Stage d'Algorithmique : Chiffre(s) romain(s)

P. Dusart

12 novembre 2009

1 Introduction

1.1 Algorithme

Un algorithme est un processus systématique de résolution d'un problème en présentant les étapes vers le résultat à quelqu'un d'autre (généralement un calculateur). En d'autres termes, un algorithme est un énoncé d'une suite d'opérations permettant de donner la réponse à un problème.

L'algorithmique a été systématisée par le mathématicien perse Al Khuwarizmi (né vers 780 - mort vers 850), auteur d'un ouvrage (souvent traduit par L'algèbre et le balancement) qui décrit des méthodes de calculs algébriques (en plus d'introduire le zéro des Indiens) et qui a laissé son nom (latinisé au Moyen Âge en Algoritmi) à ce processus.

1.2 Chiffre(s)

J'ai intitulé ce stage "chiffre(s) romain(s)" avec un pluriel éventuel. Pour son contenu, il est naturel de penser dans un premier temps à la conversion des nombres écrits en représentation romaine. Mais à l'Université de Limoges, nous avons une formation de Master en Cryptographie alors il convient de penser également à Chiffrement. A partir de là, il est évident que nous allons présenter le chiffre de César et le décodage des nombres romains.

1.3 Python

Python est un langage qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation grâce à des bibliothèques spécialisées à chaque traitement. Il est cependant particulièrement utilisé comme langage de script pour automatiser des tâches simples mais fastidieuses comme automatiser certains enchaînements d'actions répétitives.

On l'utilise également comme langage de développement de prototype lorsqu'on a besoin d'une application fonctionnelle avant de l'optimiser avec un langage de plus bas niveau.

Python est un langage de programmation interprété et orienté objet. Il fonctionne sous la plupart des environnements (Unix, Windows, Mac).

On éditera les fichiers scripts dans un éditeur de texte (ex : script.py) puis ils seront interprétés dans une fenêtre "Invite de commande" avec la commande :
`python script.py`

Il executera alors le programme ou affichera les erreurs de compilation.

2 Chiffre de César

Le chiffre de César consiste simplement à décaler les lettres de l'alphabet de 3 rangs vers la droite pour rendre le message illisible (chiffré). Ainsi la lettre A devient un D et ainsi de suite.

Algorithm 1 Chiffre de César

Entrées: Le texte clair
Entrées: La table de permutation (décalage de 3 lettres vers la droite)
Sorties: Le texte chiffré

```
Clair ← "A Demander ?"
TexteChiffre ←"" # Chaine vide
pour tout les lettres c dans le clair faire
    # On cherche la lettre c dans la table de permutation
    i ← 0
    tantque (i < 26) et (Table[i] ≠ c) faire
        i ← i + 1
    fin tantque
    si i est égal à 26 alors
        # Je suis arrivé à la fin de ma table sans trouver la lettre
        afficher Caractere non reconnu
    sinon {Caractère reconnu}
        # Je prend le caractère correspondant dans la table chiffrée
        CaractèreChiffré←TableChiffrée[i]
        # Je rajoute ce caractère à la fin de la chaine (concaténation)
        TexteChiffre←[TexteChiffre,CaractèreChiffré]
    finsi
fin pour
afficher TexteChiffre
```

Voici le programme correspondant en Python :

Algorithm 2 (PYTHON) Chiffre de César

```
Table='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
TableChiffree='DEFGHIJKLMNOPQRSTUVWXYZABC'
clair="BONJOUR"
cesar="" # initialisation chaine vide
for c in clair:
  i=0
  while ((i<26)  and (Table[i]<>c)):
    i=i+1
  if (i==26):
    print('caractere non reconnu : '+c)
  else:
    cesar=cesar+TableChiffree[i]
print(cesar)
```

Remarque : il suffit de modifier la table de permutation pour obtenir un programme de chiffrement par substitution monoalphabétique.

Algorithm 3 (PYTHON) Chiffre de César (plus informatique)

```
print "Entrez votre message : "
# Par exemple : Bonjour
saisie = raw_input()      # retourne toujours une chaine
saisie = saisie.upper()     # on met la chaine en majuscule
saisie = saisie.rstrip(' ') # on enleve les espaces
# print saisie
cesar = "" # initialisation chaine vide
for c in saisie:
  # concatenation de chaine
  cesar = cesar + chr((ord(c)-ord('A')+3) %26 +ord('A'))
print(cesar)
```

3 Conversion de nombres (représentation romaine)

Référence : `http://fr.wikipedia.org/wiki/Num%C3%A9ration_romaine`

Les chiffres romains étaient un système de numération utilisé par les Romains de l'Antiquité pour, à partir de seulement sept lettres, écrire des nombres entiers (mais pas le zéro, qu'ils ne connaissaient pas ; ou plus exactement qu'ils ne considéraient pas comme un nombre).

La numérotation a été normalisée dans l'usage actuel et repose sur quatre principes :
- Toute lettre placée à la droite d'une autre figurant une valeur supérieure ou égale à la sienne s'ajoute à celle-ci.
- Toute lettre d'unité placée immédiatement à la gauche d'une lettre plus forte qu'elle, indique que le nombre qui lui correspond doit être retranché au nombre qui suit.
- Les valeurs sont groupées en ordre décroissant, sauf pour les valeurs à retrancher selon la règle précédente.
- La même lettre ne peut pas être employée quatre fois consécutivement sauf M.

Lettre d'unité : I est une unité pour V et X, X est une unité pour L et C, C est une unité pour D et M.

Chiffre romain	I	V	X	L	C	D	M
Valeur	1	5	10	50	100	500	1000

Exemple :
"MCMLXXV" = 1 000 + (1 000 - 100) + 50 + 10+ 10 + 5 = 1975

Idée générale : le décodage s'effectue lettre après lettre de la gauche vers la droite. Nous ajoutons ou retranchons la lettre précédente selon la valeur de lettre en cours de decodage.

3.1 Version Itérative

Algorithm 4 (Itératif) Algorithme **convertir**

```
Entrées: romain : chaine
Sorties: entier
  tableau←['M','D','C','L','X','V','I']
  valeur←[1000,500,100,50,10,5,1]
  anc←-1
  valeur_lettre←0
  romain←0
  pour i de 0 à taille(romain)-1 faire
    cpt←0
    trouve←faux
    tantque (cpt<taille(tableau)) et non trouve faire
      si nombre[i]≠tableau[cpt] alors
        cpt←cpt+1
      sinon
        trouve← vrai
      finsi
    fin tantque
    si cpt = longueur(tableau)+1 # ou trouve = faux alors
      afficher  lettre non reconnue
    sinon
      afficher  valeur_lettre
      superieur←(anc≤cpt)
      si superieur alors
        romain←romain+valeur_lettre
      sinon
        romain←romain-valeur_lettre
      finsi
      valeur_lettre←valeur[cpt]
      anc←cpt
    finsi
  fin pour
  romain←romain+valeur_lettre
  afficher  "Valeur decimale :", romain
```

Algorithm 5 (PYTHON) Conversion Chiffres Romains

```
# nombre="MDCCCCXII"
nombre="CMXCVIII" # ou nombre="IIM" qui ne convient pas car non normalise
#nombre="XLIV"
tableau=['M','D','C','L','X','V','I']
valeur=[1000,500,100,50,10,5,1]
anc=-1
valeur_lettre=0
romain=0

for i in range(len(nombre)):
  #recherche du caractere dans le tableau
  cpt=0
  trouve=0
  while (cpt<len(tableau)) and not(trouve):
    if nombre[i]<>tableau[cpt]:
      cpt=cpt+1
    else:
      trouve=1
  if cpt==len(tableau)+1: # ou trouve ==0
    print('lettre non reconnue')
  else:
    print(valeur_lettre)
    superieur=(anc<=cpt)
    if superieur:
      romain=romain+valeur_lettre
    else:
      romain=romain-valeur_lettre
    valeur_lettre=valeur[cpt]
    anc=cpt
romain=romain+valeur_lettre
print("Valeur decimale ", romain)
```

3.2 Version Récursive

La fonction est récursive si elle s'appelle elle-même. L'exemple classique est le suivant

```
# Fonction factorielle en python
 def factorielle(x):
     if x == 0:
         return 1
     else:
         return x * factorielle(x-1)
```

Algorithm 6 Algorithme **decode**

Entrées: un chiffre *c* romain
Sorties: La valeur décimale correspondante

```
si c est égal à 'M' alors
  renvoyer(1000)
finsi
si c est égal à 'D' alors
  renvoyer(500)
finsi
si c est égal à 'C' alors
  renvoyer(100)
finsi
si c est égal à 'L' alors
  renvoyer(50)
finsi
si c est égal à 'X' alors
  renvoyer(10)
finsi
si c est égal à 'V' alors
  renvoyer(5)
finsi
si c est égal à 'I' alors
  renvoyer(1)
finsi
```

Algorithm 7 (Récursif) Algorithme **convertir**

Entrées: romain : chaine
Sorties: entier

```
si Longueur(chaine)=1 alors
  Renvoyer(decode(romain))
sinon
  si decode(romain[0])<decode(romain[1]) alors
    Renvoyer(convertir(romain[1 : fin])-decode(romain[0])
  sinon
    Renvoyer(convertir(romain[1 : fin])+decode(romain[0])
  finsi
finsi
```

Algorithm 8 (PYTHON) Récursif : Conversion Chiffres Romains

```
nombre="CMXCVIII"
tableau=['M','D','C','L','X','V','I']
valeurs=[1000,500,100,50,10,5,1]

def valeur(r):
  if r in tableau:
    i=0
    while (r<>tableau[i]):
      i=i+1
    return(valeurs[i])
  else:
    print("caractere non reconnu")
    return(0)


def convertir(romain):
  if len(romain)==1:
    return valeur(romain)
  elif valeur(romain[0])<valeur(romain[1]):
    return convertir(romain[1:])-valeur(romain[0])
  else:
    return convertir(romain[1:])+valeur(romain[0])

print(valeur("C"))
print(convertir(nombre))

# ***********************************************
# Programmation Alternative de la fonction Valeur
# ***********************************************
def valeur(r):
  if   r=='M':
   return(1000)
  elif r=='D':
   return(500)
  elif r=='D':
   return(500)
  elif r=='C':
   return(100)
  elif r=='L':
   return(50)
  elif r=='X':
   return(10)
  elif r=='V':
   return(5)
  elif r=='I':
   return(1)
  else:
    print("caractere non reconnu")
    return(0)
```

3.3 Fonction réciproque

Algorithm 9 (PYTHON) Transformer une valeur décimale en chiffres romains

```
# -*- coding: latin-1 -*-
tableau=['M','CM','D','CD','C','XC','L','XL','X','IX','V','IV','I']
valeur=[1000,900,500,400,100,90,50,40,10,9,5,4,1]

print("Valeur décimale ?")
chaine_decimale=raw_input()
# decimal=1912 # valeur à convertir en chiffres romains
decimal=int(chaine_decimale)
romain=""

for i in range(len(valeur)):
  repetition=decimal / valeur[i]
  for cpt in range(repetition):
    romain=romain+tableau[i]
  decimal=decimal - repetition*valeur[i]
print(romain)
```

4 Références utiles

- Livre conseillé :
  Algorithme, Raisonner pour concevoir
  Christophe HARO, Editions ENI, Mai 2009, 45 euros
  ISBN 978-2-7460-4844-7
- Télécharger Python : `http ://www.python.org`
  Emplacement exact pour Windows :
  `http ://www.python.org/ftp/python/2.6.3/python-2.6.3.msi`
- Livre conseillé pour Python :
  Introduction à Python,
  Lutz & Ascher, Editions O'Reilly,
  ISBN 2-84177-089-3
- Tutoriel Python de Gérard Swinnen
  `http ://www.framasoft.net/IMG/pdf/python_notes-2.pdf`
  `http ://python.ftp-developpez.com/cours/TutoSwinnen/fichiers/python_notes.pdf`