Faculté des sciences
Université d'Aix*Marseille

Programmation 1
samedi 24 février 2018

Université Nazi Boni
UNB

Mathématiques discrètes, master de mathématiques, 1re année, Thierry Coulbois

Calcul avec les grands nombres.
Le but de ce TP est de pouvoir effectuer des calculs tels que

$$328910383922011847560152 64 \times 33429322399191033124857663320539209 8$$

Dans ce TP un nombre est stocké comme un tableau de chiffres décimaux. Dont la taille maximale est fixée dans une constante `define MAX_LENGTH 400`. Chaque case du tableau contient exactement un chiffre.
Notre programme doit fonctionnner avec des nombres entiers positifs. Vous pourrez l'étendre pour travailler aussi avec les nombres négatifs.
Rappelons que `17%10` est 7 le reste de la division euclidienne de 17 par 10. Le quotient est `17/10` en `C` et `17//10` en python.
Vous écrirez, testerez, documenterez successivement des fonctions :

1. `void affiche(int nombre[])` qui prend un tableau de chiffres en entrée et affiche correctement le nombre (on pourra introduire un espace tous les trois caractères).
2. `void somme(int nombre1[], int nombre2[], int resultat[])` qui additionne deux tableaux de chiffres.
3. `void produit(int nombre1[], int nombre2[], int resultat[])` qui multiplie deux tableaux de chiffres.
4. `void lire(int nombre[], char mot[])` qui lit une chaine de caractère et la transforme en un tableau de chiffres.

5. Écrivez une fonction qui convertit un tableau de chiffres décimaux en un tableau de chiffres dans une base quelconque.
6. Convertissez des écritures dans une base dans une écriture dans n'importe quelle base.

7. La cryptographie RSA est basée sur les nombres premiers et les calculs modulo n. Écrivez encore des fonctions pour pouvoir effectuer la division euclidienne.
8. Écrivez une fonction pour pouvoir élever à une puissance modulo n. (le nombre, l'exposant et le modulo sont tous les trois des grands nombres sous la forme de tableaux de chiffres).

Voici ci-contre une clé RSA

Le fichier ci-contre est écrit dans le standard `base-64-encoded-key`.
Chaque caractère est un des 64 caractères ASCII :

```
AB...ZO1...9ab...z/+
```

Il faut donc d'abord convertir le fichier ci-contre en un nombre en base 64.
Il faut ensuite convertir ce nombre en un nombre en base 256 (ou en hexadécimal au choix). Remarquez que 4 caractères du fichier initial codent 3 octets. Par exemple `MIIB` code `30 82 01` (en hexadécimal).
Le modulo commence au 13e caractère : `+xGZ`... et se termine pour laisser 5 octets à la fin (les trois derniers octets pour l'exposant).

Améliorez votre programme pour qu'il lise cette clé RSA, affiche le modulo et l'exposant.

```
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEA+xGZ/wcz9ugFpP07Nspo6U17l0YhFiFpxxU4pTk3Lifz9R3zsIsu
ERwta7+fWIfxOo2O8ett/jhskiVodSEt3QBGh4XBipyWopKwZ93HHaDVZAALi/2A
+xTBtWdEo7XGUujKDvC2/aZKukfjpOiUI8AhLAfjmlcD/UZ1QPhOmHsglRNCmpCw
mwSXA9VNmhz+PiB+Dml4WWnKW/VHo2ujTXxq7+efMU4H2fny3Se3KYOsFPFGZ1TN
QSYlFuShWrHPtiLmUdPoP6CV2mML1tk+l7DIIqXrQhLUKDACeM5roMx0kLhUWB8P
+0uj1CNlNN4JRZlC7xFfqiMbFRU9Z4N6YwIDAQAB
-----END RSA PUBLIC KEY-----
```