POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5) **Programmation Structurée 2011/2012**
http://laure.gonnord.org/pro/teaching/

TP - exos supplémentaires

Objectifs

Ce TP contient des exercices supplémentaires à faire en TP *après qu'un enseignant ait validé votre solution pour le tp courant.*

1 Algorithmique sans tableau ni pointeur

Exercice 1 *On va améliorer le calcul de x^n en faisant moins de n opérations. La méthode expliquée ici a pour nom* **exponentiation rapide**. *Voici un exemple :*

$$x^{23} = x * (x^2)^{11} = (x * x^2) * (x^4)^5 = (x * x^2 * x^4) * (x^8)^2 = x * x^2 * x^4 * x^{16}$$

Donc, si on calcule les puissances paires de x au fur et à mesure, on réalise moins d'opérations (combien, sur l'exemple ?).

1. *En nommant les différentes parties de l'égalité :* `res`, `puiss` *et* `k` *judicieusement, obtenir l'invariant de boucle suivant : «A chaque tour de boucle, on a $res*puiss^k = x^n$». Comment obtenir res, puiss, k à la boucle suivante ? Quand s'arrête-t-on ?*
2. *Écrire la fonction* `puiss_dicho` *qui prend en argument les entiers x et k et qui calcule x^k avec cet algorithme. En pseudo code, puis en C.*
3. *Donner en fonction de n le nombre de multiplications $C(n)$ que réalise cet algorithme. On trouvera $C(n) \leq 2 \log n$. On commencera par établir $C(n) \leq 2 + C(n/2)$ puis on posera $k = 2^n$.*
4. *Écrire le même algo en récursif.*

2 Tableaux

2.1 Tableaux d'entiers

Exercice 2 (Tri bulle) *(Après le tp de tris uniquement)*
Concevoir, implémenter, tester et évaluer la complexité de la variante du tri-sélection suivant : Au lieu de :

"chercher le minimum du sous-tableau restant (à droite de l'indice courant) PUIS le permuter avec la case d'indice courant"

on fait :

"Faire des permutations de voisins à partir de la case $N-1$ jusqu'à la case d'indice courant, si les voisins ne sont pas dans le bon ordre".

Trouver un invariant qui permet de prouver la correction de l'algorithme.

EXERCICE 3 *Coder en C le tri rapide.*

EXERCICE 4 (MÉDIANE) *1. Écrire un algorithme pour calculer le médian d'un tableau (autant d'éléments supérieurs qu'inférieurs). Quelle est la complexité en nombre de comparaisons ?*

2. (Difficile) Sans trier, écrire un algorithme pour le médian, en O(n). On pourra s'inspirer du tri rapide.

2.2 Chaînes de caractères

(après le TP4 uniquement)

EXERCICE 5 *Écrire une fonction qui prend en argument deux chaînes de caractères (sous forme de tableaux de caractères de taille* `MAX` *dans lesquels les fins de chaînes sont repérées par* `'\0'`*) et qui dit si ces deux chaînes sont anagrammes l'unes de l'autre. Par exemple,* `marine` *et* `irmane` *sont anagrammes.*

EXERCICE 6 *Sur le modèle du tp4, écrire les programmes suivants (en utilisant des fonctions/actions !) :*

1. *Déterminer et afficher le nombre d'apparitions d'une lettre donnée par l'utilisateur dans un tableau. On utilisera* `getchar()` *:*

   ```
   On trouve la lettre 'c' 4 fois.
   ```

2. *Déterminer le premier indice d'apparition de chacune des lettres et afficher sous la forme :*

   ```
   'a' n'apparait jamais
   'b' apparait pour la première fois en position 0
   'c' apparait pour la première fois en position 35
   ...
   ```

   On n'utilisera pas de tableau, et pour chaque lettre on utilisera une boucle while.

3. *Déclarer et initialiser un tableau* `occurrence` *qui va nous servir à compter le nombre d'apparitions de chaque lettre. De quelle taille est-il ?*

4. *Remplir le tableau* `occurrences` *avec le nombre d'apparition de chaque lettre dans le tableau* `tab`*, et afficher le résultat ensuite sous la forme :*

   ```
   'a' apparait 0 fois
   'b' apparait 1 fois
   'c' apparait 4 fois
   ...
   ```

5. *Afficher le contenu du tableau* `tab` *dans l'ordre alphabétique. On utilisera le tableau* `occurrences`*, et si le caractère* `'a'` *n'apparaît pas, on n'affichera pas* `'a`*.*

6. *Utiliser le tableau* `occurrences` *pour modifier le tableau* `tab` *afin qu'il soit trié par ordre alphabétique, puis afficher le tableau* `tab` *afin de vérifier qu'il est trié. Indication : avec le tableau* `occurrences`*, on connaît le nombre d'occurrences de chaque caractères. On peut donc écraser le contenu du tableau* `tab` *sans perdre la moindre donnée.*

EXERCICE 7 *Sur le modèle du TP8 :*

1. *Écrire un programme qui écrit un fichier à l'envers. Sur le fichier* `toto`*, le programme donnera :*

   ```
   fuolp
   eludib
   curt
   ```

   Indication : On se demandera comment mémoriser les caractères du fichier
2. *Écrire un programme qui dit si un certain mot (un "motif") est présent dans un fichier.*
3. *Écrire un programme qui dit si les parenthèses d'un fichier sont bien imbriquées. On ignore les autres caractères. Dans* `(())()` *les parenthèses sont bien imbriquées, mais pas dans* `(((((`.

3 Pointeurs

3.1 Pointeurs d'entiers

Exercice 8 *Écrire un algorithme qui prend en paramètre un tableau et qui calcule le min et le max de ce tableau (de façon simultanées). Tester dans le main.*

3.2 Fonctions de la lib standard pour les chaînes

Les exercices de cette section seront réalisés avec l'aide des fonctions de la librairie `string.h`. Les chaînes de caractères auront une taille statique déclarée à l'avance : `char ch1[N],ch2[N]` lorsque c'est possible. **Créez un nouveau .c pour cette section**

Exercice 9 *À l'aide de* `fgets` *(*`man 3 fgets`*), demander une chaîne de caractères à l'utilisateur, et imprimer la taille de celle-ci (*`strlen`*). Remarquer que* `fgets` *récupère aussi le saut de ligne, et écrire une fonction* `char* monfgets()` *qui retourne la chaîne donnée par l'utilisateur, sans le saut de ligne.*

Exercice 10 *À l'aide de la fonction précédente, demander deux chaînes de charactères à l'utilisateur (de taille max $NMAX$ constante). Utiliser* `strcat` *pour concaténer ces deux chaînes et afficher le résultat. La chaîne résultat est forcément un pointeur.*

Exercice 11 *Écrire une fonction qui détermine si deux chaînes de caractères sont préfixes l'une de l'autre, en utilisant* `strcmp`*. Tester.*

Exercice 12 *Écrire une fonction qui détermine si une chaîne de caractères est une sous-chaîne d'une deuxième. On utilisera* `index` *et un pointeur pour parcourir la deuxième chaîne.*