

Travaux pratiques n°3

Algorithmique

F. CUVELLIER, J. TANOÏ

M.P.S.I. 1, 2014–2015

Algorithmique

L'*algorithmique* est la science des algorithmes. Un *algorithme* est une suite d'instructions élémentaires (c'est-à-dire compréhensibles de l'utilisateur), dont l'exécution ordonnée permet d'atteindre un objectif donné. Ce peut être une recette de cuisine, le mode d'emploi d'un appareil électronique. En informatique, les instructions les plus courantes sont les suivantes.

- Entrer zéro, un ou plusieurs paramètres ;
- Effectuer un calcul, une opération (par exemple, affecter une variable, ajouter ou multiplier deux nombres, afficher un résultat partiel, etc.) ;
- Tester une condition, et exécuter une action dans le cas favorable, éventuellement une autre dans le cas contraire ;
- Répéter une opération précédente.
- Donner une réponse, un résultat (*sortie*). On dit alors qu'on *retourne* un résultat.

Par exemple, pour calculer la factorielle d'un entier naturel n , on peut effectuer les opérations suivantes.

1. Affecter la variable f de la valeur 1 et la variable k de la valeur 0.
2. Si k est plus grand que n , retourner f ; sinon,
 - (a) Augmenter la variable k de 1.
 - (b) Affecter la variable f du produit de sa valeur par k .
 - (c) Répéter l'instruction 2.

Programmes en Python

Les deux programmes suivants sont des implémentations de l'algorithme pour la factorielle de n .

```
# example of a while loop
def fact(n):
    f = 1
    k = 1
```

```

while k < n:
    k += 1
    f *= k
return f

# example of a for loop
def Fact(n):
    f = 1
    for k in range(1, n+1):
        f *= k
    return f

```

Les commentaires sont précédés du signe `#`. Tous les caractères qui le suivent jusqu'à la fin de la ligne sont ignorés par Python.

La fonction `range()` a trois paramètres, *start*, *end*, *step*, et produit l'intervalle formé des nombres de *start* (paramètre optionnel — par défaut : 0) à *end*, exclu, incrémentés avec le pas *step* (paramètre optionnel — par défaut : 1). La commande `return` force la sortie de la fonction et affiche ce qui suit.

Ce programme-ci implémente la factorielle à partir de sa définition récursive

$$n! = \begin{cases} 1 & \text{pour } n = 0, \\ (n-1)!n & \text{pour } n > 0. \end{cases}$$

```

# recursive call
def rfact(n):
    if n == 0: return 1
    return rfact(n-1)*n

```

Dans les exercices suivants, chaque algorithme sera formulé en français. On pourra aussi en donner une traduction en Python.

Exercice 1.

1. Donner un algorithme de calcul de la puissance d'exposant n d'un entier naturel a . (Imiter l'algorithme précédent.) Combien de multiplications met-il en jeu ?
2. Proposer un algorithme de calcul de a^{2^n} qui met en jeu n multiplications.

Exercice 2.

Un nombre entier $b > 1$ est fixé.

1. Donner un algorithme qui teste la parité d'un entier naturel n .
2. Donner un algorithme qui retourne le reste de la division euclidienne d'un entier naturel n par b utilisant exclusivement des additions (ou des soustractions).
3. Donner un algorithme qui retourne le quotient de la division euclidienne d'un entier naturel n par b utilisant exclusivement des additions (ou des soustractions).

Exercice 3. Donner un algorithme qui retourne la suite des chiffres de la représentation en base b d'un entier naturel n . Dans le cas où $n = 0$, il doit retourner 0. Dans le cas où n est non nul, il doit afficher les chiffres jusqu'au dernier non nul, et s'arrêter.

Exercice 4. Donner un algorithme qui retourne la forme irréductible d'un nombre rationnel (par exemple sous forme d'un couple constitué du numérateur et du dénominateur).

Exercice 5. Donner un algorithme qui calcule le nombre de jours entre deux dates du calendrier grégorien (après le 15 octobre 1582). On rappelle que sont bissextiles les années multiples de 400 et les années multiples de 4 qui ne sont pas multiples de 100. Les autres sont normales.

Exercice 6. Un nombre entier naturel non nul plus petit que 5000 se représente en chiffres romains de la façon suivante. S'il s'écrit $c_3c_2c_1c_0$ en base 10, on juxtapose c_3 lettres M, c_2 lettres C, c_1 lettres X et c_0 lettres I.

1. Donner un algorithme qui retourne la représentation en chiffres romains d'un nombre entier naturel non nul plus petit que 5000.
2. En réalité, pour réduire le nombre de lettres de la représentation, on se sert d'autres lettres qui abrègent les groupes de cinq lettres identiques : un V remplace le groupe des cinq premiers I, un L celui des cinq premiers X et un D celui des cinq premiers C.

Donner un autre algorithme qui retourne la représentation en chiffres romains d'un nombre entier naturel non nul plus petit que 5000.

3. En pratique, les groupes de quatre et de neuf lettres identiques sont aussi abrégés. On commence par remplacer les groupes de neuf premières lettres identiques, puis ceux de cinq et enfin ceux de quatre : neuf I sont remplacés par IX, neuf X par XC, neuf C par CM, cinq I par V, cinq X par L, cinq C par D, quatre I par IV, quatre X par XL, quatre X par CD.

Donner un algorithme qui retourne cette dernière représentation en chiffres romains d'un nombre entier naturel non nul plus petit que 5000.

Programmes en Python

```
def power(a,n):
    p = 1
    k = 0
    while k < n:
        k += 1
        p *= a
    return p

def Power(a,n):
    p = 1
    for k in range(1, n+1):
        p *= a
    return p

def parity(n):
    m = n
    while m > 1:
        m -= 2
    if m == 0: return "pair"
    return "impair"

def remainder(a,b):
    r = a
    while r >= b:
        r -= b
    return r

def quotient(a,b):
    r = a
    q = 0
    while r >= b:
        r -= b
        q += 1
    return q

def digits(n,b):
    if n == 0: return 0
    m = n
    while m > 0:
        r = remainder(m,b)
        m = quotient(m,b)
        print r

def gcd(a,b):
    if a*b == 0: return a+b
    if a > b: return gcd(b,a-b)
    return gcd(a,b-a)
```

```
def irreducibleform(a,b):  
    d = gcd(a,b)  
    return a/d, b/d
```