POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5) **Programmation Structurée 2012/2013**

http://laure.gonnord.org/pro/teaching/

TD/TP - exos supplémentaires

Objectifs

Cette feuille contient des exercices supplémentaires à faire en TD/TP *après qu'un enseignant ait validé votre solution pour le td/tp courant.* Les exercices ne sont pas dans l'ordre croissant de difficulté. Les algorithmes seront *expliqués* et correctement indentés.

1 Algos sans tableaux

Exercice 1 (TD) *Écrire un algorithme, qui étant donnés nbH, nbM, nbS dénotant respectivement un nombre d'heures, minutes, secondes, décompte cette durée (en affichant tous les couples (h, m, s)) et affiche "BOUM" en fin de décompte.*

Exercice 2 (TD) *Division euclidienne :*

- *Comment calculer le quotient et le reste de la division euclidienne de a par b, en n'effectuant que des soustractions/additions et des tests.*
- *Écrire l'algorithme en pseudo langage, a et b sont des données (avec $a \geq b$) ; on demande d'imprimer les résultats sur la sortie standard.*
- *Pourquoi ne peut-on pas utiliser de fonction ?*

Exercice 3 (TD/TP) *Écrire une version itérative du calcul de la suite de Fibonacci.*

Exercice 4 (FS pour GIS,TD) *Écrire un algorithme permettant à l'utilisateur de saisir une suite d'entiers au clavier (terminée par 0) et de déterminer le nombre de fois où un entier n_1 donné est immédiatement suivi d'un entier n_2 donné aussi (on n'utilisera pas de tableau).*

Exercice 5 (TD/TP) *(Syracuse) Soit la suite définie par $u_0 = N$ (entier positif) et $u_{n+1} = \frac{u_n}{2}$ si u_n est pair et $u_{n+1} = 3u_n + 1$ sinon. La conjecture de Syracuse dit que cette suite passe par 1. Écrire un algorithme qui vérifie cette conjecture pour tous les N de min à max.*

Exercice 6 (TD) *Écrire un algorithme* `logint` *qui prend en paramètre deux entiers n et p, et retourne la plus grande puissance q de p dans n. Cet algo calcule en outre le coefficient multiplicateur d et le reste r tels que $n = d \times p^q + r$ avec $r < p^q$ et $d < p$. r et d seront passés en paramètre-résultats. Il est interdit d'utiliser la fonction* `log`*. On testera à l'aide des égalités $27 = 1 \times 2^4 + 11$, $98 = 9 \times 10^1 + 8$.*

Exercice 7 (Quick 2010, TD) *Écrire un algorithme qui demande un entier à l'utilisateur et qui lui donne le nombre de chiffres que contient cet entier (dans la représentation décimale). Rappel : $153\%10 = 3$ et $153/10 = 15$. Quelle est le nombre d'opérations effectuées par votre algorithme ?*

Exercice 8 (TD) *Écrire un algorithme récursif permettant de calculer le PGCD de deux entiers positifs donnés.*

Indication : *Soient a et b deux entiers positifs. On a :*

- *pgcd(a,b) = a si b = 0,*
- *pgcd(a,b) = pgcd(b,reste(a,b)) si b ≠ 0 avec reste(a,b) le reste de la division entière de a par b.*

Exercice 9 (Exponentiation rapide, TD/TP) *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.*

Exercice 10 (TD) *Soient a et b deux entiers positifs. Ecrire un algorithme permettant de calculer $a \times b$ en utilisant l'addition.*

Exercice 11 (TD) *Écrire un algorithme permettant de calculer le salaire hebdomadaire d'un ouvrier connaissant le nombre d'heures effectuées dans la semaine et le salaire horaire.*

Les 35 premières heures sont payées au salaire horaire, les 10 suivantes avec un supplément de 50% et les autres avec un supplément de 75%.

2 Tableaux

2.1 Tableaux d'entiers

Exercice 12 (TD) *Calculer la somme des éléments d'un tableau d'entiers donné.*

Exercice 13 (TD) *Retourner l'indice du premier élément nul d'un tableau.*

Exercice 14 (TD) *Compter le nombre d'éléments nulls d'un tableau.*

Exercice 15 (TD) *Calculer simultanément :*

- *La somme des éléments positifs d'un tableau ;*
- *La somme des élément négatifs d'un tableau.*

Exercice 16 (TD) *Écrire un algorithme qui calcule le min et le max d'un tableau d'entiers donné, puis modifier cet algorithme pour calculer le nombre d'occurences du minimum et du maximum.*

Exercice 17 (Quick 2010) *Écrire une* **fonction** *qui prend un tableau d'entiers en paramètres et répond true si ce tableau est trié croissant, false sinon.*

Exercice 18 (médiane, TD)
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.*

Exercice 19 (Quick 2010) *Écrire un algorithme qui calcule le schtroumpf de deux tableaux (qui ne sont pas de même taille !) passés en paramètre . Pour calculer le schtroumpf, il faut multiplier chaque élément du tableau 1 par chaque élément du tableau 2, et additionner le tout. Par exemple si l'on a :*
- *Tableau 1 : 4 8 7 12*
- *Tableau 2 : 3 6*

Le Schtroumpf sera :

$$3*4+3*8+3*7+3*12+6*4+6*8+6*7+6*12=279$$

Deux paramètres pourront être ajoutés : les tailles des deux tableaux d'entrée.

Exercice 20 (Source : JF pour GIS) *Tester si un tableau est symétrique. Par exemple, les tableaux suivants sont symétriques :*
- *3,4,1,4,3*
- *1,2,2,1*

Exercice 21 (TD) *Source : FS pour GIS*
Soient n un entier et $M[n][n]$ une matrice d'entiers. Déterminer l'indice de la colonne dont la somme des cœfficients est la plus grande.

Exercice 22 (TD) *(même source) Soient n un entier et $M[n][n]$ une matrice de booléens représentant une relation R sur les entiers de la façon suivante : $M[i][j] = VRAI$ si et seulement si iRj Écrire des algorithmes permettant de déterminer si R est réflexive, symétrique, transitive.*

Exercice 23 (recherche dichotomique,TD) *Écrire un algorithme de recherche dans un tableau trié croissant en suivant la méthode suivante :*
- *regarder la case du milieu*
- *si l'élément n'est pas dans cette case, soit il est inférieur, à ce moment on recherche dans le sous tableau de gauche, soit il est supérieur (et alors ? ? ?)*

On s'inspirera fortement de l'écriture du trifusion, faite dans le cours, et on se demandera quand est-ce qu'on s'arrête.

Exercice 24 (TD) *Écrire un algorithme pour renverser un tableau d'entiers* en place.

Exercice 25 (TD) *Écrire un algorithme pour dire si un tableau d'entiers est un palindrôme (c'est-à-dire que la lecture des cases de gauche à droite et de droite à gauche donne un résultat identique)*

Exercice 26 (Tri bulle TD/TP) *(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 27 (TP) *Coder en C le tri rapide.*

2.2 Chaînes de caractères

(après le TP4 uniquement)

Exercice 28 (TD, TP) *Écrire un algorithme 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 29 (TP) *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 30 (TP) *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

Exos à ne faire qu'après avoir eu le cours.

Exercice 31 (TP) *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.*

Exercice 32 (TD) *Soit le programme ci-dessous constitué d'une action* `conversion_hms` *prenant en paramètre un nombre de secondes à convertir en heures, minutes et secondes et du* `main()` *appelant cette action.*

Faire le schéma d'exécution du programme en prenant pour valeur saisie par l'utilisateur 3732.

```
void conversion_hms(int *pH, int *pM, int *pS){
   *pH = *pS / 3600;
   *pM = (*pS % 3600) / 60;
   *pS = *pS % 60;
}
int main(){
   int h, m, s;
   printf("Veuillez saisir le nombre de secondes à convertir : ");
   scanf("%d",&s);
   conversion_hms(&h, &m, &s);
   printf("Le resultat de la conversion donne : %d h %d mns %d s\n", h,m,s);
}
```

3.1 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 33 *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 34 *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 35 *Écrire une fonction qui détermine si deux chaînes de caractères sont préfixes l'une de l'autre, en utilisant* `strcmp`*. Tester.*

Exercice 36 *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.*