Faculté des sciences Dhar EL-Mehraz
Département d'Informatique
SMI - S3
Prof. Noureddine En-nahnahi

Algorithmique II
Exercices de TP 1-

L'objectif de ce TP est de permettre aux étudiants de métriser la compétence d'analyse d'un problème et sa décomposition en sous-problèmes et en écrire un algorithme qui fait appel aux sous-programmes ou fonctions et procédures.

Commencer dans tous les exercices par l'écriture d'un programme de test.

Exercice1:

Echange dans l'ordre croissant

1) Ecrire une procédure *Echange2* sur 2 réels a, b qui échange éventuellement a et b, pour que l'état de sortie soit $a \leq b$. Ecrire un algorithme principal d'appel.

2) Ecrire une procédure *Echange3* sur 3 réels a, b, c qui appelle Echange2. L'état de sortie est $a \leq b \leq c$. Ecrire plusieurs versions.

3) Ecrire une procédure *Echange4* sur 4 réels a, b, c, d qui appelle Echange2 et Echange3. L'état de sortie est $a \leq b \leq c \leq d$. Ecrire plusieurs versions.

Exercice2:

Exponentielle

Faire une fonction expn(x,n) qui calcule la valeur approchée de e^x en utilisant la formule :

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Exercice 3:

Nombres parfaits

1) Ecrire une fonction qui retourne le nombre des entiers parfaits entre deux entiers passés en paramètres. NB : Un nombre parfait est un entier qui est égal à la somme de ses diviseurs (Exemple : le nombre 6 est parfait car 6=1+2+3)

Faculté des sciences Dhar EL-Mehraz
Département d'Informatique
SMI - S3

Algorithmique II
Exercices de TD 2

Les Fonctions (Série 2/2)

Prof. Noureddine En-nahnahi

Exercice 1: Minimum et omme d'un tableau

a) Écrire une fonction récursive Min qui détermine la valeur minimale un tableau d'entiers T comportant N éléments, et défini sur l'intervalle [0..Nmax-1].

b) Ecrire une fonction qui a pour paramètre un tableau de nombres et qui renvoie la valeur de la somme de ses éléments.

Exercice 2: PGCD

Écrire la fonction récursive PGCD qui calcule le plus grand commun diviseur de deux entiers naturels. Les relations suivantes permettant le calcul du PGCD :

PGCD(a, a) = a

PGCD(a, b) = PGCD(a-b, b) si a > b

PGCD(a, b) = PGCD(a, b-a) si a < b

PGCD(a, b) = PGCD(r, b) si a > b et a = b * q + r

PGCD(a, b) = PGCD (b, a)

si a % b = 0 alors PGCD(a,b)=b

Exercice 3: Recherche

a) Écrire la fonction récursive ExisteCar qui teste la présence d'un caractère dans une chaîne de caractères :

ExisteCar : fonction (C : caractère, X : chaîne) ⟶ booléen

{ ExisteCar(C,X) désigne vrai si le caractère C est présent dans la chaîne X }

Exercice 4: Chaine Palindrome

Ecrire une fonction récursive qui détermine si une chaîne donnée est un palindrome :

Estpalindrome : fonction (X : chaîne) ⟶ booléen

{ Estpalindrome(X) désigne vrai si la chaîne X est un palindrome }

Exemple : Estpalindrome("TRALALA") = faux

Estpalindrome("LAVAL") = vrai

Faculté des sciences Dhar EL-Mehraz
Département d'Informatique
SMI - S3

Algorithmique II
Travaux dirigés et travaux pratiques sur les Tris

Prof. Noureddine En-nahnahi

Exercice 1 : Tri à bulles

L'algorithme de tri à bulles parcourt la liste, et compare les couples d'éléments successifs. Lorsque deux éléments successifs ne sont pas dans l'ordre croissant, ils sont échangés. Après chaque parcours complet de la liste, l'algorithme recommence l'opération. Lorsqu'aucun échange n'a lieu pendant un parcours, cela signifie que la liste est triée : l'algorithme peut s'arrêter.

1) (TD) Triez le tableau suivant en donnant toutes les étapes intermédiaires à l'aide du tri à bulles.

3	5	2	0	1	4

2) (TD) Donnez en pseudo-code, l'implémentation de la fonction `tri_bulles( tableau )` qui prend en paramètre un tableau et le tri à l'aide de l'algorithme de tri à bulles.

3) (TD) Donnez la complexité de cet algorithme.

4) (TP) Implémentez en Java la fonction `void tri_bulles( int tableau[] )` qui ordonne les éléments du tableau suivant le tri à bulle

Exercice 2 : Tri par sélection

Il consiste en la rechercher du plus grand élément (ou le plus peit) dans le tableau que l'on va replacer à sa position finale c'est-à-dire en dernière position (ou en première), puis on recherche le second plus grand élément (ou le second plus petit) que l'on va replacer également à sa position finale c'est-à-dire en avant-dernière position (ou en seconde), etc., jusqu'à ce que le tableau soit entièrement trié.

1) (TD) Triez le tableau suivant en donnant toutes les étapes intermédiaires à l'aide du tri par sélection.

3	5	2	0	1	4

2) (TD) Donnez en pseudo-code, l'implémentation de la fonction `tri_bulle( tableau )` qui prend en paramètre un tableau et le tri à l'aide de l'algorithme de tri par sélection.

3) (TD) Donnez la complexité de cet algorithme.

4) (TP) Implémentez en Java la fonction `void tri_bulle( int tableau[] )` qui ordonne les éléments du tableau suivant le tri par sélection.

Exercice 3 : Tri par insertion

Le tri par insertion consiste à trier la liste au fur et à mesure que l'on ajoute un élément. Pour ce faire, on parcourt la liste en selectionnant les éléments dans l'ordre. Pour chaque élément selectionner, on le compare avec les éléments précédents qui ont déjà été ordonné jusqu'à trouver sa place. Il ne reste plus qu'à décaler les éléments du tableau pour insérer l'élément considéré à sa place dans la partie déjà triée.

1) (TD) Triez le tableau suivant en donnant toutes les étapes intermédiaires à l'aide du tri par insertions.

3	5	2	0	1	4

2) (TD) Donnez en pseudo-code, l'implémentation de la fonction `tri_bulles( tableau )` qui prend en paramètre un tableau et le tri à l'aide de l'algorithme de tri par insertions.

3) (TD) Donnez la complexité de cet algorithme.

4) (TP) Implémentez en Java la fonction `void tri_bulles( int tableau[] )` qui ordonne les éléments du tableau suivant le tri par insertions.