Techniques Algorithmiques et Programmation

TP noté – 2h40

Consignes

Vous avez le droit de consulter les notes de cours [1] sur Internet. C'est la seule ressource documentaire que vous êtes autorisé à consulter sur Internet. Vous pouvez utiliser vos notes personnelles (TDs, vos programmes, vos notes de cours). C'est une épreuve individuelle, vous n'avez pas le droit de communiquer avec vos voisins.

Commencer par télécharger et décompresser l'archive `tp1.tgz` disponible à la l'adresse :

http://dept-info.labri.fr/~gavoille/tp1.tgz

En fin d'épreuve, il faudra renommer le fichier `tp1.c` avec votre NOM et Prénom, par exemple `GAVOILLE_Cyril.c`. Vous devrez envoyer par courriel votre programme ainsi renommé au responsable de votre groupe :

- A1 ou A5 : gavoille@labri.fr
- A2 ou A4 : henri.derycke@labri.fr
- A3 : thomas.bellitto@labri.fr

Enfin, avant de partir, signer la feuille d'émargement et vérifier que votre courriel a bien été reçu.

La notation prendra en compte :

- la lisibilité de votre code (commentaires)
- l'abscence de fuite mémoire
- les performances, testables avec la commande `time tp1 ...`

Sujet : multiplication rapide

Dans ce TP noté il s'agit de coder les algorithmes de multiplications de grands nombres entiers vus en cours. On utilisera le type prédéfini `number` qui correspond à un tableau de chiffres (`digit[]`) avec un certain nombre de chiffres (`n`), voir `tp1.h`. Les nombres sont codés dans une certaine base (`BASE`) qui est une variable globale. Elle peut être modifiée mais c'est toujours un entier entre 2 et 10. Le point important est que l'indice le plus faible dans le tableau de chiffres correspond aux unités (`digit[0]`). Par exemple, le tableau de 4 chiffres `digit[]={5,3,0,1}` en `BASE=10` représentera l'entier

$$x = 1035 = 5 \cdot 10^0 + 3 \cdot 10^1 + 0 \cdot 10^2 + 1 \cdot 10^3 \ .$$

Vous avez à programmer cinq opérations arithmétiques sur les entiers. L'addition, la soustraction, et trois variantes de la multiplication : la multiplication standard (celle apprise à l'école – non-récursive avec deux boucles `for`), la multiplication récursive avec 4 appels récursifs et la multiplication selon la méthode de Karastuba avec 3 appels récursifs. Dans tous les cas, les entiers manipulés (les opérandes) seront positifs ou nuls. Donc vous n'aurez pas à vous préoccuper du signe.

Le fichier que vous avez à éditer et à remettre à la fin TP est `tp1.c`. Ce programme que vous avez à compléter doit, s'il est exécuté sur la ligne de commande avec un nom de fichier test, permettre

1. http://dept-info.labri.fr/~gavoille/UE-TAP-cours.pdf

d'appliquer une des cinq opérations sur deux nombres contenus dans le fichier et d'afficher le résultat. Le programme a donc pour premier objectif d'afficher le résultat d'une des cinq opérations. Le format des fichiers tests est détaillé dans le `.h`, mais en gros il comporte trois lignes comme celles-ci :

```
10 a
1234
 345
```

Le `10` indique la base, `a` l'opération d'addition, `1234` et `345` étant les deux opérandes. Les caractères non numériques (comme les espaces) dans les opérandes sont simplement ignorés. Le point important est que les nombres sont écrits dans le fichiers de manière standard : l'unité est à droite. Le premier opérande représente donc $x = 1234$ et le second $y = 345$. Normalement, le résultat devrait être :

```
> tp1 n1.txt
> 1579
```

La fonction permettant de lire un tel fichier de test est déjà écrite (`read()`). De même, dans le squelette du programme, le `main()` permet de lire le fichier (avec `read()`), d'exécuter l'une des cinq opérations et d'afficher le résultat (avec `write()` qui est déjà écrite). Vous n'avez donc à programmer que les cinq opérations demandées. Vous pouvez bien sûr utiliser plus de cinq fonctions. Le correcteur doit pouvoir tester vos cinq opérations en jouant sur ses propres fichiers de tests.

Il est possible de générer des fichiers de tests aléatoires avec une commande du type :

```
> tp1 2 m 20 1 > n2.txt
> cat n2.txt
2 m
11110011010110000001
1
```

qui a pour effet de générer deux nombres en base `2` : le premier de `20` chiffres, et le second d'`1` seul chiffre. Le paramètre `m` signifie la multiplication standard (le codage de l'opération est dans le `.h` et aussi dans le `main()`). Cela vous permettra de tester chacune de vos fonctions, en particulier pour de grands nombres. Vous pouvez bien sûr toujours éditer à la main des fichiers tests.