` pour créer des interfaces graphiques.
– Dans le fichier contenant la définition de la fonction `base` (Ex.1.1), saisir le code suivant puis lancer son éxécution :

Complément. Enoncé

```
# Interface graphique
from tkinter import * # Module pour la creation d'interface graphique

def evaluer(event): # action lors de l'appui de la touche 'Entree'
    resultat = "s'ecrit en binaire : "
    resultat += base(eval(entree.get()),2) # Conversion en binaire
    sortie.configure(text = resultat) # Affichage du resultat

def effacer(event): # pour effacer au clic
    entree.delete(0,END) # Effacer du premier au dernier caractere
    sortie.configure(text = '') # Effacer le texte resultat

fenetre = Tk() # cree une fenetre graphique
entree = Entry(fenetre, width = 30) #cree un champ de saisie dans fenetre
entree.insert(0,"Saisir le nombre a convertir ici")
entree.bind("<Return>", evaluer) # Lie l'entree avec la touche 'Entree'
entree.bind("<Button-1>", effacer) # Lie l'entree avec le CLIC gauche
sortie = Label(fenetre) # cree un champ texte dans fenetre
entree.pack() # affiche le champ de saisie
sortie.pack() # affiche le champ texte
fenetre.mainloop() # ouvre la fenetre graphique
```

On pourra ne pas saisir les commentaires.

• Modifier le code pour obtenir une interface graphique qui donne aussi la conversion en héxadécimal.

Complément. Enoncé

```
# Modifications :
def evaluer(event): # action lors de l'appui de la touche 'Entree'
    resultat = "s'ecrit en binaire : "
    resultat += base(eval(entree.get()),2) # Conversion binaire
    sortie.configure(text = resultat) # Affichage du resultat
    resultat2 = "s'ecrit en hexadecimal : "
    resultat2 += base(eval(entree.get()),16) # Conversion héxadécimale
    sortie2.configure(text = resultat2) # Affichage du resultat

def effacer(event): # pour effacer au clic
    entree.delete(0,END) # Effacer du premier au dernier caractere
    sortie.configure(text = '') # Effacer le texte resultat

fenetre = Tk() # cree une fenetre graphique
entree = Entry(fenetre, width = 30) #cree un champ de saisie dans fenetre
entree.insert(0,"Saisir le nombre a convertir ici")
entree.bind("<Return>", evaluer) # Lie l'entree avec la touche 'Entree'
entree.bind("<Button-1>", effacer) # Lie l'entree avec le CLIC gauche
sortie = Label(fenetre) # cree un champ texte dans fenetre
sortie2 = Label(fenetre) # cree un 2eme champ texte
entree.pack() # affiche le champ de saisie
sortie.pack() # affiche le champ texte
sortie2.pack() # affiche le champ 2eme texte
fenetre.mainloop() # ouvre la fenetre graphique
```